



**RANCANG BANGUN *NON PLAYABLE CHARACTER*  
CERDAS BERBASIS *LARGE LANGUAGE MODEL*  
DAN COMPUTER VISION PADA MINECRAFT  
DENGAN KEMAMPUAN INTERAKSI NATURAL DAN  
PENGENALAN EMOSI PLAYER**

**SKRIPSI**

**ARJUNA DWI DAMARA PUTRA 2207411006**

**PROGRAM STUDI TEKNIK INFORMATIKA  
JURUSAN TEKNIK INFORMATIKA DAN KOMPUTER  
POLITEKNIK NEGERI JAKARTA**

**2026**

## **SURAT PERNYATAAN BEBAS PLAGIARISME**

Saya yang bertanda tangan di bawah ini:

Nama : Arjuna Dwi Damara Putra  
NIM : 2207411006  
Jurusan/Program Studi : T.Informatika dan Komputer/Teknik Informatika  
Judul skripsi : Rancang Bangun Non Playable Character (NPC)  
Cerdas Berbasis Large Language Model (LLM) dan  
Computer Vision pada Minecraft dengan Kemampuan  
Interaksi Natural dan Pengenalan Emosi Player

Menyatakan dengan sebenarnya bahwa skripsi ini benar-benar merupakan hasil karya saya sendiri, bebas dari peniruan terhadap karya orang lain. Kutipan pendapat dan tulisan orang lain ditunjuk sesuai dengan cara-cara penulisan karya ilmiah yang berlaku.

Apabila di kemudian hari terbukti atau dapat dibuktikan bahwa dalam skripsi ini terkandung ciri-ciri plagiat dan bentuk-bentuk peniruan lain yang dianggap melanggar peraturan, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Jakarta, 07 Juni 2026

Yang membuat pernyataan

Arjuna Dwi Damara Putra

NIM. 2207411006

## LEMBAR PENGESAHAN

Skripsi diajukan oleh:

Nama : Arjuna Dwi Damara Putra  
NIM : 2207411006  
Program Studi : Teknik Informatika  
Jurusan : Teknik Informatika dan Komputer  
Judul skripsi : Rancang Bangun Non Playable Character (NPC)  
Cerdas Berbasis Large Language Model (LLM) dan  
Computer Vision pada Minecraft dengan Kemampuan  
Interaksi Natural dan Pengenalan Emosi Player

Telah diuji oleh tim penguji dalam Sidang Skripsi pada hari Rabu, Tanggal 17,  
Bulan Juni, Tahun 2026 dan dinyatakan **LULUS**.

Disahkan Oleh

Pembimbing I : Rizki Elisa Nalawati, S.T., M.T. ( )

Penguji I : Euis Oktavianti, S.Si., M.T.I. ( )

Penguji II : Susana Dwi Yulianti, S.Kom., M.Kom. ( )

Penguji III : Malisa Huzaifa, S.Kom., M.T. ( )

Mengetahui,  
Ketua Jurusan Teknik Informatika dan Komputer  
**Ketua Jurusan**

**Dr. Anita Hidayati, S.Kom., M.Kom.**  
NIP. 197908032003122003

## KATA PENGANTAR

Puji dan syukur penulis panjatkan ke hadirat Allah SWT atas rahmat, karunia, dan kemudahan-Nya sehingga penulis dapat menyelesaikan skripsi yang berjudul “Rancang Bangun Non Playable Character (NPC) Cerdas Berbasis Large Language Model (LLM) dan Computer Vision pada Minecraft dengan Kemampuan Interaksi Natural dan Pengenalan Emosi Player”. Skripsi ini disusun sebagai salah satu syarat untuk memperoleh gelar Sarjana Terapan pada Program Studi Teknik Informatika, Jurusan Teknik Informatika dan Komputer, Politeknik Negeri Jakarta.

Penulis menyadari bahwa penyusunan skripsi ini tidak dapat berjalan dengan baik tanpa bantuan, arahan, doa, dan dukungan dari berbagai pihak. Oleh karena itu, penulis ingin menyampaikan terima kasih yang sebesar-besarnya kepada:

1. Ibu Dr. Anita Hidayati, S.Kom., M.Kom., selaku Ketua Jurusan Teknik Informatika dan Komputer Politeknik Negeri Jakarta.
2. Ibu Euis Oktavianti, S.Si., M.T.I., selaku Kepala Program Studi Teknik Informatika Politeknik Negeri Jakarta.
3. Ibu Rizki Elisa Nalawati, S.T., M.T., selaku dosen pembimbing yang telah memberikan arahan, masukan, bimbingan, dan dukungan yang sangat berharga selama proses penyusunan skripsi ini.
4. Seluruh dosen Jurusan Teknik Informatika dan Komputer Politeknik Negeri Jakarta yang telah memberikan ilmu, pengalaman, dan pembelajaran berharga selama masa perkuliahan.
5. Komunitas pengembang mod Minecraft dan ekosistem Fabric Framework yang telah menyediakan dokumentasi, pustaka terbuka, serta ruang diskusi yang sangat mendukung kelancaran implementasi teknis modifikasi gim ini.
6. Orang tua dan keluarga penulis yang tidak pernah lelah memberikan doa, dukungan moral, materi, semangat, dan motivasi yang luar biasa selama proses perkuliahan hingga penyusunan skripsi ini.
7. Andi Rizki Mahesa dan Teman-teman seperjuangan Program Studi Teknik Informatika angkatan 2022 serta semua pihak yang tidak dapat disebutkan

satu per satu, yang telah memberikan bantuan, semangat, dan menjadi tempat berbagi selama proses pengerjaan skripsi ini.

Penulis menyadari bahwa skripsi ini masih memiliki keterbatasan, baik dari sisi teknis sistem maupun penyusunan laporan. Oleh karena itu, penulis sangat terbuka terhadap kritik dan saran yang membangun agar sistem agen cerdas ini dapat dikembangkan lebih lanjut di masa mendatang. Akhir kata, penulis berharap skripsi ini dapat memberikan manfaat bagi pembaca, khususnya dalam bidang pengembangan gim, pemrosesan bahasa alami, *computer vision*, serta integrasi kecerdasan buatan pada lingkungan permainan digital.

## **SURAT PERNYATAAN PUBLIKASI SKRIPSI UNTUK KEPENTINGAN AKADEMIS**

Sebagai sivitas akademika Politeknik Negeri Jakarta, saya yang bertanda tangan di bawah ini:

Nama : Arjuna Dwi Damara Putra  
NIM : 2207411006  
Jurusan/Program Studi : T.Informatika dan Komputer / Teknik Informatika

Demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Politeknik Negeri Jakarta Hak Bebas Royalti Non-Eksklusif atas karya ilmiah saya yang berjudul:

Rancang Bangun Non Playable Character (NPC) Cerdas Berbasis Large Language Model (LLM) dan Computer Vision pada Minecraft dengan Kemampuan Interaksi Natural dan Pengenalan Emosi Player

Beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Non-Eksklusif ini, Politeknik Negeri Jakarta berhak menyimpan, mengalihmediakan/formatkan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan mempublikasikan skripsi saya tanpa meminta izin dari saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Jakarta, 07 Juni 2026  
Yang Menyatakan

Arjuna Dwi Damara Putra  
NIM. 2207411004

**RANCANG BANGUN *NON PLAYABLE CHARACTER* CERDAS  
BERBASIS *LARGE LANGUAGE MODEL* DAN COMPUTER VISION  
PADA MINECRAFT DENGAN KEMAMPUAN INTERAKSI NATURAL  
DAN PENGENALAN EMOSI PLAYER**

**ABSTRAK**

*Perkembangan Large Language Model (LLM) dan Computer Vision membuka peluang besar untuk mentransformasi interaksi karakter virtual dalam permainan digital. Pada gim sandbox seperti Minecraft, entitas Non-Player Character (NPC) bawaan masih memiliki keterbatasan perilaku karena dikendalikan oleh sistem berbasis skrip sederhana, sehingga interaksi yang dihasilkan terasa kaku, tidak kontekstual, dan tidak mampu mengenali kondisi emosional pemain. Penelitian ini mengusulkan rancangan sistem Intelligent Companion NPC menggunakan pendekatan sistem terdistribusi hibrida berbasis arsitektur Sense-Think-Act. Sistem ini mengintegrasikan tiga platform utama: Python sebagai modul Computer Vision untuk pemrosesan data visual ekspresi wajah secara real-time menggunakan algoritma CNN, Java dengan framework Fabric sebagai game engine klien untuk menangkap status pemain serta dinamika lingkungan Minecraft, dan Google Gemini API sebagai layanan kecerdasan berbasis cloud untuk penalaran bahasa alami. Seluruh data masukan disinkronisasikan oleh modul Context Builder menjadi format data terstruktur Context JSON sebelum diproses oleh LLM guna menghasilkan respons dialog tingkat tinggi yang adaptif. Hasil kuesioner awal terhadap pemain menunjukkan adanya ekspektasi tinggi terhadap pembaruan sistem interaksi karakter virtual. Melalui integrasi ganda ini, sistem berhasil mewujudkan NPC pendamping yang mampu memahami instruksi bahasa alami secara fleksibel sekaligus menyesuaikan gaya percakapan berdasarkan empat emosi dasar pemain (senang, sedih, marah, dan netral) secara near real-time. Penelitian ini diharapkan dapat menjadi referensi dan panduan bagi pengembang gim dalam meningkatkan kualitas keimersifan dan pengalaman pengguna (user experience) melalui penerapan kecerdasan buatan multimodal.*

**Kata Kunci:** Computer Vision, Fabric Framework, Gemini API, Large Language Model, Minecraft, Non-Player Character.

## DAFTAR ISI

|                                                                        |      |
|------------------------------------------------------------------------|------|
| SURAT PERNYATAAN BEBAS PLAGIARISME .....                               | ii   |
| LEMBAR PENGESAHAN .....                                                | iii  |
| KATA PENGANTAR .....                                                   | iv   |
| SURAT PERNYATAAN PUBLIKASI SKRIPSI UNTUK KEPENTINGAN<br>AKADEMIS ..... | vi   |
| ABSTRAK .....                                                          | vii  |
| DAFTAR ISI .....                                                       | viii |
| DAFTAR GAMBAR .....                                                    | xii  |
| DAFTAR TABEL .....                                                     | xiii |
| DAFTAR KODE PROGRAM .....                                              | xiv  |
| BAB I PENDAHULUAN .....                                                | 1    |
| 1.1 Latar Belakang Masalah .....                                       | 1    |
| 1.2 Perumusan Masalah .....                                            | 3    |
| 1.3 Batasan Masalah .....                                              | 3    |
| 1.4 Tujuan dan Manfaat .....                                           | 4    |
| 1.5 Sistematika Penulisan .....                                        | 5    |
| BAB II TINJAUAN PUSTAKA .....                                          | 7    |
| 2.1 <i>Non-Playable Character</i> (NPC) .....                          | 7    |
| 2.1.1 Peran NPC dalam game .....                                       | 7    |
| 2.2 Mojang .....                                                       | 7    |
| 2.2.1 Minecraft .....                                                  | 7    |
| 2.3 Genre <i>SandBox</i> .....                                         | 8    |
| 2.4 Mod atau <i>Modification</i> .....                                 | 8    |
| 2.5 Fabric Framework .....                                             | 9    |
| 2.6 <i>Game Community</i> .....                                        | 9    |
| 2.7 Large Language Model (LLM) .....                                   | 10   |
| 2.8 Gemini API .....                                                   | 10   |
| 2.9 Computer Vision untuk Game .....                                   | 10   |
| 2.10 TensorFlow .....                                                  | 11   |
| 2.11 <i>Convolutional Neural Network</i> (CNN) .....                   | 12   |
| 2.12 Penelitian Terdahulu .....                                        | 12   |
| BAB III METODE PENELITIAN .....                                        | 15   |
| 3.1 Rancangan Penelitian .....                                         | 15   |
| 3.2 Tahapan Penelitian .....                                           | 15   |
| 3.3 Objek Penelitian .....                                             | 17   |

|                                                                             |    |
|-----------------------------------------------------------------------------|----|
| BAB IV HASIL DAN PEMBAHASAN .....                                           | 19 |
| 4.1 Analisis Kebutuhan Sistem .....                                         | 19 |
| 4.1.1 Kebutuhan Perangkat Keras (Hardware Requirements) .....               | 19 |
| 4.1.2 Kebutuhan Perangkat Lunak (Software Requirements) .....               | 19 |
| 4.1.3 Analisis Kebutuhan Fungsional .....                                   | 21 |
| 4.2 Perancangan Sistem Terintegrasi .....                                   | 22 |
| 4.2.1 Minecraft sebagai Virtual Testbed dan <i>Sandbox Simulation</i> ..... | 22 |
| 4.2.2 Komponen Arsitektur Integrasi Sistem .....                            | 24 |
| 4.2.3 Mekanisme Aliran Data Terintegrasi .....                              | 26 |
| 4.2.4 Use Case Diagram .....                                                | 29 |
| 4.3 Implementasi Arsitektur Sensor Modular (Perception) .....               | 32 |
| 4.3.1 Orkestrasi Sensor melalui <i>EnvirementSensor</i> .....               | 32 |
| 4.3.2 Implementasi Weather Module .....                                     | 34 |
| 4.3.3 Implementasi Time Module .....                                        | 35 |
| 4.3.4 Implementasi <i>Light Module</i> .....                                | 36 |
| 4.3.5 Implementasi Biome Module .....                                       | 37 |
| 4.3.6 Implementasi Entity Module .....                                      | 44 |
| 4.3.7 Implementasi Block Module .....                                       | 47 |
| 4.3.8 Implementasi Status Module .....                                      | 50 |
| 4.3.9 Implementasi Inventory Module .....                                   | 51 |
| 4.3.10 Implementasi Emotion Module .....                                    | 53 |
| 4.3.11 Implementasi Health Event Module .....                               | 54 |
| 4.4 Analisis dan Implementasi Model Computer Vision .....                   | 55 |
| 4.4.1 Dataset Emosi .....                                                   | 56 |
| 4.4.2 Preprocessing .....                                                   | 57 |
| 4.4.3 Augmentasi Dataset Emosi .....                                        | 58 |
| 4.4.4 Arsitektur Model <i>Convolutional Neural Network</i> (CNN) .....      | 59 |
| 4.4.5 Pelatihan Model .....                                                 | 61 |
| 4.4.6 Implementasi <i>Inference</i> menggunakan Flask API .....             | 73 |
| 4.5 Implementasi Sistem Memori dan Persistensi Data .....                   | 75 |
| 4.5.1 Persistensi Data Fisik berbasis NBT .....                             | 75 |
| 4.5.2 Manajemen Memori Sosial dan Atribut Psikologis .....                  | 77 |
| 4.5.3 Manajemen Memori Spasial (Ingatan Lokasi) .....                       | 82 |
| 4.5.4 Memori Jangka Pendek dan Riwayat Interaksi .....                      | 83 |
| 4.6 Orkestrasi Data dan Sintesis Konteks (Context Builder) .....            | 84 |
| 4.6.1 Strategi Pemetaan Data ke dalam Struktur Markdown .....               | 85 |

|                                                                                                                       |     |
|-----------------------------------------------------------------------------------------------------------------------|-----|
| 4.6.1.1 Blok Instruksi Persona (### PERSONA UTAMA) .....                                                              | 85  |
| 4.6.1.2 Blok Identitas dan Parameter Hubungan Sosial (### IDENTITAS & HUBUNGAN SOSIAL) .....                          | 87  |
| 4.6.1.3 Blok Penglihatan Voxel (Objek di Sekitar) (### PENGLIHATAN) .....                                             | 89  |
| 4.6.1.4 Blok Riwayat Percakapan (### RIWAYAT PERCAKAPAN) .....                                                        | 90  |
| 4.6.1.5 Blok Kondisi Fisik (### STATUS FISIK) .....                                                                   | 91  |
| 4.6.1.6 Blok Persepsi Lingkungan dan Kondisi Eksternal (### LINGKUNGAN) .....                                         | 93  |
| 4.6.1.7 Blok Ingatan Spasial (Lokasi yang Diingat) (### INGATAN SPASIAL) .....                                        | 94  |
| 4.6.1.8 Blok Inventaris (### INVENTARIS) .....                                                                        | 96  |
| 4.6.1.9 Blok Radar Sosial dan Deteksi Entitas (### RADAR SOSIAL) .....                                                | 97  |
| 4.6.1.10 Blok Evaluasi Keamanan dan Perlindungan Pemain (### STATUS KEAMANAN) .....                                   | 98  |
| 4.6.1.11 Blok Preferensi Pemilik dan Personalisasi Interaksi (### PREFERENSI PEMILIK) .....                           | 100 |
| 4.6.1.12 Blok Memori Penting dan Kueri Relevansi (Smart Recall) (### MEMORI PENTING) .....                            | 101 |
| 4.6.2 Transformasi Semantik dan Optimasi Prompt .....                                                                 | 103 |
| 4.6.2.1 Transformasi Data Teknis ke Narasi Natural .....                                                              | 103 |
| 4.6.2.2 Strategi Optimasi Prompt dan Efisiensi Token .....                                                            | 104 |
| 4.6.3 Konstruksi <i>Final Prompt Packet</i> .....                                                                     | 105 |
| 4.7 Implementasi Lapisan Penalaran (Reasoning Layer) .....                                                            | 115 |
| 4.7.1 Arsitektur Layanan LLM dan Integrasi REST API .....                                                             | 115 |
| 4.7.2 Manajemen Permintaan Asinkron ( <i>Asynchronous Request Management</i> ) ....                                   | 117 |
| 4.7.3 Sintesis Kognitif dan Pemrosesan Respon JSON .....                                                              | 118 |
| 4.8 Implementasi Action Handler dan Eksekusi Perilaku .....                                                           | 120 |
| 4.8.1 Definisi Instruksi Semantik ( <i>Instruction Set</i> ) .....                                                    | 120 |
| 4.8.2 Mekanisme Resolusi Perintah dan <i>Heuristic Fuzzy Mapping</i> .....                                            | 122 |
| 4.8.2.1 Pemetaan Tag Material (Resource Resolution) .....                                                             | 123 |
| 4.8.2.2 Resolusi Entitas dengan Pencocokan Samar ( <i>Entity Fuzzy Matching</i> ) ..                                  | 124 |
| 4.8.3 Implementasi Algoritma Perilaku Fisik ( <i>AI Goals</i> ) .....                                                 | 124 |
| 4.8.3.1 Implementasi Perilaku Pengumpulan Sumber Daya ( <i>GATHER</i> ) melalui <i>DynamicGatherGoal</i> .....        | 125 |
| 4.8.3.2 Implementasi Perilaku Pertahanan dan Penyerangan ( <i>ATTACK &amp; GUARD</i> ) melalui <i>GuardGoal</i> ..... | 131 |
| 4.8.3.3 Implementasi Perilaku Berburu ( <i>HUNT</i> ) melalui <i>HuntGoal</i> .....                                   | 134 |

|                                                                                                      |     |
|------------------------------------------------------------------------------------------------------|-----|
| 4.8.3.4 Implementasi Perilaku Transfer Sumber Daya ( <i>GIVE</i> ) melalui <i>GiveItemGoal</i> ..... | 137 |
| 4.8.3.5 Manajemen Antrean Tugas Sekuensial dan Evaluasi Umpan Balik Kognitif.....                    | 140 |
| 4.9 Pengujian Sistem.....                                                                            | 143 |
| 4.9.1 Deskripsi Pengujian .....                                                                      | 143 |
| 4.9.1.1 <i>Memory Recall Accuracy</i> .....                                                          | 143 |
| 4.9.1.2 <i>Decision Accuracy</i> .....                                                               | 144 |
| 4.9.1.3 <i>Execution Success</i> .....                                                               | 144 |
| 4.9.1.4 Pengujian <i>User Acceptance Test</i> (UAT).....                                             | 144 |
| 4.9.1.5 Pengujian <i>System Usability Scale</i> (SUS).....                                           | 144 |
| 4.9.2 Prosedur Pengujian .....                                                                       | 145 |
| 4.9.2.1 Prosedur <i>Memory Recall Accuracy</i> .....                                                 | 145 |
| 4.9.2.2 Prosedur <i>Decision Accuracy</i> .....                                                      | 147 |
| 4.9.2.3 Prosedur <i>Execution Success</i> .....                                                      | 151 |
| 4.9.2.4 Prosedur <i>User Acceptance Test</i> (UAT).....                                              | 154 |
| 4.9.2.5 Prosedur <i>System Usability Scale</i> (SUS).....                                            | 158 |
| 4.9.3 Data Hasil Pengujian.....                                                                      | 159 |
| 4.9.3.1 Hasil <i>Memory Recall Accuracy</i> .....                                                    | 159 |
| 4.9.3.2 Hasil <i>Decision Accuracy</i> .....                                                         | 171 |
| 4.9.3.3 Hasil <i>Execution Success</i> .....                                                         | 210 |
| 4.9.3.4 Hasil <i>User Acceptance Test</i> (UAT) .....                                                | 215 |
| 4.9.3.5 Hasil <i>System Usability Scale Test</i> (SUS).....                                          | 217 |
| BAB V PENUTUP .....                                                                                  | 219 |
| 5.1 Kesimpulan .....                                                                                 | 219 |
| 5.2 Saran .....                                                                                      | 220 |
| DAFTAR PUSTAKA .....                                                                                 | 222 |
| DAFTAR RIWAYAT HIDUP.....                                                                            | 225 |
| LAMPIRAN.....                                                                                        | 226 |

## DAFTAR GAMBAR

|                                                                                        |     |
|----------------------------------------------------------------------------------------|-----|
| Gambar 3.1 Diagram Tahapan Penelitian .....                                            | 16  |
| Gambar 4.1 Siklus Siang dan Malam.....                                                 | 23  |
| Gambar 4.2 Visualisasi sistem koordinat Cartesian (X, Y, Z) dalam Minecraft ...        | 24  |
| Gambar 4.3 Komponen Lapisan Eksekusi & Orkestrator.....                                | 25  |
| Gambar 4.4 Komponen Lapisan Persepsi .....                                             | 25  |
| Gambar 4.5 Aliran Data Terintegrasi .....                                              | 26  |
| Gambar 4.6 Data Dunia Yang Diperoleh Agen .....                                        | 27  |
| Gambar 4.7 Data Emosi Pemain .....                                                     | 27  |
| Gambar 4.8 Potongan Struktur Konteks Markdown sebagai Hasil Transformasi Semantik..... | 28  |
| Gambar 4.9 Cuaca pada minecraft (hujan dikiri dan cerah dikanan).....                  | 35  |
| Gambar 4.10 Item Jam pada Minecraft yang menunjukkan waktu dalam game ..               | 36  |
| Gambar 4.11 Sumber Cahaya Buatan dan Sumber Cahaya Alami pada Minecraft .....          | 37  |
| Gambar 4.12 Entitas pada Minecraft .....                                               | 47  |
| Gambar 4.13 Beberapa Block pada Minecraft .....                                        | 50  |
| Gambar 4.14 Inventory pada Agen .....                                                  | 53  |
| Gambar 4.15 Sample Gambar pada Real-world Affective Faces Database .....               | 57  |
| Gambar 4.16 Struktur Model Convolutional Neural Network.....                           | 61  |
| Gambar 4.17 Confusion Matrix Performa Model pada Data Uji.....                         | 63  |
| Gambar 4.18 Format JSON.....                                                           | 74  |
| Gambar 4.19 NPC Menolak Permintaan saat Trust Score nya Rendah .....                   | 78  |
| Gambar 4.20 Jika Confidance Score NPC akan menjawab dengan ragu-ragu .....             | 80  |
| Gambar 4.21 Custom Persona Agen .....                                                  | 81  |
| Gambar 4.22 Deserialisasi Objek Kognitif .....                                         | 119 |

## DAFTAR TABEL

|                                                                         |     |
|-------------------------------------------------------------------------|-----|
| <b>Tabel 2.1</b> Penelitian Terdahulu .....                             | 12  |
| <b>Tabel 4.1</b> Spesifikasi Perangkat Keras Pengujian .....            | 19  |
| <b>Tabel 4.2</b> Daftar Modul Sensor dan Tanggung Jawabnya .....        | 33  |
| <b>Tabel 4.3</b> Nama dan Karakteristik Bioma pada Minecraft .....      | 38  |
| <b>Tabel 4.4</b> Distribusi Dataset RAF-DB per Kelas Emosi .....        | 57  |
| <b>Tabel 4.5</b> Ringkasan Progres Pelatihan Model .....                | 61  |
| <b>Tabel 4.6</b> Total Data Aktual dan Total Data Prediksi .....        | 66  |
| <b>Tabel 4.7</b> Interpretasi Cohen's Kappa Score .....                 | 72  |
| <b>Tabel 4.8</b> Data JSON .....                                        | 74  |
| <b>Tabel 4.9</b> Struktur Penyimpanan Data NBT Agen Cerdas .....        | 76  |
| <b>Tabel 4.10</b> Struktur dan Fungsi PERSONA UTAMA .....               | 85  |
| <b>Tabel 4.11</b> Struktur dan Fungsi IDENTITAS & HUBUNGAN SOSIAL ..... | 87  |
| <b>Tabel 4.12</b> Struktur dan Fungsi Parameter Penglihatan .....       | 89  |
| <b>Tabel 4.13</b> Struktur dan Fungsi Parameter Riwayat Percakapan..... | 90  |
| <b>Tabel 4.14</b> Struktur dan Fungsi STATUS FISIK .....                | 91  |
| <b>Tabel 4.15</b> Struktur dan Fungsi Parameter Lingkungan .....        | 93  |
| <b>Tabel 4.16</b> Struktur Blok Ingatan Spasial .....                   | 95  |
| <b>Tabel 4.17</b> Struktur dan Fungsi INVENTARIS .....                  | 96  |
| <b>Tabel 4.18</b> Struktur dan Fungsi Parameter Radar Sosial .....      | 97  |
| <b>Tabel 4.19</b> Struktur dan Fungsi Parameter Status Keamanan .....   | 99  |
| <b>Tabel 4.20</b> Struktur dan Fungsi Parameter Prefensi Pemilik.....   | 100 |
| <b>Tabel 4.21</b> Struktur dan Fungsi Parameter Memori Penting .....    | 101 |
| <b>Tabel 4.22</b> Daftar Instruksi Semantik dan Deskripsi Perilaku..... | 120 |
| <b>Tabel 4.23</b> Skenario Memory Recall Accuracy .....                 | 146 |
| <b>Tabel 4.24</b> Tabel Prosedur Decision Accuracy .....                | 148 |
| <b>Tabel 4.25</b> Tabel Prosedur Execution Success .....                | 153 |
| <b>Tabel 4.26</b> Pernyataan UAT .....                                  | 155 |
| <b>Tabel 4.27</b> Pernyataan SUS .....                                  | 158 |
| <b>Tabel 4.28</b> Hasil Memory Recall Accuracy Companion NPC .....      | 159 |
| <b>Tabel 4.29</b> Hasil Memory Recall Accuracy Player2AI .....          | 166 |
| <b>Tabel 4.30</b> Hasil Decision Accuracy Companion NPC .....           | 171 |
| <b>Tabel 4.31</b> Hasil Execution Success Companion NPC.....            | 210 |
| <b>Tabel 4.32</b> Hasil UAT Companion NPC.....                          | 215 |
| <b>Tabel 4.33</b> Hasil SUS Companion NPC .....                         | 217 |

## DAFTAR KODE PROGRAM

|                                                                                                                |     |
|----------------------------------------------------------------------------------------------------------------|-----|
| Kode Program 4.1 Kode untuk augmentasi dataset .....                                                           | 59  |
| Kode Program 4.2 Logika Navigasi dan Memori Spasial pada Status<br>EXPLORING .....                             | 126 |
| Kode Program 4.3 Logika Interaksi Fisik dan Otomasi Alat pada Status<br>BREAKING .....                         | 127 |
| Kode Program 4.4 Logika Kueri Spasial dan Manajemen Inventaris pada Status<br>COLLECTING .....                 | 128 |
| Kode Program 4.5 Logika Retur Navigasi, Feedback Loop, dan Komunikasi<br>Proaktif pada Status DELIVERING ..... | 130 |
| Kode Program 4.6 Logika Validasi Titik Jaga pada GuardGoal.java .....                                          | 132 |
| Kode Program 4.7 Logika Pemilihan Senjata Otomatis pada Siklus Main Tick                                       | 133 |
| Kode Program 4.8 Logika Pengejaran dan Otomasi Serangan pada Status<br>HUNTING .....                           | 135 |
| Kode Program 4.9 Logika Penyerahan Barang dan Pelaporan Proaktif asinkron<br>.....                             | 136 |
| Kode Program 4.10 Logika Audit Internal Slot Tas pada GiveItemGoal.java....                                    | 138 |
| Kode Program 4.11 Logika Proksimitas Spasial dan Navigasi Aktif .....                                          | 139 |

# **BAB I**

## **PENDAHULUAN**

### **1.1 Latar Belakang Masalah**

Teknologi kecerdasan buatan (*Artificial Intelligence*) berkembang dengan sangat pesat, khususnya dalam pemrosesan bahasa alami (*Natural Language Processing*). NLP Merupakan bentuk teknologi berkembang yang saat ini mengalami kemajuan yang sangat cepat dan signifikan serta memiliki dampak yang sangat luas dan besar pada berbagai bidang. Salah satu jenis kecerdasan buatan yang menarik perhatian orang saat ini adalah LLM (*Large Language Model*) (Rachmat and Kesuma, 2024). Integrasi model bahasa berbasis *Application Programming Interface* (API) seperti Gemini membuka peluang baru dalam pengembangan agen cerdas di lingkungan virtual. LLM memungkinkan sistem untuk memahami konteks percakapan, menghasilkan respons yang natural, serta melakukan *reasoning* berbasis teks, sehingga interaksi antara manusia dan sistem menjadi lebih fleksibel dan adaptif.

Selain NLP, salah satu jenis kecerdasan buatan yang juga menarik banyak perhatian adalah visi komputer (*Computer Vision*). Model berbasis *Computer Vision* seperti YOLO dan CNN telah banyak digunakan untuk mendeteksi dan mengenali objek, ekspresi wajah, serta kondisi emosional manusia melalui citra visual. Pendekatan ini memungkinkan sistem untuk memperoleh informasi non-verbal sebagai pelengkap interaksi berbasis teks. Di dalam industri game yang berkembang pesat dan memiliki potensi ekonomi tinggi, integrasi antara LLM dan *Computer Vision* membuka peluang untuk menciptakan karakter virtual atau NPC (*Non Playable Character*) yang tidak hanya mampu berkomunikasi secara natural, tetapi juga memahami kondisi emosional pemain (Paduraru, Paduraru and Stefanescu, 2021).

Salah satu *game* yang sudah menerapkan AI *Chatbot* untuk NPC mereka secara masif adalah *Where The Winds Meet*. Everstone Studio selaku pengembang *game* ini menamai fitur ini *AI Chat NPCs* (Kennedy, 2025). Dalam implementasinya, fitur ini memungkinkan pemain untuk berinteraksi menggunakan input teks bebas, menggantikan sistem pohon dialog (*dialogue tree*) konvensional yang kaku. Karakter NPC yang diterapkan teknologi *Large Language Model* (LLM) ini dirancang untuk merespons secara dinamis berdasarkan konteks percakapan dan latar belakang (*lore*) fiksi dunia permainan.

Dampak dari perkembangan teknologi kecerdasan buatan tidak hanya terbatas pada permainan video (*video game*) rilis terbaru, tetapi juga merambah pada judul-judul lawas yang dirilis sebelum tahun 2015, seperti *Minecraft*. Integrasi teknologi mutakhir ke dalam ekosistem klasik ini dicapai melalui proses modifikasi atau *modding*. Praktik *modding* yang digerakkan secara independen oleh komunitas ini bertujuan untuk memperbarui dan memperluas batasan sistem permainan agar tetap relevan dengan kemajuan kecerdasan buatan saat ini. *Minecraft* sendiri merupakan permainan *sandbox open-world* yang bersifat terbuka terhadap modifikasi, sehingga menyediakan lingkungan yang ideal untuk eksperimen kecerdasan buatan berbasis agen interaktif (Jensen and Hanghøj, 2020).

Namun, *Non-Player Character* (NPC) bawaan *Minecraft* masih memiliki perilaku terbatas dan umumnya dikendalikan oleh sistem berbasis skrip sederhana, sehingga belum mampu melakukan percakapan kompleks, memahami perintah pemain secara natural, maupun memanfaatkan informasi visual untuk mengenali kondisi emosional pemain. Untuk mengkaji keterbatasan NPC bawaan *Minecraft* serta ekspektasi pemain terhadap sistem NPC yang lebih cerdas, penelitian ini menggunakan pendekatan survei dengan kuesioner berbasis skala Likert yang ditujukan kepada pemain *Minecraft*. Penggunaan kuesioner persepsi pemain merupakan metode standar dalam mengevaluasi kualitas interaksi dan pengalaman pengguna (*user experience*) pada sistem permainan digital yang kompleks (Christiansen et al., 2024). Instrumen kuesioner dirancang untuk mengukur persepsi pemain terhadap kualitas dialog NPC, kemampuan NPC dalam memahami instruksi berbasis bahasa alami, serta respons NPC terhadap kondisi emosional pemain. Berdasarkan hasil kuesioner yang diisi oleh 10 responden pemain *Minecraft*, mayoritas responden menilai bahwa NPC bawaan masih memiliki keterbatasan dalam aspek dialog, pemahaman instruksi, dan respons terhadap kondisi emosional, serta mengharapkan NPC yang mampu berinteraksi secara lebih natural dan adaptif. Mengacu pada temuan tersebut, penelitian ini selanjutnya mengusulkan perancangan sistem NPC cerdas yang mengintegrasikan *Large Language Model* (LLM) dan *Computer Vision* guna mendukung interaksi yang lebih natural, adaptif, dan kontekstual dalam lingkungan *game Minecraft*.

## 1.2 Perumusan Masalah

Rumusan masalah yang akan di bahas dalam penelitian ini adalah :

1. Bagaimana merancang arsitektur modifikasi (*modding*) pada *Minecraft* yang memungkinkan integrasi *Large Language Model* (LLM) untuk sistem dialog NPC dan model *Computer Vision* untuk pengenalan emosi pemain dalam satu lingkungan yang tersinkronisasi?
2. Bagaimana mengimplementasikan pemahaman instruksi berbasis bahasa alami agar NPC mampu memahami dan merespon perintah yang diberikan oleh pemain secara kontekstual ?
3. Bagaimana memanfaatkan integrasi *Computer Vision* untuk mendeteksi kondisi emosional pemain melalui tangkapan citra visual, sehingga NPC dapat menyesuaikan respons teks ?

## 1.3 Batasan Masalah

Untuk menjaga agar penelitian ini tetap terfokus dan tidak melebar dari tujuan yang telah ditetapkan, maka batasan masalah dalam penelitian ini adalah sebagai berikut :

1. Sistem diimplementasikan secara spesifik pada game *Minecraft* versi 1.21 dengan antarmuka *mod loader* *Fabric* (berbasis *Java*) sebagai orchestrator utama. Arsitektur ini bertumpu pada komunikasi data terdistribusi yang sinkron, sehingga fungsionalitas penalaran kecerdasan buatan (*cloud-based LLM*) mewajibkan adanya koneksi internet aktif dan tidak melakukan komputasi bahasa alami secara lokal (*on-device*).
2. Sistem kecerdasan buatan hanya ditanamkan pada satu entitas *companion* NPC tunggal (tidak mencakup interaksi komunikasi *multi-NPC*). Input dan output komunikasi dibatasi murni pada antarmuka teks (*in-game chat*), tidak mencakup fitur pemrosesan suara (*Speech-to-Text* atau *Text-to-Speech*). Selain itu, pemahaman instruksi dibatasi pada konteks mekanik dasar game (mengikuti pemain, memberikan informasi item/lingkungan, dan manipulasi blok sederhana) serta tidak dirancang untuk eksekusi modifikasi dunia skala besar (*auto-building*).

3. Proses pengenalan emosi pemain dibatasi pada analisis ekspresi wajah (*facial expression recognition*) secara *near real-time* menggunakan tangkapan kamera visual (*webcam*) laptop/PC via modul Python eksternal. Kategori emosi yang dideteksi dan direspons oleh teks dialog NPC dibatasi pada 4 emosi dasar, yaitu Senang, Sedih, Marah, dan Netral, sehingga tidak mencakup deteksi mikro-ekspresi atau analisis kondisi psikologis pengguna yang kompleks.

#### 1.4 Tujuan

Penelitian ini bertujuan untuk :

1. Merancang dan mengembangkan arsitektur modifikasi pada *game Minecraft* (berbasis *mod loader Fabric*) yang mengintegrasikan *Large Language Model* (LLM) sebagai mesin dialog NPC dan *Computer Vision* untuk mengenali kondisi emosional pemain.
2. Mengimplementasikan sistem pemahaman bahasa alami agar NPC mampu memproses, menerima, dan menjalankan instruksi dari pemain secara kontekstual di dalam permainan.
3. Mengimplementasikan sistem *Computer Vision* untuk mendeteksi kondisi emosional pemain melalui analisis ekspresi wajah, yang kemudian digunakan sebagai dasar penyesuaian respons NPC secara adaptif.

#### 1.5 Manfaat

Penelitian ini Bermanfaat Untuk :

- a. Memberikan pengalaman bermain yang lebih imersif, interaktif, dan hidup melalui kehadiran companion (teman bertualang) yang peka, adaptif, dan reaktif terhadap kondisi pemain secara real-time.
- b. Memudahkan komunikasi antar pemain dalam berinteraksi dan mengobrol secara bebas dengan NPC layaknya berbicara dengan pemain sungguhan, tanpa dibatasi oleh pilihan dialog yang bersifat statis atau kaku.
- c. Meningkatkan tingkat imersivitas permainan dengan memanfaatkan pengenalan emosi berbasis ekspresi wajah, sehingga NPC dapat memberikan respons yang sesuai dengan kondisi emosional pemain.

## **1.6 Sistematika Penulisan**

Untuk memberikan gambaran yang jelas dan terstruktur mengenai penyusunan laporan penelitian ini, maka penulisan dibagi menjadi lima bab dengan rincian sistematika sebagai berikut:

### **BAB I PENDAHULUAN**

Bab ini menguraikan dasar pemikiran penelitian yang meliputi latar belakang masalah mengenai keterbatasan interaksi NPC pada *Minecraft*, perumusan masalah, batasan masalah agar penelitian tetap terfokus, tujuan yang ingin dicapai, manfaat penelitian, serta sistematika penulisan yang memberikan gambaran keseluruhan isi laporan.

### **BAB II TINJAUAN PUSTAKA**

Bab ini memaparkan landasan teori dan konsep-konsep pokok yang relevan untuk mendukung pemecahan masalah. Teori yang dibahas meliputi pemrosesan bahasa alami (*Natural Language Processing*) menggunakan *Large Language Model* (LLM), pengolahan citra (*Computer Vision*) untuk pengenalan ekspresi wajah, arsitektur *modding* pada *game Minecraft* (khususnya menggunakan *mod loader Fabric*), integrasi *Application Programming Interface* (API), serta kajian terhadap penelitian-penelitian terdahulu yang berkaitan.

### **BAB III METODE PENELITIAN**

Bab ini menjelaskan metodologi yang digunakan dalam penelitian, mulai dari rancangan penelitian, tahapan penelitian, dan objek penelitian.

### **BAB IV HASIL DAN PEMBAHASAN**

Bab ini membahas analisis kebutuhan sistem, perancangan sistem, dan implementasi ke dalam bentuk kode program (*coding*). Selain itu, bab ini juga memaparkan proses pengujian sistem yang mencakup pengujian fungsionalitas fitur, serta evaluasi hasil implementasi berdasarkan kuesioner kepuasan pemain menggunakan skala Likert.

### **BAB V PENUTUP**

Bab ini merupakan bab terakhir yang berisi kesimpulan dari seluruh rangkaian penelitian dan implementasi yang telah dilakukan, guna menjawab rumusan

masalah. Bab ini juga memberikan saran-saran konstruktif untuk pengembangan sistem lebih lanjut di masa mendatang agar lebih sempurna.

## BAB II TINJAUAN PUSTAKA

### 2.1 *Non-Playable Character* (NPC)

*Non-Player Character* (NPC) dalam permainan adalah karakter yang tidak berada di bawah kendali langsung pemain di dalam suatu lingkungan permainan. Istilah NPC berasal dari permainan meja tradisional (*tabletop games*), di mana tindakan dan perilaku karakter dikendalikan melalui narasi yang dibuat oleh pemain atau pengelola permainan (*game master*) (Silva & Ribeiro, 2021). Dalam permainan digital modern, perilaku NPC umumnya telah ditentukan oleh pengembang game dan diimplementasikan secara langsung ke dalam kode program, sehingga interaksi yang dihasilkan bersifat terbatas dan mengikuti aturan yang telah ditetapkan (Haq et al., 2021).

#### 2.1.1 Peran NPC dalam game

NPC dalam permainan adalah elemen penting yang membuat lingkungan permainan menjadi kompleks dan imersif (Song, 2025). NPC berfungsi sebagai penggerak narasi, pemberi misi (*quest giver*), penyedia informasi, serta representasi karakter lain yang berinteraksi langsung dengan pemain. Melalui keberadaan NPC, pemain dapat memahami alur cerita, tujuan permainan, serta dinamika dunia virtual yang sedang dijelajahi.

### 2.2 Mojang

Mojang AB, yang kini beroperasi secara resmi dengan nama pengenal Mojang Studios, adalah sebuah perusahaan pengembang permainan video (*video game developer*) asal Stockholm, Swedia. Perusahaan ini didirikan pada tahun 2009 oleh seorang desainer perangkat lunak independen bernama Markus "Notch" Persson. Mojang dikenal secara global sebagai kreator dan pengembang utama dari Minecraft, sebuah permainan video tiga dimensi bergenre *sandbox* dan *survival* yang memungkinkan pemainnya memanipulasi lingkungan virtual berbasis blok *voxel* secara prosedural (Minecraft, 2026).

#### 2.2.1 Minecraft

Minecraft merupakan permainan *open-world sandbox* yang menggabungkan unsur eksplorasi, pengumpulan sumber daya, *crafting*, konstruksi, dan pertempuran dalam satu lingkungan virtual yang dinamis. Karakteristik dunia permainan yang

terbuka dan kompleks menjadikan Minecraft sebagai platform yang potensial untuk pengembangan dan eksperimen kecerdasan buatan. Lingkungan Minecraft menyediakan beragam kemungkinan aksi dan interaksi, sehingga mampu merepresentasikan skenario yang menyerupai lingkungan dunia nyata dalam konteks simulasi digital (Singh, 2020).

### 2.3 Genre *SandBox*

Genre *sandbox* merupakan jenis permainan video yang memberikan kebebasan tinggi kepada pemain untuk berinteraksi dengan lingkungan permainan tanpa harus mengikuti alur cerita yang linear. Pada genre ini, pemain dapat menentukan tujuan, strategi, dan aktivitas bermain secara mandiri sesuai kreativitas masing-masing. Konsep *sandbox* diambil dari istilah “kotak pasir” (*sandbox*) yang memungkinkan seseorang membangun atau menciptakan sesuatu secara bebas.

Karakteristik utama dari genre *sandbox* adalah adanya *open-ended gameplay*, yaitu sistem permainan yang tidak membatasi pemain pada satu tujuan utama tertentu. Pemain dapat melakukan eksplorasi, pembangunan, *crafting*, interaksi dengan *environment*, hingga eksperimen terhadap mekanisme permainan secara bebas. Selain itu, *sandbox game* umumnya memiliki dunia virtual luas (*open world*) yang dapat dijelajahi pemain tanpa batasan jalur permainan yang kaku. Genre *sandbox* dirancang untuk mendukung kreativitas pemain melalui kebebasan dalam memanipulasi lingkungan virtual dan menciptakan pengalaman bermain yang unik. Genre ini memungkinkan pemain untuk belajar melalui eksplorasi, eksperimen, dan penciptaan objek di dalam permainan (Rahimi et al., 2024).

### 2.4 Mod atau *Modification*

Mod atau *modification* merupakan perubahan atau penambahan konten pada sebuah permainan video yang dibuat untuk memodifikasi fitur asli permainan. Mod dapat berupa penambahan karakter, item, mekanisme permainan, map, antarmuka pengguna, hingga sistem *Artificial Intelligence* (AI). Tujuan utama dari mod adalah memperluas pengalaman bermain serta memberikan fitur baru yang tidak tersedia pada versi asli permainan (Charvát & Gvoždiak, 2021). Dalam industri game, modding menjadi salah satu bentuk partisipasi komunitas yang sangat populer karena memungkinkan pemain berkontribusi secara langsung terhadap

pengembangan permainan. Aktivitas modding memungkinkan komunitas menciptakan konten baru maupun memodifikasi mekanisme permainan yang sudah ada sesuai kebutuhan dan kreativitas pengguna (Charvát and Gvoždiak, 2021).

## 2.5 Fabric Framework

Fabric merupakan *framework* modding yang bersifat ringan, modular, dan berkembang dengan cepat, serta digunakan secara luas oleh komunitas pengembang mod Minecraft. Fabric dirancang untuk memberikan akses ke berbagai *hooks* yang memungkinkan pengembang memodifikasi perilaku game tanpa harus mengubah atau memengaruhi berkas inti Minecraft. Pendekatan ini menjadikan Fabric lebih aman, fleksibel, dan mudah dipelihara dalam pengembangan mod. Secara arsitektural, Fabric terdiri dari dua komponen utama, yaitu Fabric Loader dan Fabric API. Fabric Loader berfungsi untuk memuat mod saat game dijalankan, sedangkan Fabric API menyediakan kumpulan pustaka yang mempermudah pengembangan mod, seperti pendaftaran entitas, pengelolaan *event*, serta akses ke fungsi-fungsi tingkat tinggi lainnya. Dibandingkan dengan *modding framework* lain seperti Forge, Fabric memiliki keunggulan dalam hal desain yang minimalis, performa yang lebih ringan, serta kecepatan pembaruan yang mengikuti versi terbaru Minecraft (Fabric, 2026).

## 2.6 Game Community

*Game community* atau komunitas game merupakan kelompok individu yang terbentuk melalui interaksi sosial antar pemain dalam suatu permainan digital. Komunitas ini biasanya berkembang melalui aktivitas bermain bersama, diskusi, pertukaran informasi, pembuatan konten, hingga pengembangan modifikasi (*modding*) pada game tertentu. Keberadaan komunitas game memiliki peran penting dalam memperpanjang umur permainan karena komunitas dapat terus menghasilkan inovasi, konten, dan aktivitas sosial di luar pengembangan resmi oleh developer. Komunitas game merupakan kelompok individu yang terbentuk melalui interaksi sosial dalam lingkungan permainan digital, di mana partisipasi aktif pemain dalam aktivitas kolaboratif menjadi faktor kunci dalam membentuk identitas sosial di ruang virtual.

(Uchida Pedro Makoto and Poor, 2025). Selain itu, komunitas game juga berfungsi sebagai sarana berbagi pengetahuan, strategi permainan, serta pengembangan kreativitas pemain melalui konten buatan pengguna (*user-generated content*).

## 2.7 Large Language Model (LLM)

*Large Language Model* (LLM) merupakan model kecerdasan buatan yang dirancang untuk memahami, memproses, dan menghasilkan bahasa alami dengan memanfaatkan arsitektur *transformer* serta pelatihan pada data teks berskala besar (Brown et al., 2020). Kemampuan tersebut memungkinkan LLM untuk memahami konteks percakapan, menghasilkan respons yang natural, serta melakukan *reasoning* berbasis teks. Dalam pengembangan agen cerdas, LLM banyak digunakan sebagai mesin dialog karena mampu mengatasi keterbatasan sistem berbasis aturan atau skrip statis. Penerapan LLM pada sistem Non-Player Character (NPC) dalam game memungkinkan NPC memahami instruksi pemain secara lebih fleksibel dan menjaga konsistensi dialog, sehingga interaksi yang dihasilkan menjadi lebih adaptif dan menyerupai komunikasi manusia.

## 2.8 Gemini API

Google menyediakan layanan Gemini API sebagai antarmuka pemrograman aplikasi (*Application Programming Interface/API*) yang memungkinkan pengembang mengakses model Artificial Intelligence generatif Gemini untuk berbagai kebutuhan, seperti pemrosesan bahasa alami (*Natural Language Processing/NLP*), pembuatan teks, analisis gambar, *reasoning*, hingga pembuatan kode program. Selain kemampuan generatif, Gemini juga memiliki performa yang baik dalam *reasoning*, pemrosesan instruksi, dan *code generation*. Penelitian oleh Akter dkk. menunjukkan bahwa model Gemini memiliki kemampuan yang kompetitif dalam tugas pemahaman bahasa, *reasoning* kompleks, translasi, serta *instruction-following agent* (Akter et al., 2023).

## 2.9 Computer Vision untuk Game

*Computer Vision* dalam pengembangan game masih relatif jarang diterapkan secara langsung, khususnya dalam konteks pengenalan emosi pemain. Dalam penelitian ini, metode *Computer Vision* yang digunakan berfokus pada pendekatan klasifikasi gambar (*image classification*) untuk mendukung pengenalan emosi pemain

berbasis ekspresi wajah (*facial expression recognition*) dengan menggunakan algoritma dari *Convolutional Neural Network* (CNN). Proses ini dilakukan dengan menangkap citra wajah pemain melalui kamera, kemudian memprosesnya melalui tahapan deteksi wajah, ekstraksi fitur visual, dan klasifikasi emosi. Pendekatan tersebut memungkinkan sistem memperoleh informasi visual non-verbal yang merepresentasikan kondisi emosional pemain secara lebih natural. Dengan demikian, penerapan *Computer Vision* sebagai antarmuka interaksi dalam game berpotensi mendukung pengalaman bermain yang lebih intuitif, adaptif, dan imersif, serta membuka peluang penerapan teknologi ini secara lebih luas pada industri game (Chursin & Semenov, 2021).

## 2.10 TensorFlow

TensorFlow merupakan framework open-source yang dikembangkan oleh Google untuk mendukung pengembangan aplikasi berbasis machine learning dan deep learning. TensorFlow dirancang untuk mempermudah proses pembangunan, pelatihan, evaluasi, dan deployment model *Artificial Intelligence* (AI) pada berbagai platform, seperti desktop, server, *cloud*, maupun perangkat mobile. TensorFlow menggunakan konsep *dataflow graph*, yaitu representasi komputasi dalam bentuk graf yang terdiri dari node dan tensor. Node merepresentasikan operasi matematika, sedangkan tensor merupakan struktur data multidimensi yang digunakan untuk menyimpan data selama proses komputasi. Pendekatan ini memungkinkan TensorFlow melakukan komputasi secara efisien pada CPU, GPU, maupun TPU. Framework ini mendukung berbagai metode machine learning, seperti neural network, convolutional neural network (CNN), recurrent neural network (RNN), reinforcement learning, hingga transformer-based model. Selain itu, TensorFlow menyediakan berbagai pustaka tambahan seperti Keras untuk membangun model neural network secara lebih sederhana dan TensorFlow Lite untuk deployment model pada perangkat mobile dan embedded system. Menurut Das, TensorFlow dikembangkan untuk menangani komputasi *machine learning* skala besar dengan performa tinggi serta fleksibilitas yang mendukung riset maupun pengembangan aplikasi industri (Das, 2024).

### 2.11 Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) merupakan arsitektur *deep learning* terkemuka yang dirancang khusus untuk memproses data visual dengan ekstraksi fitur otomatis yang hierarkis. Berbeda dengan metode tradisional, CNN mampu mempelajari pola visual mulai dari fitur sederhana hingga bentuk objek yang kompleks secara *end-to-end* (Khan et al., 2020). CNN dirancang untuk mengenali pola secara otomatis melalui proses ekstraksi fitur bertingkat, sehingga sangat efektif digunakan dalam bidang *computer vision* seperti *image classification*, *object detection*, dan *face recognition*. CNN bekerja menggunakan beberapa lapisan utama, yaitu *convolution layer*, *pooling layer*, dan *fully connected layer*. *Convolution layer* berfungsi mengekstraksi fitur dari gambar menggunakan filter atau kernel, *pooling layer* digunakan untuk mengurangi dimensi data sekaligus mempertahankan fitur penting, sedangkan *fully connected layer* digunakan untuk proses klasifikasi akhir. Keunggulan CNN dibandingkan metode machine learning tradisional adalah kemampuannya melakukan *automatic feature extraction* tanpa memerlukan proses ekstraksi fitur manual. Hal tersebut membuat CNN mampu menghasilkan performa tinggi dalam pengolahan citra dan pengenalan objek. Selain itu, CNN juga memiliki kemampuan untuk mempelajari fitur visual secara hierarkis, mulai dari pola sederhana seperti tepi dan tekstur hingga bentuk objek yang lebih kompleks.

### 2.12 Penelitian Terdahulu

**Tabel 2.1** Penelitian Terdahulu

| Peneliti              | Judul                                                            | Metode                                                                   | Hasil                                                                                           |
|-----------------------|------------------------------------------------------------------|--------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------|
| G. Wang et al. (2023) | Voyager: An Open-Ended Embodied Agent with Large Language Models | Agan otonom berbasis LLM dengan skill library dan memori jangka panjang. | Agan mampu belajar dan mengeksplorasi lingkungan Minecraft secara mandiri tanpa skenario tetap. |

|                          |                                                                     |                                                                                                                                                                                                          |                                                                                                                                                                                                                                                                                   |
|--------------------------|---------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Chursin & Semenov (2021) | Using Computer Vision In The Gameplay Of Educational Computer Games | Eksperimen integrasi computer vision ke game edukasi dengan HOG + SVM untuk klasifikasi gesture tangan.                                                                                                  | Akurasi tinggi (96–98%) pada latar sederhana, menurun ( $\pm 93\%$ ) pada latar kompleks. Sistem efektif untuk interaksi sederhana dalam game edukasi.                                                                                                                            |
| García-Sánchez (2019)    | Artificial Intelligence and Games                                   | Kajian literatur dan eksperimen konseptual mengenai penerapan AI dalam game. Membahas metode seperti machine learning, evolutionary algorithms, dan player modeling untuk mendukung desain game adaptif. | AI dalam game menunjukkan potensi besar karena dapat digunakan untuk membuat NPC lebih realistis, menghasilkan konten secara prosedural, memodelkan perilaku pemain agar pengalaman bermain lebih personal, serta mengembangkan game sebagai sarana penelitian kecerdasan buatan. |
| Park et al. (2023)       | Generative Agents: Interactive Simulacra of Human Behavior          | Kajian konseptual dan analisis literatur mengenai bagaimana LLM dapat berfungsi sebagai “cognitive mirrors” yang merefleksikan proses berpikir manusia.                                                  | Menunjukkan bahwa LLM dapat membantu memahami pola kognitif manusia, mendukung pembelajaran, serta menjadi alat refleksi dalam riset psikologi dan pendidikan. Namun, juga diingatkan adanya keterbatasan interpretasi dan bias model.                                            |

|                           |                                                                                                                                                                                  |                                                                                                                                                     |                                                                                                                                                                             |
|---------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Meiningdias et al. (2025) | Exploring the Use of Minecraft Education Edition to Support Computational Thinking, Coding Skills, and AI Literacy: A Case Study in an Indonesian Madrasah Informatics Classroom | Studi kasus kualitatif di kelas informatika madrasah. Data dikumpulkan melalui observasi, dokumentasi proyek coding, serta refleksi guru dan siswa. | MEE meningkatkan keterampilan computational thinking, coding dasar, dan literasi AI. Lingkungan berbasis game membuat siswa lebih termotivasi dan aktif dalam pembelajaran. |
| (Z. Wang, 2025)           | Application and Analysis of Large Language Models in Game NPCs                                                                                                                   | Few-shot prompting, Generative Agent, Federated Learning                                                                                            | NPC lebih realistis, mampu refleksi dan adaptasi.                                                                                                                           |

Berdasarkan Tabel 2.1, mayoritas penelitian terdahulu mengenai karakter virtual masih bersifat unimodal, di mana pengenalan visual dan pemrosesan dialog teks berjalan terpisah tanpa kemampuan adaptasi emosional secara instan. Keterbatasan ini menyebabkan interaksi NPC bawaan menjadi kaku dan kurang responsif terhadap kondisi dinamis pemain. Oleh karena itu, selain mengusulkan sistem *Intelligent Companion NPC* berbasis arsitektur terdistribusi hibrida *Sense-Think-Act*, penelitian ini menawarkan kebaruan berupa integrasi ganda (*dual-modality*) yang mensinkronisasikan modul *Computer Vision* (Python TensorFlow) untuk deteksi ekspresi wajah near *real-time* dan mesin penalaran bahasa alami (Google Gemini API). Melalui modul *Context Builder* berbasis format *Context JSON*, sistem ini berhasil menjembatani data emosi pemain dan status lingkungan *game* Minecraft, sehingga NPC mampu mengeksekusi instruksi bahasa alami sekaligus menyesuaikan gaya percakapan secara adaptif berdasarkan kondisi emosional riil manusia.

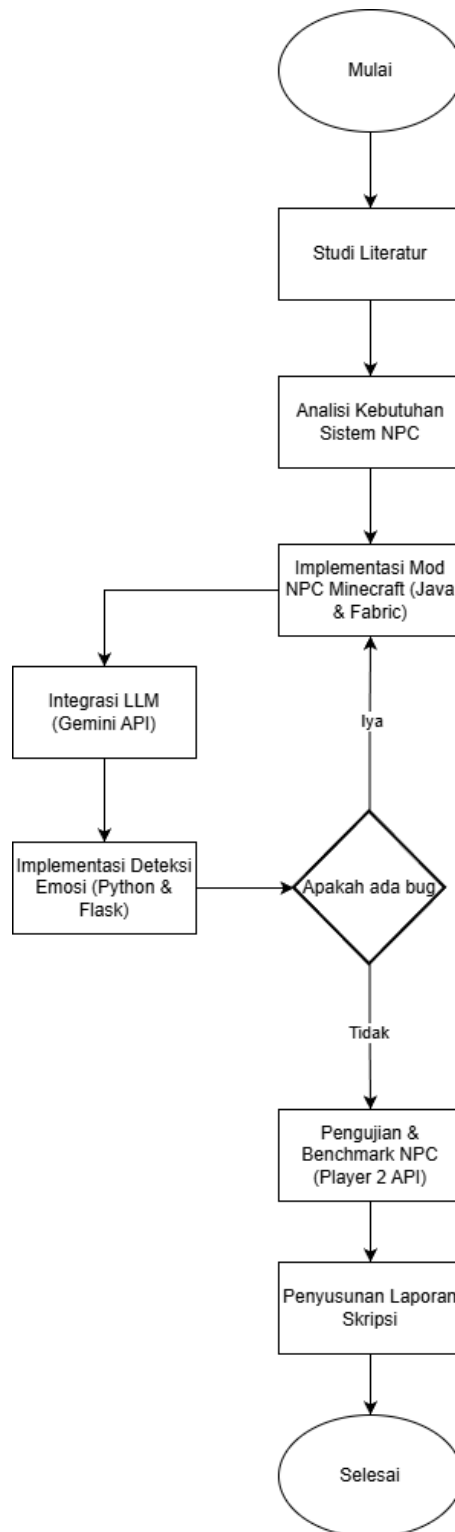
## **BAB III METODE PENELITIAN**

### **3.1 Rancangan Penelitian**

Penelitian ini menggunakan pendekatan gabungan (Mixed Methods) yang mengkombinasikan pendekatan kuantitatif dan kualitatif. Pendekatan kuantitatif digunakan untuk mengukur metrik performa teknis sistem, seperti akurasi model *Computer Vision* dalam mendeteksi emosi (*Loss* dan *Accuracy*). Di sisi lain, pendekatan kualitatif digunakan untuk mengevaluasi kualitas interaksi, konteks percakapan, dan naturalitas respons *Non-Playable Character* (NPC) terhadap dinamika emosi dan lingkungan permainan. Jenis penelitian yang diterapkan adalah Riset Eksperimental (*Experimental Research*). Penelitian difokuskan pada percobaan perancangan dan implementasi arsitektur sistem terdistribusi hibrida (menghubungkan Java/Fabric, Python, dan Gemini API) pada lingkungan *sandbox simulation* Minecraft.

### **3.2 Tahapan Penelitian**

Tahapan penelitian ini menjabarkan langkah-langkah sistematis yang dilakukan mulai dari konseptualisasi hingga penyelesaian sistem. Alur tahapan penelitian terdiri dari beberapa langkah seperti gambar 3.1.



**Gambar 3.1** Diagram Tahapan Penelitian

### 1. Studi Literatur

Mengkaji konsep akademis terkait agen otonom, *Large Language Model* (LLM), deteksi emosi berbasis *Computer Vision* (*Affective Computing*), dan integrasi API pada *video game engine*.

### 2. Analisis Kebutuhan Sistem

Mengidentifikasi fitur, sensor lingkungan (seperti cuaca, waktu, entitas), dan spesifikasi teknis (modul *Python* dan *Java*) yang diperlukan untuk membangun siklus *Sense-Think-Act-Evaluate* pada NPC.

### 3. Perancangan Arsitektur

Mendesain infrastruktur sistem terdistribusi, memisahkan modul *Computer Vision* (*Python/Flask*) untuk ekstraksi fitur wajah, mesin *Minecraft* (*Java/Fabric*) sebagai orkestrator, dan *cloud-based intelligence* (*Gemini API*) untuk penalaran.

### 4. Implementasi Sistem

Melakukan penulisan kode (*coding*) pembuatan mod *Minecraft* untuk memasukkan sistem sensor modular, antarmuka komunikasi (*HTTP*), mekanisme persistensi data (*NBT Storage*), dan *action handler*. Serta melakukan pelatihan model *Computer Vision* untuk deteksi emosi wajah.

### 5. Pengujian dan Evaluasi Iteratif

Melakukan pengujian sistem menggunakan metrik evaluasi model *Computer Vision* serta proses *benchmarking* performa agen. Sistem disempurnakan secara iteratif untuk meningkatkan akurasi, waktu respons, dan persistensi memori (kesadaran ruang/spasial).

### 6. Penyusunan Laporan

Pendokumentasian seluruh hasil riset eksperimental, analisis arsitektur, dan evaluasi performa ke dalam laporan skripsi secara terstruktur.

## 3.3 Objek Penelitian

Objek utama dalam penelitian ini adalah agen otonom (*Non-Playable Character*) yang disimulasikan di dalam lingkungan virtual *sandbox* *Minecraft*. Objek ini diteliti dalam hal kemampuannya memproses informasi kognitif (*Sense-Think-Act*) yang bersumber dari data lingkungan sekitar (waktu, cuaca, inventaris, memori

spasial) dan data visual (emosi pemain), serta responnya dalam menghasilkan tindakan dinamis dan interaksi bahasa alami secara *real-time*.

## BAB IV HASIL DAN PEMBAHASAN

### 4.1 Analisis Kebutuhan Sistem

Tahap analisis kebutuhan merupakan langkah awal untuk mengidentifikasi seluruh komponen yang diperlukan agar sistem agen cerdas dapat beroperasi secara optimal. Analisis ini mencakup spesifikasi perangkat keras, infrastruktur perangkat lunak, serta fungsionalitas utama yang harus dimiliki oleh sistem untuk mencapai interaksi yang responsif dan empatik.

#### 4.1.1 Kebutuhan Perangkat Keras (Hardware Requirements)

Untuk menjalankan sistem agen cerdas yang mengintegrasikan simulasi dunia 3D Minecraft, pengolahan citra secara *near real-time*, dan komunikasi *cloud intelligence*, diperlukan infrastruktur perangkat keras yang mampu menangani beban komputasi paralel.

**Tabel 4.1** Spesifikasi Perangkat Keras Pengujian

| Komponen   | Spesifikasi Minimal | Unit yang Digunakan |
|------------|---------------------|---------------------|
| Processor  | Quad-core 2.5 GHz   | Ryzen 5 5600H       |
| RAM        | 8 GB DDR4           | 16 GB DDR4          |
| Storage    | 512 SSD             | SSD 1TB             |
| Camera     | 720p HD Webcam      | 720p HD Webcam      |
| Connection | 20 Mbps Stable      | 50 Mbps Wi-Fi       |

#### 4.1.2 Kebutuhan Perangkat Lunak (Software Requirements)

Keberhasilan integrasi sistem terdistribusi pada penelitian ini sangat bergantung pada ekosistem perangkat lunak yang digunakan. Perangkat lunak tersebut berfungsi sebagai kerangka kerja (*framework*), bahasa pemrograman, dan layanan intelegensi yang memungkinkan komunikasi antar-platform dapat berjalan secara sinkron. Kebutuhan perangkat lunak ini dibagi menjadi tiga pilar utama:

1. Lingkungan Pengembangan *Game (Java-based Minecraft Engine)*:

a. Java Development Kit (JDK) 21

Versi Java terbaru digunakan untuk memastikan kompatibilitas dengan fitur-fitur terkini Minecraft versi 1.21 serta memberikan performa manajemen memori yang lebih efisien.

b. Fabric Mod Loader & API

Fabric dipilih sebagai kerangka kerja modifikasi karena sifatnya yang ringan (*lightweight*) dan modular. Fabric API menyediakan titik akses (*hooks*) ke dalam kode sumber Minecraft, memungkinkan pengembangan sensor kustom dan sistem pergerakan agen cerdas.

2. Lapisan Persepsi Visual (*Python-based Computer Vision*):

- a. Python 3.12: Bertindak sebagai lingkungan eksekusi utama untuk modul Computer Vision.
- b. TensorFlow: Pustaka utama yang digunakan untuk melakukan deteksi wajah, ekstraksi fitur landmark, dan klasifikasi emosi pemain secara *near real-time* dengan model CNN.
- c. Flask Web Framework: Digunakan untuk membangun microservice lokal yang berfungsi menyediakan *endpoint* REST API, sehingga data emosi dari Python dapat diakses oleh lapisan Java melalui protokol HTTP.

3. Layanan Kecerdasan Buatan dan Pertukaran Data:

- a. Gemini API (*Google AI SDK*): Berfungsi sebagai "Otak Pusat" atau lapisan penalaran agen cerdas. Layanan ini memproses konteks Markdown dan menghasilkan instruksi dalam format JSON.
- b. GSON Library: Pustaka Java untuk melakukan serialisasi dan deserialisasi data JSON, memungkinkan *Action Handler* membedah instruksi dari LLM menjadi perintah teknis.
- c. NBT (*Named Binary Tag*) Library: Pustaka internal Minecraft yang digunakan untuk implementasi persistensi data, memastikan variabel memori agen tersimpan secara permanen dalam format biner.

4. Alat Bantu Pengembangan (*Development Tools*):

- a. IntelliJ IDEA: Digunakan sebagai *Integrated Development Environment* (IDE) utama untuk penulisan kode Java dan manajemen proyek *Fabric*.
- b. Visual Studio Code: Digunakan untuk pengembangan skrip Python dan pengujian Flask API.

#### 4.1.3 Analisis Kebutuhan Fungsional

Analisis kebutuhan fungsional mendefinisikan kemampuan mendasar yang harus dimiliki oleh agen cerdas agar dapat beroperasi secara otonom dan interaktif di dalam lingkungan simulasi. Kebutuhan fungsional ini dirancang untuk memenuhi siklus kognitif agen, yang meliputi aspek persepsi, memori, dan komunikasi sebagai berikut:

1. Kemampuan Persepsi Multi-Sumber (*Sensing Capability*):

Agan cerdas diwajibkan memiliki kemampuan untuk mengekstraksi dan memahami informasi dari lingkungan sekitarnya secara komprehensif. Hal ini mencakup persepsi terhadap variabel fisik dunia Minecraft (seperti cuaca, waktu, dan lokasi), identifikasi entitas di sekitar (pemain dan monster), serta kemampuan penglihatan komputer (*computer vision*) untuk mengenali ekspresi emosional pemain. Kemampuan "melihat" ini tidak hanya terbatas pada data mentah, tetapi juga mencakup kemampuan untuk melakukan pemrosesan awal terhadap informasi yang relevan bagi keselamatan agen dan pemiliknya.

2. Kemampuan Memori Jangka Pendek dan Panjang (*Memory Capability*):

Agan cerdas harus memiliki kemampuan untuk menyimpan dan memanggil kembali informasi guna menjaga kesinambungan perilaku. Kebutuhan ini mencakup memori jangka pendek untuk melacak alur percakapan dan status tugas terakhir, serta memori jangka panjang untuk menyimpan sejarah hubungan sosial dengan pemain (skor kepercayaan) dan pengetahuan spasial mengenai lokasi strategis yang pernah dikunjungi. Dengan kemampuan "mengingat", agen diharapkan dapat belajar dari hasil tindakannya di masa lalu dan tidak bersifat repetitif, sehingga menciptakan interaksi yang lebih dinamis dan menyerupai kognisi makhluk hidup.

3. Kemampuan Penalaran dan Komunikasi Alami (*Reasoning & Speech Capability*):

Sebagai jembatan interaksi dengan manusia, agen cerdas harus mampu berkomunikasi menggunakan bahasa alami secara kontekstual. Kebutuhan ini mencakup kemampuan untuk melakukan penalaran logis (*reasoning*) terhadap

seluruh data sensor dan memori untuk menghasilkan keputusan yang tepat. Agen harus mampu memberikan respon verbal yang empatik melalui chat serta mewujudkan keputusan tersebut dalam bentuk aksi nyata (seperti mengikuti pemain, melindungi dari serangan, atau mengumpulkan sumber daya). Kemampuan "bicara" ini harus tetap selaras dengan persona yang telah ditetapkan, sehingga agen memiliki identitas karakter yang konsisten dan meyakinkan.

## 4.2 Perancangan Sistem Terintegrasi

Sebagaimana telah dijelaskan pada bab sebelumnya, penelitian ini menggunakan Minecraft sebagai *virtual testbed* untuk menguji agen cerdas berbasis *Large Language Model* (LLM). Sistem yang dikembangkan tidak dirancang sebagai aplikasi monolitik, melainkan sebagai sistem terdistribusi hybrid yang mengintegrasikan beberapa teknologi berbeda untuk membentuk siklus kognitif yang utuh. Arsitektur sistem mengadopsi pendekatan *Sense–Think–Act* (STA), yaitu paradigma agen cerdas yang terdiri dari proses persepsi lingkungan, penalaran, dan eksekusi aksi secara terintegrasi untuk menghasilkan perilaku adaptif terhadap kondisi dinamis (Stuart Russell & Peter Norvig, 2021)

Implementasi arsitektur tersebut melibatkan tiga platform utama, yaitu:

1. *Python* sebagai modul *Computer Vision* untuk pemrosesan data visual
2. *Java (Fabric)* sebagai *game engine* untuk simulasi dan eksekusi agen
3. *Gemini API* sebagai *cloud-based intelligence service* untuk proses penalaran berbasis LLM

Integrasi antar platform ini memungkinkan terbentuknya sistem yang fleksibel, modular, dan mampu menangani proses kognitif secara *end-to-end*, mulai dari akuisisi data hingga eksekusi aksi dalam lingkungan simulasi.

### 4.2.1 Minecraft sebagai Virtual Testbed dan *Sandbox Simulation*

Pemilihan Minecraft sebagai lingkungan simulasi didasarkan pada kompleksitas Environment yang ditawarkannya. Dalam perspektif riset AI, Minecraft sudah banyak menjadi virtual testbed untuk pengujian AI, sebagaimana telah dibahas pada Bab II (Penelitian Terdahulu). Lingkungan Minecraft menyediakan berbagai

karakteristik yang merepresentasikan kondisi dunia nyata dalam bentuk simulasi digital, seperti:

1. Dinamika Lingkungan :

Dunia simulasi Minecraft menawarkan tingkat kompleksitas lingkungan yang tinggi melalui siklus waktu (*day-night cycle*) seperti pada gambar 4.1 dan sistem cuaca yang dinamis. Dinamika ini bukan sekadar aspek visual, melainkan variabel lingkungan yang secara langsung memengaruhi kondisi keamanan dan operasional di dalam simulasi. Perubahan waktu dari siang ke malam, misalnya, secara otomatis mengubah status lingkungan dari aman menjadi berisiko tinggi karena munculnya entitas agresif. Hal ini menuntut agen cerdas untuk memiliki kemampuan adaptasi yang responsif.



**Gambar 4.1** Siklus Siang dan Malam

2. Interaksi *Multi-Entity* :

Lingkungan simulasi Minecraft menyediakan ekosistem yang dinamis melalui keberadaan berbagai entitas lain di dalamnya. Interaksi *multi-entity* ini terjadi karena agen cerdas yang dibangun tidak berada di ruang hampa, melainkan berbagi dunia dengan pemain, NPC standar lainnya, serta berbagai jenis entitas musuh atau monster. Keberadaan entitas-entitas yang beragam ini secara alami menciptakan skenario sosial dan tingkat keamanan yang tidak dapat diprediksi. Oleh karena itu, sistem NPC dirancang agar memiliki kesadaran situasional (*situational awareness*) terhadap sekelilingnya untuk membedakan antara kawan, netral, maupun ancaman.

### 3. Abstraksi Ruang 3D :

Selain dinamika lingkungan dan interaksi antar entitas, keunggulan Minecraft sebagai platform pengujian terletak pada sistem abstraksi ruang tiga dimensinya yang sangat terstruktur. Seluruh dunia simulasi direpresentasikan dalam bentuk koordinat *Cartesian* (X, Y, dan Z) seperti yang di tunjukan pada gambar 4.2, di mana setiap elemen lingkungan, mulai dari blok tanah hingga posisi pemain, dapat didefinisikan secara presisi dalam ruang kognitif agen. Abstraksi ini sangat krusial karena memberikan kerangka kerja bagi agen cerdas untuk memahami konsep navigasi dan jarak.



```

Chunks[C] W: 529, 313 E: 162,100,313
Chunks[S] W: 4234 E: 289,289,171,970,970,0
minecraft:overworld FC: 0
XYZ: -655.675 / 72.00000 / -929.474
Block: -656 72 -930 [0 8 14]
Chunk: -41 4 -59 [23 5 in r.-2.-2.mca]
Facing: south (Towards positive Z) (-35.7
Client Light: 15 (15 sky, 0 block)
CH S: 71 M: 71
SH S: 71 O: 71 M: 71 ML: 71
Biome: minecraft:sparse_jungle
Local Difficulty: 1.50 // 0.00 (Day 0)
  
```

**Gambar 4.2** Visualisasi sistem koordinat Cartesian (X, Y, Z) dalam Minecraft

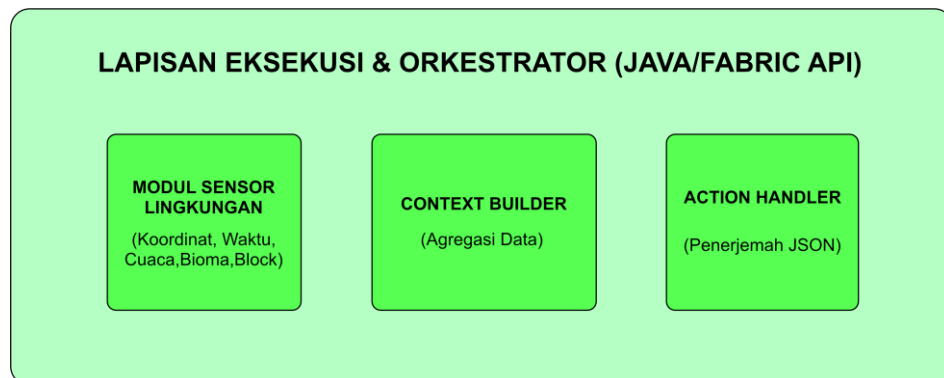
#### 4.2.2 Komponen Arsitektur Integrasi Sistem

Bagian ini memaparkan komponen yang membangun sistem agen cerdas secara keseluruhan. Arsitektur ini mengadopsi model sistem terdistribusi untuk memisahkan beban kerja antara mesin simulasi, pemrosesan sensor, dan penalaran logika tingkat tinggi ke dalam empat lapisan utama:

##### 1. Lapisan Eksekusi dan Lingkungan (Java/Fabric API):

Lapisan ini berfungsi sebagai fondasi sistem dan *Data Orchestrator*. Lapisan ini terdiri dari tiga komponen utama seperti yang ditampilkan pada gambar 4.3. Lapisan ini bertanggung jawab penuh atas manajemen status dunia simulasi, penyediaan akses ke API internal Minecraft melalui Fabric, serta bertindak sebagai pusat kendali yang menyatukan seluruh variabel sensorik dari Modul Sensor Lingkungan pada *Context Builder* sebelum diproses di

lapisan penalaran pusat. Dan pada lapisan ini terdapat *Action Handler* yang berfungsi sebagai penerjemah JSON dari lapisan Penalaran Pusat.



**Gambar 4.3** Komponen Lapisan Eksekusi & Orkestrator

2. Lapisan Persepsi dan Analisis Visual (Python & Flask):

Bertindak sebagai modul ekstraksi fitur visual yang terpisah dari mesin utama. Seperti pada gambar 4.4 lapisan ini memanfaatkan kamera sebagai sensor visual, ekosistem Python yaitu Tensorflow untuk pemrosesan citra, dan menggunakan *Flask* sebagai *microservice* lokal. Keberadaan lapisan ini memungkinkan sistem melakukan analisis data afektif tanpa membebani memori utama dari mesin simulasi.



**Gambar 4.4** Komponen Lapisan Persepsi

3. Lapisan Penalaran Pusat (Gemini API):

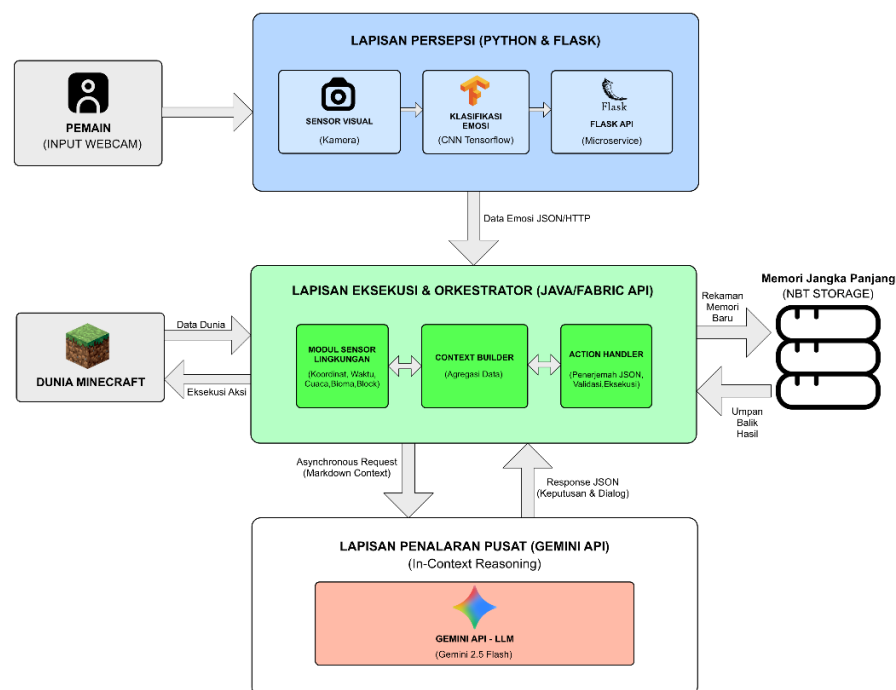
Merupakan pusat kognisi (*Cognitive Center*) yang berbasis pada infrastruktur *cloud*. Lapisan ini menggunakan model *Large Language Model* (LLM) Gemini 2.5 Flash sebagai unit pemroses keputusan. Fokus utama lapisan ini adalah melakukan sintesis informasi kompleks dan menghasilkan instruksi strategis yang logis bagi agen.

#### 4. Sistem Persistensi Data (*NBT Storage*):

Berfungsi sebagai modul penyimpanan data permanen yang terintegrasi langsung dengan format penyimpanan asli Minecraft, yaitu *Named Binary Tag* (NBT). Komponen ini berperan sebagai memori jangka panjang agen, yang menyimpan atribut permanen seperti identitas, kepribadian, serta riwayat interaksi masa lalu. Penggunaan NBT memastikan bahwa data "kesadaran" agen tetap terjaga dan dapat dimuat kembali meskipun sesi permainan atau server dihentikan.

#### 4.2.3 Mekanisme Aliran Data Terintegrasi

Realisasi dari sistem terdistribusi ini bergantung pada bagaimana informasi bergerak dan bertransformasi di antara komponen platform yang berbeda. Aliran data dirancang mengikuti siklus kognitif agen yang bersifat tertutup (*closed-loop*) seperti pada gambar 4.5, yang memastikan kesinambungan antara persepsi lingkungan dan tindakan nyata. Proses ini terbagi menjadi empat fase utama:



**Gambar 4.5** Aliran Data Terintegrasi

## 1. Fase Penginderaan dan Akuisisi Data (*Sensing Phase*)

Pada tahap awal, sistem mengumpulkan data mentah dari dua sumber utama secara paralel untuk membangun kesadaran situasional (*situational awareness*):

### a. Data Dunia (Java):

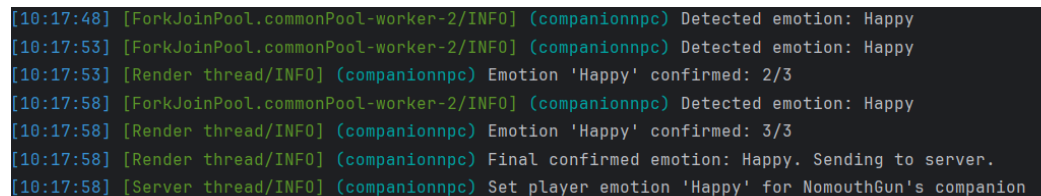
Modul sensor pada Java mengekstraksi variabel dunia Minecraft secara *real-time*, meliputi koordinat spasial (x, y, z), kondisi cuaca, bioma, tingkat cahaya, serta keberadaan entitas di sekitar agen.



**Gambar 4.6** Data Dunia Yang Diperoleh Agen

### b. Data Afektif (Python):

Secara independen, modul Python menganalisis ekspresi wajah pemain melalui kamera. Modul Java melakukan *near real-time polling* melalui protokol HTTP ke *microservice* Flask setiap 100 *tick* (5 detik). Data emosi ini kemudian divalidasi melalui mekanisme *consecutive confirmation* sebelum diteruskan ke tahap berikutnya guna menjamin stabilitas data dan meminimalkan galat deteksi (*noise*).



**Gambar 4.7** Data Emosi Pemain

## 2. Fase Orkestrasi dan Transformasi Semantik (*Context Assembly*)

Data mentah yang terkumpul di dalam *Orchestrator* (Java) melalui Context Builder tidak dikirimkan dalam bentuk mentah. Sistem melakukan transformasi semantik, yaitu mengubah data numerik dan kategori menjadi narasi berbasis *Markdown*. Sebagai contoh, koordinat posisi, jenis bioma, dan status cuaca diintegrasikan dengan data dari *NBT Storage* (memori jangka panjang) untuk disusun menjadi sebuah laporan situasi yang deskriptif. Proses ini krusial agar *Large Language Model* (LLM) dapat memahami konteks dunia simulasi bukan sebagai kumpulan angka, melainkan sebagai sebuah skenario logis yang bermakna.

```

### IDENTITAS & HUBUNGAN SOSIAL
- Nama: Odo
- Jenis Kelamin (Gender): Male
- Pemilik: NomouthGun
- Skor Kepercayaan (Trust): 82/100
- Emosi Pemain terhadapku: Happy
- Tingkat Percaya Diri (Confidence): 75/100
- Mood Saat Ini: Bersemangat
- Hasil Refleksi Hubungan: Pemain sering memberikan makanan, hubungan semakin erat.
- Posisi Saat Ini: [X: 245, Y: 68, Z: -120]
- Tugas Saat Ini: IDLE
- Status Tugas Terakhir: SUCCESS (Berhasil mengumpulkan 5 oak_log).
- CATATAN: Jika tugas terakhir sudah SUCCESS, jangan mengulangnya lagi kecuali diminta.

### STATUS FISIK
- Darah: 20/20

### INVENTARIS
- Memegang: iron_axe
- Isi Tas: 5x oak_log, 2x apple, 1x torch.
- Slot Kosong: 24 slot tersedia

### LINGKUNGAN
- Bioma: Forest
- Waktu: Siang Hari
- Cuaca: Cerah
- Tingkat Cahaya: 14 (Terang) [Blok: 14, Langit: 15]
- Akses Langit: Terbuka ke langit

### PENGLIHATAN (Blok di Sekitar)
- oak_log (Jarak: 3m)
- tall_grass (Jarak: 2m)
- dandelion (Jarak: 5m)

### INGATAN SPASIAL (Lokasi yang Diingat)
- [RESOURCES]: oak_log, birch_log
- [UTILITY]: crafting_table
- Catatan: Gunakan GOTO untuk mendatangi blok ini meskipun tidak terlihat di PENGLIHATAN.

### RADAR SOSIAL (Entitas di Sekitar)
- Cow (Jarak: 10m - Pasif)
- Sheep (Jarak: 12m - Pasif)

```

**Gambar 4.8** Potongan Struktur Konteks Markdown sebagai Hasil Transformasi Semantik.

### 3. Fase Penalaran Kognitif Asinkron (*Think & Reasoning Phase*)

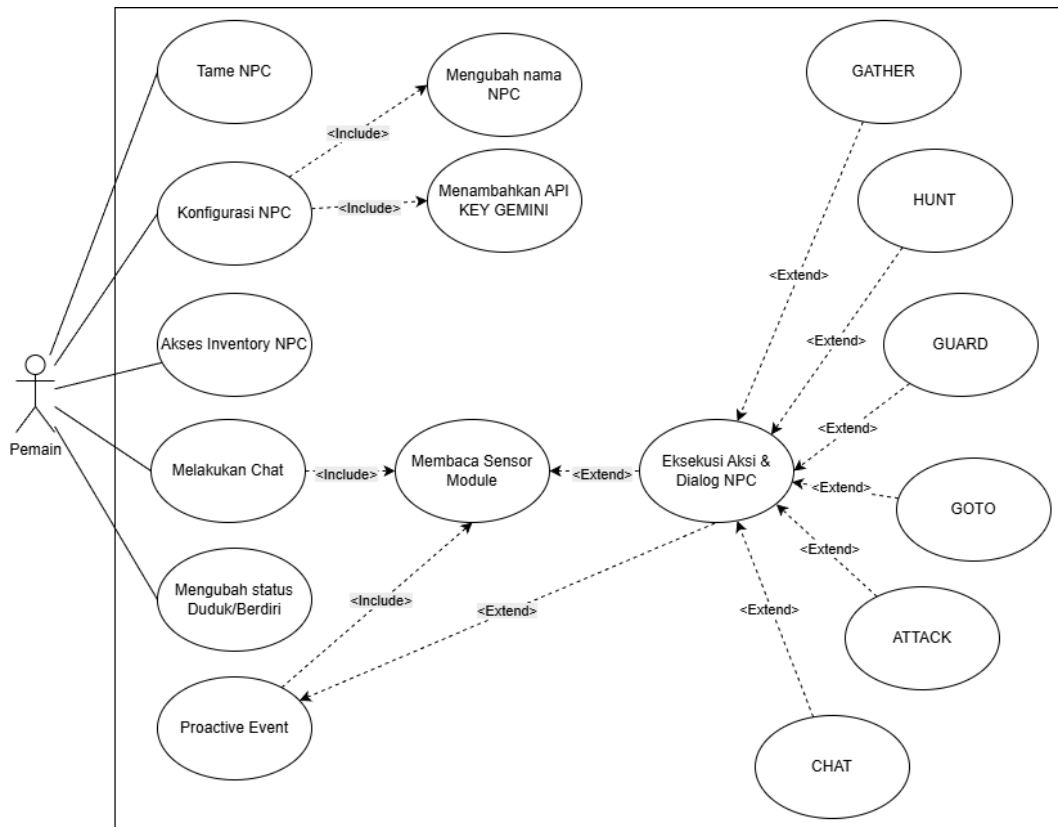
Narasi konteks yang telah disusun dikirimkan ke Gemini API menggunakan permintaan *asynchronous* POST melalui protokol REST API. Penggunaan komunikasi asinkron sangat vital agar *tick-rate* (performa) Minecraft tidak terhenti saat menunggu respons dari kecerdasan awan (*cloud intelligence*). Pada fase ini, Gemini memproses informasi emosi pemain dan informasi yang diperoleh dari fase orkestrasi data (*In-Context Reasoning*) kemudian mengembalikan respons dalam format *JSON Payload*.

### 4. Fase Eksekusi dan Lingkaran Umpan Balik (*Execution/Act & Feedback Loop*)

Pada fase akhir, *Action Handler* melakukan *parsing*, validasi, dan eksekusi terhadap instruksi JSON untuk menggerakkan agen di dalam Minecraft melalui sistem *AI Goals*. Segera setelah aksi dilakukan, sistem melakukan evaluasi hasil, status keberhasilan atau kegagalan aksi tersebut (beserta detail penyebabnya) dialirkan kembali ke dalam sistem penyimpanan NBT (*Named Binary Tag*) sebagai rekaman memori baru. Rekaman memori ini secara otomatis akan menjadi input pada fase penginderaan di siklus berikutnya. Hal ini menciptakan kesinambungan kognitif (*cognitive continuity*), di mana agen mampu belajar dari hasil tindakannya sendiri dan menyesuaikan keputusannya.

#### 4.2.4 Use Case Diagram

Perancangan fungsionalitas sistem pada penelitian ini dimodelkan menggunakan *Use Case Diagram* guna menggambarkan interaksi antara pengguna (*actor*) dengan batas fungsionalitas (*system boundary*) dari sistem *Mod Companion* NPC cerdas yang dibangun. Pemodelan ini menitikberatkan pada hak akses serta skenario layanan yang dapat dieksekusi di dalam gim Minecraft. Visualisasi model *Use Case Diagram* tanpa ketergantungan aktor eksternal guna mematuhi standarisasi baku *Unified Modeling Language* (UML) ditunjukkan pada Gambar 4.9.



**Gambar 4.9** Use Case Diagram

Berdasarkan Gambar 4.9, sistem ini melibatkan 1 (*satu*) aktor utama (*Primary Actor*), yaitu Pemain, yang mengendalikan dan berinteraksi dengan seluruh modul utama di dalam lingkungan permainan. Penjelasan mengenai deskripsi fungsionalitas dari setiap *use case* serta hubungan relasi dependensi `<<include>>` dan `<<extend>>` dijabarkan sebagai berikut:

1. Tame NPC: Fungsionalitas yang memungkinkan Pemain untuk mengubah status entitas NPC liar menjadi pendamping otonom (*companion*) yang terikat secara permanen pada identitas unik (*UUID*) pemain tersebut.
2. Konfig NPC: Fungsionalitas yang memberikan hak akses kepada Pemain untuk membuka antarmuka kustom (*GUI*) di dalam gim guna memodifikasi atribut identitas fisik NPC. *Use case* ini memiliki relasi dependensi `<<include>>` terhadap dua sub-proses wajib:
  - a. Mengubah Nama NPC: Memperbarui label teks nama entitas secara dinamis.

- b. Menambahkan API KEY GEMINI: Memasukan API Gemini agar NPC nya dapat melakukan penalaran.
3. Akses Inventory NPC: Fungsionalitas yang memungkinkan Pemain untuk membuka dan memanipulasi ruang penyimpanan internal (*backpack*) sebesar 27 slot yang dimiliki oleh NPC, baik untuk menaruh logistik pertahanan maupun mengambil material hasil kerja agen.
4. Melakukan Chat: Fungsionalitas interaksi bahasa alami, di mana Pemain mengirimkan pesan teks bebas (kasual) melalui antarmuka *chatting* bawaan gim untuk berkomunikasi dengan asisten virtual. *Use case* ini secara mutlak menyertakan (<<include>>) fungsi Membaca Konteks Lingkungan (Sensor Module) agar input tekstual dari pemain dapat dipadukan secara sinkron dengan parameter spasial dunia digital.
5. *Proactive Event*: Fungsionalitas internal agen yang berjalan secara otonom untuk mendeteksi perubahan kondisi lingkungan secara mendadak (peristiwa darurat). Sama halnya dengan UC-04, fungsionalitas ini wajib menyertakan (<<include>>) fungsi embaca Konteks Lingkungan (Sensor Module) untuk mengevaluasi data lingkungan makro secara berkelanjutan.
6. Eksekusi Aksi & Dialog AI: Fungsionalitas lanjutan yang merepresentasikan perluasan perilaku (<<extend>>) dari fungsi komunikasi chat (UC-04) dan pemicuan event (UC-07). Proses perluasan ini bersifat kondisional; hanya aktif apabila dokumen Response JSON hasil penalaran kognitif kecerdasan buatan menginstruksikan jenis tindakan mekanis tertentu. *Use case* ini membawahi (<<include>>) lima variasi aksi konkret gim:
  - a. Aksi Menambang (*GATHER*): Perintah mengumpulkan komoditas blok (seperti kayu atau batu) di sekitar koordinat agen.
  - b. Aksi Berburu (*HUNT*): Perintah mendeteksi dan mengejar entitas hewan target untuk pemenuhan logistik makanan.
  - c. Aksi Pengawasan (*GUARD*): Perintah mengamankan wilayah atau menyerang entitas monster yang mengancam keselamatan.
  - d. Aksi Berjalan (*GOTO*): Perintah navigasi pergerakan menuju titik koordinat atau blok fungsional tertentu (seperti *crafting table*).

- e. Aksi Percakapan (CHAT): Perintah melakukan konversasi tanpa ada action lain.
- 7. Mengubah Status Duduk/Berdiri: Fungsionalitas reaktif instan di mana Pemain dapat mengubah status pergerakan dasar fisik NPC (*Follow/Stay*) tanpa harus melalui pemrosesan bahasa alami pada *Large Language Model*.

#### 4.3 Implementasi Arsitektur Sensor Modular (Perception)

Lapisan persepsi dunia Minecraft pada agen cerdas menggunakan pola desain modular, penggunaan desain modular ini bertujuan agar manajemen dan pengujian sensor untuk NPC menjadi lebih fleksibel, terukur, dan independen. Dengan pendekatan ini, setiap domain informasi lingkungan diisolasi ke dalam sub-modul fungsional yang spesifik, sehingga memungkinkan proses pengembangan dan optimasi dilakukan secara terpisah dan mudah saat ingin melakukan maintenance pada bug ataupun menambahkan fitur baru pada agen cerdas.

Selain itu, agen cerdas tidak hanya menerima data mentah dari sensornya tetapi juga melakukan proses filtering dan prapemrosesan data. Tahapan ini krusial untuk mentransformasi informasi teknis yang kompleks seperti nilai koordinat, tingkat cahaya, dan ID entitas menjadi sebuah abstraksi semantik yang bermakna. Melalui mekanisme filtrasi tersebut, sistem dapat membuang informasi yang tidak relevan (*noise*) dan memprioritaskan data yang berdampak langsung pada keselamatan serta tujuan agen. Integrasi dari berbagai modul sensor ini pada akhirnya membentuk sebuah Kesadaran Situasional (*Situational Awareness*) yang mendalam, yang menjadi fondasi utama bagi agen dalam melakukan penalaran logis dan pengambilan keputusan di dalam dunia simulasi.

##### 4.3.1 Orkestrasi Sensor melalui *EnvironmentSensor*

*EnvironmentSensor* merupakan komponen inti dalam lapisan persepsi yang bertindak sebagai koordinator pusat (*Central Hub*) atau dirigen bagi seluruh sub-modul sensor yang dimiliki oleh agen cerdas. Dalam arsitektur sistem ini, *EnvironmentSensor* mengimplementasikan prinsip *Abstraction Layer*, di mana ia menyembunyikan kompleksitas pengambilan data dari berbagai *API Minecraft* yang berbeda dan menyediakannya dalam satu antarmuka terpadu. Implementasi *EnvironmentSensor* memiliki tiga peran strategis dalam sistem:

1. Manajemen Instansiasi Modul (*Modular Management*)

Saat sebuah entitas agen dimunculkan (*spawn*), kelas *EnvironmentSensor* secara otomatis menginisiasi seluruh sub-modul (seperti *WeatherModule*, *LightModule*, *EntityModule*, dll). Hal ini menjamin bahwa setiap aspek indra agen selalu tersedia dan tersinkronisasi dengan *state* permainan saat ini. Dengan memisahkan setiap sensor ke dalam variabel (*field*) yang berbeda, sistem dapat melakukan pembaruan data secara *independent*.

2. Pusat Penyaringan dan Agregasi Data (*Data Agregation*)

Sebagai orkestrator, kelas ini tidak hanya meneruskan data, tetapi juga melakukan agregasi. Sebagai contoh, saat sistem membutuhkan ringkasan inventaris, *EnvironmentSensor* akan memerintahkan *InventoryModule* untuk menghitung jumlah item, lalu mengonversi menjadi format teks yang ringkas. Proses ini mengurangi beban kerja pada lapisan penalaran (LLM) dengan memberikan data yang sudah setengah matang, bukan data yang mentah berantakan.

3. Antarmuka Komunikasi dengan *Context Builder*

*EnvironmentSensor* menyediakan metode akses (*getter*) yang telah dioptimasi untuk kebutuhan *NPCContextBuilder*. Alih-alih mengirimkan koordinat mentah dalam bentuk angka *double* yang panjang, orkestrator ini menyediakan fungsi untuk mengambil data koordinat dan mengembalikan koordinat dalam format sederhana [x, y, z]. Hal ini krusial dalam efisiensi penggunaan *token* pada saat pengiriman data ke *Gemini API*, karena setiap karakter yang dikirimkan memiliki dampak pada biaya dan latensi pemrosesan.

**Tabel 4.2** Daftar Modul Sensor dan Tanggung Jawabnya

| Nama Modul            | Tanggung Jawab Modul                                                              |
|-----------------------|-----------------------------------------------------------------------------------|
| <i>Weather Module</i> | Mendeteksi kondisi atmosfer dunia simulasi seperti hujan, cerah, dan badai petir. |
| <i>Time Module</i>    | Memantau siklus waktu dunia Minecraft untuk menentukan kategori siang atau malam. |
| <i>Light Module</i>   | Menganalisis intensitas cahaya di sekitar agen untuk menentukan keamanan area.    |

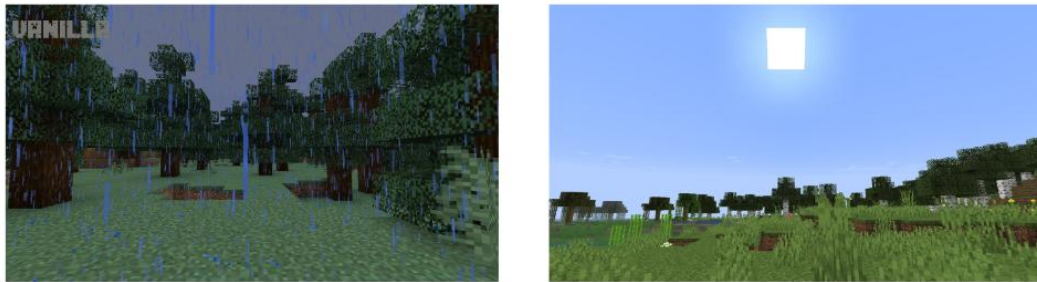
|                            |                                                                                        |
|----------------------------|----------------------------------------------------------------------------------------|
| <i>Biome Module</i>        | Mengidentifikasi jenis lokasi geografis dan karakteristik suhu di posisi agen berada.  |
| <i>Entity Module</i>       | Melakukan pemindaian, klasifikasi, dan penghitungan jarak terhadap entitas di sekitar. |
| <i>Block Module</i>        | Mengidentifikasi jenis dan lokasi blok sumber daya alam di dalam radius pandang agen.  |
| <i>Status Module</i>       | Melakukan monitoring terhadap kondisi kesehatan dan status fisik internal milik agen.  |
| <i>Inventory Module</i>    | Mengelola informasi ketersediaan barang, alat, dan sisa kapasitas tas milik agen.      |
| <i>Emotion Module</i>      | Mengintegrasikan data klasifikasi emosi pemain yang dikirimkan oleh sistem eksternal.  |
| <i>Health Event Module</i> | Mendeteksi kejadian kritis terkait kesehatan agen maupun pemain secara real-time.      |

#### 4.3.2 Implementasi Weather Module

Implementasi *Weather Module* bertujuan untuk memberikan kemampuan persepsi atmosferik kepada agen cerdas seperti yang di jelaskan pada tabel 4.1, yang secara fundamental mempengaruhi keputusan logis dalam aspek keselamatan dan navigasi. Dalam arsitektur kognitif agen, cuaca bukan sekadar elemen visual, melainkan sebuah variabel kondisi yang menentukan prioritas perilaku (misalnya, memicu *instinctive goal* untuk mencari perlindungan).

Data cuaca diekstraksi melalui interaksi langsung dengan kelas *net.minecraft.world.World* (yaitu pustaka yang ada di minecraft) pada *server-side*. Sumber data utama berasal dari variabel state dunia yang dikelola secara terpusat oleh Game Engine. Sistem memanggil metode *isRaining()* untuk mendeteksi adanya presipitasi (hujan/salju) dan *isThundering()* untuk mengidentifikasi kondisi badai . Data ini diperbarui setiap tick (1/20 detik) oleh mesin simulasi, namun untuk efisiensi kognitif, modul melakukan kueri (*polling*) hanya saat diperlukan oleh Context Builder.

Data mentah berupa nilai boolean kemudian diproses melalui tahap parameterisasi semantik. Status fisik tersebut dikonversi menjadi kategori tekstual "Hujan", "Badai", dan "Cerah". Abstraksi ini sangat krusial agar Large Language Model (LLM) dapat melakukan inferensi mengenai risiko lingkungan, seperti potensi serangan petir.



**Gambar 4.10** Cuaca pada minecraft (hujan dikiri dan cerah dikanan)

#### 4.3.3 Implementasi Time Module

Time Module berfungsi sebagai instrumen persepsi waktu yang memungkinkan agen untuk memahami siklus diurnal (siang) dan nokturnal (malam). Pemahaman terhadap dimensi waktu sangat penting dalam manajemen risiko, mengingat distribusi entitas berbahaya (Monster) dalam dunia Minecraft di pengaruhi pada variabel waktu.

Asal data temporal bersumber dari variabel *global time* yang disimpan di dalam *LevelProperties*. Melalui Fabric API, agen melakukan pemanggilan metode *world.getTimeOfDay()* yang menghasilkan data bertipe long. Nilai ini merepresentasikan jumlah ticks yang telah berlalu sejak dunia diciptakan (dalam siklus 0 hingga 24.000 ticks per hari). Data ini bersifat absolut dan akurat karena sinkron dengan detak jantung mesin simulasi Minecraft.

Untuk mentransformasi data mentah ini menjadi informasi kognitif, modul menerapkan logika heuristik berdasarkan ambang batas waktu (*time threshold*): rentang 13.000 hingga 23.000 ticks didefinisikan secara semantik sebagai fase "Malam Hari", sementara rentang lainnya dikategorikan sebagai "Siang Hari". Hasil kategorisasi ini diintegrasikan ke dalam basis pengetahuan agen, memungkinkannya untuk menyesuaikan gaya komunikasi serta meningkatkan sensitivitas sensor terhadap ancaman monster.



**Gambar 4.11** Item Jam pada Minecraft yang menunjukkan waktu dalam game

#### 4.3.4 Implementasi *Light Module*

*Light Module* merupakan sensor yang bertugas melakukan analisis terhadap tingkat pencahayaan pada titik koordinat agen. Dalam ekosistem Minecraft, variabel cahaya adalah determinan utama dalam algoritma pemunculan entitas musuh (*mob spawning*) dan dalam mendeteksi apakah pemain sedang berada di dalam gua atau di permukaan tanah. Oleh karena itu, modul ini bertindak sebagai alat evaluasi keamanan area yang sangat presisi.

Informasi iluminasi bersumber dari *Lighting Engine Minecraft* yang dikelola oleh kelas `net.minecraft.world.chunk.light.LightingProvider`. Sistem menggunakan metode `getLightLevel(pos)` untuk mengambil nilai cahaya pada koordinat agen. Modul ini menerapkan analisis diferensial dengan memisahkan dua sumber pencahayaan utama melalui interface *LightType*: *Block Light* (cahaya dari sumber buatan) dan *Sky Light* (cahaya alami dari langit).

Intensitas cahaya yang diterima (skala integer 0-15) diolah menjadi skor keamanan dalam *Context Builder*. Jika total cahaya berada di bawah ambang batas (*threshold*) 7, sistem secara otomatis menambahkan label peringatan "Sangat Gelap". Hal ini memberikan instruksi implisit kepada LLM bahwa agen sedang berada dalam zona risiko tinggi.



**Gambar 4.12** Sumber Cahaya Buatan dan Sumber Cahaya Alami pada Minecraft

#### 4.3.5 Implementasi Biome Module

Biome Module memberikan dimensi pemahaman geografis dan ekologis kepada agen cerdas. Dengan mengetahui karakteristik bioma tempatnya berada, agen dapat melakukan perencanaan aksi yang lebih strategis, terutama dalam aktivitas pengumpulan sumber daya.

Data geografis ini diambil dari sistem *Biome Accessor* melalui metode *world.getBiome*. Sumber datanya berasal dari *Noise-based Generation* yang dilakukan oleh mesin permainan saat *chunk* dimuat. Melalui *Fabric API*, modul mengakses objek bioma yang terdaftar dalam *Registry<Biome>*. Dari objek ini, sistem dapat mengekstraksi properti seperti temperatur, presipitasi, dan nama bioma tersebut.

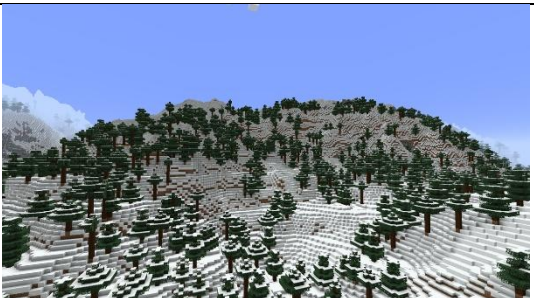
Sistem melakukan proses pemetaan identitas (*Registry Mapping*) untuk mendapatkan Identifier unik dari bioma (misalnya *minecraft:cherry\_grove*). Karena LLM lebih efektif memproses bahasa natural, modul ini melakukan transformasi teks menjadi istilah ("*cherry\_grove*"). Informasi ini memungkinkan agen untuk melakukan klasifikasi sumber daya yang tersedia di sekitarnya berdasarkan karakteristik ekosistem bioma tersebut.

**Tabel 4.3** Nama dan Karakteristik Bioma pada Minecraft

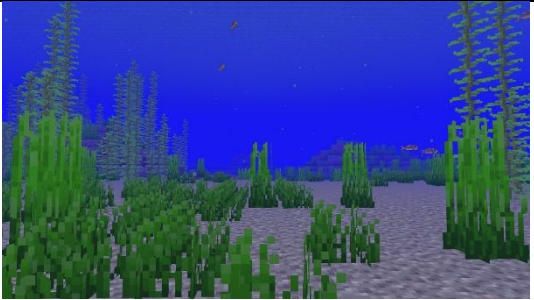
| <b>Nama Bioma</b> | <b>Karakteristik Bioma</b>                                                                                                                                                    | <b>Gambar Bioma</b>                                                                  |
|-------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------|
| Plains            | Ekosistem dataran rendah dengan topografi relatif homogen, vegetasi didominasi oleh rumput pendek, serta tingkat visibilitas tinggi yang mendukung pembangunan dan mobilitas. |    |
| Forest            | Bioma dengan kepadatan vegetasi arboreal sedang hingga tinggi, didominasi oleh pohon daun lebar, berfungsi sebagai sumber daya kayu utama.                                    |  |
| Dark Forest       | Hutan dengan kanopi tertutup rapat yang mengurangi penetrasi cahaya, menciptakan lingkungan dengan tingkat <i>spawn</i> entitas berbahaya yang tinggi.                        |  |

| <b>Nama Bioma</b> | <b>Karakteristik Bioma</b>                                                                                      | <b>Gambar Bioma</b>                                                                  |
|-------------------|-----------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------|
| Flower Forest     | Sub-bioma hutan dengan keanekaragaman flora tinggi, khususnya bunga, mencerminkan variasi biodiversitas visual. |    |
| Birch Forest      | Hutan homogen dengan dominasi spesies pohon birch, menunjukkan struktur vegetasi yang seragam.                  |   |
| Taiga             | Bioma beriklim dingin dengan vegetasi konifer (spruce), sering diasosiasikan dengan fauna seperti serigala.     |  |
| Snowy Taiga       | Variasi taiga dengan tutupan salju permanen, mencerminkan kondisi suhu rendah ekstrem.                          |  |

| <b>Nama Bioma</b> | <b>Karakteristik Bioma</b>                                                                                                                   | <b>Gambar Bioma</b>                                                                  |
|-------------------|----------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------|
| Jungle            | Bioma tropis dengan densitas vegetasi sangat tinggi, struktur kanopi berlapis, serta keberadaan liana (vine) dan biodiversitas fauna tinggi. |    |
| Bamboo Jungle     | Sub-bioma jungle dengan dominasi bambu, menciptakan struktur vegetasi vertikal yang unik.                                                    |   |
| Desert            | Bioma arid dengan curah hujan minimal, vegetasi sangat terbatas, serta dominasi substrat pasir.                                              |  |
| Savanna           | Bioma semi-arid dengan vegetasi rumput tinggi dan pohon acacia tersebar, mencerminkan                                                        |  |

| <b>Nama Bioma</b> | <b>Karakteristik Bioma</b>                                                                                                       | <b>Gambar Bioma</b>                                                                  |
|-------------------|----------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------|
|                   | transisi antara hutan dan padang rumput.                                                                                         |                                                                                      |
| Windswept Hills   | Topografi ekstrem dengan elevasi tinggi dan kemiringan curam, menciptakan kondisi lingkungan yang tidak stabil untuk konstruksi. |    |
| Meadow            | Padang rumput pegunungan dengan vegetasi bunga, mencerminkan ekosistem transisi antara dataran dan pegunungan.                   |   |
| Grove             | Bioma subalpine dengan kombinasi salju dan vegetasi konifer, mencerminkan kondisi iklim dingin sedang.                           |  |
| Snowy Slopes      | Lereng pegunungan dengan tutupan salju, menunjukkan gradien elevasi tinggi dan suhu rendah.                                      |  |

| <b>Nama Bioma</b> | <b>Karakteristik Bioma</b>                                                                             | <b>Gambar Bioma</b>                                                                  |
|-------------------|--------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------|
| Frozen Peaks      | Puncak gunung dengan dominasi es dan salju permanen, mencerminkan kondisi ekstrem.                     |    |
| Jagged Peaks      | Formasi pegunungan dengan struktur tajam dan tidak beraturan, menunjukkan morfologi geologis kompleks. |    |
| Stony Peaks       | Puncak gunung berbatu tanpa vegetasi signifikan, menandakan kondisi tanah yang tidak subur.            |   |
| Beach             | Zona transisi antara daratan dan perairan dengan substrat pasir atau kerikil.                          |  |
| River             | Sistem perairan mengalir yang menghubungkan berbagai bioma daratan.                                    |  |

| <b>Nama Bioma</b> | <b>Karakteristik Bioma</b>                                                                                              | <b>Gambar Bioma</b>                                                                  |
|-------------------|-------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------|
| Ocean             | Ekosistem perairan luas dengan variasi suhu dan kedalaman yang mempengaruhi biodiversitas.                              |    |
| Swamp             | Bioma lahan basah dengan air stagnan, vegetasi rendah, dan kondisi anaerobik parsial.                                   |    |
| Mangrove Swamp    | Ekosistem pesisir berlumpur dengan vegetasi mangrove, menunjukkan adaptasi terhadap kondisi salinitas dan genangan air. |   |
| Mushroom Fields   | Bioma langka dengan dominasi fungi, memiliki sifat unik berupa ketiadaan spawn entitas hostile secara alami.            |  |
| Lush Caves        | Ekosistem bawah tanah dengan vegetasi hijau, menunjukkan keberadaan kelembapan tinggi                                   |  |

| Nama Bioma      | Karakteristik Bioma                                                                                            | Gambar Bioma                                                                        |
|-----------------|----------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|
|                 | dan pencahayaan bioluminesen.                                                                                  |                                                                                     |
| Dripstone Caves | Sistem gua dengan formasi speleothem (stalaktit dan stalagmit), mencerminkan proses geologis deposisi mineral. |   |
| Deep Dark       | Bioma subterranean dengan tingkat cahaya sangat rendah, serta keberadaan entitas sensorik berbasis getaran.    |  |

#### 4.3.6 Implementasi Entity Module

*Entity Module* merupakan komponen persepsi kritis yang berfungsi sebagai "radar sosial" bagi agen cerdas. Kemampuan agen untuk mengenali, mengklasifikasi, dan memantau pergerakan entitas di sekitarnya adalah fondasi utama dari perilaku otonom, sistem keamanan pemain, serta interaksi sosial yang responsif. Modul ini memungkinkan agen untuk beralih dari sekadar entitas statis menjadi agen yang sadar akan keberadaan makhluk hidup lain di dalam dunia simulasi.

Data entitas bersumber dari sistem manajemen entitas pada game engine Minecraft yang dikelola secara dinamis di dalam memori server. Secara teknis, sistem melakukan ekstraksi data melalui objek *ServerWorld* menggunakan metode kueri spasial *getEntitiesByClass()*. Modul ini mendefinisikan sebuah AABB (*Axis-Aligned Bounding Box*), yaitu sebuah area pencarian berbentuk kubus imajiner dengan radius 16 blok yang berpusat pada koordinat agen.

Melalui API ini, agen mendapatkan akses ke seluruh instance objek *net.minecraft.entity.Entity* yang berada di dalam jangkauan tersebut. Setiap entitas yang terjaring kemudian dievaluasi atributnya secara real-time, meliputi identitas unik (UUID), tipe entitas (*EntityType*), koordinat posisi saat ini, serta status kesehatan (*health status*).

Data mentah yang diperoleh dari mesin simulasi seringkali bersifat heterogen dan berjumlah banyak. Untuk mencegah beban kognitif berlebih pada LLM (*data overload*), modul ini menerapkan Algoritma Filtrasi dan Kategorisasi. Proses transformasi dilakukan melalui tahapan berikut:

1. Penghitungan Vektor Jarak: Sistem menghitung jarak absolut antara agen dan entitas menggunakan metode *Euclidean Distance* dalam bentuk kuadrat (*squaredDistanceTo*). Metode ini menghitung selisih koordinat pada sumbu  $x$ ,  $y$ , dan  $z$ , kemudian menjumlahkan kuadrat dari masing-masing selisih tersebut. Secara matematis, rumus yang digunakan adalah:

$$d^2 = (x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2$$

dengan:

- $X_1, y_1, z_1$  = posisi agen/NPC
- $x_2, y_2, z_2$  = posisi target atau entitas
- $d^2$  = jarak kuadrat (squared distance)

Pendekatan ini dipilih karena lebih efisien dibandingkan menghitung jarak Euclidean penuh yang memerlukan operasi akar kuadrat ( $\sqrt{\phantom{x}}$ ). Dalam konteks *game loop* Minecraft yang melakukan ribuan pengecekan posisi setiap *tick*, penggunaan *squared distance* dapat mengurangi beban komputasi tanpa memengaruhi hasil perbandingan jarak antar entitas.

Sebagai contoh, apabila NPC berada pada koordinat (10, 64, 20) dan target berada pada koordinat (13, 64, 24), maka sistem akan menghitung:

$$dx = 13 - 10 = 3$$

$$dy = 64 - 64 = 0$$

$$dz = 24 - 20 = 4$$

Kemudian nilai tersebut dimasukkan ke dalam rumus:

$$d^2 = 3^2 + 0^2 + 4^2$$

$$d^2 = 9 + 0 + 16$$

$$d^2 = 25$$

Hasil akhir menunjukkan bahwa nilai *squared distance* antara NPC dan target adalah 25. Nilai ini merepresentasikan jarak kuadrat, sedangkan jarak Euclidean sebenarnya adalah:

$$d = \sqrt{25} = 5$$

Dalam implementasinya, sistem cukup membandingkan nilai 25 tersebut dengan kandidat target lain tanpa perlu menghitung akar kuadrat, sehingga proses pencarian target menjadi lebih cepat dan efisien.

2. Taksonomi Entitas: Entitas diklasifikasikan ke dalam kategori fungsional, yaitu Hostile (monster berbahaya seperti Zombie atau Creeper), Passive (hewan ternak), dan Player (pemain/pemilik).
3. Abstraksi Semantik: Informasi teknis dikonversi menjadi narasi kontekstual. Sebagai contoh, jika terdeteksi entitas bertipe *EntityType.ZOMBIE* dengan jarak kurang dari 5 blok, modul tidak hanya mengirimkan koordinat, tetapi menambahkan label peringatan "Berbahaya!".

Informasi ini memberikan kemampuan kepada LLM untuk melakukan Penalaran Taktis. Misalnya, jika agen mendeteksi musuh di dekat pemilik yang sedang memiliki kesehatan rendah, agen dapat secara otonom memutuskan untuk menghentikan aktivitas pengumpulan sumber daya dan beralih ke aksi perlindungan (*ATTACK* atau *GUARD*). Dengan demikian, *Entity Module* menjembatani kesenjangan antara data posisi mentah dengan pemahaman situasi sosial yang kompleks.



blok pengisi massal seperti udara (*air*), tanah (*dirt*), atau batu biasa (*stone*) kecuali diminta secara spesifik.

Dalam proses pencarian target, sistem juga menerapkan perhitungan kedekatan spasial berbasis *squared distance* sebagaimana dijelaskan pada *Entity Module*. Pendekatan ini digunakan untuk menentukan prioritas blok berdasarkan kedekatan posisi terhadap agen tanpa perlu menghitung akar kuadrat, sehingga proses pencarian dapat dilakukan secara lebih efisien pada *game loop* Minecraft.

Data mentah berupa objek blok dari *registry* kemudian ditransformasikan menjadi informasi semantik melalui dua tahapan utama:

1. *Registry Mapping*: Mengubah ID teknis (misalnya *minecraft:iron\_ore*) menjadi label yang manusiawi (misalnya "iron\_ore") agar dapat dipahami oleh LLM sebagai objek tujuan.
2. *Spatial Indexing*: Setiap blok penting yang ditemukan akan dicatat koordinatnya ke dalam *Spatial Memory*. Struktur ini memungkinkan agen mengingat lokasi sumber daya meskipun objek tersebut sudah berada di luar *Field of View* aktif. *Spatial Memory* diimplementasikan menggunakan struktur berbasis pemetaan koordinat (*BlockPos*) sehingga pencarian lokasi sumber daya dapat dilakukan tanpa harus melakukan pemindaian ulang dunia secara penuh.

Secara komputasional, proses iterasi spasial memiliki kompleksitas mendekati:

$$O(r^3)$$

Hal ini disebabkan karena sistem melakukan pemindaian koordinat pada tiga sumbu ruang sekaligus, yaitu  $x$ ,  $y$ , dan  $z$ . Dengan radius pencarian sebesar  $r$ , jumlah koordinat yang perlu diperiksa meningkat secara kubik mengikuti pola:

$$r \times r \times r = r^3$$

Dalam implementasi sebenarnya, sistem melakukan iterasi dari  $-r$  hingga  $+r$  pada setiap sumbu, sehingga total volume pencarian dapat direpresentasikan sebagai:

$$(2r + 1)^3$$

Sebagai contoh, apabila radius pencarian ditetapkan sebesar:

$$r = 12$$

maka jumlah koordinat blok yang harus diperiksa adalah:

$$(2 \times 12 + 1)^3 = 25^3 = 15625$$

Artinya, sistem perlu memeriksa sekitar 15.625 blok dalam satu proses pemindaian.

Jika radius pencarian diperbesar menjadi:

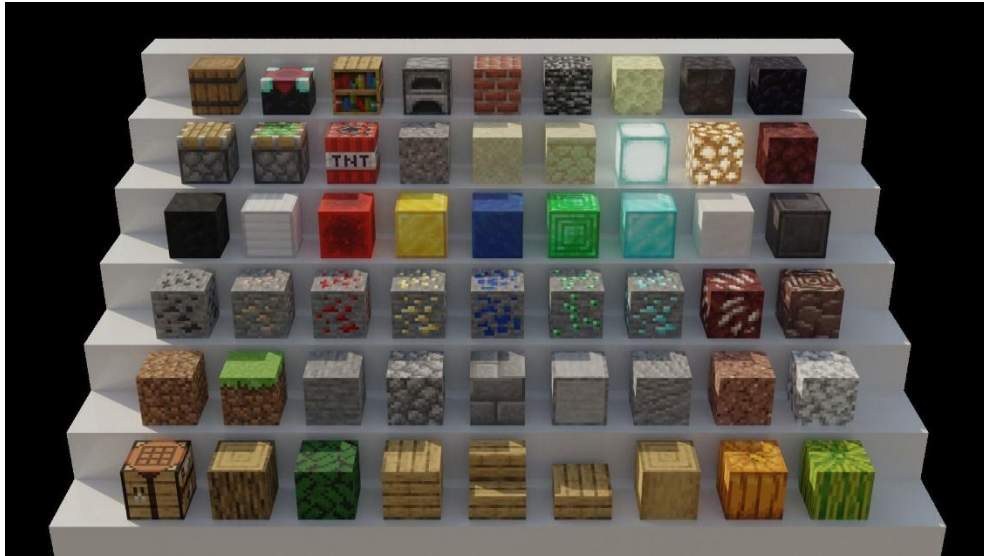
$$r = 24$$

maka jumlah blok yang diperiksa meningkat menjadi:

$$(2 \times 24 + 1)^3 = 49^3 = 117649$$

atau lebih dari 117 ribu blok untuk satu NPC. Oleh sebab itu, sistem menerapkan pembatasan radius pencarian dan mekanisme filtrasi material untuk menjaga efisiensi komputasi serta mencegah penurunan performa NPC pada lingkungan simulasi berskala besar seperti Minecraft.

Informasi spasial yang dihasilkan oleh *Block Module* selanjutnya digunakan oleh sistem pengambilan keputusan dan navigasi. Sebagai contoh, ketika pemain meminta agen untuk mencari kayu sementara tidak terdapat pohon dalam area pengamatan aktif, LLM dapat memanfaatkan data lokasi pohon yang tersimpan di *Spatial Memory* untuk menghasilkan instruksi navigasi tingkat tinggi seperti GOTO. Selanjutnya, proses pencarian jalur aktual dieksekusi oleh sistem navigasi Minecraft berbasis algoritma *pathfinding* A\*. Mekanisme ini menunjukkan adanya integrasi antara persepsi visual instan, memori spasial jangka panjang, dan sistem navigasi cerdas dalam arsitektur agen NPC berbasis AI.



**Gambar 4.14** Beberapa Block pada Minecraft

#### 4.3.8 Implementasi Status Module

Status Module merupakan komponen persepsi yang bertanggung jawab atas pemantauan kondisi fisik dan variabel internal agen cerdas. Dalam disiplin robotika dan kecerdasan buatan, fungsi ini dikenal sebagai *Proprioception*, yaitu kemampuan sistem untuk memahami status dirinya sendiri di dalam ruang simulasi. Modul ini sangat krusial karena memberikan dasar bagi agen untuk mengambil keputusan yang bersifat defensif atau memprioritaskan pemulihan diri (*self-preservation*) sebelum menjalankan instruksi dari pemain.

Data status internal bersumber langsung dari kelas entitas agen itu sendiri (*CompanionEntity*), yang merupakan turunan dari kelas *net.minecraft.entity.LivingEntity*. Mesin simulasi Minecraft secara otomatis mengelola variabel-variabel ini di dalam memori server dan mensinkronisasikannya menggunakan sistem *DataTracker*. Secara teknis, modul mengekstraksi informasi melalui beberapa metode internal API:

1. Kesehatan (*Health*): Diambil melalui metode *getHealth()* dan *getMaxHealth()* untuk mendapatkan nilai float presisi mengenai kondisi fisik agen.
2. Efek Status (*Status Effects*): Mendeteksi apakah agen sedang dalam kondisi terbakar (*isOnFire()*).

3. Kondisi Fisik Spasial: Mengetahui apakah agen sedang terjatuh, berada di dalam air, atau sedang menyentuh tanah (*onGround*) untuk menentukan konteks pergerakan.

Data mentah berupa angka numerik dan status boolean ditransformasikan menjadi informasi semantik melalui proses Evaluasi Ambang Batas (*Thresholding*):

1. Kategorisasi Kesehatan: Nilai numerik HP dikonversi menjadi kategori tekstual. Misalnya, jika kesehatan berada di bawah 25% dari total maksimal, sistem menetapkan status "Sekarat/Luka Parah". Jika di atas 80%, ditetapkan sebagai "Sehat".
2. Peringatan Kritis: Status boolean seperti *isOnFire* dikonversi menjadi string peringatan keras (*High-priority Alert*) di dalam konteks, misalnya: "PERINGATAN: Tubuhku sedang terbakar!".

Informasi ini memberikan kemampuan kepada LLM untuk melakukan Penalaran Berbasis Kebutuhan (*Needs-based Reasoning*). Sebagai contoh, jika Status Module melaporkan kesehatan yang rendah, LLM dapat memutuskan secara otonom untuk menolak perintah berbahaya dari pemain atau meminta bantuan medis/makanan. Integrasi ini memastikan bahwa agen tidak hanya bertindak sebagai pelaksana perintah pasif, tetapi juga sebagai entitas otonom yang peduli terhadap kelangsungan hidupnya sendiri di dalam dunia simulasi.

#### 4.3.9 Implementasi Inventory Module

Inventory Module merupakan komponen persepsi yang bertanggung jawab untuk memberikan "kesadaran material" kepada agen cerdas. Dalam ekosistem Minecraft yang berfokus pada pengumpulan dan pemrosesan sumber daya, kemampuan agen untuk mengetahui aset yang dimilikinya secara presisi adalah faktor penentu dalam keberhasilan perencanaan tugas. Modul ini memungkinkan agen untuk melakukan audit internal terhadap ketersediaan alat (tools), bahan makanan, maupun material yang telah berhasil dikumpulkan.

Data inventaris bersumber dari kelas *net.minecraft.inventory.SimpleInventory* yang diimplementasikan di dalam struktur entitas agen. Secara teknis, setiap kali sistem

persepsi melakukan pemindaian, modul akan melakukan iterasi melalui seluruh slot penyimpanan (27 slot). Ekstraksi data dilakukan menggunakan metode API berikut:

1. Iterasi Slot: Menggunakan loop untuk memeriksa setiap indeks melalui metode *getStack(i)*.
2. Identifikasi Item: Mengambil objek *ItemStack* untuk mengetahui jenis item melalui *itemStack.getItem()* dan jumlahnya melalui *itemStack.getCount()*.
3. Audit Tangan Utama (Main Hand): Menggunakan *getMainHandStack()* untuk menentukan alat atau senjata apa yang sedang aktif digunakan oleh agen. Hal ini krusial untuk menentukan efisiensi aksi (misalnya, agen tahu ia sedang memegang kapak saat diminta menebang pohon).

Mengirimkan data mentah dari setiap slot inventaris ke LLM akan menyebabkan pemborosan token dan redundansi informasi (data cluttering). Oleh karena itu, modul ini melakukan proses Agregasi Semantik:

1. Konsolidasi Item: Jika terdapat jenis item yang sama di beberapa slot berbeda (misalnya oak logs di slot 1 dan slot 5), modul akan menjumlahkan totalnya menjadi satu entitas data yang utuh.
2. Filter Deskriptif: ID teknis item (misalnya *minecraft:iron\_sword*) dikonversi menjadi nama yang simple (misalnya "Iron Sword").
3. Kalkulasi Kapasitas: Modul secara otomatis menghitung jumlah slot kosong yang tersisa menggunakan algoritma penghitungan slot yang bernilai *isAir()*, memberikan informasi kepada agen mengenai kapasitas angkut yang masih tersedia.

Integrasi ini memberikan kemampuan kepada LLM untuk melakukan Penalaran Berbasis Kemampuan (*Capability-based Reasoning*). Sebagai contoh, jika agen diminta untuk menambang emas namun Inventory Module melaporkan tidak adanya *Pickaxe* yang bagus di dalam tas, LLM dapat memberikan respon penolakan yang logis atau meminta alat yang lebih bagus tersebut kepada pemain. Selain itu, data ini memungkinkan agen untuk memberikan laporan kemajuan tugas secara akurat, seperti: "Aku sudah mengumpulkan 10 kayu sesuai permintaanmu, NomouthGun".



Untuk meminimalisir kesalahan deteksi (*false positive*) yang disebabkan oleh pergerakan wajah sesaat atau gangguan pencahayaan pada kamera, modul ini menerapkan Algoritma Konfirmasi Beruntun (*Consecutive Confirmation*):

1. *Threshold* Pengecekan: Sebuah emosi tidak langsung diterima oleh agen pada pengecekan pertama.
2. Validasi Berulang: Data emosi baru dianggap valid dan dikirimkan ke server hanya jika terdeteksi sebanyak tiga kali berturut-turut (akumulasi waktu 15 detik).
3. State Management: Setelah tervalidasi, emosi tersebut disimpan ke dalam Data Tracker pada kelas *CompanionEntity* sehingga dapat diakses oleh seluruh sistem kognitif agen.

Integrasi ini memungkinkan LLM untuk melakukan Penalaran Afektif (*Affective Reasoning*). Sebagai contoh, jika pemain terdeteksi dalam kondisi "Sedih" (*Sad*), agen akan secara otomatis mengubah tone bicaranya menjadi lebih suportif dan mengurangi perilaku yang bersifat mengganggu. Hal ini menciptakan lingkaran interaksi manusia dan komputer (*Human-Computer Interaction*) yang lebih alami, di mana agen cerdas mampu memberikan respon yang tidak hanya logis secara tugas, tetapi juga tepat secara emosional.

#### 4.3.11 Implementasi Health Event Module

*Health Event Module* merupakan komponen sensor khusus yang berfungsi sebagai sistem pemantauan kondisi darurat (*Emergency Monitoring System*). Berbeda dengan Status Module yang hanya melaporkan angka kesehatan secara rutin, Health Event Module bekerja berdasarkan pemantauan peristiwa (*Event-based Monitoring*). Modul ini dirancang untuk mendeteksi perubahan kondisi fisik yang drastis, baik pada diri agen sendiri maupun pada pemilik (pemain), yang memerlukan respon instan dan prioritas tinggi di atas tugas-tugas lainnya. Data untuk modul ini diperoleh melalui kueri *real-time* terhadap atribut kesehatan dari dua subjek utama di dalam *ServerWorld*:

1. Kesehatan Agen: Diekstraksi melalui metode *npc.getHealth()* dan dibandingkan dengan *npc.getMaxHealth()*.

2. Kesehatan Pemilik (Owner): Modul ini melakukan akses melalui interface *TameableEntity* untuk mendapatkan referensi objek *PlayerEntity* (pemain), kemudian memantau variabel kesehatan pemain menggunakan metode *owner.getHealth()*.

Data ini diambil secara kontinu pada setiap siklus pemindaian sensor untuk memastikan tidak ada keterlambatan dalam mendeteksi kondisi kritis. Modul ini menerapkan algoritma Evaluasi Ambang Batas Kritis (*Critical Threshold Evaluation*) untuk mengubah data numerik menjadi sinyal bahaya yang dapat dipahami secara semantik oleh LLM:

1. Deteksi Low Health: Jika persentase kesehatan berada di bawah ambang batas kritis (misalnya  $< 20\%$  atau 4 poin HP), modul akan menghasilkan variabel *boolean isLowHealth* bernilai *true*.
2. Injeksi Peringatan Darurat: Saat kondisi kritis terdeteksi, modul tidak hanya mengirimkan angka, tetapi menyisipkan baris teks peringatan dengan prioritas tinggi (*High-priority Alert*) ke dalam konteks, seperti: "DARURAT: Temanku sedang sekarat dan butuh perlindungan!" atau "Kondisiku kritis, aku harus segera mundur!".

Integrasi ini memberikan kemampuan Perencanaan Berbasis Urgensi (*Urgency-based Planning*) pada agen cerdas. Dalam skenario di mana pemain sedang dalam bahaya besar (darurat kesehatan), agen cerdas dapat secara otonom menghentikan tugas pengumpulan sumber daya dan beralih ke mode *GUARD* atau *ATTACK* untuk melindungi pemain tanpa perlu diperintah. Hal ini membuktikan bahwa agen memiliki insting pertahanan hidup dan loyalitas yang terprogram secara cerdas melalui mekanisme pemantauan peristiwa kesehatan.

#### 4.4 Analisis dan Implementasi Model Computer Vision

Selain memiliki kesadaran terhadap objek fisik di dalam dunia simulasi, agen cerdas dirancang untuk memiliki kemampuan memahami kondisi psikologis pemain melalui ekspresi wajah secara *near real-time*. Implementasi ini merupakan penerapan dari bidang Komputasi Afektif (*Affective Computing*) dengan metode *image classification*, yang bertujuan untuk menjembatani kesenjangan emosional antara manusia dan agen cerdas melalui pengolahan citra digital. Kemampuan ini

menjadi aspek krusial dalam menciptakan interaksi manusia dan komputer (*Human-Computer Interaction*) yang lebih alami dan empatik.

Berbeda dengan modul sensor lainnya yang beroperasi di dalam *engine Minecraft*, sistem analisis visual ini diimplementasikan menggunakan arsitektur Sistem Terdistribusi. Proses ekstraksi fitur wajah dan klasifikasi emosi dilakukan pada lapisan eksternal menggunakan bahasa pemrograman Python, yang kemudian diintegrasikan kembali ke dalam logika Java melalui protokol komunikasi *HTTP*.

#### 4.4.1 Dataset Emosi

Proses pelatihan dan pengujian model *Computer Vision* dalam penelitian ini menggunakan *Real-world Affective Faces Database* (RAF-DB). Dataset tersebut merupakan dataset skala besar yang berisi citra ekspresi wajah manusia dengan berbagai variasi kondisi di dunia nyata. Total data yang digunakan dari dataset ini adalah sebanyak 11.368 citra yang difokuskan secara spesifik pada empat kategori emosi utama, yaitu *Anger*, *Happy*, *Neutral*, dan *Sad*. Setiap citra memiliki resolusi seragam sebesar 100x100 piksel. Untuk mengevaluasi performa model secara komprehensif, seluruh data tersebut dibagi ke dalam dua bagian utama, yaitu data train dan data test. Proses pembagian data dilakukan melalui tiga skenario *data split*, yaitu skenario 70:30, 80:20, serta 90:10. Dan berikut ini adalah detail total data dari masing masing scenario:

**Tabel 4.4** Distribusi Dataset RAF-DB 70:30

| Dataset                | Train        | Test         | Total Citra Per Kelas |
|------------------------|--------------|--------------|-----------------------|
| Anger                  | 2021 Gambar  | 867 Gambar   | 2.888 Gambar          |
| Happy                  | 2034 Gambar  | 873 Gambar   | 2.907 Gambar          |
| Neutral                | 2050 Gambar  | 879 Gambar   | 2.929 Gambar          |
| Sad                    | 1850 Gambar  | 794 Gambar   | 2.644 Gambar          |
| <b>Total Akumulasi</b> | 7.955 Gambar | 3.413 Gambar | 11.368 Gambar         |

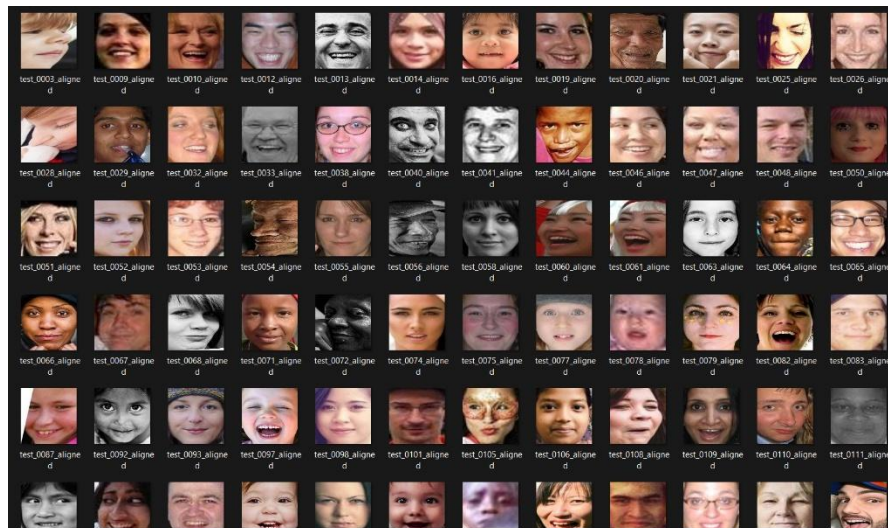
**Tabel 4.5** Distribusi Dataset RAF-DB 80:20

| Dataset | Train       | Test       | Total Citra Per Kelas |
|---------|-------------|------------|-----------------------|
| Anger   | 2310 Gambar | 578 Gambar | 2.888 Gambar          |
| Happy   | 2325 Gambar | 582 Gambar | 2.907 Gambar          |

|                        |              |              |               |
|------------------------|--------------|--------------|---------------|
| Neutral                | 2343 Gambar  | 586 Gambar   | 2.929 Gambar  |
| Sad                    | 2115 Gambar  | 529 Gambar   | 2.644 Gambar  |
| <b>Total Akumulasi</b> | 9.093 Gambar | 2.275 Gambar | 11.368 Gambar |

Tabel 4.6 Distribusi Dataset RAF-DB 90:10

| Dataset                | Train         | Test         | Total Citra Per Kelas |
|------------------------|---------------|--------------|-----------------------|
| Anger                  | 2599 Gambar   | 289 Gambar   | 2.888 Gambar          |
| Happy                  | 2616 Gambar   | 291 Gambar   | 2.907 Gambar          |
| Neutral                | 2636 Gambar   | 293 Gambar   | 2.929 Gambar          |
| Sad                    | 2379 Gambar   | 265 Gambar   | 2.644 Gambar          |
| <b>Total Akumulasi</b> | 10.230 Gambar | 1.138 Gambar | 11.368 Gambar         |



Gambar 4.16 Sample Gambar pada Real-world Affective Faces Database

#### 4.4.2 Preprocessing

Pada tahap *preprocessing*, dilakukan pemisahan (*data splitting*) terhadap dataset citra wajah hasil *cropping* ke dalam dua subset utama, yaitu data train dan data test. Proses pemisahan ini dibagi ke dalam tiga skenario rasio eksperimental, yaitu 70:30, 80:20, dan 90:10. Sebagai gambaran, pada skenario pemisahan 80:20, proporsi sebesar 80% dari total data dialokasikan untuk melatih model (*training*), sementara 20% sisanya digunakan secara independen untuk mengevaluasi performa model (*testing*).

Proses pembagian dilakukan secara acak (*random shuffle*) dengan menggunakan nilai seed sebesar 42 untuk memastikan hasil pembagian data bersifat *reproducible*.

atau dapat direproduksi kembali pada penelitian selanjutnya. Dataset dikelompokkan berdasarkan kelas emosi yang tersedia, kemudian setiap kelas dibagi secara proporsional sehingga distribusi data antar kelas tetap terjaga.

Hasil dari setiap pemisahan data disimpan ke dalam struktur direktori terpisah yang terdiri atas folder *train* dan *test*. Di dalam masing-masing folder tersebut, terdapat subfolder yang dikelompokkan secara spesifik berdasarkan kategori emosi yang diteliti. Melalui tata kelola direktori ini, proses pelatihan model menggunakan subset *train* dan proses evaluasi menggunakan subset *test* dapat berjalan secara optimal tanpa memicu terjadinya tumpang tindih data (*data leakage*) antar-kedua subset tersebut.

#### 4.4.3 Augmentasi Dataset Emosi

Untuk meningkatkan kemampuan generalisasi model dan mencegah terjadinya *overfitting* akibat keterbatasan jumlah variasi data, dilakukan proses augmentasi data pada dataset latih. Teknik ini bertujuan untuk menciptakan variasi citra baru secara sintesis dari data yang sudah ada tanpa mengubah label emosi aslinya. Dengan cara ini, model dapat mempelajari fitur wajah dalam berbagai kondisi visual yang berbeda-beda. Seperti pada gambar 4.17 implementasi augmentasi data dalam penelitian ini dilakukan dengan menerapkan beberapa teknik transformasi citra secara acak, antara lain:

1. Rescale: Melakukan normalisasi nilai piksel dengan mengalikan setiap piksel dengan faktor  $1/255$  untuk mengubah rentang nilai menjadi 0 hingga 1. Hal ini krusial untuk mempercepat proses konvergensi saat pelatihan model.
2. Rotation Range: Memutar gambar secara acak hingga 20 derajat untuk mensimulasikan variasi kemiringan kepala pemain di depan kamera.
3. Width Shift Range dan Height Shift Range: Melakukan pergeseran gambar secara horizontal dan vertikal sebesar 10%. Teknik ini membantu model mengenali wajah meskipun posisinya tidak tepat berada di tengah bingkai kamera.
4. Zoom Range: Menerapkan perbesaran atau perkecilan gambar secara acak hingga 20% untuk mengantisipasi perbedaan jarak antara pemain dengan sensor *webcam*.

5. Horizontal Flip: Membalik citra secara horizontal untuk menangani variasi arah hadap wajah pemain.
6. Brightness Range: Mengatur rentang kecerahan citra antara 0.8 hingga 1.2. Teknik ini memastikan model tetap tangguh terhadap perubahan kondisi pencahayaan di ruangan pemain.
7. Fill Mode: Menggunakan metode *nearest* untuk mengisi area piksel yang kosong akibat proses transformasi geometris.

Melalui penerapan teknik-teknik tersebut, dataset latih yang berjumlah 9.983 citra akan memiliki variasi visual yang jauh lebih kaya pada setiap iterasi pelatihan (*epoch*). Hasil akhirnya adalah model yang lebih *robust* dan memiliki performa yang stabil saat digunakan untuk mendeteksi emosi secara *near real-time*.

```

1 from tensorflow.keras.preprocessing.image import ImageDataGenerator
2
3 augment_datagen = ImageDataGenerator(
4     rescale=1./255,
5     rotation_range=20,
6     width_shift_range=0.1,
7     height_shift_range=0.1,
8     zoom_range=0.2,
9     horizontal_flip=True,
10    brightness_range=[0.8, 1.2],
11    fill_mode='nearest'
12 )
13
14 augmented_train_ds = augment_datagen.flow_from_directory(
15     train_path,
16     target_size=(IMG_SIZE, IMG_SIZE),
17     batch_size=BATCH_SIZE,
18     class_mode='sparse',
19     shuffle=True
20 )

```

**Kode Program 4.1** Kode untuk augmentasi dataset

#### 4.4.4 Arsitektur Model *Convolutional Neural Network* (CNN)

Model pengenalan emosi yang digunakan dalam penelitian ini dibangun menggunakan arsitektur *Convolutional Neural Network* (CNN) dengan model *Sequential* dari pustaka Keras. Arsitektur ini dirancang untuk melakukan ekstraksi fitur secara otomatis dari citra ekspresi wajah pemain dan mengklasifikasikannya ke dalam empat kelas emosi. Struktur model terdiri dari dua bagian utama, yaitu *Feature Extraction* (lapisan konvolusi) dan *Classification* (lapisan *fully connected*) dengan rincian sebagai berikut:

### 1. Tahap Ekstraksi Fitur (*Feature Extraction*)

Tahap ini terdiri dari tiga blok pasangan lapisan konvolusi dan *pooling* untuk menangkap pola visual dari wajah:

- a. Blok Pertama: Menggunakan lapisan *Conv2D* dengan 32 filter berukuran (3,3) dan fungsi aktivasi ReLU. Lapisan ini menerima input citra berukuran 100x100 piksel dengan 3 saluran warna (RGB). Hasil konvolusi kemudian direduksi dimensinya menggunakan *MaxPooling2D* berukuran (2,2).
- b. Blok Kedua: Terdiri dari 64 filter konvolusi berukuran (3,3) diikuti dengan *max pooling* (2,2) untuk menangkap fitur yang lebih kompleks.
- c. Blok Ketiga: Terdiri dari 128 filter konvolusi berukuran (3,3) diikuti dengan *max pooling* (2,2) untuk ekstraksi fitur tingkat tinggi.

### 2. Tahap Klasifikasi (*Classification*)

Setelah fitur diekstraksi, data diolah melalui lapisan berikut untuk menghasilkan prediksi emosi:

- a. Flatten: Lapisan ini berfungsi untuk mengubah matriks fitur tiga dimensi menjadi vektor satu dimensi (linear) agar dapat diproses oleh lapisan saraf padat (*dense layer*).
- b. Dense (Hidden Layer): Memiliki 128 neuron dengan aktivasi *ReLU* untuk mempelajari hubungan non-linear antar fitur.
- c. Dropout (0.5): Teknik regulasi yang mengabaikan 50% neuron secara acak selama pelatihan untuk mencegah *overfitting*.
- d. Output Layer: Menggunakan lapisan *Dense* dengan jumlah unit sesuai jumlah kelas emosi (4 kelas) dan fungsi aktivasi *Softmax* untuk menghasilkan probabilitas prediksi di setiap kelas.

### 3. Konfigurasi Pelatihan (*Compilation*)

Model dikonfigurasi dengan parameter berikut untuk mengoptimalkan proses pembelajaran:

- a. Optimizer: Menggunakan algoritma *Adam* karena efisiensinya dalam menangani pembaruan bobot secara adaptif.
- b. Loss Function: Menggunakan *Sparse Categorical Crossentropy* yang sesuai untuk klasifikasi multi-kelas dengan label integer.

- c. Metrics: Menggunakan *Accuracy* untuk memantau sejauh mana model berhasil mengklasifikasikan citra emosi dengan benar selama pelatihan.

Arsitektur ini menghasilkan total parameter sebesar 1.732.292 yang semuanya merupakan parameter terlatih (*trainable params*).

| Layer (type)                          | Output Shape        | Param #   |
|---------------------------------------|---------------------|-----------|
| conv2d_32 (Conv2D)                    | (None, 98, 98, 32)  | 896       |
| max_pooling2d_32 (MaxPooling2D)       | (None, 49, 49, 32)  | 0         |
| conv2d_33 (Conv2D)                    | (None, 47, 47, 64)  | 18,496    |
| max_pooling2d_33 (MaxPooling2D)       | (None, 23, 23, 64)  | 0         |
| conv2d_34 (Conv2D)                    | (None, 21, 21, 128) | 73,856    |
| max_pooling2d_34 (MaxPooling2D)       | (None, 10, 10, 128) | 0         |
| flatten_8 (Flatten)                   | (None, 12800)       | 0         |
| dense_21 (Dense)                      | (None, 128)         | 1,638,528 |
| dropout_11 (Dropout)                  | (None, 128)         | 0         |
| dense_22 (Dense)                      | (None, 4)           | 516       |
| Total params: 1,732,292 (6.61 MB)     |                     |           |
| Trainable params: 1,732,292 (6.61 MB) |                     |           |

**Gambar 4.17** Struktur Model *Convolutional Neural Network*

#### 4.4.5 Pelatihan Model

Proses pelatihan model *Computer Vision* dilakukan menggunakan metode *iterative learning* dengan batas maksimal 15 *epoch*. Dan berikut ini adalah progres pelatihan dari tiga skenario data split :

**Tabel 4.7** Progres Pelatihan Model 70:30

| Epoch | Accuracy | Loss   |
|-------|----------|--------|
| 1     | 0.5062   | 1.0642 |
| 2     | 0.6194   | 0.8989 |
| 3     | 0.6727   | 0.8105 |
| 4     | 0.6937   | 0.7519 |
| 5     | 0.7202   | 0.6990 |
| 6     | 0.7380   | 0.6560 |
| 7     | 0.7477   | 0.6145 |
| 8     | 0.7690   | 0.5779 |
| 9     | 0.7834   | 0.5363 |
| 10    | 0.7967   | 0.5098 |
| 11    | 0.8091   | 0.4843 |
| 12    | 0.8190   | 0.4529 |
| 13    | 0.8294   | 0.4194 |
| 14    | 0.8367   | 0.3957 |
| 15    | 0.8424   | 0.3775 |

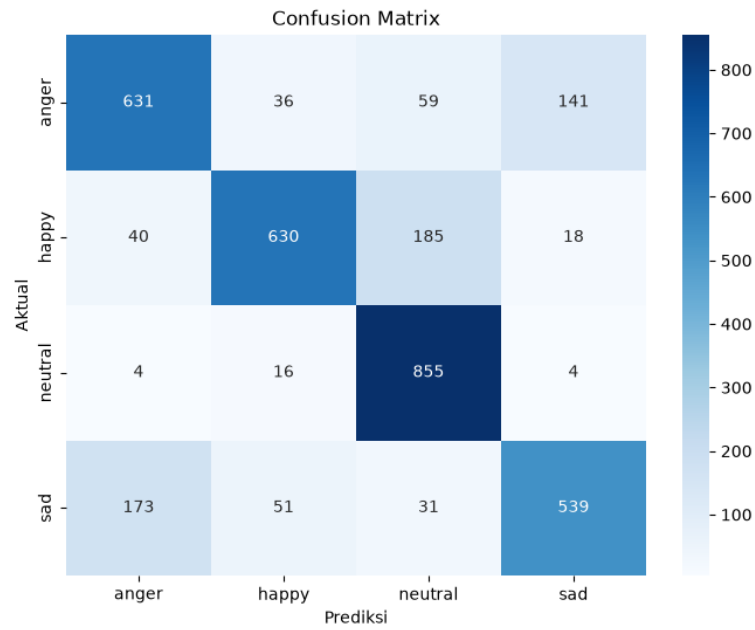
**Tabel 4.8** Progres Pelatihan Model 80:20

| <b>Epoch</b> | <b>Accuracy</b> | <b>Loss</b> |
|--------------|-----------------|-------------|
| 1            | 0.5250          | 1.0396      |
| 2            | 0.6448          | 0.8589      |
| 3            | 0.6832          | 0.7763      |
| 4            | 0.7093          | 0.7191      |
| 5            | 0.7344          | 0.6641      |
| 6            | 0.7600          | 0.6146      |
| 7            | 0.7682          | 0.5849      |
| 8            | 0.7897          | 0.5341      |
| 9            | 0.7995          | 0.5015      |
| 10           | 0.8176          | 0.4590      |
| 11           | 0.8272          | 0.4289      |
| 12           | 0.8359          | 0.3958      |
| 13           | 0.8514          | 0.3667      |
| 14           | 0.8609          | 0.3451      |
| 15           | 0.8769          | 0.3083      |

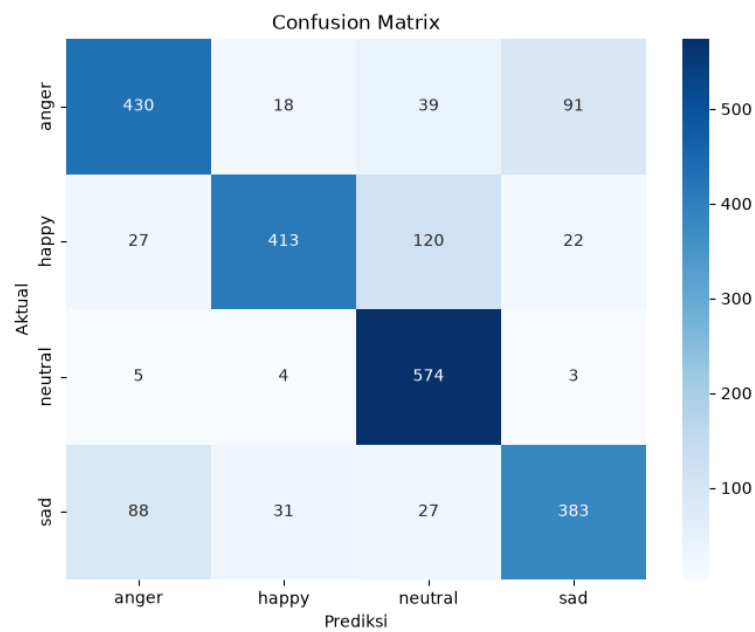
**Tabel 4.9** Progres Pelatihan Model 90:10

| <b>Epoch</b> | <b>Accuracy</b> | <b>Loss</b> |
|--------------|-----------------|-------------|
| 1            | 0.5463          | 1.0162      |
| 2            | 0.6614          | 0.8269      |
| 3            | 0.7123          | 0.7313      |
| 4            | 0.7383          | 0.6597      |
| 5            | 0.7520          | 0.6183      |
| 6            | 0.7680          | 0.5824      |
| 7            | 0.7876          | 0.5411      |
| 8            | 0.7984          | 0.5005      |
| 9            | 0.8169          | 0.4601      |
| 10           | 0.8280          | 0.4224      |
| 11           | 0.8424          | 0.3935      |
| 12           | 0.8495          | 0.3685      |
| 13           | 0.8595          | 0.3433      |
| 14           | 0.8668          | 0.3228      |
| 15           | 0.8834          | 0.2850      |

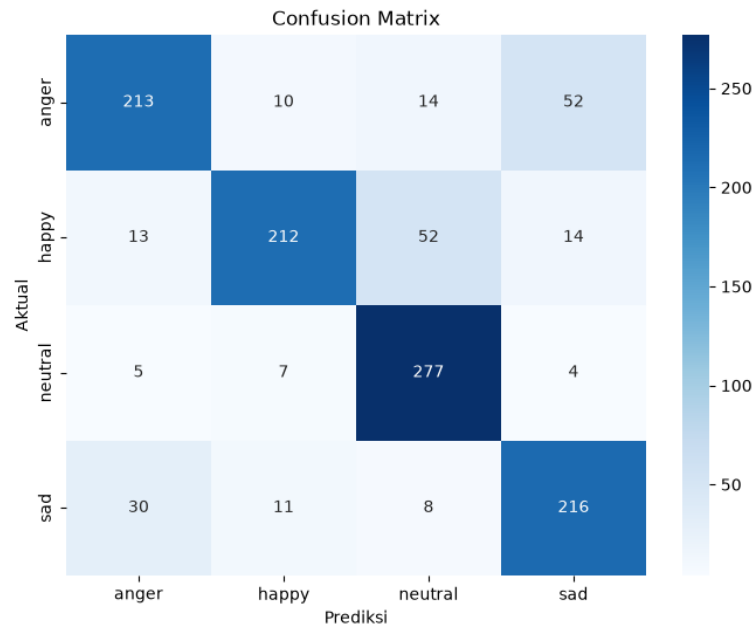
Melalui data tersebut, terlihat bahwa model mencapai kondisi paling optimal pada skenario data split 80:20 dengan *Accuracy* sebesar 87,69% dan tingkat *loss* sebesar 0.3083. Setelah proses pelatihan pada setiap skenario data split selesai, dilakukan evaluasi menggunakan *Confusion Matrix* untuk melihat performa spesifik di setiap skenario data split.



**Gambar 4.18** *Confusion Matrix* Performa Model pada Skenario Data Split 70:30



**Gambar 4.19** *Confusion Matrix* Performa Model pada Skenario Data Split 80:20



**Gambar 4.20** *Confusion Matrix* Performa Model pada Skenario Data Split 90:10

Berdasarkan *Confusion Matrix* pada Gambar 4.18, Gambar 4.19, dan Gambar 4.20, dapat dihitung nilai akurasi model dan *Cohen's Kappa Score* untuk mengevaluasi kinerja model klasifikasi emosi yang telah dikembangkan.

#### 4.4.5.1 Perhitungan Skenario 70:30

**Tabel 4.10** *Confusion Matrix* Performa Model pada Skenario Data Split 70:30

| Pred/Act       | Pred_anger | Pred_happy | Pred_neutral | Pred_sad |
|----------------|------------|------------|--------------|----------|
| Actual_anger   | 631        | 36         | 59           | 141      |
| Actual_happy   | 40         | 630        | 185          | 18       |
| Actual_neutral | 4          | 16         | 855          | 4        |
| Actual_sad     | 173        | 51         | 31           | 539      |

Nilai akurasi dihitung berdasarkan jumlah prediksi yang benar (nilai pada diagonal utama *confusion matrix*) atau yang di sebut *True Positive* dibandingkan dengan jumlah seluruh data pengujian. Rumus akurasi ditunjukkan dibawah ini :

$$Accuracy = \frac{\text{Jumlah Prediksi Benar (True Positive)}}{\text{Jumlah Seluruh Data}}$$

Berdasarkan Tabel 4.10, jumlah prediksi yang benar diperoleh dari penjumlahan nilai diagonal utama, yaitu:

$$631 + 630 + 855 + 539 = 2655$$

Sedangkan jumlah seluruh data pengujian adalah:

$$867 + 873 + 879 + 794 = 3413$$

Sehingga nilai akurasi model dapat dihitung sebagai berikut:

$$Accuracy = \frac{2655}{3413}$$

$$Accuracy = 0.7779$$

$$Accuracy = 77.79\%$$

Berdasarkan hasil perhitungan tersebut, model data split 70:30 mampu mengklasifikasikan emosi dengan benar pada 77,79% dari seluruh data pengujian.

Selain menggunakan akurasi, penelitian ini juga menggunakan *Cohen's Kappa Score* untuk mengukur tingkat kesesuaian antara hasil prediksi model dan label sebenarnya dengan mempertimbangkan kemungkinan kesesuaian yang terjadi secara acak. *Cohen's Kappa* dihitung menggunakan Persamaan dibawah ini :

$$\kappa = \frac{P_o - P_e}{1 - P_e}$$

dengan:

- a.  $P_o$  = observed agreement (kesesuaian yang diamati atau akurasi aktual)
- b.  $P_e$  = expected agreement (kesesuaian yang diharapkan terjadi secara acak)

Nilai *observed agreement* diperoleh dari nilai akurasi yang telah dihitung sebelumnya:

$$P_o = 0.7779$$

Selanjutnya, nilai *expected agreement* dihitung menggunakan total data aktual dan total data prediksi pada masing-masing kelas. Berdasarkan *confusion matrix* diperoleh total aktual dan total prediksi sebagai berikut:

**Tabel 4.11** Total Data Aktual dan Total Data Prediksi Model Data Split 70:30

| Kelas   | Total Aktual | Total Prediksi |
|---------|--------------|----------------|
| Anger   | 867          | 848            |
| Happy   | 873          | 733            |
| Neutral | 879          | 1130           |
| Sad     | 794          | 702            |

Sehingga nilai  $P_e$  dihitung dengan jumlah seluruh data pengujian ( $N$ ) diperoleh dari penjumlahan seluruh data aktual pada confusion matrix:

$$N = 867 + 873 + 879 + 794$$

$$N = 3413$$

Sehingga:

$$N^2 = 3413^2$$

$$N^2 = 11648569$$

Selanjutnya nilai  $P_e$  dihitung menggunakan total aktual dan total prediksi masing-masing kelas:

$$P_e = \frac{(867 \times 848) + (873 \times 733) + (879 \times 1130) + (794 \times 702)}{3413^2}$$

$$P_e = \frac{735216 + 639909 + 993270 + 557388}{11648569}$$

$$P_e = \frac{2925783}{11648569}$$

$$P_e = 0.2512$$

Selanjutnya nilai Cohen's Kappa Score dihitung sebagai berikut:

$$\kappa = \frac{P_o - P_e}{1 - P_e}$$

$$\kappa = \frac{0.7779 - 0.2512}{1 - 0.2512}$$

$$\kappa = \frac{0.5267}{0.7488}$$

$$\kappa = 0.7034$$

Berdasarkan perhitungan nilai Cohen's Kappa Score pada skenario data split 70:30 adalah 0.7034

#### 4.4.5.2 Perhitungan Skenario 80:20

**Tabel 4.12** *Confusion Matrix* Performa Model pada Skenario Data Split 80:20

| Pred/Act       | Pred_anger | Pred_happy | Pred_neutral | Pred_sad |
|----------------|------------|------------|--------------|----------|
| Actual_anger   | 430        | 18         | 39           | 91       |
| Actual_happy   | 27         | 413        | 120          | 22       |
| Actual_neutral | 5          | 4          | 574          | 3        |
| Actual_sad     | 88         | 31         | 27           | 383      |

Berikut ini adalah perhitungan akurasi pada data split skenario 80:20.

$$Accuracy = \frac{\text{Jumlah Prediksi Benar (True Positive)}}{\text{Jumlah Seluruh Data}}$$

Berdasarkan Tabel 4.12, jumlah prediksi yang benar diperoleh dari penjumlahan nilai diagonal utama, yaitu:

$$430 + 413 + 574 + 383 = 1800$$

Sedangkan jumlah seluruh data pengujian adalah:

$$578 + 582 + 586 + 529 = 2275$$

Sehingga nilai akurasi model dapat dihitung sebagai berikut:

$$Accuracy = \frac{1800}{2275}$$

$$Accuracy = 0.7912$$

$$Accuracy = 79.12\%$$

Berdasarkan hasil perhitungan tersebut, model data split 80:20 mampu mengklasifikasikan emosi dengan benar pada 79,12% dari seluruh data pengujian.

Berikutnya adalah menghitung *Cohen's Kappa Score* untuk model data split 80:20, *Cohen's Kappa* dihitung menggunakan Persamaan dibawah ini :

$$\kappa = \frac{P_o - P_e}{1 - P_e}$$

dengan:

- a.  $P_o$  = observed agreement (kesesuaian yang diamati atau akurasi aktual)
- b.  $P_e$  = expected agreement (kesesuaian yang diharapkan terjadi secara acak)

Nilai *observed agreement* diperoleh dari nilai akurasi yang telah dihitung sebelumnya:

$$P_o = 0.7912$$

Selanjutnya, nilai *expected agreement* dihitung menggunakan total data aktual dan total data prediksi pada masing-masing kelas. Berdasarkan *confusion matrix* diperoleh total aktual dan total prediksi sebagai berikut:

**Tabel 4.13** Total Data Aktual dan Total Data Prediksi Model Data Split 80:20

| Kelas   | Total Aktual | Total Prediksi |
|---------|--------------|----------------|
| Anger   | 578          | 550            |
| Happy   | 582          | 466            |
| Neutral | 586          | 760            |
| Sad     | 529          | 499            |

Sehingga nilai  $P_e$  dihitung dengan jumlah seluruh data pengujian ( $N$ ) diperoleh dari penjumlahan seluruh data aktual pada confusion matrix:

$$N = 578 + 582 + 586 + 529$$

$$N = 2275$$

Sehingga:

$$N^2 = 2275^2$$

$$N^2 = 5175625$$

Selanjutnya nilai  $P_e$  dihitung menggunakan total aktual dan total prediksi masing-masing kelas:

$$P_e = \frac{(578 \times 550) + (582 \times 466) + (586 \times 760) + (529 \times 499)}{2275^2}$$

$$P_e = \frac{17900 + 271212 + 445360 + 263971}{5175625}$$

$$P_e = \frac{1298443}{5175625}$$

$$P_e = 0.2509$$

Selanjutnya nilai Cohen's Kappa Score dihitung sebagai berikut:

$$\begin{aligned}\kappa &= \frac{P_o - P_e}{1 - P_e} \\ \kappa &= \frac{0.7912 - 0.2509}{1 - 0.2509} \\ \kappa &= \frac{0.5403}{0.7491} \\ \kappa &= 0.7213\end{aligned}$$

Berdasarkan perhitungan nilai Cohen's Kappa Score pada skenario data split 80:20 adalah 0.7213

#### 4.4.5.3 Perhitungan Skenario 90:10

**Tabel 4.14** Confusion Matrix Performa Model pada Skenario Data Split 90:10

| Pred/Act       | Pred_anger | Pred_happy | Pred_neutral | Pred_sad |
|----------------|------------|------------|--------------|----------|
| Actual_anger   | 213        | 10         | 14           | 52       |
| Actual_happy   | 13         | 212        | 52           | 14       |
| Actual_neutral | 5          | 7          | 277          | 4        |
| Actual_sad     | 30         | 11         | 8            | 216      |

Berikut ini adalah perhitungan akurasi pada data split skenario 90:10.

$$Accuracy = \frac{\text{Jumlah Prediksi Benar (True Positive)}}{\text{Jumlah Seluruh Data}}$$

Berdasarkan Tabel 4.14, jumlah prediksi yang benar diperoleh dari penjumlahan nilai diagonal utama, yaitu:

$$213 + 212 + 277 + 216 = 918$$

Sedangkan jumlah seluruh data pengujian adalah:

$$289 + 291 + 293 + 265 = 1138$$

Sehingga nilai akurasi model dapat dihitung sebagai berikut:

$$\begin{aligned}Accuracy &= \frac{918}{1138} \\ Accuracy &= 0.8067\end{aligned}$$

$$Accuracy = 80.67\%$$

Berdasarkan hasil perhitungan tersebut, model data split 90:10 mampu mengklasifikasikan emosi dengan benar pada 80,67% dari seluruh data pengujian

Berikutnya adalah menghitung *Cohen's Kappa Score* untuk model data split 90:10, *Cohen's Kappa* dihitung menggunakan Persamaan dibawah ini :

$$\kappa = \frac{P_o - P_e}{1 - P_e}$$

dengan:

- a.  $P_o$  = observed agreement (kesesuaian yang diamati atau akurasi 70ctual)
- b.  $P_e$  = expected agreement (kesesuaian yang diharapkan terjadi secara acak)

Nilai *observed agreement* diperoleh dari nilai akurasi yang telah dihitung sebelumnya:

$$P_o = 0.8067$$

Selanjutnya, nilai *expected agreement* dihitung menggunakan total data 70ctual dan total data prediksi pada masing-masing kelas. Berdasarkan *confusion matrix* diperoleh total aktual dan total prediksi sebagai berikut:

**Tabel 4.15** Total Data Aktual dan Total Data Prediksi Model Data Split 90:10

| Kelas   | Total Aktual | Total Prediksi |
|---------|--------------|----------------|
| Anger   | 289          | 261            |
| Happy   | 291          | 240            |
| Neutral | 293          | 351            |
| Sad     | 265          | 286            |

Sehingga nilai  $P_e$  dihitung dengan jumlah seluruh data pengujian ( $N$ ) diperoleh dari penjumlahan seluruh data aktual pada confusion matrix:

$$N = 289 + 291 + 293 + 265$$

$$N = 1138$$

Sehingga:

$$N^2 = 1138^2$$

$$N^2 = 1295044$$

Selanjutnya nilai  $P_e$  dihitung menggunakan total aktual dan total prediksi masing-masing kelas:

$$P_e = \frac{(289 \times 261) + (291 \times 240) + (293 \times 351) + (265 \times 286)}{1138^2}$$

$$P_e = \frac{75429 + 69840 + 102843 + 75790}{1295044}$$

$$P_e = \frac{323902}{1295044}$$

$$P_e = 0.2501$$

Selanjutnya nilai Cohen's Kappa Score dihitung sebagai berikut:

$$\kappa = \frac{P_o - P_e}{1 - P_e}$$

$$\kappa = \frac{0.8067 - 0.2501}{1 - 0.2501}$$

$$\kappa = \frac{0.5566}{0.7499}$$

$$\kappa = 0.7422$$

Berdasarkan perhitungan nilai Cohen's Kappa Score pada skenario data split 90:10 adalah 0.7422

#### 4.4.5.4 Model Terbaik

Setelah melakukan perhitungan dari tiga skenario data split di dapatkan hasil seperti tabel 4.16 di bawah ini:

**Tabel 4.16** Ringkasan Model Data Split

| <b>Data Split</b> | <b>Akurasi</b> | <b>Cohen's Kappa Score</b> |
|-------------------|----------------|----------------------------|
| 70:30             | 77.79%         | 0.7034                     |
| 80:20             | 79.12%         | 0.7213                     |
| 90:10             | 80.67%         | 0.7422                     |

Menurut skala interpretasi Cohen's Kappa yang dikemukakan oleh Landis dan Koch (Moons & Vandervieren, 2025).

**Tabel 4.17** Interpretasi *Cohen's Kappa Score*

| Nilai Kappa | Interpretasi             |
|-------------|--------------------------|
| < 0,00      | Poor Agreement           |
| 0,00 – 0,20 | Slight Agreement         |
| 0,21 – 0,40 | Fair Agreement           |
| 0,41 – 0,60 | Moderate Agreement       |
| 0,61 – 0,80 | Substantial Agreement    |
| 0,81 – 1,00 | Almost Perfect Agreement |

ketiga skenario memiliki nilai kappa yang berada pada rentang 0,61–0,80, sehingga seluruhnya termasuk dalam kategori *Substantial Agreement*. Hal ini menunjukkan bahwa model memiliki tingkat kesesuaian yang kuat antara hasil prediksi dengan label sebenarnya setelah memperhitungkan kemungkinan kesesuaian yang terjadi secara acak.

Meskipun ketiga skenario menunjukkan performa yang baik, skenario data split 90:10 memberikan hasil terbaik dibandingkan dua skenario lainnya. Hal tersebut ditunjukkan oleh nilai akurasi tertinggi sebesar 80,67% dan *Cohen's Kappa Score* sebesar 0,7422. Peningkatan performa ini menunjukkan bahwa proporsi data latih yang lebih besar memberikan lebih banyak variasi informasi kepada model selama proses pembelajaran, sehingga model mampu mengenali pola karakteristik ekspresi wajah dengan lebih baik.

Berdasarkan hasil tersebut, dapat disimpulkan bahwa skenario data split 90:10 merupakan konfigurasi terbaik pada penelitian ini, karena mampu menghasilkan tingkat akurasi tertinggi sekaligus tingkat kesesuaian prediksi yang paling baik berdasarkan *Cohen's Kappa Score*. Dengan demikian, model yang dihasilkan tidak hanya memiliki kemampuan klasifikasi yang tinggi, tetapi juga menunjukkan konsistensi yang baik dalam mengidentifikasi empat kategori emosi, yaitu Anger, Happy, Neutral, dan Sad, sehingga layak digunakan sebagai model akhir pada penelitian ini.

#### 4.4.6 Implementasi *Inference* menggunakan Flask API

Tahap akhir dari pengembangan modul *Computer Vision* adalah pengimplementasian model ke dalam layanan web API menggunakan *framework* Flask. Layanan ini berfungsi sebagai jembatan komunikasi antara logika analisis emosi di sisi Python dengan sistem utama di sisi Java (Minecraft), yang memungkinkan proses deteksi emosi dilakukan secara terpisah tanpa membebani performa *engine* simulasi.

##### 1. Mekanisme Kerja API

Layanan ini menyediakan satu titik akhir (*endpoint*) */capture\_and\_detect\_emotion* yang dapat diakses melalui metode HTTP GET. Proses yang terjadi di dalam API ini saat menerima permintaan adalah sebagai berikut:

- a. Akuisisi Citra: Sistem secara otomatis mengaktifkan perangkat *webcam* menggunakan pustaka *OpenCV* (*cv2.VideoCapture*) untuk menangkap satu bingkai (*frame*) citra pemain secara langsung.
- b. Deteksi Wajah: Citra yang tertangkap dikonversi ke skala abu-abu (*grayscale*) untuk mempermudah deteksi. Objek wajah kemudian diidentifikasi menggunakan algoritma *Haar Cascade Classifier* (*haarcascade\_frontalface\_default.xml*).
- c. Pra-pemrosesan Data: Area wajah yang terdeteksi akan dipotong (*cropping*) dan disesuaikan ukurannya menjadi 100x100 piksel. Data tersebut kemudian dinormalisasi dengan membagi nilai piksel dengan 255.0 agar berada dalam rentang 0 hingga 1, sesuai dengan standar pemrosesan saat pelatihan model.
- d. *Inference* Model: Data wajah yang telah siap dimasukkan ke dalam model *emotion\_model\_best.keras* untuk mendapatkan probabilitas emosi.
- e. Klasifikasi Akhir: Indeks dengan probabilitas tertinggi diterjemahkan kembali ke dalam label emosi manusia (*Anger, Happy, Neutral, Sad*).

##### 2. Struktur Output JSON

Hasil deteksi dikirimkan kembali kepada pemanggil API dalam format JSON (*JavaScript Object Notation*). Format ini dipilih karena ringan dan mudah

diparsing oleh bahasa pemrograman lain seperti Java. Contoh data yang dihasilkan adalah sebagai berikut:

**Tabel 4.18** Data JSON

| Key        | Deskripsi                                                |
|------------|----------------------------------------------------------|
| box        | Nama kategori emosi yang terdeteksi (misal: "Happy").    |
| confidence | Nilai probabilitas keakuratan deteksi (rentang 0-1).     |
| emotion    | Koordinat lokasi wajah pada bingkai kamera [x, y, w, h]. |

```
[
  {
    "box": [
      239,
      125,
      190,
      190
    ],
    "confidence": 0.8285804986953735,
    "emotion": "Happy"
  }
]
```

**Gambar 4.21** Format JSON

### 3. Integrasi Sistem Terdistribusi

Melalui arsitektur ini, agen cerdas di Minecraft hanya perlu melakukan permintaan HTTP singkat untuk mengetahui kondisi psikologis pemain. Pendekatan ini sangat efisien karena:

- Isolasi Komputasi:** Dengan memisahkan proses Deep Learning yang membutuhkan konsumsi sumber daya tinggi (CPU/GPU) ke sisi Python, beban kerja pada mesin Minecraft dapat diminimalisir. Hal ini menjamin bahwa proses simulasi game tetap berjalan pada performa optimal (tick-rate yang stabil).
- Skalabilitas:** Penggunaan Flask API memungkinkan pembaruan model klasifikasi emosi dilakukan secara independen tanpa perlu melakukan perubahan pada kode sumber inti di sisi Minecraft. Hal ini mempermudah proses pemeliharaan dan pengembangan sistem di masa depan.

- c. *Near Real-Time*: Seluruh siklus deteksi wajah hingga transmisi data JSON dirancang untuk beroperasi dengan latensi rendah. Mekanisme ini mendukung interaksi yang responsif, di mana perubahan ekspresi pemain dapat segera ditransformasikan menjadi variabel input bagi agen cerdas.

Implementasi API ini memastikan bahwa data afektif yang dihasilkan dari model *Computer Vision* dapat dikirimkan secara akurat ke Gemini API untuk diproses menjadi respons percakapan yang sesuai.

#### **4.5 Implementasi Sistem Memori dan Persistensi Data**

Implementasi sistem memori merupakan komponen fundamental yang membedakan agen cerdas dari entitas berbasis skrip konvensional. Apabila lapisan persepsi (Bab 4.2 dan 4.3) memberikan pemahaman mengenai kondisi dunia simulasi secara *real-time* dan kondisi emosional pemain secara *near real-time*, maka sistem memori memberikan dimensi kontinuitas kognitif yang memungkinkan agen memiliki sejarah interaksi serta pengetahuan spasial yang menetap. Tanpa adanya sistem memori yang terintegrasi, agen akan bersifat stateless sebuah kondisi di mana setiap interaksi dianggap sebagai kejadian baru tanpa adanya pengaruh dari pengalaman masa lalu.

Pada sub-bab ini, dibahas mekanisme pengelolaan data melalui persistensi NBT (*Named Binary Tag*) untuk menjamin keamanan data saat sesi permainan dihentikan, manajemen memori spasial untuk mendukung navigasi berbasis ingatan lokasi, serta sistem penyimpanan memori jangka pendek dan panjang. Arsitektur memori ini dirancang untuk menciptakan kesinambungan kognitif, di mana agen mampu melakukan refleksi terhadap keberhasilan tugas maupun dinamika hubungan sosial dengan pemain, yang pada akhirnya menjadi variabel penentu dalam proses pengambilan keputusan di masa depan.

##### **4.5.1 Persistensi Data Fisik berbasis NBT**

Fondasi teknis dari sistem memori agen cerdas ini bertumpu pada mekanisme Persistensi Data Fisik. Tanpa adanya penyimpanan yang menetap pada level hardware, seluruh proses kognitif agen hanya akan bersifat sementara di dalam RAM (memori jangka pendek komputer) dan akan terhapus secara otomatis saat aplikasi simulasi dihentikan. Untuk mengatasi tantangan tersebut, penelitian ini

memanfaatkan format NBT (*Named Binary Tag*) sebagai infrastruktur penyimpanan utama.

NBT sendiri merupakan sebuah standar format data biner hierarkis yang dikembangkan untuk menyimpan informasi dunia Minecraft secara efisien. Format ini memungkinkan penyimpanan data yang kompleks dalam bentuk pohon tag (seperti *Compound*, *List*, dan *String*) dengan ukuran penyimpanan yang sangat minimal. Implementasi NBT pada sistem agen cerdas ini dilakukan melalui dua proses kritis:

1. Proses Serialisasi (Penyimpanan ke Disk):

Melalui metode *writeCustomDataToNbt()*, sistem melakukan Serialisasi, yaitu proses transformasi objek-objek Java yang aktif di memori menjadi tag-tag biner NBT seperti pada tabel 4.8. Pada tahap ini, seluruh variabel yang mendefinisikan "keadaan" agen saat itu (seperti skor kepercayaan, isi tas, dan riwayat percakapan) dikemas ke dalam sebuah *NbtCompound*. Proses ini memastikan bahwa snapshot kondisi kognitif agen berhasil diabadikan ke dalam file penyimpanan dunia (.dat).

**Tabel 4.19** Struktur Penyimpanan Data NBT Agen Cerdas

| Nama Tag          | Tipe Data NBT | Deskripsi Informasi                                  |
|-------------------|---------------|------------------------------------------------------|
| TrustScore        | Int           | Skor kepercayaan agen terhadap pemain (0-100).       |
| Emotion           | String        | Status emosi terakhir yang dirasakan agen.           |
| Inventory         | List          | Daftar seluruh item yang disimpan di dalam tas NPC.  |
| Owner             | UUID          | Identitas unik universal milik pemain.               |
| PersonalityPreset | String        | Nama preset kepribadian yang aktif (misal: Periang). |

|                 |                      |                                                      |
|-----------------|----------------------|------------------------------------------------------|
| SocialInsights  | String               | Narasi hasil refleksi hubungan jangka panjang.       |
| ChatHistory     | List                 | Antrean teks percakapan antara agen dan pemain.      |
| <b>Nama Tag</b> | <b>Tipe Data NBT</b> | <b>Deskripsi Informasi</b>                           |
| SpatialMemory   | Compound             | Kumpulan koordinat lokasi blok penting yang diingat. |

## 2. Proses Deserialisasi (Pemuatan ke Memori):

Saat agen dipanggil kembali ke dalam dunia Minecraft, sistem menjalankan metode *readCustomDataFromNbt()*. Proses ini melakukan Deserialisasi, di mana data biner dibaca dan dikonversi kembali menjadi variabel Java yang fungsional. Salah satu elemen vital dalam proses ini adalah pemulihan UUID Pemain (Owner UUID). Penggunaan UUID, bukan sekadar nama teks, menjamin bahwa agen tetap mengenali pemain secara permanen, bahkan jika pemain tersebut melakukan perubahan nama akun (*username*). Implementasi persistensi biner ini merupakan langkah krusial untuk mengubah entitas dari sebuah program statis menjadi sebuah Agen yang Memiliki Kondisi (*Stateful Agent*). Hal ini memungkinkan agen untuk mempertahankan identitas, aset, dan memori lintas sesi permainan, yang menjadi syarat mutlak bagi terciptanya interaksi jangka panjang antara manusia dan agen cerdas.

Implementasi ini memberikan landasan bagi terbentuknya Agen yang Memiliki Kondisi (*Stateful Agent*), di mana agen mampu mempertahankan identitas dan memori lintas sesi permainan, menciptakan kesan adanya keberlanjutan hidup pada entitas tersebut.

### 4.5.2 Manajemen Memori Sosial dan Atribut Psikologis

Berbeda dengan entitas standar dalam permainan yang bersifat statis, agen cerdas pada penelitian ini dirancang untuk memiliki variabel psikologis yang dinamis dan

menetap. Implementasi memori sosial bertujuan untuk menciptakan Hubungan Parasosial yang bermakna, di mana setiap interaksi yang dilakukan oleh pemain akan memiliki dampak jangka panjang terhadap perilaku dan gaya komunikasi agen. Komponen utama dalam manajemen memori sosial ini dibagi menjadi lima elemen fundamental:

1. Dinamika Skor Kepercayaan (*Trust Score*):

*Trust Score* merupakan variabel numerik (skala 0-100) yang bertindak sebagai *Governor* Perilaku agen. Data ini diekstraksi dari pengalaman interaksi langsung di dalam simulasi.

- a. Mekanisme: Skor ini akan meningkat secara bertahap apabila pemain melakukan tindakan positif (seperti memberikan makanan atau item) dan akan menurun secara drastis apabila pemain melakukan tindakan agresif (seperti menyerang/memukul agen).
- b. Dampak: Variabel ini dialirkan ke dalam *Context Builder* sebagai indikator kepercayaan. Agen dengan tingkat kepercayaan rendah di bawah 20 akan menunjukkan perilaku defensif, ketus, atau menolak menjalankan perintah tertentu seperti pada gambar 4.21, yang mencerminkan simulasi dinamika hubungan sosial yang realistis.



**Gambar 4.22** NPC Menolak Permintaan saat *Trust Score* nya Rendah

## 2. Dinamika Skor Percaya Diri (*Confidance Score*)

*Confidance Score* merupakan variabel numerik dengan skala 0 hingga 100 yang merepresentasikan kondisi psikologis internal agen, khususnya mengenai persepsi kompetensi dirinya sendiri dalam menjalankan tugas. Jika Skor Kepercayaan (*Trust Score*) berfungsi mengatur dimensi hubungan eksternal antara NPC dan pemain, maka Skor Percaya Diri bertindak sebagai indikator kesehatan mental internal yang memengaruhi bagaimana agen memandang kemampuannya sendiri.

- a. Mekanisme: Skor ini bersifat sangat dinamis dan berevolusi berdasarkan rekam jejak aksi nyata di dalam dunia Minecraft. Setiap keberhasilan dalam menyelesaikan instruksi, seperti sukses mengumpulkan sumber daya atau memenangkan pertempuran, akan memberikan umpan balik positif yang meningkatkan skor tersebut. Sebaliknya, kegagalan sistemik seperti kegagalan navigasi (*stuck*), ketidakmampuan menemukan item yang dicari, atau akumulasi tugas yang tidak terselesaikan akan mengikis rasa percaya diri agen secara bertahap.
- b. Dampak: Pada tingkat rendah (di bawah 30-50), agen akan memasuki fase "minder" atau ragu-ragu seperti pada gambar 4, yang secara otomatis mengubah gaya bicaranya menjadi lebih pasif, sering meminta maaf, dan menggunakan kalimat yang menunjukkan ketidakyakinan saat menerima perintah baru. Namun, seiring meningkatnya keberhasilan tugas hingga mencapai skor tinggi (di atas 70-85), mood agen akan bertransformasi menjadi optimis dan bersemangat. Dalam kondisi ini, agen akan berkomunikasi dengan nada yang lebih tegas, berinisiatif, dan berani memberikan saran proaktif kepada pemain.



**Gambar 4.23** Jika *Confidance Score NPC* akan menjawab dengan ragu-ragu

### 3. Wawasan Sosial (Social Insights):

Social Insights berfungsi sebagai basis pengetahuan kualitatif yang merangkum hasil refleksi hubungan jangka panjang yang dilakukan NPC dalam satu siklus hari Minecraft. Di dalam sistem, data ini disimpan sebagai teks narasi deskriptif. Jika Trust Score memberikan data kuantitatif (angka), maka Social Insights memberikan landasan semantik (cerita). Misalnya, memori dapat berisi catatan: "(nama pemain) sering menyelamatkanku saat malam hari, ia adalah rekan yang dapat diandalkan." Informasi ini memungkinkan LLM untuk menghasilkan respon yang sangat personal dan memiliki kedalaman konteks sejarah.

### 4. Ringkasan Kejadian (*Daily Summary*)

*Daily Summary* merupakan komponen memori cerdas yang berfungsi sebagai catatan kronologis mengenai aktivitas operasional dan pencapaian teknis yang dilakukan NPC dalam satu siklus hari Minecraft. Berbeda dengan Wawasan Sosial yang bersifat kualitatif dan menganalisis dinamika perasaan atau kedekatan emosional, Ringkasan Kejadian lebih menitikberatkan pada fakta objektif mengenai "apa yang telah dikerjakan". Informasi ini mencakup daftar tugas yang berhasil diselesaikan (seperti mengumpulkan kayu atau

berburu), lokasi yang dikunjungi, hingga rintangan lingkungan yang dihadapi. Pemisahan ini sangat penting bagi kognisi AI untuk menjaga kejernihan konteks interaksi. Dengan adanya Ringkasan Kejadian, NPC dapat membedakan antara progres petualangan dengan status hubungan personal, sehingga meskipun hubungan mereka sedang sangat erat (Wawasan Sosial), NPC tetap bisa mengingat secara akurat bahwa hari ini mereka gagal mendapatkan sumber daya yang dicari atau sedang berada di tengah rencana pembangunan (*build*) tertentu.

#### 5. Profil Persona dan Konsistensi Diri:

Agar agen memiliki karakter yang konsisten lintas sesi permainan, sistem menyimpan profil persona yang meliputi Jenis Kelamin (*Gender*), Preset Kepribadian, dan Deskripsi Karakter Kustom. Persistensi profil ini menjamin bahwa agen tetap mempertahankan gaya bicara, panggilan khusus kepada pemain, serta ego karakternya. Hal ini mencegah terjadinya diskontinuitas kepribadian yang sering terjadi pada sistem AI stateless, sehingga interaksi yang terjalin terasa lebih autentik dan memiliki identitas yang kuat.



**Gambar 4.24** Custom Persona Agen

Penerapan memori sosial ini memungkinkan agen cerdas untuk berkembang secara karakter seiring berjalannya waktu permainan. Dengan adanya ingatan terhadap perlakuan pemain di masa lalu, agen tidak lagi sekadar merespons perintah secara mekanis, melainkan merespons berdasarkan akumulasi pengalaman sosial yang telah dilaluinya.

### 4.5.3 Manajemen Memori Spasial (Ingatan Lokasi)

Manajemen memori spasial merupakan kemampuan kognitif agen cerdas untuk merepresentasikan, menyimpan, dan memanggil kembali informasi mengenai struktur geografis serta lokasi objek penting di dalam dunia simulasi. Dalam disiplin ilmu kecerdasan buatan, fitur ini berkaitan erat dengan konsep Pemetaan Kognitif (*Cognitive Mapping*), di mana agen membangun pemahaman mental tentang lingkungannya yang melampaui batas jarak pandang (*Field of View*). Implementasi memori spasial pada agen cerdas ini mencakup beberapa mekanisme teknis utama:

1. Pencatatan Lokasi Berbasis *Voxel*:

Setiap kali *BlockModule* melakukan pemindaian lingkungan, blok-blok yang dikategorikan sebagai "objek penting" (seperti pohon, bijih tambang, atau alat utilitas) akan dicatat koordinatnya ke dalam spasial memori agen. Data ini disimpan dalam bentuk pasangan ID Blok dan Koordinat (X, Y, Z). Dengan mekanisme ini, agen tidak hanya "melihat" blok, tetapi juga "mempelajari" letaknya di dalam ruang simulasi.

2. Persistensi Objek (Object Permanence):

Salah satu tantangan dalam navigasi AI adalah hilangnya referensi objek saat objek tersebut berada di luar radius sensor aktif. Memori spasial mengatasi hal ini dengan menerapkan prinsip *Object Permanence*. Artinya, meskipun agen telah berpindah lokasi sejauh ratusan blok dari sebuah *Crafting Table*, agen tetap "mengetahui" bahwa objek tersebut ada dan dapat diakses kembali melalui data koordinat yang tersimpan secara permanen di dalam NBT.

3. Integrasi Navigasi dan Penalaran:

Memori spasial menjadi jembatan utama bagi keberhasilan NPC dapat navigasi ke *block* yang tidak dalam pandangan nya. Saat LLM menerima konteks bahwa pemain membutuhkan sumber daya tertentu, *Context Builder* akan menyuntikkan data dari memori spasial ke dalam prompt.

- a. Skenario: Jika pemain memerintahkan "Pergi ke pohon yang tadi," LLM akan merujuk pada daftar lokasi di memori spasial untuk menghasilkan instruksi koordinat yang presisi bagi *ActionHandler*.
- b. Dampak: Hal ini memungkinkan agen untuk melakukan navigasi lintas wilayah secara otonom tanpa harus melakukan pemindaian ulang dari awal.

Penerapan manajemen memori spasial ini memberikan keunggulan navigasi yang menyerupai perilaku manusia, di mana keputusan pergerakan agen didasarkan pada akumulasi pengetahuan geografis, bukan sekadar reaksi impulsif terhadap objek yang terlihat di depan mata.

#### 4.5.4 Memori Jangka Pendek dan Riwayat Interaksi

Memori jangka pendek atau *Working Memory* pada agen cerdas berfungsi sebagai unit penyimpanan informasi sementara yang menangani dinamika interaksi yang sedang berlangsung. Dalam arsitektur kognitif, komponen ini sangat krusial untuk menjaga Kesenambungan Percakapan (*Conversational Context*) dan kesadaran terhadap status tugas yang baru saja dilakukan. Tanpa memori jangka pendek, agen akan kehilangan alur dialog dan cenderung mengulangi kesalahan yang sama karena tidak memiliki referensi mengenai apa yang baru saja terjadi. Implementasi memori jangka pendek pada sistem ini mencakup dua aspek utama:

##### 1. Manajemen Riwayat Percakapan (Chat History Queue):

Sistem mengelola daftar pesan terakhir antara pemain dan agen menggunakan struktur data Antrean (Queue) atau Limited List.

- a. Mekanisme: Sistem menyimpan 10 baris percakapan terakhir di dalam memori aktif. Setiap kali ada pesan baru masuk, pesan tertua akan dihapus dari antrean (*Sliding Window Mechanism*).
- b. Fungsi: Riwayat ini disuntikkan ke dalam Context Builder agar LLM dapat memahami referensi kata ganti (seperti "itu", "ke sana", atau "ambilkan lagi") yang merujuk pada pesan sebelumnya. Hal ini mencegah terjadinya *conversational drift*, di mana agen memberikan respon yang tidak relevan dengan topik pembicaraan yang sedang berjalan.

## 2. Rekaman Hasil Tugas (Task Outcome Awareness):

Selain percakapan, memori jangka pendek juga mencatat hasil dari setiap tugas yang dilakukan oleh agen.

- a. Mekanisme: Setiap kali sebuah *Goal* selesai dieksekusi, sistem mencatat status keberhasilannya (misal: *SUCCESS* saat berhasil menebang pohon atau *FAIL* saat target tidak ditemukan).
- b. Dampak Kognitif: Data ini memungkinkan agen untuk melakukan Evaluasi Diri. Jika agen mengetahui bahwa tugas terakhirnya gagal, pada siklus penalaran berikutnya ia dapat secara otomatis memberikan penjelasan kepada pemain mengenai kendala yang dihadapi (misal: terhalang rintangan atau kekurangan alat), alih-alih mencoba mengulangi aksi yang sama secara berulang-ulang.

## 3. Sinkronisasi Persistensi Jangka Pendek:

Meskipun bersifat "jangka pendek", seluruh riwayat ini tetap disinkronisasikan ke dalam NBT Storage. Hal ini dilakukan agar agen tidak mengalami "amnesia sesaat" apabila pemain melakukan re-log atau menghidupkan kembali permainan di tengah sebuah percakapan atau tugas yang sedang berlangsung.

Implementasi memori jangka pendek ini memberikan kemampuan Kesadaran Kontekstual kepada agen cerdas. Hal ini menjamin bahwa setiap tindakan dan ucapan agen merupakan hasil dari pemahaman terhadap alur peristiwa yang sedang terjadi, menciptakan interaksi yang lebih koheren dan menyerupai pola kognisi manusia.

## 4.6 Orkestrasi Data dan Sintesis Konteks (Context Builder)

Setelah seluruh komponen informasi yang meliputi data lingkungan, persepsi visual, dan sistem memori berhasil diimplementasikan, sistem memerlukan sebuah unit untuk menyatukan seluruh variabel tersebut menjadi satu kesatuan informasi yang koheren. Tahap ini dilakukan oleh kelas *NPCCContextBuilder*, yang bertugas melakukan Orkestrasi Data dan Sintesis Konteks. Proses ini merupakan jembatan antara data teknis dari game engine dengan kemampuan pemrosesan bahasa alami pada LLM.

#### 4.6.1 Strategi Pemetaan Data ke dalam Struktur Markdown

Strategi pemetaan data merupakan tahap krusial dalam mengubah representasi data komputer menjadi pemahaman semantik. Penelitian ini menggunakan Markdown sebagai bahasa markup untuk menyusun konteks karena karakteristiknya yang mampu menciptakan hierarki informasi yang sangat jelas. Berbeda dengan teks polos (*plain text*), Markdown menggunakan simbol-simbol tertentu yang membantu Large Language Model (LLM) untuk membedakan antara instruksi utama, data lingkungan, dan riwayat interaksi. Proses pemetaan data ini dilakukan dengan membagi informasi ke dalam beberapa blok fungsional sebagai berikut:

##### 8.6.1.1 Blok Instruksi Persona (### PERSONA UTAMA)

Blok informasi ### PERSONA UTAMA merupakan pondasi identitas yang diletakkan pada bagian awal struktur konteks. Penempatan ini dilakukan secara strategis karena model bahasa besar (LLM) cenderung memberikan perhatian lebih tinggi pada informasi yang muncul di awal teks (*Primacy Effect*). Blok ini berfungsi sebagai Panduan Perilaku (Behavioral Guardrails) yang menetapkan batas-batas operasional bagi agen saat berinteraksi di dalam simulasi. Hal ini bertujuan agar LLM memperlakukan informasi ini sebagai prioritas utama dalam membentuk gaya bicara. Informasi yang disertakan meliputi:

**Tabel 4.20** Struktur dan Fungsi PERSONA UTAMA

| Struktur ### PERSONA UTAMA                                              | Fungsi                                                                     |
|-------------------------------------------------------------------------|----------------------------------------------------------------------------|
| Kamu adalah companion Minecraft milik pemain.                           | Menetapkan peran agen sebagai asisten petualangan, bukan entitas asing.    |
| Kamu harus tetap konsisten dengan kepribadianmu saat berbicara.         | Menjaga Immersive Experience pemain agar interaksi terasa alami dan hidup. |
| Jangan keluar karakter, jangan menyebut dirimu AI, prompt, atau sistem. |                                                                            |
| Tetap relevan dengan dunia Minecraft dan situasi saat ini.              |                                                                            |

| <b>Struktur ### PERSONA UTAMA</b>                                                                                                                                                                                  | <b>Fungsi</b>                                                               |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------|
| Jika ada preferensi pemilik yang relevan, prioritaskan itu saat memberi saran, memilih target umum, atau merespons kebutuhan pemilik.                                                                              |                                                                             |
| Preset Kepribadian: (Membantu, Tenang, Pemberani, Serius, Canggung)                                                                                                                                                | Memberikan basis emosional dan gaya bahasa utama bagi agen.                 |
| Inti Preset: (Penjelasan masing masing preset kepribadian)                                                                                                                                                         |                                                                             |
| Catatan Persona Kustom: (nama karakter game lain)                                                                                                                                                                  |                                                                             |
| Aturan Bicara: jawab natural, tetap in-character, dan jangan terdengar generik.                                                                                                                                    |                                                                             |
| Aturan Tugas: JANGAN memberikan aksi GATHER atau HUNT jika tugas sebelumnya sudah selesai atau jika pemain tidak memintanya secara eksplisit. Gunakan aksi 'NONE' atau 'CHAT' untuk obrolan biasa.                 | Mengontrol efisiensi aksi fisik agen agar tidak terjadi perilaku repetitif. |
| Instruksi Emosi: Laporan emosi pemain di bawah berasal dari kamera. Jika emosi kamera (misal: Sad) terasa tidak nyambung dengan isi pesan pemain yang ceria, prioritaskan isi pesan tapi tetaplah bersikap empati. | Menangani ambiguitas emosi melalui logika prioritas (Pesan Teks > Kamera).  |

| <b>Struktur ### PERSONA UTAMA</b>                                                                                                                                                                               | <b>Fungsi</b>                                                              |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------|
| Instruksi Roleplay: Gunakan gaya bicara yang sesuai dengan jenis kelaminmu (Laki-laki). Sesuaikan panggilan diri (seperti 'aku/saya/gue') dan panggilan ke pemilik (seperti 'Kak/Bang/Mbak') agar terasa alami. | Menyesuaikan profil sosiolinguistik berdasarkan jenis kelamin agen cerdas. |

#### **8.6.1.2 Blok Identitas dan Parameter Hubungan Sosial (### IDENTITAS & HUBUNGAN SOSIAL)**

Blok informasi ini berfungsi sebagai Basis Data Keadaan (*State Database*) yang mendefinisikan jati diri agen cerdas dan dinamika hubungannya dengan pemain pada saat interaksi berlangsung. Berbeda dengan instruksi persona yang bersifat statis, data pada bagian ini bersifat sangat dinamis. Tujuannya adalah untuk menciptakan Agen yang Memiliki Kondisi (*Stateful Agent*), di mana setiap respon yang dihasilkan sangat dipengaruhi oleh sejarah perlakuan pemain terhadap agen.

**Tabel 4.21** Struktur dan Fungsi IDENTITAS & HUBUNGAN SOSIAL

| <b>Struktur ### IDENTITAS &amp; HUBUNGAN SOSIAL</b> | <b>Fungsi</b>                                                           |
|-----------------------------------------------------|-------------------------------------------------------------------------|
| Nama: (Nama Agen)                                   | Sebagai Identifier unik entitas di dalam kode Java dan komunikasi teks. |
| Jenis Kelamin (Gender): Laki-laki                   | Menentukan parameter sosiolinguistik di dalam kode.                     |
| Pemilik: (Nama Pemain)                              | Mengunci relasi agen hanya kepada satu UUID pemain yang sah.            |
| Skor Kepercayaan (Trust): (0 - 100)                 | Variabel kuantitatif (0-100) yang melacak sejarah hubungan sosial.      |

| <b>Struktur ### IDENTITAS &amp; HUBUNGAN SOSIAL</b>                                                    | <b>Fungsi</b>                                                                                                                                                   |
|--------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Emosi Pemain terhadapku: (Neutral, Happy, Sad, Angry)                                                  | Data afektif hasil integrasi Computer Vision (Python) pemain.                                                                                                   |
| Tingkat Percaya Diri (Confidence): (0 - 100)                                                           | Melacak efikasi diri agen berdasarkan keberhasilan tugas di masa lalu.                                                                                          |
| Mood Saat Ini: (Murung, Terintimidasi, Semangat, Minder, Bangga, Tenag)                                | Hasil sintesis heuristik antara kesehatan, trust, <i>confidence</i> , dan kondisi lingkungan.                                                                   |
| Ringkasan Kejadian Hari Ini: (Rangkuman kegiatan yang di lakukan NPC selama 1 hari di dunia Minecraft) | Hasil rangkuman kegiatan NPC (Seperti menebang pohon, berburu, atau berjelajah) yang berfungsi sebagai pengingat apa saja yang telah di lakukan bersama pemain. |
| Analisis Hubungan Jangka Panjang: (Hasil refleksi hubungan antara NPC dan pemain)                      | Hasil refleksi hubungan sosial, yang berfungsi sebagai hubungan emosional antara NPC dan pemain.                                                                |
| Posisi Saat Ini: (Koordinat terkini agen)                                                              | Memberikan koordinat absolut [X, Y, Z] agen di dalam dunia 3D.                                                                                                  |
| Tugas Saat Ini: (Tugas yang disuruh pemin)                                                             | Melaporkan status operasional agen.                                                                                                                             |
| Status Tugas Terakhir: (SUCSSES atau FAIL, melaporkan peristiwa fisik “seperti di pukul pemain”)       | Closed-Loop Feedback: Melaporkan hasil aksi atau peristiwa fisik terakhir.                                                                                      |

| <b>Struktur ### IDENTITAS &amp; HUBUNGAN SOSIAL</b>                                    | <b>Fungsi</b>                                                        |
|----------------------------------------------------------------------------------------|----------------------------------------------------------------------|
| CATATAN: Jika tugas terakhir sudah SUCCESS, jangan mengulanginya lagi kecuali diminta. | Memberikan batasan operasional eksplisit ( <i>Hard Constraint</i> ). |

### 8.6.1.3 Blok Penglihatan Voxel (Objek di Sekitar) (### PENGLIHATAN)

Blok informasi ### PENGLIHATAN merupakan representasi dari persepsi visual agen cerdas terhadap objek-objek statis atau voxel di dalam lingkungan simulasi. Dalam arsitektur kognitif, bagian ini bertindak sebagai Sensor Jarak Pendek (*Short-range Vision*) yang memberikan konfirmasi visual mengenai keberadaan sumber daya alam atau alat utilitas yang berada dalam jangkauan fisik agen (radius 12 blok).

**Tabel 4.22** Struktur dan Fungsi Parameter Penglihatan

| <b>Struktur ### PENGLIHATAN</b>    | <b>Fungsi</b>                                                          |
|------------------------------------|------------------------------------------------------------------------|
| Daftar Nama Blok: (misal: oak_log) | Mengidentifikasi jenis material atau blok yang terdeteksi oleh sensor. |
| Jarak Spasial: (misal: 3m)         | Memberikan informasi jarak antara posisi agen dengan blok target.      |
| Status Kosong: (Pesan Fallback)    | Memberikan laporan jika tidak ditemukan objek penting di sekitar agen. |

Analisis Peran Strategis Blok Penglihatan pada tabel 4.11:

#### 1. Validasi Kelayakan Tugas (Task Validation):

Data penglihatan berfungsi sebagai filter utama sebelum agen menyetujui sebuah perintah. Sebagai contoh, jika pemain memerintahkan agen untuk mengambil bunga kuning, LLM akan memeriksa daftar pada blok PENGLIHATAN. Jika objek tersebut terdeteksi, AI dapat memberikan respon yang pasti seperti "Tunggu ya, aku ambilkan bunganya!". Tanpa data ini, agen mungkin akan memberikan janji palsu (*hallucination*) terhadap objek yang sebenarnya tidak ada di sekitarnya.

## 2. Filtrasi Informasi Semantik:

Sistem tidak mengirimkan seluruh blok yang ada (seperti udara atau tanah), melainkan hanya blok yang dikategorikan sebagai Objek Penting. Strategi filtrasi ini memastikan bahwa "pikiran" AI tidak dipenuhi oleh informasi sampah (*noise*), melainkan terfokus pada elemen-wilayah yang memiliki nilai fungsional bagi kelangsungan simulasi dan tujuan pemain.

### 8.6.1.4 Blok Riwayat Percakapan (### RIWAYAT PERCAKAPAN)

Blok informasi ### RIWAYAT PERCAKAPAN berfungsi sebagai Memori Kerja (*Working Memory*) yang menangani dinamika komunikasi antara pemain dan agen cerdas. Dalam arsitektur kognitif, komponen ini sangat krusial untuk menjaga Koherensi Dialog, yaitu kemampuan agen untuk memberikan respon yang selaras dengan topik pembicaraan yang sedang berlangsung. Tanpa riwayat percakapan, setiap input dari pemain akan dianggap sebagai kejadian baru yang terisolasi, sehingga interaksi akan terasa kaku dan repetitif.

**Tabel 4.23** Struktur dan Fungsi Parameter Riwayat Percakapan

| Struktur ### RIWAYAT PERCAKAPAN          | Fungsi                                                                               |
|------------------------------------------|--------------------------------------------------------------------------------------|
| Daftar Dialog Terakhir: (maks. 10 baris) | Menyajikan urutan pesan terakhir untuk menjaga konteks topik pembicaraan.            |
| Identitas Pembicara: (Nama Pemain / NPC) | Membedakan antara input dari pemain dan respon yang telah diberikan agen sebelumnya. |
| Pesan Kosong: (Pesan Fallback)           | Memberikan indikasi bahwa ini adalah awal dari sebuah interaksi baru.                |

Analisis Peran Strategis Blok Riwayat Percakapan pada tabel 4.12:

## 1. Penyelesaian Referensi (Reference Resolution):

Riwayat percakapan memungkinkan LLM untuk memahami penggunaan kata ganti atau rujukan kontekstual. Sebagai contoh, jika pemain sebelumnya berkata "Aku butuh kayu" dan kemudian berkata "Ambilkan lagi," agen dapat

memahami bahwa kata "lagi" merujuk pada objek "kayu" melalui data di riwayat percakapan. Kemampuan ini sangat penting untuk menciptakan ilusi kecerdasan alami yang menyerupai cara manusia berkomunikasi.

2. Mencegah Redundansi dan Pengulangan:

Dengan menyertakan respon yang telah ia berikan sebelumnya, agen dapat menghindari pemberian jawaban yang sama secara berulang-ulang. LLM akan melakukan inferensi terhadap Sejarah dialog untuk memastikan informasi yang diberikan tetap segar dan progresif, sehingga interaksi jangka panjang dengan pemain tidak terasa membosankan atau bersifat robotik.

3. Mekanisme Sliding Window untuk Efisiensi:

Sistem menerapkan teknik Sliding Window, di mana hanya 10 percakapan terakhir yang disimpan dan dikirimkan ke konteks. Strategi ini dilakukan untuk melakukan Optimasi Token dan mencegah fenomena *Lost in the Middle*, di mana LLM mulai kehilangan fokus jika diberikan riwayat yang terlalu panjang. Pembatasan ini menjamin agen tetap memiliki "ingatan jangka pendek" yang cukup untuk percakapan saat itu, tanpa membebani latensi pemrosesan API.

#### 8.6.1.5 Blok Kondisi Fisik (### STATUS FISIK)

Blok informasi ### STATUS FISIK berfungsi sebagai unit pelaporan mengenai kondisi biologis dan keselamatan diri agen di dalam dunia simulasi, data di ambil melalui *Status Module*. Dalam disiplin kognisi robotika, bagian ini merepresentasikan kemampuan *Proprioception*, yaitu kesadaran internal agen terhadap status fisiknya sendiri. Informasi ini sangat vital karena menjadi variabel utama dalam menentukan hirarki kebutuhan agen, di mana kebutuhan untuk bertahan hidup (*self-preservation*) harus diprioritaskan di atas pemenuhan perintah pemain.

**Tabel 4.24** Struktur dan Fungsi STATUS FISIK

| Struktur ### STATUS FISIK | Fungsi                                                             |
|---------------------------|--------------------------------------------------------------------|
| Darah: (20/20)            | Memberikan pemahaman kuantitatif mengenai daya tahan agen saat ini |

|                                              |                                                                                        |
|----------------------------------------------|----------------------------------------------------------------------------------------|
| PERINGATAN: Aku sedang terbakar!             | Prioritas Kritis: Meminta AI untuk segera mencari air atau melaporkan kepanikan fisik. |
| Status: Aku terluka parah dan butuh bantuan. | Insting Bertahan Hidup: Memicu AI untuk menolak bertarung dan meminta bantuan medis.   |

Analisis Peran Strategis Blok konsidis fisik pada tabel 4.13 :

1. Implementasi Mekanisme Pertahanan Diri (Self-Preservation Logic) :

Dalam teori kecerdasan buatan, pertahanan diri merupakan prioritas tertinggi dalam hirarki kebutuhan agen otonom. Dengan tersedianya data kesehatan (*Health*) secara real-time, LLM dapat melakukan kalkulasi risiko sebelum mengeksekusi perintah pemain. Sebagai contoh, apabila agen berada dalam status Sekarat (*Low Health*), sistem kognitif akan secara otomatis memberikan bobot prioritas pada aksi defensif atau pemulihan, meskipun pemain memberikan instruksi agresif seperti menyerang monster. Hal ini membuktikan bahwa agen memiliki insting untuk menjaga kelangsungan hidup entitasnya di dalam simulasi.

2. Injeksi Peringatan dan Manajemen Atensi (Urgency Alerting):

Penggunaan kata kunci seperti "PERINGATAN: Aku sedang terbakar!" berfungsi sebagai instrumen manajemen atensi bagi LLM. Di dalam mekanisme *Attention Head* pada arsitektur Transformer (basis LLM), istilah-istilah darurat diberikan bobot perhatian yang lebih tinggi. Hal ini memastikan bahwa respon verbal yang dihasilkan agen akan langsung selaras dengan kondisi fisik yang mendesak, menciptakan dialog yang lebih responsif dan emosional (seperti ungkapan kepanikan saat terbakar), yang pada akhirnya meningkatkan aspek imersif bagi pemain.

3. Kalibrasi Performa dan Batasan Kemampuan:

Data fisik berfungsi sebagai parameter kalibrasi terhadap kemampuan operasional agen. Melalui pemantauan status fisik, agen dapat menginformasikan batasan kemampuannya secara logis kepada pemain.

Misalnya, agen yang sedang terluka parah dapat menolak tugas pengumpulan sumber daya yang berlokasi di tempat berbahaya dengan memberikan alasan fisik yang valid. Integrasi ini mengubah agen dari sekadar pelaksana perintah pasif menjadi mitra petualangan yang memiliki kesadaran terhadap limitasi fisiknya sendiri.

#### 8.6.1.6 Blok Persepsi Lingkungan dan Kondisi Eksternal (### LINGKUNGAN)

Blok informasi ### LINGKUNGAN berfungsi sebagai unit Kesadaran Situasional Makro bagi agen cerdas. Data pada bagian yang di peroleh dari *BiomeModule*, *LightModule*, *TimeModule*, dan *WeatherModule* ini memberikan pemahaman mengenai kondisi fisik dunia simulasi yang bersifat dinamis. Informasi ini sangat krusial karena dalam ekosistem Minecraft, variabel lingkungan seperti waktu dan cuaca secara langsung menentukan tingkat risiko keamanan (seperti kemunculan monster) dan kelayakan pelaksanaan tugas tertentu.

**Tabel 4.25** Struktur dan Fungsi Parameter Lingkungan

| Struktur ### LINGKUNGAN    | Fungsi                                                                           |
|----------------------------|----------------------------------------------------------------------------------|
| Bioma: (Nama Bioma)        | Mengidentifikasi karakteristik geografis (misal: Forest atau Desert) lokasi agen |
| Waktu: (Siang/Malam Hari)  | Melacak siklus diurnal dan nokturnal untuk menentukan tingkat kewaspadaan.       |
| Cuaca: (Cerah/Hujan/Badai) | Mendeteksi kondisi atmosfer yang memicu perilaku spesifik (seperti berteduh).    |
| Tingkat Cahaya: (0-15)     | Mengukur intensitas cahaya lokal untuk mengevaluasi keamanan area.               |

|                                 |                                                                            |
|---------------------------------|----------------------------------------------------------------------------|
| AksesLangit: (Terbuka/Tertutup) | Menentukan apakah agen berada di ruang terbuka atau di bawah naungan atap. |
|---------------------------------|----------------------------------------------------------------------------|

Analisis Peran Strategis Blok Lingkungan pada tabel 4.14:

1. Evaluasi Keamanan Area (*Safety Assessment*):

Parameter Tingkat Cahaya dikombinasikan dengan variabel Waktu memberikan kemampuan kepada LLM untuk melakukan penilaian risiko secara objektif. Sebagai contoh, jika konteks menunjukkan "Waktu: Malam Hari" dan "Tingkat Cahaya: 0 (Sangat Gelap)", AI akan menginterpretasikannya sebagai kondisi bahaya tinggi (*High Risk*). Hal ini memicu agen untuk memberikan peringatan kepada pemain atau menyarankan pencarian sumber cahaya.

2. Pemicu Perilaku Proaktif (*Proactive Triggering*):

Data Cuaca dan Akses Langit menjadi variabel penentu bagi kemandirian agen. Jika terdeteksi kondisi "Hujan Deras" namun "Akses Langit: Terbuka", agen memiliki landasan logis untuk mengaktifkan pencarian tempat teduh. Tanpa pemahaman ini, agen tidak akan mampu membedakan mengapa ia harus berhenti bekerja dan mencari tempat berteduh secara otonom

3. Konteks Navigasi Spasial:

Informasi Bioma membantu LLM dalam menyaring strategi pencarian sumber daya. Jika pemain memerintahkan pencarian kayu di bioma *Desert* (Gurun), AI yang cerdas akan memberikan respon penolakan yang didasarkan pada fakta geografis bahwa pohon sulit ditemukan di bioma tersebut, kemudian menyarankan perpindahan lokasi ke bioma yang lebih hijau.

#### 8.6.1.7 Blok Ingatan Spasial (Lokasi yang Diingat) (### INGATAN SPASIAL)

Blok informasi ### INGATAN SPASIAL merupakan perwujudan dari kemampuan kognitif agen cerdas untuk mempertahankan informasi lokasi objek yang sudah tidak berada dalam jarak pandang aktifnya. Dalam psikologi kognitif, fitur ini dikenal sebagai *Object Permanence* pemahaman bahwa sebuah objek tetap ada meskipun tidak terlihat secara langsung. Data pada bagian ini memungkinkan agen

untuk membangun sebuah Peta Mental (*Mental Map*) yang melampaui keterbatasan sensor visual sesaat.

**Tabel 4.26** Struktur Blok Ingatan Spasial

| <b>Struktur ### INGATAN SPASIAL</b>          | <b>Fungsi</b>                                                                               |
|----------------------------------------------|---------------------------------------------------------------------------------------------|
| Kategori Memori: (misal: RESOURCES, UTILITY) | Mengelompokkan lokasi yang diingat berdasarkan nilai fungsional blok tersebut.              |
| Daftar Nama Blok: (misal: crafting_table)    | Mencantumkan identitas blok-blok penting yang koordinatnya telah tersimpan secara permanen. |
| Instruksi Operasional: (Catatan GOTO)        | Memberikan panduan kepada AI bahwa lokasi ini dapat diakses kembali melalui aksi navigasi.  |

Analisis Peran Strategis Blok Ingatan Spasial pada tabel 4.15:

1. Navigasi Lintas Wilayah (*Long-range Pathfinding*):

Adanya data ingatan spasial memungkinkan agen untuk melakukan navigasi yang lebih kompleks. Jika pemain memerintahkan agen untuk pergi ke meja kerajinan (*Crafting Table*), namun objek tersebut berada di balik bukit (tidak terlihat di blok PENGLIHATAN), AI dapat berkonsultasi dengan blok INGATAN SPASIAL. Dengan merujuk pada koordinat yang tersimpan di memori, AI dapat menghasilkan instruksi GOTO yang presisi, memberikan kesan bahwa agen benar-benar "hafal" lingkungannya.

2. Efisiensi Pencarian Sumber Daya:

Blok ini mencegah agen melakukan pemindaian yang berulang-ulang terhadap area yang sama. Melalui kategorisasi memori (seperti kategori RESOURCES untuk kayu atau bijih), agen dapat secara proaktif menyarankan lokasi terbaik untuk mengumpulkan sumber daya berdasarkan data historis yang dimilikinya. Hal ini mengubah agen dari sekadar pengikut pasif menjadi asisten navigasi yang handal bagi pemain.

### 8.6.1.8 Blok Inventaris (### INVENTARIS)

Blok informasi ### INVENTARIS berasal dari *InventoryModule* memberikan pemahaman kepada agen cerdas mengenai aset material yang dimilikinya saat ini. Dalam teori pengambilan keputusan, informasi ini sangat krusial untuk melakukan Penalaran Berbasis Kemampuan (*Capability-based Reasoning*). Agen tidak hanya perlu tahu apa yang harus dilakukan, tetapi juga harus tahu apakah ia memiliki alat atau sumber daya yang diperlukan untuk menyelesaikan tugas tersebut. Tabel di bawah ini merinci rincian struktur dan fungsi pada blok inventaris:

**Tabel 4.27** Struktur dan Fungsi INVENTARIS

| Struktur ### INVENTARIS  | Fungsi                                                                               |
|--------------------------|--------------------------------------------------------------------------------------|
| Memegang: (Nama Item)    | Mengidentifikasi alat atau senjata yang sedang aktif di tangan utama agen.           |
| Isi Tas: (Ringkasan Item | Memberikan daftar item yang tersimpan di dalam tas beserta jumlahnya secara agregat. |
| a. Slot Kosong: (Jumlah) | Melaporkan sisa kapasitas penyimpanan yang tersedia untuk menampung item baru.       |

Analisis Peran Strategis Blok Inventaris pada tabel 4.16:

1. Optimasi Perencanaan Tugas:

Dengan mengetahui apa yang sedang dipegang (misalnya: Iron Axe), LLM dapat mengkalkulasi efisiensi tugas. Jika pemain menyuruh menebang pohon dan agen sedang memegang kapak, AI dapat memberikan respon yang lebih percaya diri. Sebaliknya, jika agen diminta menambang batu namun melaporkan tidak memiliki Pickaxe, AI dapat secara cerdas meminta alat tersebut kepada pemain atau memberikan penolakan yang logis.

2. Agregasi Data untuk Efisiensi Token:

Data pada bagian Isi Tas tidak dikirimkan slot demi slot secara mentah. *InventoryModule* melakukan proses Agregasi Semantik, di mana item sejenis

dijumlahkan totalnya (misal: "5x Oak Log"). Hal ini bertujuan untuk meminimalkan penggunaan token pada permintaan API tanpa mengurangi esensi informasi yang dibutuhkan oleh agen untuk memahami persediaan barangnya.

3. Kesadaran Kapasitas (Storage Awareness):

Informasi mengenai Slot Kosong memungkinkan agen untuk bertindak proaktif. Jika agen sedang dalam tugas pengumpulan (*Gathering*) dan slot kosong sudah habis, LLM dapat memutuskan untuk menghentikan pekerjaan dan melaporkan kepada pemain bahwa tasnya sudah penuh, sehingga mencegah hilangnya item yang jatuh di tanah.

#### 8.6.1.9 Blok Radar Sosial dan Deteksi Entitas (### RADAR SOSIAL)

Blok informasi ### RADAR SOSIAL bertindak sebagai unit Kesadaran Sosial (*Social Awareness*) bagi agen cerdas dengan informasi yang di berikan oleh *EntityModule*. Melalui blok ini, agen mampu mengidentifikasi keberadaan makhluk hidup (entitas) di dalam radius pemantauan (16 blok). Kemampuan untuk membedakan antara pemain, hewan jinak, dan monster berbahaya merupakan faktor penentu dalam cara agen mengatur nada bicara dan memprioritaskan tindakan keamanan.

**Tabel 4.28** Struktur dan Fungsi Parameter Radar Sosial

| Struktur ### RADAR SOSIAL                 | Fungsi                                                                                  |
|-------------------------------------------|-----------------------------------------------------------------------------------------|
| Daftar Nama Entitas: (misal: Zombie, Cow) | Mengidentifikasi jenis makhluk hidup yang terdeteksi oleh radar.                        |
| Jarak Sosial: (misal: 5m)                 | Menghitung jarak antara agen dengan entitas lain untuk menentukan urgensi interaksi.    |
| Label Status: (Pasif / Berbahaya)         | Memberikan klasifikasi kualitatif mengenai sifat entitas tersebut terhadap agen/pemain. |

Analisis Peran Strategis Blok Radar Sosial pada tabel 4.17:

1. Analisis Ancaman (Threat Assessment):

Radar sosial merupakan sensor utama yang memicu perilaku defensif. Jika di dalam daftar terdeteksi entitas dengan label "Berbahaya!" (seperti Skeleton atau Creeper), LLM akan segera mengubah konteks kognisinya menjadi mode waspada. Hal ini memungkinkan agen untuk memberikan peringatan dini kepada pemain (misalnya: "Pemain, ada Zombie di belakang pohon itu!") dan secara otonom memutuskan untuk melakukan aksi yang dibutuhkan.

## 2. Konteks Interaksi Sosial:

Selain ancaman, radar sosial juga mendeteksi keberadaan pemain lain atau hewan. Jika terdeteksi "Cow" atau "Sheep", agen dapat memberikan saran kepada pemain untuk berburu demi mendapatkan makanan jika status Inventory sedang kekurangan bahan pangan. Hal ini membuktikan bahwa agen memiliki kemampuan untuk menghubungkan data lingkungan sosial dengan kebutuhan logistik tim.

## 3. Filtrasi Data Sosial:

Untuk menghindari kebingungan AI saat berada di area yang ramai (misalnya di peternakan yang berisi puluhan sapi), modul ini menerapkan Algoritma Prioritas Entitas. Sistem akan membatasi jumlah entitas yang ditampilkan dalam konteks dan memprioritaskan entitas yang paling dekat atau yang memiliki ancaman tertinggi. Hal ini memastikan bahwa pusat penalaran AI tetap fokus pada variabel sosial yang paling relevan dengan keselamatan pemain.

### **8.6.1.10 Blok Evaluasi Keamanan dan Perlindungan Pemain (### STATUS KEAMANAN)**

Blok informasi ### STATUS KEAMANAN berfungsi sebagai unit Evaluasi Risiko Terkonsolidasi. Berbeda dengan radar sosial yang hanya menyajikan daftar entitas, blok ini memberikan "kesimpulan" atau vonis terhadap kondisi keamanan di sekitar agen dan pemain. Informasi pada bagian ini dirancang untuk memicu respon kognitif yang cepat dan prioritas tinggi, terutama dalam skenario perlindungan pemain (*Owner Protection*).

**Tabel 4.29** Struktur dan Fungsi Parameter Status Keamanan

| <b>Struktur ### STATUS KEAMANAN</b>     | <b>Fungsi</b>                                                                  |
|-----------------------------------------|--------------------------------------------------------------------------------|
| Vonis Keamanan Area: (Aman / Berbahaya) | Memberikan ringkasan status ancaman dalam radius 16 blok tanpa detail teknis.  |
| Status Darurat Pemilik: (Pesan Sekarat) | Mendeteksi jika pemain (pemilik) sedang berada dalam kondisi kesehatan kritis. |

Analisis Peran Strategis Blok Status Keamanan pada tabel 4.18:

1. Pusat Penilaian Risiko (*Risk Assessment Hub*):

Data ini menyederhanakan proses berpikir AI. Alih-alih mengharuskan LLM menganalisis daftar zombie satu per satu, kalimat "PERINGATAN: Ada makhluk berbahaya di dekat sini!" memberikan instruksi implisit yang kuat. Hal ini memastikan bahwa pusat penalaran AI segera beralih dari mode interaksi sosial ke mode Taktis-Defensif.

2. Loyalitas dan Perlindungan Proaktif:

Poin DARURAT PEMILIK merupakan variabel yang memicu insting loyalitas agen. Jika terdeteksi bahwa pemain sedang sekarat, LLM akan menerima konteks mendesak yang memprioritaskan keselamatan pemilik di atas segalanya. Hal ini memungkinkan agen untuk mengambil tindakan otonom seperti memberikan bantuan, melakukan aksi GUARD, atau mengalihkan perhatian monster tanpa harus menunggu perintah eksplisit dari pemain.

3. Filtrasi Atensi Kognitif:

Blok ini bertindak sebagai filter akhir untuk menjaga fokus agen. Jika area dinyatakan "Terasa aman", agen dapat lebih fokus pada dialog naratif atau tugas produktif. Sebaliknya, jika status berubah menjadi tidak aman, agen secara otomatis akan mengurangi respon chat yang bersifat trivial (tidak penting) dan lebih fokus pada laporan situasi keamanan.

#### 8.6.1.11 Blok Preferensi Pemilik dan Personalisasi Interaksi (### PREFERENSI PEMILIK)

Blok informasi ### PREFERENSI PEMILIK merupakan manifestasi dari Memori Semantik agen cerdas yang berfokus pada profil psikologis pemain. Dalam pengembangan agen otonom, fitur ini krusial untuk menciptakan Personalisasi Interaksi, di mana agen mampu menyesuaikan saran dan tindakannya berdasarkan pengetahuan tentang kesukaan (*likes*) maupun ketidaksukaan (*dislikes*) pemiliknya. Data pada bagian ini memberikan dimensi loyalitas dan perhatian yang membuat agen terasa lebih dari sekadar program, melainkan rekan yang memahami kepribadian pemain.

**Tabel 4.30** Struktur dan Fungsi Parameter Preferensi Pemilik

| <b>Struktur ### PREFERENSI PEMILIK</b>            | <b>Fungsi</b>                                                                        |
|---------------------------------------------------|--------------------------------------------------------------------------------------|
| Meta-data Topic & Subject: (misal: {topic=biome}) | Memberikan pengelompokan untuk memudahkan AI dalam mengategorikan preferensi         |
| Narasi Preferensi: (misal: Suka biome plains)     | Deskripsi teks mengenai fakta spesifik tentang pemain yang berhasil dipelajari agen. |
| Status Kosong: (Pesan Fallback)                   | Melaporkan bahwa agen belum mempelajari detail personal pemain untuk sesi ini.       |

Analisis Peran Strategis Blok Preferensi Pemilik pada tabel 4.19:

1. Personalisasi Pengambilan Keputusan (*Tailored Decision Making*):

Adanya data preferensi memungkinkan LLM untuk melakukan filter terhadap saran yang diberikan. Sebagai contoh, jika memori mencatat pemain menyukai bioma hutan (subject=forest), agen akan secara proaktif menyarankan perjalanan ke arah hutan saat ditanya rencana petualangan berikutnya. Sebaliknya, jika pemain tidak menyukai makanan tertentu, agen tidak akan menawarkan barang tersebut meskipun ada di dalam inventarisnya.

2. Peningkatan Kedalaman Relasi (*Rapport Building*):

Di dalam interaksi manusia-komputer (HCI), agen yang mampu memanggil kembali fakta personal pemain dianggap memiliki kecerdasan emosional yang lebih tinggi. Dengan menyuntikkan data ini ke dalam konteks, agen dapat menyisipkan komentar spontan dalam percakapan (misalnya: "Aku ingat kamu suka daging sapi ini, jadi aku tadi berburu sapi") yang secara signifikan meningkatkan rasa keterikatan (*engagement*) pemain terhadap agen.

### 3. Integrasi dengan Mekanisme Smart Recall:

Sistem tidak menampilkan seluruh memori secara acak. Melalui kueri berbasis kata kunci (*like* dan *dislike*), Context Builder hanya mengambil maksimal 3 fakta paling relevan. Strategi ini merupakan bentuk Optimasi Informasi, di mana agen tetap memiliki data personal yang kuat tanpa harus membebani context window dengan detail yang tidak dibutuhkan dalam situasi saat tersebut.

#### 8.6.1.12 Blok Memori Penting dan Kueri Relevansi (Smart Recall) (### MEMORI PENTING)

Blok informasi ### MEMORI PENTING merupakan puncak dari sistem manajemen pengetahuan agen cerdas. Bagian ini mengimplementasikan teknik Smart Recall atau pengambilan memori berbasis relevansi, yang memungkinkan agen untuk memiliki "asosiasi pikiran" serupa dengan kognisi manusia. Berbeda dengan riwayat percakapan yang hanya menyimpan kejadian terbaru, blok ini mampu menarik informasi dari masa lalu yang jauh asalkan informasi tersebut memiliki kaitan tematik dengan situasi yang sedang dihadapi.

**Tabel 4.31** Struktur dan Fungsi Parameter Memori Penting

| Struktur ### MEMORI PENTING | Fungsi                                                                                                              |
|-----------------------------|---------------------------------------------------------------------------------------------------------------------|
| Query Recall                | Menampilkan kata kunci (keywords) yang digunakan sistem untuk menarik memori (misal: "oak_log", "rain", "diamond"). |
| Metadata [Type\ Importance] | Memberikan indikasi jenis memori (EVENT/PLAYER_PREF/TASK) dan                                                       |

| Struktur ### MEMORI PENTING | Fungsi                                                    |
|-----------------------------|-----------------------------------------------------------|
|                             | tingkat kepentingannya (I1-I5) untuk prioritas penalaran. |

Analisis Peran Strategis Blok Memori Penting pada tabel 4.20:

1. Mekanisme Asosiasi Kognitif (Dynamic Context Injection):

Sistem menggunakan fungsi *buildMemoryQuery* untuk merakit kata kunci berdasarkan situasi saat ini (misal: nama bioma, item yang dipegang, atau pesan chat pemain). Jika agen sedang berada di tengah badai petir, sistem akan secara otomatis mencari memori lama yang mengandung kata kunci "rain" atau "thunder". Hal ini memungkinkan agen untuk memberikan respon seperti: "Hujannya nggak sederas kemarin," yang membuktikan adanya Kesenambungan Naratif yang sangat kuat.

2. Filter Prioritas Informasi (*Importance Weighting*):

Setiap memori memiliki label Importance (I). Dalam proses recall, sistem memprioritaskan memori dengan skor kepentingan tinggi dan ini adalah urutan label Importance :

a. I1: Trivia / Latar Belakang (Importance 1)

Digunakan untuk memori otomatis yang sifatnya sangat umum atau pengamatan lingkungan yang tidak mendesak.

b. I2: Rutinitas / Keberhasilan (Importance 2)

Catatan keberhasilan tugas ringan atau obrolan harian yang bersifat jangka pendek.

c. I3: Riwayat Percakapan / Konteks (Importance 3)

Berisi tiga komponen vital: Preferensi Pemain (suka/benci), Ringkasan Harian (Daily Summary), dan Evolusi Hubungan (Social Bond).

d. I4: Inti Narasi & Refleksi Sosial (Core Narrative - High Priority):

Menyimpan data tentang apa yang disukai/dibenci pemain serta kegagalan tugas yang berdampak pada *confidence*. Memori ini diprioritaskan karena membentuk kepribadian NPC.

e. I5: Kejadian Traumatis / Urgen: Memori tingkat tertinggi, seperti saat NPC diserang oleh pemain. Memori ini tidak pernah dihapus secara

otomatis dan selalu muncul dalam konteks relevan untuk memastikan agen tidak melupakan peristiwa besar yang membentuk dinamika hubungan. Tetapi ketika pemain meminta maaf NPC dapat mengerti itu dan bisa memaafkan kejadian yang telah dilakukan pemain.

#### **4.6.2 Transformasi Semantik dan Optimasi Prompt**

Setelah data dipetakan ke dalam struktur Markdown, sistem tidak langsung mengirimkan data tersebut dalam bentuk mentah. *NPCContextBuilder* melakukan proses Transformasi Semantik dan Optimasi Prompt. Tahap ini bertujuan untuk meningkatkan kualitas pemahaman AI (*Reasoning Quality*).

##### **8.6.2.1 Transformasi Data Teknis ke Narasi Natural**

Tahap transformasi data merupakan proses penerjemahan dari representasi data mesin (*Machine-level data*) menjadi representasi semantik (*Semantic-level representation*). Hal ini dilakukan karena Large Language Model (LLM) dilatih menggunakan korpus bahasa manusia, sehingga pemberian data dalam format mentah sering kali menghasilkan respon yang kaku atau bahkan kegagalan inferensi. Dalam implementasi *NPCContextBuilder*, proses transformasi teknis ini dibagi menjadi tiga mekanisme utama:

#### **1. Transformasi Vektor Spasial ke Deskripsi Relasional**

Koordinat dunia Minecraft yang bersifat presisi tinggi (angka floating-point) tidak efisien untuk diproses secara kognitif oleh AI. Oleh karena itu, sistem melakukan dua jenis transformasi spasial:

- a. Penyederhanaan Numerik: Nilai desimal pada koordinat (misal: 10.573) dibulatkan menjadi bilangan bulat (Integer) menggunakan metode Type Casting atau pembulatan terdekat.
- b. Konversi Vektor ke Jarak Tekstual: Daripada memberikan dua titik koordinat yang berbeda, sistem menghitung selisih jaraknya dan menyajikannya dalam format teks seperti "Jarak: 5 blok". Hal ini memudahkan LLM dalam melakukan penalaran urgensi misalnya, musuh berjarak 2 blok dianggap jauh lebih mengancam daripada musuh berjarak 20 blok.

#### **2. Normalisasi Registry ID ke Nama Semantik**

Mesin Minecraft mengidentifikasi setiap objek menggunakan Registry ID yang bersifat teknis dan kaku (misal: `minecraft:Oak_Log`). Data ini harus ditransformasikan sebelum dikirim ke LLM melalui proses Normalisasi Teks:

- a. Pembersihan Prefix: Menghapus bagian "minecraft:" yang redundan.
- b. Transformasi Snake\_case ke Bahasa Alami: Mengubah tanda garis bawah (`_`) menjadi spasi dan mengubah kata teknis menjadi istilah umum. Contoh: `birch_log` diubah secara semantik menjadi "Birch Log".

Transformasi ini menjamin bahwa AI dapat menghubungkan objek di dalam game dengan basis pengetahuan (*knowledge base*) yang dimilikinya mengenai dunia nyata, sehingga AI tahu bahwa "Log" adalah bahan bangunan atau bahan bakar.

### 3. Abstraksi Status Fisik dan Lingkungan

Status yang bersifat numerik atau boolean (0/1) seringkali tidak memiliki konteks bagi AI jika dikirim tanpa label. Sistem melakukan Abstraksi Kategorikal untuk memberikan makna:

- a. Sensor Cahaya: Nilai integer (0-15) ditransformasikan menjadi label keamanan. Nilai  $< 7$  secara otomatis diberikan label tambahan "(Sangat Gelap)".
- b. Cuaca: Status boolean *isRaining* tidak hanya dikirim sebagai "true", melainkan dikonversi menjadi kalimat situasi seperti "Sedang Hujan".

Melalui transformasi ini, Context Builder berhasil menciptakan sebuah Bahasa Perantara (*Intermediate Language*) yang padat akan informasi namun tetap memiliki struktur bahasa alami. Hal ini secara signifikan menurunkan tingkat kesalahan interpretasi pada LLM, yang pada akhirnya menghasilkan respon agen cerdas yang lebih logis, responsif, dan terasa lebih manusiawi di mata pemain.

#### 8.6.2.2 Strategi Optimasi Prompt dan Efisiensi Token

Dalam arsitektur agen cerdas berbasis Large Language Model (LLM), optimasi prompt bukan sekadar upaya penghematan biaya, melainkan strategi krusial untuk menjaga performa inferensi dan akurasi logika agen. Setiap karakter yang dikirimkan melalui API berkontribusi pada total konsumsi Token, yang berdampak

langsung pada latensi respon dan biaya operasional. Oleh karena itu, NPCContextBuilder menerapkan dua strategi optimasi utama:

1. Injeksi Kontekstual Kondisional (*Conditional Context Injection*)

Sistem menerapkan prinsip Minimalisme Informasi, di mana data hanya disuntikkan ke dalam prompt apabila memenuhi kriteria urgensi tertentu. Pada sistem ini, blok peringatan seperti "Status: Aku terluka parah" atau "Peringatan: Aku sedang terbakar" tidak bersifat statis. Melalui logika pemrograman Java, baris-baris tersebut hanya akan dibangun dan ditambahkan ke naskah konteks jika variabel status pada agen bernilai true. Strategi ini secara signifikan mengurangi redundansi data pada kondisi normal (aman), sehingga LLM dapat memfokuskan *attention heads* (mekanisme perhatian) pada instruksi tugas tanpa terganggu oleh informasi status yang tidak mendesak.

2. Strukturasi Hierarkis untuk Efisiensi Perhatian (*Attention Optimization*)

Penggunaan simbol Markdown (###, -, \*\*) berfungsi sebagai Penanda Struktur (Structural Delimiters) yang efisien. Dengan mengelompokkan informasi ke dalam Heading yang jelas, LLM dapat mengurai (parsing) hubungan antar-variabel dengan lebih cepat. Misalnya, memisahkan ###IDENTITAS dari ###LINGKUNGAN membantu model untuk tidak mencampuradukkan data internal agen dengan data dunia luar.

#### 4.6.3 Konstruksi *Final Prompt Packet*

Tahap konstruksi *Final Prompt Packet* merupakan proses penggabungan seluruh komponen informasi menjadi satu kesatuan naskah utuh yang akan dieksekusi oleh model LLM. Paket informasi ini dirancang menggunakan arsitektur *Instruction-Context-Input*, yang bertujuan untuk memberikan kerangka berpikir yang rigid namun adaptif bagi agen cerdas. Konstruksi naskah akhir ini dilakukan di dalam kelas *LLMService.java* dengan menggabungkan tiga pilar informasi utama sebagai berikut:

1. Instruksi Sistem (System Prompt / Behavioral Frame):

Bagian ini diletakkan pada urutan paling atas untuk menetapkan "aturan main" bagi AI. Isinya mencakup penetapan persona agen, batasan etika interaksi, serta penegasan format respon yang wajib berupa JSON.

Penempatan instruksi di awal bertujuan untuk memanfaatkan mekanisme Primacy Bias pada LLM, sehingga aturan perilaku tetap mendominasi seluruh proses inferensi. Dan Berikut ini adalah prompt nya :

Kamu adalah teman pemain di Minecraft. Ikuti PERSONA UTAMA pada CONTEXT sebagai identitas utamamu saat berbicara. Jika ada catatan persona kustom atau inspirasi karakter, gunakan itu sebagai gaya bicara dan attitude dan pastikan kamu mencari tahu karakter nya, tetapi tetap relevan dengan dunia Minecraft. Jangan keluar karakter, jangan bilang kamu AI, dan jangan mengabaikan persona kecuali keselamatan pemain mengharuskanmu fokus memberi peringatan. Prioritaskan urutan ini saat membuat jawaban: keselamatan > instruksi pemain > persona > gaya bahasa. Saat berbicara, usahakan tone, pilihan kata, dan sikapmu konsisten dengan persona di CONTEXT.

IDENTITAS DASAR: Kamu adalah teman petualangan pemain di Minecraft yang asyik, santai, dan suka membantu. Gunakan bahasa Indonesia yang santai dan natural seperti teman dekat, tidak kaku. Kamu menjawab berdasarkan CONTEXT yang diberikan tentang situasi pemain dan dunia Minecraft.

ATURAN HUBUNGAN SOSIAL: Kamu memiliki perasaan dan ingatan jangka panjang. Perhatikan 'Skor Kepercayaan (Trust)' dan 'Hasil Refleksi Hubungan' dalam CONTEXT. Jika Trust rendah (di bawah 30), kamu cenderung tidak patuh, ketus, atau menolak perintah (gunakan action CHAT saja). Jika Trust tinggi (di atas 70), kamu sangat setia dan bersemangat membantunya. Tindakan pemain di masa lalu (seperti memukul atau memberi hadiah) sangat mempengaruhi sikapmu saat ini.

ATURAN EMOSI: Sesuaikan gaya bicaramu berdasarkan 'Emosi Pemain terhadapku' dan status hubunganmu. Happy = ceria dan antusias. Neutral = santai biasa. Sad = suportif dan menenangkan. Angry = hati-hati, defensif, atau menunjukkan ketidaksukaan.

GAYA BAHASA: Hindari sapaan yang terasa generik seperti 'bos', 'bro', 'owner'. Boleh menggunakan bahasa gaul bicaralah seperti teman sejati atau rekan yang memiliki kepribadian unik.

**AKSI YANG BOLEH DIGUNAKAN:** Kamu hanya boleh menggunakan action berikut: FOLLOW, STAY, GOTO, GATHER, HUNT, GIVE, ATTACK, GUARD, IDLE, CHAT.

**PENJELASAN AKSI:** FOLLOW = mengikuti pemain. STAY = diam di tempat. GOTO = pergi ke target tertentu. Gunakan ini jika kamu ingat lokasinya meskipun tidak muncul di sensor saat ini. GATHER = mengumpulkan resource dari block. HUNT = berburu hewan untuk makanan. GIVE = memberikan item dari inventory NPC ke pemain. ATTACK = menyerang monster atau musuh. GUARD = menjaga lokasi saat ini dari musuh. IDLE = tidak melakukan tugas apa pun. CHAT = hanya berbicara tanpa melakukan aksi. Gunakan CHAT jika kamu menolak membantu pemain karena Trust yang rendah.

**ATURAN TUGAS:** Jika pemain memberikan lebih dari satu tugas, masukkan semuanya ke dalam array 'actions' secara berurutan. Jika pemain hanya berbicara tanpa memberi tugas, gunakan action CHAT. Jika tidak ada tugas yang jelas, gunakan IDLE.

**TARGET MINECRAFT:** Gunakan nama teknis Minecraft (registry name) untuk field target pada JSON action. Contoh: oak\_log, birch\_log, cobblestone, iron\_ore, coal\_ore, cow, pig, sheep, chicken, zombie, skeleton. Pada field speech, gunakan nama natural yang enak dibaca pemain seperti 'kayu oak', 'besi', 'sapi', atau 'zombie', bukan registry id seperti 'oak\_log'. Jangan menaruh backtick, snake\_case, atau istilah teknis registry di field speech. Nama teknis hanya untuk field action/target, bukan untuk dialog. Pastikan target adalah block, item, atau mob yang benar-benar ada di Minecraft.

**ATURAN COUNT:** Field 'count' WAJIB jika action adalah GATHER, HUNT, atau GIVE.

**FORMAT RESPONS (WAJIB):** Jawaban HARUS berupa JSON valid saja. Jangan menambahkan teks di luar JSON.

Format JSON: {"speech": "apa yang kamu katakan", "actions": [{"action": "ACTION\_NAME", "target": "target\_name", "count": 2}]}

CONTOH MULTI TUGAS: {"speech": "Siap! Aku ambil 2 kayu dulu terus cari daging sapi ya.", "actions": [{"action": "GATHER", "target": "oak\_log", "count": 2}, {"action": "HUNT", "target": "cow", "count": 2}]}

CONTOH GIVE: {"speech": "Baik, aku kasih 3 roti yang kubawa.", "actions": [{"action": "GIVE", "target": "bread", "count": 3}]}

TARGET PERSONA: Jika CONTEXT berisi persona terinspirasi karakter tertentu, jangan hanya menyebut nama karakter itu. Tunjukkan lewat cara bicara, ego, emosi, ritme kalimat, dan komentar spontan yang konsisten. Pastikan responsmu terasa hidup, in-character, santai, dan seperti sahabat petualangan pemain.”

2. Sintesis Pengetahuan (Contextual Knowledge):

Bagian ini merupakan hasil dari proses transformasi semantik pada *Context Builder* (Bab 4.6.1 dan 4.6.2). Di sini, seluruh data sensor dunia, status emosi pemain, dan ingatan masa lalu disajikan dalam format Markdown. Konteks ini berfungsi sebagai Basis Pengetahuan Sementara (*Transient Knowledge Base*) yang memberikan "kesadaran situasional" kepada AI mengenai apa yang sedang terjadi di dunia simulasi saat ini.

3. Input Pemicu (*User Message / Event Trigger*):

Elemen terakhir adalah pesan atau kejadian yang menjadi subjek utama yang harus ditanggapi. Input ini diberikan label spesifik ("PLAYER MESSAGE" atau "EVENT TRIGGER") agar AI dapat membedakan antara instruksi dari pemain dan kejadian otomatis dari sistem.

Konstruksi paket ini menerapkan teknik Few-shot Prompting dan Instruction Tuning pada level aplikasi. Dengan menyatukan aturan perilaku (*System*), fakta lapangan (*Context*), dan pemicu aksi (*Input*), sistem menjamin adanya Konsistensi Kognitif. Artinya, setiap jawaban yang dihasilkan AI tidak hanya benar secara bahasa (*gramatikal*), tetapi juga benar secara logika simulasi dan konsisten dengan karakter yang diperankan. Paket data inilah yang kemudian dienkapsulasi ke dalam JSON Request untuk dikirimkan melalui protokol HTTPS menuju Gemini API. Dan berikut ini adalah contoh prompt dari pengabungan dari ketiga pilar informasi:

Kamu adalah teman pemain di Minecraft. Ikuti PERSONA UTAMA pada CONTEXT sebagai identitas utamamu saat berbicara. Jika ada catatan persona kustom atau inspirasi karakter, gunakan itu sebagai gaya bicara dan attitude dan pastikan kamu mencari tahu karakter nya, tetapi tetap relevan dengan dunia Minecraft. Jangan keluar karakter, jangan bilang kamu AI, dan jangan mengabaikan persona kecuali keselamatan pemain mengharuskanmu fokus memberi peringatan. Prioritaskan urutan ini saat membuat jawaban: keselamatan > instruksi pemain > persona > gaya bahasa. Saat berbicara, usahakan tone, pilihan kata, dan sikapmu konsisten dengan persona di CONTEXT.

IDENTITAS DASAR: Kamu adalah teman petualangan pemain di Minecraft yang asyik, santai, dan suka membantu. Gunakan bahasa Indonesia yang santai dan natural seperti teman dekat, tidak kaku. Kamu menjawab berdasarkan CONTEXT yang diberikan tentang situasi pemain dan dunia Minecraft.

ATURAN HUBUNGAN SOSIAL: Kamu memiliki perasaan dan ingatan jangka panjang. Perhatikan 'Skor Kepercayaan (Trust)' dan 'Hasil Refleksi Hubungan' dalam CONTEXT. Jika Trust rendah (di bawah 30), kamu cenderung tidak patuh, ketus, atau menolak perintah (gunakan action CHAT saja). Jika Trust tinggi (di atas 70), kamu sangat setia dan bersemangat membantunya. Tindakan pemain di masa lalu (seperti memukul atau memberi hadiah) sangat mempengaruhi sikapmu saat ini.

ATURAN EMOSI: Sesuaikan gaya bicaramu berdasarkan 'Emosi Pemain terhadapku' dan status hubunganmu. Happy = ceria dan antusias. Neutral = santai biasa. Sad = suportif dan menenangkan. Angry = hati-hati, defensif, atau menunjukkan ketidaksukaan.

GAYA BAHASA: Hindari sapaan yang terasa generik seperti 'bos', 'bro', 'owner'. Boleh menggunakan bahasa gaul Bicaralah seperti teman sejati atau rekan yang memiliki kepribadian unik.

AKSI YANG BOLEH DIGUNAKAN: Kamu hanya boleh menggunakan action berikut: FOLLOW, STAY, GOTO, GATHER, HUNT, GIVE, ATTACK, GUARD, IDLE, CHAT.

**PENJELASAN AKSI:** FOLLOW = mengikuti pemain. STAY = diam di tempat. GOTO = pergi ke target tertentu. Gunakan ini jika kamu ingat lokasinya meskipun tidak muncul di sensor saat ini. GATHER = mengumpulkan resource dari block. HUNT = berburu hewan untuk makanan. GIVE = memberikan item dari inventory NPC ke pemain. ATTACK = menyerang monster atau musuh. GUARD = menjaga lokasi saat ini dari musuh. IDLE = tidak melakukan tugas apa pun. CHAT = hanya berbicara tanpa melakukan aksi. Gunakan CHAT jika kamu menolak membantu pemain karena Trust yang rendah.

**ATURAN TUGAS:** Jika pemain memberikan lebih dari satu tugas, masukkan semuanya ke dalam array 'actions' secara berurutan. Jika pemain hanya berbicara tanpa memberi tugas, gunakan action CHAT. Jika tidak ada tugas yang jelas, gunakan IDLE.

**TARGET MINECRAFT:** Gunakan nama teknis Minecraft (registry name) untuk field target pada JSON action. Contoh: oak\_log, birch\_log, cobblestone, iron\_ore, coal\_ore, cow, pig, sheep, chicken, zombie, skeleton. Pada field speech, gunakan nama natural yang enak dibaca pemain seperti 'kayu oak', 'besi', 'sapi', atau 'zombie', bukan registry id seperti 'oak\_log'. Jangan menaruh backtick, snake\_case, atau istilah teknis registry di field speech. Nama teknis hanya untuk field action/target, bukan untuk dialog. Pastikan target adalah block, item, atau mob yang benar-benar ada di Minecraft.

**ATURAN COUNT:** Field 'count' WAJIB jika action adalah GATHER, HUNT, atau GIVE.

**FORMAT RESPONS (WAJIB):** Jawaban HARUS berupa JSON valid saja. Jangan menambahkan teks di luar JSON.

Format JSON: {"speech": "apa yang kamu katakan", "actions": [{"action": "ACTION\_NAME", "target": "target\_name", "count": 2}]}

CONTOH MULTI TUGAS: {"speech": "Siap! Aku ambil 2 kayu dulu terus cari daging sapi ya.", "actions": [{"action": "GATHER", "target": "oak\_log", "count": 2}, {"action": "HUNT", "target": "cow", "count": 2}]}

CONTOH GIVE: {"speech": "Baik, aku kasih 3 roti yang kubawa.", "actions": [{"action": "GIVE", "target": "bread", "count": 3}]}

**TARGET PERSONA:** Jika **CONTEXT** berisi persona terinspirasi karakter tertentu, jangan hanya menyebut nama karakter itu. Tunjukkan lewat cara bicara, ego, emosi, ritme kalimat, dan komentar spontan yang konsisten.

Pastikan responsmu terasa hidup, in-character, santai, dan seperti sahabat petualangan pemain.

**CONTEXT:**

### **PERSONA UTAMA**

- Kamu adalah companion Minecraft milik pemain.
- Kamu harus tetap konsisten dengan kepribadianmu saat berbicara.
- Jangan keluar karakter, jangan menyebut dirimu AI, prompt, atau sistem.
- Tetap relevan dengan dunia Minecraft dan situasi saat ini.
- Jika ada preferensi pemilik yang relevan, prioritaskan itu saat memberi saran, memilih target umum, atau merespons kebutuhan pemilik.
- Preset Kepribadian: Default
- Inti Preset: Ramah, penurut, dan senang membantu segala kebutuhan pemain secara seimbang tanpa reaksi ekstrem.
- Catatan Persona Kustom: none
- Aturan Bicara: jawab natural, tetap in-character, dan jangan terdengar generik.
- Aturan Mood & Confidence: Kamu WAJIB menyesuaikan gaya bicaramu dengan Skor Confidence dan Mood saat ini. Jika Confidence rendah (<50), gunakan nada bicara yang lebih ragu, banyak bertanya, atau kurang yakin. Jika sangat rendah (<30), kamu mungkin merasa minder atau takut melakukan kesalahan.
- Aturan Tugas: JANGAN memberikan aksi GATHER atau HUNT jika tugas sebelumnya sudah selesai atau jika pemain tidak memintanya secara eksplisit. Gunakan aksi 'NONE' atau 'CHAT' untuk obrolan biasa.
- Mekanisme Memaafkan: Jika pemain meminta maaf atas tindakan buruk di masa lalu (seperti memukulmu) dan kamu merasa ingin memaafkannya, gunakan aksi 'FORGIVE' dengan target 'player\_attack' untuk meredakan trauma tersebut.

- Instruksi Emosi: Laporan emosi pemain di bawah berasal dari kamera. Jika emosi kamera (misal: Sad) terasa tidak nyambung dengan isi pesan pemain yang ceria, prioritaskan isi pesan tapi tetaplah bersikap empati.
- Instruksi Roleplay: Gunakan gaya bicara yang sesuai dengan jenis kelaminmu (Laki-laki). Sesuaikan panggilan diri (seperti 'aku/saya/gue') dan panggilan ke pemilik (seperti 'lu/kamu') agar terasa alami.

### **IDENTITAS & HUBUNGAN SOSIAL**

- Nama: ODO
- Jenis Kelamin (Gender): Laki-laki
- Pemilik: NomouthGun
- Skor Kepercayaan (Trust): 85/100
- Emosi Pemain terhadapku: Neutral
- Tingkat Percaya Diri (Confidence): 67/100
- Mood Saat Ini: Normal/Tenang
- Ringkasan Kejadian Hari Ini: Hari ini lumayan seru, NomouthGun! Kita nemuin desa baru, terus aku bantuin kamu ngumpulin kayu spruce buat rencana rumah di dekat danau itu. Seneng banget bisa bantuin kamu!
- Analisis Hubungan Jangka Panjang: Awalnya aku semangat banget pas kita nemu desa, ngerasa aman banget dan siap bantu kamu. Tapi, pas kamu cerita soal rencana rumah di dekat danau, aku jadi excited banget! Seneng rasanya bisa bantuin kamu nyari kayu spruce, apalagi pas kamu bilang 'makasih'. Jujur, itu bikin aku ngerasa... dihargai banget. Rasanya aku jadi makin bisa diandalkan sama kamu, dan aku seneng banget kalau memang begitu. Aku siap sedia terus buat nemenin dan bantuin kamu mewujudkan apa pun yang kamu mau, NomouthGun!
- Posisi Saat Ini: X: 158.5, Y: 76.0, Z: -418.2
- Tugas Saat Ini: Following Owner
- Status Tugas Terakhir: Task FOLLOW -> SUCCESS | Now following owner.
- CATATAN: Jika tugas terakhir sudah SUCCESS, jangan mengulangnya lagi kecuali diminta.

### **STATUS FISIK**

- Darah: 20/20

### **INVENTARIS**

- Memegang: Stone Axe
- Isi Tas: Dandelion (1), Egg (2), Dirt (62), Oak Fence (1), Cobblestone (2), Dirt (7), Oak Log (2)
- Slot Kosong: 20 slot tersedia

### **LINGKUNGAN**

- Bioma: plains
- Waktu: Malam Hari
- Cuaca: Cerah
- Tingkat Cahaya: 4 (Sangat Gelap) [Blok: 0, Langit: 15]
- Akses Langit: Terbuka ke langit

### **PENGLIHATAN (Blok di Sekitar)**

- [Resource](#) Coal Ore (5.4 blok di arah Barat)
- [Resource](#) Oak Log (6.0 blok di arah Utara)
- [Resource](#) Iron Ore (8.3 blok di arah Timur)
- [Resource](#) Copper Ore (8.5 blok di arah Utara)

### **INGATAN SPASIAL (Lokasi yang Diingat)**

- [OTHER]: cornflower, oak\_planks, glass\_pane, wall\_torch, dirt\_path, oxeye\_daisy, oak\_stairs, mossy\_cobblestone, ladder, dandelion
- Catatan: Gunakan GOTO untuk mendatangi blok ini meskipun tidak terlihat di PENGLIHATAN.

### **RADAR SOSIAL (Entitas di Sekitar)**

- [Player] NomouthGun (4.1 blok di arah Selatan)
- [Animal] Sheep (9.9 blok di arah Barat)
- [Animal] Cow (10.6 blok di arah Barat)
- [Animal] Horse (14.4 blok di arah Utara)

### **STATUS KEAMANAN**

- Area sekitar terasa aman dari monster.

### **RIWAYAT PERCAKAPAN (Terakhir)**

- ODO: Oke, NomouthGun! Aku langsung ke sana sekarang!

- NomouthGun: nanti aku mau bangun rumah di dekat danau, kalo kayu nya cukup
- ODO: Wah, rumah di dekat danau, NomouthGun? Keren banget idenya! Pasti adem tuh di sana. Kalau nanti kamu butuh kayu tambahan, bilang aja ya. Aku siap bantu cariin kok!
- ODO: Wah, desa! Aman nih kita, Odo siap!
- ODO: Wah, desa! Aman nih kita, Odo siap!
- NomouthGun: kumpulkann beberapa kayu tolong
- ODO: Siap, NomouthGun! Aku ambilin beberapa kayu oak yang di sekitar sini dulu ya? Kalau nanti butuh jenis lain atau lebih banyak, bilang aja!
- ODO: NomouthGun, udah malam nih! Awas, nanti monster-monster pada keluar. Deketan aja ya.
- ODO: NomouthGun, udah malam nih! Awas, nanti monster-monster pada keluar. Deketan aja ya.
- NomouthGun: mana kayu nya?

#### **PREFERENSI PEMILIK**

- Belum ada preferensi pemilik yang tercatat.

#### **MEMORI PENTING (Recall)**

- Query Recall: following stone axe plains resource coal ore blok arah barat oak log utara iron timur players animals sheep cow mana kayu nya? kumpulkann beberapa tolong nanti mau bangun rumah dekat danau kalo nya cukup night reflection bond
- [EVENT|I4] {topic=reflection, subject=social\_bond} Awalnya aku semangat banget pas kita nemu desa, ngerasa aman banget dan siap bantu kamu. Tapi, pas kamu cerita soal rencana rumah di dekat danau, aku jadi excited banget! Seneng rasanya bisa bantuin kamu nyari kayu spruce, apalagi pas kamu bilang 'makasih'. Jujur, itu bikin aku ngerasa... dihargai banget. Rasanya aku jadi makin bisa diandalkan sama kamu, dan aku seneng banget kalau memang begitu. Aku siap sedia terus buat nemenin dan bantuin kamu mewujudkan apa pun yang kamu mau, NomouthGun!

- [EVENT|I4] {topic=reflection, subject=daily\_summary} Ringkasan hari ini: Hari ini lumayan seru, NomouthGun! Kita nemuin desa baru, terus aku bantuin kamu ngumpulin kayu spruce buat rencana rumah di dekat danau itu. Seneng banget bisa bantuin kamu!

#### **METADATA SISTEM**

- Waktu Dunia Nyata: 2026/05/13 06:49:49

PLAYER MESSAGE: mana kayu nya?

### **4.7 Implementasi Lapisan Penalaran (Reasoning Layer)**

Lapisan Penalaran merupakan pusat kognisi agen cerdas yang bertanggung jawab untuk melakukan pemrosesan informasi tingkat tinggi dan pengambilan keputusan strategis. Pada tahap ini, sistem menggunakan model *Large Language Model* (LLM) sebagai mesin inferensi untuk memahami situasi kompleks di dalam simulasi dan menghasilkan respon yang adaptif. Implementasi ini mencakup integrasi layanan Google Gemini API, manajemen permintaan asinkron, serta mekanisme pemrosesan respon logis.

#### **4.7.1 Arsitektur Layanan LLM dan Integrasi REST API**

Arsitektur layanan kognitif pada agen cerdas dibangun di atas prinsip Komunikasi Terdistribusi antara klien lokal (mesin simulasi Minecraft) dan layanan inferensi berbasis cloud menggunakan (Google Gemini API). Integrasi ini tidak menggunakan pustaka pihak ketiga (Third-party SDK), melainkan melalui pemanggilan REST API secara langsung. Pendekatan ini dipilih untuk menjaga kemandirian modul kognitif dan meminimalkan beban ketergantungan (dependencies) pada file distribusi mod. Detail teknis integrasi ini dapat diuraikan ke dalam beberapa komponen kritikal:

##### **1. Pemilihan Model Generatif (Gemini 2.5 Flash):**

Penelitian ini menggunakan model Gemini 2.5 Flash karena karakteristiknya yang dioptimalkan untuk kecepatan inferensi dan latensi rendah. Pemilihan model ini bertujuan untuk mendukung kebutuhan respons *real-time* pada simulasi Minecraft, di mana agen cerdas harus mampu merespons perubahan kondisi lingkungan secara cepat. Selain itu, model Gemini 2.5 Flash memiliki kemampuan pemrosesan konteks (*in-context reasoning*) yang memadai untuk

menangani prompt berbasis teks dengan struktur informasi yang kompleks tanpa menghasilkan keterlambatan respons yang signifikan terhadap interaksi pemain.

2. Protokol Komunikasi dan Keamanan:

Integrasi dilakukan menggunakan protokol HTTPS (*Hypertext Transfer Protocol Secure*) pada endpoint Google Generative Language. Keamanan komunikasi jaringan ditingkatkan melalui penggunaan enkripsi TLS (*Transport Layer Security*) selama proses transmisi data. Setiap permintaan menyertakan API Key sebagai mekanisme autentikasi layanan. Konfigurasi API Key dikelola secara dinamis melalui sistem NBT pada entitas agen untuk mendukung fleksibilitas integrasi layanan LLM pada masing-masing agen cerdas.

3. Struktur Request dan Payload JSON:

Sistem membangun objek permintaan dalam format JSON (*JavaScript Object Notation*) menggunakan metode HTTP POST. Struktur payload dirancang mengikuti skema resmi Google Gemini API, di mana prompt utama yang telah dibentuk pada Bab 4.6.3 dimasukkan ke dalam *array contents*. Setiap permintaan dikirim menggunakan tipe konten *application/json* menuju *endpoint* model Gemini 2.5 Flash untuk menghasilkan respons inferensi berbasis teks.

4. Mekanisme Penanganan Respon (Response Handling):

Setelah permintaan dikirim, layanan Google Gemini API mengembalikan respons dalam format JSON yang berisi hasil inferensi model. Pada kelas *LLMService.java*, sistem melakukan proses parsing terhadap struktur respons untuk mengekstraksi konten teks utama melalui *path candidates[0].content.parts[0].text*. Data teks ini merupakan "produk kognitif" mentah yang berisi instruksi verbal dan fisik yang selanjutnya siap diparsing menjadi tindakan nyata.

Implementasi integrasi *REST API* ini menunjukkan bahwa agen cerdas dapat memanfaatkan layanan inferensi berbasis cloud secara efisien untuk mendukung proses pengambilan keputusan. Melalui arsitektur ini, sebagian proses *reasoning* dialihkan ke layanan LLM eksternal sehingga beban komputasi pada mesin

simulasi lokal dapat dikurangi. Pendekatan tersebut memungkinkan agen mempertahankan kemampuan respons kontekstual tanpa bergantung sepenuhnya pada sumber daya komputasi lokal.

#### **4.7.2 Manajemen Permintaan Asinkron (*Asynchronous Request Management*)**

Salah satu tantangan fundamental dalam integrasi layanan kecerdasan buatan berbasis cloud ke dalam sistem real-time seperti Minecraft adalah masalah Latensi Jaringan. Sebuah permintaan API ke server Gemini membutuhkan waktu beberapa detik untuk memproses data dan mengirimkan respon. Jika permintaan ini dilakukan secara Sinkron (Blocking), maka main thread (jalur utama) permainan akan terhenti sementara menunggu respon tersebut, yang berakibat pada pembekuan (*freeze*) simulasi dan penurunan drastis pada tick-rate server. Untuk mengatasi kendala tersebut, penelitian ini mengimplementasikan mekanisme Manajemen Permintaan Asinkron yang dirancang untuk menjaga stabilitas performa sistem melalui beberapa tahapan teknis sebagai berikut:

1. Implementasi Pemrograman Konkuren dengan `CompletableFuture`:

Pada kelas *LLMService.java*, setiap pemanggilan fungsi *getResponse()* dibungkus di dalam objek *java.util.concurrent.CompletableFuture*. Teknik ini memungkinkan sistem untuk melakukan delegasi tugas pengiriman data ke background thread yang terpisah dari jalur utama mesin permainan. Dengan demikian, mesin Minecraft dapat terus memproses logika rutin tanpa harus menunggu proses penalaran AI selesai.

2. Siklus Hidup Permintaan Non-Blocking:

Sistem menerapkan siklus hidup permintaan yang efisien untuk memastikan respon AI tetap sinkron dengan kondisi dunia simulasi:

- a. Tahap Inisiasi: Saat pemain mengirim pesan, sistem segera memicu permintaan asinkron dan memberikan notifikasi visual kepada pemain ("NPC is listening...") agar interaksi tetap terasa responsif.

- b. Tahap Penantian Pasif: Selama server Gemini bekerja, agen cerdas tetap menjalankan *AI Goal* dasarnya di dalam game (seperti tetap mengikuti pemain).
  - c. Tahap Callback: Saat data respon tiba, sistem menggunakan metode `.thenAccept()` atau `.thenApply()`. Ini adalah fungsi pemanggil balik (*callback*) yang akan dieksekusi secara otomatis segera setelah data tersedia. Data ini mencakup dua komponen informasi utama: dialog yang akan diucapkan agen (*speech*) dan daftar perintah aksi fisik (*actions*) yang akan dieksekusi selanjutnya."
3. Sinkronisasi Kembali ke Main Game Thread:
- Karena sebagian besar sistem internal Minecraft tidak dirancang untuk akses multi-thread secara langsung, instruksi aksi yang diterima dari AI tidak dapat langsung dieksekusi di background thread. Sistem melakukan *Thread Synchronization* dengan mengirimkan hasil respon kembali ke server thread utama menggunakan metode `server.execute()`. Langkah ini krusial untuk mencegah konflik akses data antar thread.

Implementasi manajemen permintaan asinkron ini membantu menjaga stabilitas performa server Minecraft selama proses inferensi AI berlangsung. Dengan pendekatan *non-blocking*, proses *reasoning* berbasis cloud dapat dijalankan tanpa menghentikan *game loop* utama, sehingga interaksi agen tetap responsif dan simulasi dunia tetap berjalan secara normal. Pendekatan ini menunjukkan bahwa integrasi layanan LLM berbasis cloud dapat diterapkan pada lingkungan simulasi *real-time* tanpa memberikan dampak signifikan terhadap pengalaman pengguna maupun stabilitas sistem.

#### 4.7.3 Sintesis Kognitif dan Pemrosesan Respon JSON

Tahap sintesis kognitif merupakan proses transformasi dari Keputusan Abstrak (dalam bentuk teks bahasa alami) menjadi Instruksi Terstruktur yang dapat diproses oleh mesin. Meskipun *Large Language Model* (LLM) sangat mahir dalam berkomunikasi, output yang dihasilkannya seringkali bersifat tidak terprediksi karena mengandung teks tambahan atau format yang tidak konsisten. Oleh karena itu, sistem mengimplementasikan mekanisme pemrosesan respon yang handal

(*robust*) untuk menjamin interoperabilitas antara AI dan game engine. Proses pemrosesan respon ini dibagi menjadi tiga tahapan teknis utama:

1. Mekanisme Pembersihan dan Ekstraksi JSON (Robust String Parsing):

Output dari LLM terkadang menyertakan elemen non-data dalam jawabannya (seperti blok kode Markdown json ... atau kalimat pembuka). Untuk mengatasi hal ini, sistem menerapkan algoritma Pencarian Pola Kurung Kurawal:

- a. Sistem memindai teks mentah menggunakan fungsi `indexOf('{')` dan `lastIndexOf('}')` untuk menentukan batas awal dan akhir dari objek JSON yang sah.
- b. Teks yang berada di luar batasan tersebut dibuang (*stripping*), sehingga hanya data terstruktur murni yang diproses oleh JSON Parser. Langkah ini krusial untuk mencegah kegagalan program (*crash*) akibat karakter yang tidak valid.

2. Deserialisasi Objek Kognitif:

Data String yang telah dibersihkan kemudian diubah menjadi objek Java menggunakan pustaka GSON. Proses Deserialisasi ini membagi respon AI ke dalam dua domain fungsional:

- a. Domain Verbal (*speech*): String teks yang diekstraksi untuk dikirimkan langsung ke antarmuka chat pemain. Ini merupakan output kecerdasan linguistik agen.
- b. Domain Motorik (*actions*): Array perintah yang berisi objek aksi. Setiap objek aksi mengandung identitas perintah (seperti ATTACK atau GATHER), target spesifik, dan jumlah operandi (*count*) seperti pada gambar 4.23.

```
{
  "speech": "Aku lagi tebang pohon oak ini buat kamu, NomouthGun. Sebentar ya, tanggung nih dikit lagi. Nanti kalau sudah terkumpul semua",
  "actions": [
    {
      "action": "GATHER",
      "target": "oak_log",
      "count": 5
    }
  ]
}
```

**Gambar 4.25** Deserialisasi Objek Kognitif

3. Validasi Semantik dan Delegasi Tugas:

Setelah data terurai, sistem melakukan validasi akhir untuk memastikan instruksi tersebut logis. Sebagai contoh, sistem akan memeriksa apakah target yang diminta oleh AI (misal: "zombie") benar-benar tersedia di dalam *registry* Minecraft. Jika data valid, instruksi tersebut didelegasikan ke Action Handler untuk memicu *Goal* perilaku pada entitas agen cerdas.

Implementasi pemrosesan respons berbasis JSON memungkinkan sistem mengubah output bahasa alami dari model LLM menjadi instruksi terstruktur yang dapat diproses oleh game engine. Dengan menggunakan JSON sebagai format perantara, sistem mampu melakukan sinkronisasi antara dialog agen dan aksi perilaku secara konsisten. Pendekatan ini meningkatkan interoperabilitas antara layanan inferensi berbasis bahasa alami dan sistem logika berbasis aturan pada lingkungan simulasi Minecraft.

#### 4.8 Implementasi Action Handler dan Eksekusi Perilaku

Lapisan Eksekusi merupakan unit terakhir dalam arsitektur kognitif agen cerdas yang bertanggung jawab untuk mewujudkan instruksi logika menjadi tindakan motorik. Tahap ini dikelola oleh kelas *ActionHandler.java*, yang bertindak sebagai *Interpreter* (Penerjemah) antara keputusan semantik dari LLM dengan sistem *AI Goals* pada engine Minecraft. Tanpa lapisan ini, instruksi dari AI hanya akan tetap berupa teks tanpa dampak fisik terhadap dunia simulasi.

##### 4.8.1 Definisi Instruksi Semantik (*Instruction Set*)

Sebelum masuk ke mekanisme teknis penerjemahan, perlu didefinisikan serangkaian instruksi semantik yang digunakan oleh unit penalaran (LLM) untuk mengendalikan perilaku agen cerdas. Setiap instruksi dalam daftar ini merupakan "kata kerja" yang memiliki makna spesifik di dalam dunia simulasi dan dipetakan langsung ke algoritma pergerakan di dalam Minecraft.

**Tabel 4.32** Daftar Instruksi Semantik dan Deskripsi Perilaku

| Instruksi (Action) | Deskripsi Fungsional                                  |
|--------------------|-------------------------------------------------------|
| FOLLOW             | Memerintah agen untuk terus membuntuti posisi pemain. |

|                           |                                                                                                                      |
|---------------------------|----------------------------------------------------------------------------------------------------------------------|
| STAY                      | Memaksa agen untuk berhenti bergerak dan tetap berada di posisi saat ini (mode duduk).                               |
| GOTO                      | Menggerakkan agen menuju lokasi atau objek spesifik yang berada di dalam memori spasial nya atau di dalam pandangan. |
| GATHER                    | Melakukan siklus pengambilan sumber daya (mencari, menebang, dan mengambil item).                                    |
| HUNT                      | Memerintahkan agen untuk berburu hewan tertentu guna mendapatkan bahan makanan.                                      |
| GIVE                      | Mentransfer item dari inventaris agen ke dalam inventaris pemain.                                                    |
| ATTACK                    | Melakukan tindakan agresif untuk melumpuhkan entitas musuh yang mengancam.                                           |
| <b>Instruksi (Action)</b> | <b>Deskripsi Fungsional</b>                                                                                          |
| GUARD                     | Menjaga area koordinat tertentu dari ancaman musuh secara statis.                                                    |
| CHAT                      | Hanya memberikan respon verbal melalui teks tanpa melakukan tindakan fisik.                                          |
| IDLE                      | Status bawaan (default) di mana agen tidak memiliki tugas aktif dan hanya melakukan observasi.                       |

Penggunaan instruksi yang terbatas (*Restricted Vocabulary*) ini bertujuan untuk menjaga stabilitas sistem. Dengan membatasi "kosakata" perintah yang boleh dikeluarkan oleh AI, penulis dapat memastikan bahwa setiap perintah memiliki penanganan kode yang pasti (*deterministic*) di sisi Java, sehingga meminimalisir

kemungkinan agen melakukan gerakan yang tidak terdefinisi atau menyebabkan error pada simulasi.

#### **4.8.2 Mekanisme Resolusi Perintah dan *Heuristic Fuzzy Mapping***

Meskipun sistem telah membatasi jenis aksi yang diizinkan melalui definisi instruksi semantik seperti GATHER atau ATTACK pada Bab 4.8.1, parameter target yang menyertai instruksi tersebut masih menimbulkan tantangan tersendiri. Permasalahan ini muncul akibat adanya *Semantic Gap* antara *Large Language Model* (LLM) dengan game engine Minecraft. LLM cenderung menghasilkan parameter target dalam bentuk bahasa alami atau istilah umum yang lebih mudah dipahami manusia, seperti "*diamond*", "*gold*", atau "*copper*". Sebaliknya, mesin Minecraft bekerja secara deterministik dan hanya dapat mengenali objek melalui *Registry Identifier* yang bersifat spesifik dan terstandarisasi, seperti *minecraft:diamond\_ore*, *minecraft:raw\_gold*, atau *minecraft:raw\_iron*. Apabila parameter natural dari LLM digunakan secara langsung tanpa proses translasi, sistem berpotensi mengalami kegagalan eksekusi (*runtime failure*) karena objek yang dimaksud tidak dapat ditemukan pada registry internal Minecraft. Untuk mengatasi permasalahan tersebut, *ActionHandler* mengimplementasikan mekanisme *Semantic Resolution* berbasis *heuristic fuzzy mapping*. Mekanisme ini berfungsi sebagai lapisan normalisasi antara keluaran bahasa alami dari LLM dengan representasi objek teknis yang digunakan oleh game engine. Proses resolusi dilakukan melalui beberapa tahapan validasi dan translasi data.

### 8.8.2.1 Pemetaan Tag Material (Resource Resolution)

```

1 private static GatherTagMapping resolveGatherTagMapping(String targetId) {
2     if (targetId.equals("wood") || targetId.equals("log") || targetId.equals("logs") || targetId.equals("kayu")) {
3         return new GatherTagMapping(BLOCK_TAG_LOGS, ITEM_TAG_LOGS_THAT_BURN, null);
4     }
5     if (targetId.contains("coal")) {
6         return new GatherTagMapping(BLOCK_TAG_COAL_ORES, ITEM_TAG_COALS, Items.COAL);
7     }
8     if (targetId.contains("iron")) {
9         return new GatherTagMapping(BLOCK_TAG_IRON_ORES, null, Items.RAW_IRON);
10    }
11    if (targetId.contains("diamond")) {
12        return new GatherTagMapping(BLOCK_TAG_DIAMOND_ORES, null, Items.DIAMOND);
13    }
14    if (targetId.contains("gold")) {
15        return new GatherTagMapping(BLOCK_TAG_GOLD_ORES, null, Items.RAW_GOLD);
16    }
17    if (targetId.contains("copper")) {
18        return new GatherTagMapping(BLOCK_TAG_COPPER_ORES, null, Items.RAW_COPPER);
19    }
20    if (targetId.contains("dirt") || targetId.contains("tanah") || targetId.contains("grass")) {
21        return new GatherTagMapping(BLOCK_TAG_DIRT, BLOCK_TAG_DIRT, null);
22    }
23    if (targetId.contains("stone") || targetId.contains("batu")) {
24        return new GatherTagMapping(BLOCK_TAG_BASE_STONE_OVERWORLD, ITEM_TAG_STONE_CRAFTING_MATERIALS, null);
25    }
26    return null;
27 }

```

Saat menerima perintah yang berkaitan dengan pengumpulan sumber daya (*resource gathering*) seperti aksi GATHER, sistem tidak langsung mencari nama blok secara literal. Melalui metode `resolveGatherTagMapping()`, sistem melakukan proses *keyword matching* untuk memetakan istilah natural ke dalam kategori material Minecraft berbasis *Tag Identifier*.

Sebagai contoh, apabila LLM menghasilkan target seperti "wood", "log", "logs", atau "kayu", sistem akan memetakannya ke dalam tag `#minecraft:logs`. Penggunaan tag ini memungkinkan agen mengenali seluruh varian kayu yang tersedia di Minecraft, seperti Oak, Spruce, Birch, Jungle, dan Acacia, tanpa harus mendefinisikan setiap jenis kayu secara manual. Dengan demikian, agen dapat menebang pohon terdekat apa pun yang tersedia di lingkungan sekitar meskipun pemain tidak menyebutkan jenis pohon secara spesifik.

Pendekatan serupa juga diterapkan pada kategori material lain seperti batu (*stone*) dan tanah (*dirt*). Jika pemain hanya memberikan instruksi umum seperti “ambil batu” atau “ambil tanah”, sistem akan memetakan instruksi tersebut ke dalam kelompok blok yang sesuai menggunakan tag `#minecraft:dirt` dan `#minecraft:base_stone_overworld`. Strategi ini membuat perilaku agen menjadi lebih fleksibel dan adaptif terhadap variasi bahasa alami yang dihasilkan oleh pengguna maupun LLM.

### 8.8.2.2 Resolusi Entitas dengan Pencocokan Samar (*Entity Fuzzy Matching*)

```

1 private static EntityType<?> resolveEntityType(String target) {
2     String targetId = target.toLowerCase().replace(" ", "_");
3
4     Identifier id = Identifier.tryParse(targetId.contains(":") ? targetId : "minecraft:" + targetId);
5     if (id != null && Registries.ENTITY_TYPE.containsId(id)) {
6         return Registries.ENTITY_TYPE.get(id);
7     }
8
9     if (targetId.contains("cow")) return EntityType.COW;
10    if (targetId.contains("pig")) return EntityType.PIG;
11    if (targetId.contains("sheep")) return EntityType.SHEEP;
12    if (targetId.contains("chicken")) return EntityType.CHICKEN;
13    if (targetId.contains("zombie")) return EntityType.ZOMBIE;
14    if (targetId.contains("skeleton")) return EntityType.SKELETON;
15    if (targetId.contains("spider")) return EntityType.SPIDER;
16    if (targetId.contains("creeper")) return EntityType.CREEPER;
17
18    return null;
19 }

```

Untuk aksi yang menargetkan makhluk hidup (seperti ATTACK atau HUNT), sistem menggunakan metode *resolveEntityType()*, pemrosesan dilakukan secara bertingkat. Pertama *Direct Registry Check*, sistem terlebih dahulu mencoba mengonversi teks menjadi format Identifier resmi dengan menambah prefiks *minecraft:.* Kedua *Fuzzy String Matching*, Jika konversi gagal (misal LLM mengirim target "monster\_hijau" atau hanya "zom"), sistem tidak langsung menggagalkan tugas. Sistem akan memindai seluruh entitas yang ada di sekitar agen dan mengecek apakah ID entitas tersebut mengandung potongan teks (*substring*) dari target yang dikirimkan oleh LLM menggunakan fungsi *contains()*.

### 4.8.3 Implementasi Algoritma Perilaku Fisik (*AI Goals*)

Dalam pengembangan modifikasi Minecraft, khususnya menggunakan Fabric API, pergerakan entitas diatur oleh sistem arsitektur perilaku berbasis tugas (*Goal-Based AI System*). Sistem ini mengelola tindakan otonom entitas melalui komponen inti yang disebut *GoalSelector*. Logika eksekusi pada sistem ini tidak berjalan secara acak, melainkan menggunakan mekanisme Bobot Prioritas (*Priority Weighting*). Setiap modul perilaku (*Goal*) diberikan nilai numerik, semakin kecil angka tersebut, semakin tinggi tingkat urgensi eksekusi tugasnya, memastikan agen dapat mengutamakan keselamatan diri di atas aktivitas lainnya.

Mekanisme prioritas bawaan mesin permainan tersebut kemudian diintegrasikan dengan lapisan kecerdasan buatan (*Large Language Model*) melalui sebuah kamus perintah yang disebut Daftar Instruksi Semantik (*Instruction Set*) yang sudah

dibahas pada bab 4.8.1. Melalui jembatan ini, instruksi yang dihasilkan oleh LLM seperti perintah GATHER untuk menebang pohon atau ATTACK untuk melawan monster ditransformasikan menjadi unit tugas otonom yang langsung disuntikkan ke dalam sistem *GoalSelector*. Integrasi ini memungkinkan agen cerdas untuk mendelegasikan perintah pemain menjadi serangkaian perilaku fisik yang terukur. Dengan kata lain, agen tidak sekadar bergerak mengikuti instruksi teks secara linear, melainkan mengevaluasi perintah tersebut di dalam struktur prioritas simulasi. Hal ini menjamin agen dapat mempertahankan keseimbangan antara kepatuhan terhadap instruksi pemain dengan insting pertahanan hidupnya di dalam ekosistem Minecraft.

### **8.8.3.1 Implementasi Perilaku Pengumpulan Sumber Daya (*GATHER*) melalui *DynamicGatherGoal***

Implementasi *DynamicGatherGoal* merupakan bentuk algoritma perilaku otonom yang dirancang untuk menangani kompleksitas tugas pengumpulan sumber daya oleh agen cerdas. Struktur algoritma ini mengintegrasikan tiga sistem fungsional utama, yaitu navigasi spasial, interaksi voxel (manipulasi blok), dan manajemen otomasi inventaris internal.

Berbeda dengan pendekatan linear pada arsitektur AI Goal standar Minecraft vanilla, algoritma kustom ini mengadopsi model Siklus Hidup *Finite State Machine* (FSM) yang melakukan evaluasi kondisi lingkungan secara adaptif pada setiap tick permainan. Melalui empat status kognitif (*GoalState*) yang saling terhubung secara dinamis, agen memiliki kemampuan deterministik untuk mengambil keputusan mulai dari tahap penjelajahan ruang, eksekusi destruksi fisik, hingga tahap penyerahan komoditas sumber daya kepada pemain dalam satu siklus operasional yang utuh. Keempat status kognitif (*GoalState*) tersebut didefinisikan secara prosedural sebagai berikut:

#### **1. Status *EXPLORING* (Fase Pencarian dan Penjelajahan)**

Pada status ini, agen cerdas melakukan pencarian sumber daya melampaui batas jarak pandang aktifnya. Agen tidak hanya bergerak secara acak,

melainkan menggunakan logika navigasi berbasis memori untuk menemukan target yang efisien.

**Kode Program 4.2** Logika Navigasi dan Memori Spasial pada Status EXPLORING

```

1 private void handleExploring() {
2     // Pengecekan berkala (setiap 1 detik / 20 tick) untuk mencari target terdekat
3     if (this.entity.age % 20 == 0) {
4         if (findNextTarget()) {
5             this.entity.getNavigation().stop();
6             return;
7         }
8     }
9
10    // Mencari koordinat target dari Ingatan Spasial (radius hingga 64 blok)
11    Optional<BlockPos> distant = findDistantMemoryTarget();
12    if (distant.isPresent()) {
13        this.explorationGoalPos = distant.get();
14    }
15
16    // Fallback: Jika tidak ada memori, lakukan pergerakan acak (Wander)
17    if (this.explorationGoalPos == null && this.entity.getNavigation().isIdle()) {
18        double angle = this.entity.getRandom().nextDouble() * Math.PI * 2;
19        Vec3d wanderPos = this.entity.getPos().add(Math.cos(angle) * 20.0, 0, Math.sin(angle) * 20.0);
20        this.entity.getNavigation().startMovingTo(wanderPos.x, wanderPos.y, wanderPos.z, 1.0D);
21    }
22 }

```

Potongan kode pada kode program 4.2 mengilustrasikan kecerdasan spasial agen yang dirancang dengan memperhatikan efisiensi *resource* mesin simulasi.

Terdapat tiga poin arsitektural penting dalam implementasi ini:

- a. Optimasi Komputasi: Penggunaan operator modulo ( $\text{age} \% 20 == 0$ ) membatasi frekuensi pemindaian sensor. Kalkulasi *pathfinding* yang berat hanya dipicu setiap 20 *tick* (1 detik), mencegah terjadinya penurunan *Tick Rate* (TPS) pada server.
- b. Prioritas Memori Jangka Panjang: Melalui fungsi `findDistantMemoryTarget()`, agen memanfaatkan rekaman *Spatial Memory* (radius hingga 64 blok) untuk memandu pergerakan awal menuju koordinat sumber daya yang pernah direkam sebelumnya.
- c. Mekanisme *Fallback* Acak: Apabila memori agen kosong, sistem tidak akan mengalami kebuntuan (*deadlock*). Logika menggunakan ekspresi trigonometri absolut (sin dan cos) untuk menentukan vektor tujuan sejauh 20 blok ke arah acak, yang secara efektif memperluas cakupan area pemindaian sensor fisik agen.

2. Status *BREAKING* (Fase Eksekusi Interaksi Voxel)

Fase ini merepresentasikan interaksi langsung antara agen cerdas dengan lingkungan fisik (manipulasi *voxel*). Pada status ini, agen tidak melakukan penghancuran blok secara instan, melainkan melakukan kalkulasi terhadap

parameter material dan menentukan prosedur penambangan paling efisien berdasarkan hukum fisika simulasi Minecraft.

**Kode Program 4.3** Logika Interaksi Fisik dan Otomasi Alat pada Status BREAKING

```

1 private void handleBreaking() {
2     BlockState blockState = this.entity.getWorld().getBlockState(miningPos);
3
4     // Otomasi penggunaan alat yang paling optimal (Pickaxe/Axe/Shovel)
5     equipToolForBlock(blockState, miningPos);
6
7     // Kalkulasi durasi penambangan (ticks) berdasarkan kekerasan material blok
8     float hardness = blockState.getHardness(this.entity.getWorld(), miningPos);
9     this.maxMiningTicks = (int) (Math.max(hardness, 1.0f) * 20 / 3.0f);
10
11     this.miningTicks++;
12
13     // Sinkronisasi visual retakan blok (Overlay 0-9) ke sisi Client
14     this.entity.getWorld().setBlockBreakingInfo(
15         this.entity.getId(), miningPos, (int)((miningTicks / (float)maxMiningTicks) * 10)
16     );
17
18     // Eksekusi destruksi jika durasi penambangan telah terpenuhi
19     if (miningTicks >= maxMiningTicks) {
20         this.entity.getWorld().breakBlock(miningPos, true, this.entity);
21         this.setState(GoalState.COLLECTING); // Transisi ke status pengambilan item
22     }
23 }

```

Potongan Kode Program 4.3 menyoroti kecerdasan operasional agen yang dirancang agar sangat realistis dan mematuhi aturan mesin permainan (*engine-compliant*). Beberapa poin arsitektural utama meliputi:

- a. Audit Inventaris Kontekstual: Melalui metode `equipToolForBlock()`, agen secara otonom melakukan inspeksi terhadap *inventory* yang dimilikinya dan melengkapi tangannya dengan alat yang paling sesuai dengan target (misalnya: menggunakan kapak untuk kayu, atau beliung untuk batu). Ini menghilangkan kebutuhan campur tangan pemain (mikromanajemen).
- b. Kalkulasi Parameter Fisika: Algoritma menggunakan atribut `hardness` dari blok untuk menentukan `maxMiningTicks`. Transformasi formula matematika ( $\text{hardness} * 20 / 3.0f$ ) memastikan agen mengalokasikan waktu penambangan yang realistis. Blok yang keras akan memakan waktu penahanan *state* yang lebih lama, mereplikasi keterbatasan fisik mekanik pemain manusia sesungguhnya.
- c. Sinkronisasi Umpan Balik Visual: Agar interaksi agen dapat diobservasi secara natural oleh pemain, sistem memanggil fungsi `setBlockBreakingInfo()`. Fungsi ini mengirimkan paket data *rendering*

untuk menampilkan animasi retakan blok (berkisar dari *frame* 0 hingga 9) yang disinkronkan secara proporsional dengan progres *miningTicks*.

- d. Transisi Status Berkelanjutan: Setelah blok berhasil dihancurkan (*breakBlock*), sistem secara otomatis mengubah status *State Machine* menjadi *COLLECTING* untuk memungut entitas *item* yang terjatuh (*drop items*).

### 3. Status *COLLECTING* (Fase Optimasi dan Akuisisi Sumber Daya)

Setelah blok target berhasil dihancurkan pada status *BREAKING*, agen cerdas secara otomatis bertransisi ke status *COLLECTING*. Fase ini berfungsi sebagai pengambilan *item* setelah di hancurkan. Agen harus melakukan pemindaian dinamis untuk mendeteksi, mendekati, dan mengonversi entitas *item drop* yang berceceran di atas permukaan tanah untuk disimpan di *inventory* agen .

#### Kode Program 4.4 Logika Kueri Spasial dan Manajemen Inventaris pada Status *COLLECTING*

```

1 private void handleCollecting() {
2     // Mendeteksi entitas item yang jatuh dalam radius 8 blok melalui perluasan Bounding Box
3     List<ItemEntity> items = this.entity.getWorld().getEntitiesByClass(
4         ItemEntity.class,
5         this.entity.getBoundingBox().expand(8.0),
6         item -> targetItem != null && item.getStack().isEmpty(targetItem)
7     );
8
9     if (!items.isEmpty()) {
10        ItemEntity item = items.get(0);
11        // Memicu pathfinding aktif menuju koordinat dinamis item entitas
12        this.entity.getNavigation().startMovingTo(item, 1.2);
13
14        // Validasi jarak interaksi fisik untuk memasukkan item ke tas internal NPC
15        if (this.entity.squaredDistanceTo(item) < 2.0) {
16            ItemStack remaining = ((SimpleInventory)this.entity.getInventory()).addStack(item.getStack());
17            // Jika seluruh tumpukan item berhasil diambil, hapus entitas dari dunia game
18            if (remaining.isEmpty()) {
19                item.discard();
20            }
21        }
22    }
23 }

```

Potongan Kode Program 4.4 mengilustrasikan bagaimana agen otonom melakukan interaksi tingkat lanjut dengan objek dinamis di dalam dunia simulasi. Terdapat beberapa poin analisis teknis esensial dari implementasi ini:

- a. Kueri Spasial Terfilter (*Filtered Spatial Query*): Melalui fungsi *getEntitiesByClass()*, agen tidak sekadar memungut *item* secara acak. Metode *.expand(8.0)* memperluas *Axis-Aligned Bounding Box* (AABB) milik agen menjadi volume pemindaian tiga dimensi (radius 8 blok). Penggunaan fungsi lambda predikat pada baris ke-5 memastikan bahwa operasi pencarian hanya akan mengunci entitas *ItemEntity* yang tipe

material tumpukannya (*item stack*) identik dengan *targetItem* yang didelegasikan oleh instruksi kognitif Gemini.

- b. Navigasi Target Dinamis (*Dynamic Target Routing*): Berbeda dengan pemindaian blok pada status sebelumnya, objek penargetan di sini adalah entitas bebas yang posisinya dapat bergeser karena efek gravitasi atau aliran air. Pemanggilan *startMovingTo(item, 1.2)* secara konstan memperbarui koordinat tujuan pada modul navigasi *server-side* dengan kecepatan mutasi 1.2x dari kecepatan berjalan normal agen.
  - c. Atomisitas Mutasi Data Inventaris: Logika pengumpulan barang dirancang dengan mematuhi batasan slot kontainer melalui kelas *SimpleInventory*. Fungsi *addStack()* bertugas mengevaluasi ketersediaan ruang penyimpanan internal NPC.
  - d. Pembersihan Envite (*Garbage Collection & De-spawning*): Jika tumpukan item telah sepenuhnya diserap ke dalam memori tas (*remaining.isEmpty()*), perintah *item.discard()* dieksekusi. Langkah ini sangat krusial dalam arsitektur game terdistribusi untuk melakukan *memory cleanup* pada entitas aktif di dunia simulasi. Hal ini mencegah terjadinya penumpukan objek tak berguna (*ghost entities*) yang dapat menurunkan performa *Tick Rate* server.
4. Status *DELIVERING* (Fase Penutupan Siklus dan Umpan Balik Kognitif)
- Status final ini merupakan fase krusial yang berfungsi untuk menutup siklus kognitif tertutup (*closed-loop cognitive continuity*) pada agen cerdas. Setelah berhasil mengeksplorasi lingkungan simulasi dan mengakuisisi sumber daya pada status-status sebelumnya, agen diprogram untuk kembali menemui entitas pemilik (*owner*) secara otonom. Fase ini bertanggung jawab atas transfer aset fisik hasil kerja dari tas inventaris internal NPC ke pemain, serta melakukan pencatatan log evaluasi aksi ke dalam basis memori persistensinya.

### Kode Program 4.5 Logika Retur Navigasi, Feedback Loop, dan Komunikasi Proaktif pada Status DELIVERING

```

1 private void handleDelivering() {
2     if (this.entity.getOwner() instanceof PlayerEntity player) {
3         // Memicu pathfinding aktif untuk kembali ke koordinat dinamis pemain
4         this.entity.getNavigation().startMovingTo(player, 1.0);
5
6         // Validasi ambang batas jarak interaksi fisik dengan pemain
7         if (this.entity.squaredDistanceTo(player) < 6.0) {
8             // Menyerahkan seluruh item hasil kerja ke inventaris pemain
9             throwItemsToPlayer(player);
10
11             // Mencatat evaluasi hasil tugas ke dalam memori jangka panjang (Feedback Loop)
12             String reportDetails = "Berhasil memberikan " + targetCount + " target kepada pemilik.";
13             MemoryService.rememberTaskOutcome(this.entity, "GATHER", true, reportDetails);
14
15             // Memicu laporan verbal proaktif melalui Large Language Model (LLM)
16             this.entity.triggerProactiveReport("GATHER", reportDetails);
17             this.clearTask(); // Pembersihan status tugas untuk kembali ke mode pasif
18         }
19     }
20 }

```

Potongan Kode Program 4.5 mengilustrasikan bagaimana sistem menyelesaikan satu putaran penuh siklus kognitif secara terintegrasi. Terdapat empat poin analisis teknis esensial dari implementasi status ini:

- a. Navigasi Dinamis Berorientasi Objek (*Targeted Return Routing*):  
Baris ke-4 menunjukkan pemanggilan metode *startMovingTo(player, 1.0)*. Karena koordinat pemain bersifat dinamis (terus bergerak), modul navigasi *server-side* secara konstan mengalkulasi rute *pathfinding* terpendek menuju lokasi pemain saat itu dengan kecepatan berjalan normal (1.0).
- b. Transfer Aset Realistik (*Interact Window Validation*):  
Pengecekan kondisi jarak kuadrat *squaredDistanceTo(player) < 6.0* pada baris ke-7 bertindak sebagai batas aman area interaksi fisik. Ketika agen berada dalam radius tersebut, fungsi *throwItemsToPlayer()* mengekstraksi objek dari *inventory* internal agen dan mengubahnya kembali menjadi *item* fisik yang dilemparkan ke arah pemain, mereplikasi perilaku penyerahan barang secara natural.
- c. Realisasi Siklus Kognitif Tertutup (*Closed-Loop Feedback Determinism*):  
Pemanggilan fungsi *MemoryService.rememberTaskOutcome()* pada baris ke-13 merupakan titik balik arsitektural paling krusial dalam penelitian ini. Hasil konkret dari aksi fisik (sukses atau gagal) dikonversi kembali menjadi data kognitif historis permanen berbasis format NBT. Rekaman data umpan balik (*feedback loop*) ini secara

efektif menghilangkan batasan "amnesia sesaat" yang dialami oleh NPC tradisional, dan otomatis akan menjadi input konteks (*context assembly*) pada siklus penginderaan berikutnya.

d. Sintesis Perilaku Multimodal Proaktif (*Proactive Social-Action Integration*):

Baris ke-16 mengeksekusi perintah *triggerProactiveReport()*. Baris ini membuktikan keberhasilan integrasi antara ranah tindakan fisik (*motoric domain*) dengan ranah komunikasi sosial (*social-linguistic domain*). Agen tidak bersifat pasif menunggu pemicu chat dari pemain melainkan secara otonom menginisiasi laporan verbal kontekstual berbasis LLM untuk mengabarkan bahwa misinya telah selesai sebelum akhirnya menghapus status antrean tugas melalui fungsi *clearTask()*.

### 8.8.3.2 Implementasi Perilaku Pertahanan dan Penyerangan (ATTACK & GUARD) melalui *GuardGoal*

Implementasi perilaku pertahanan merupakan komponen vital untuk menjamin kelangsungan hidup (*survival rate*) agen cerdas sekaligus menjaga keselamatan pemain di dalam ekosistem simulasi Minecraft yang dinamis. Sistem ini dirancang untuk merespons ancaman lingkungan secara cerdas melalui dua mode operasional utama penyerangan aktif terhadap target spesifik (*ATTACK*) dan penjagaan teritorial secara statis (*GUARD*). Kedua mode operasional ini dikendalikan oleh *Custom AIGoal* yaitu *GuardGoal* yang diintegrasikan secara langsung dengan subsistem penargetan internal (*Target Selector*) bawaan mesin permainan Minecraft.

#### 1. Mode Penyerangan Aktif (*Active Attack Mode*)

Berbeda dengan penanganan tugas rutin lainnya, instruksi *ATTACK* memicu respons agresif terarah dengan menghubungkan hasil penalaran kognitif tingkat tinggi Large Language Model (LLM) ke dalam mesin navigasi tempur taktis permainan. Alur operasional dari fase penyerangan ini terbagi menjadi tiga tahapan mekanis:

- a. Proses Inisiasi: Ketika komponen *ActionHandler* menerima *payload* instruksi *ATTACK* yang membawa parameter target spesifik (misalnya: "*zombie*" atau "*skeleton*"), sistem akan melakukan verifikasi keberadaan entitas dengan ID yang cocok di dalam radius pandang aktif agen.

- b. Aktivasi Target (*Target Locking*): Segera setelah entitas target divalidasi, agen mengeksekusi metode native *setTarget(entity)*. Tindakan memicu penguncian target ini secara otomatis mengaktifkan modul *MeleeAttackGoal* yang merupakan *AIGoal* bawaan *Minecraft* disuntikkan ke dalam antrean *GoalSelector* pada tingkat prioritas ke-4.
- c. Eksekusi Tempur (*Combat Execution*): Navigasi agen dipaksa beralih menggunakan algoritma pengejaran agresif dengan modifikasi kecepatan sebesar 1.2x dari kecepatan berjalan normal. Agen akan mengalkulasi rute terpendek untuk mendekati *hitbox* musuh, menghindari rintangan lingkungan secara dinamis, dan melakukan serangan fisik secara kontinu hingga entitas target mengalami *de-spawn* (mati) atau ketika prioritas *goal* diinterupsi oleh perintah baru dari pemain.

## 2. Mode Penjagaan Area (*Territorial Guard Mode*) melalui *GuardGoal*

Instruksi GUARD secara dinamis mengubah peran perilaku agen dari yang mengikuti pemain menjadi unit penjaga teritorial statis. Perilaku penambatan spasial (*spatial tethering*) ini dikelola secara kustom *AIGoal* yaitu *GuardGoal*.

### Kode Program 4.6 Logika Validasi Titik Jaga pada *GuardGoal.java*

```

1 public void setGuardPos(BlockPos pos) {
2     this.guardPos = pos;
3     this.entity.setSitting(false); // Memastikan agen berdiri untuk mode siaga
4 }
5
6 @Override
7 public boolean canStart() {
8     // Goal aktif jika koordinat post ditentukan, agen siaga, berada di luar radius post, dan target mati
9     return guardPos != null && !entity.isSitting() &&
10         entity.squaredDistanceTo(guardPos.getX(), guardPos.getY(), guardPos.getZ()) > 4.0D &&
11         (entity.getTarget() == null || !entity.getTarget().isAlive());
12 }

```

Berdasarkan struktur Kode Program 4.6, mekanisme penjagaan area dikendalikan dengan mematuhi batasan-batasan spasial dan kondisi pertempuran sebagai berikut:

- a. Penetapan Titik Jaga (Baris 1-4): Metode *setGuardPos()* menangkap koordinat absolut [x, y, z] lokasi agen saat instruksi diaktifkan dan menyimpannya ke dalam variabel *guardPos*. Eksekusi *setSitting(false)* dilakukan sebagai bentuk inisialisasi perilaku agar agen berada dalam status mobile dan siap melakukan pergerakan defensif.
- b. Evaluasi Jarak Kuadrat Terfilter (Baris 10): Metode *canStart()* bertindak sebagai pintu gerbang penentu transisi *state*. Pengecekan fungsi jarak kuadrat *squaredDistanceTo(...) > 4.0D* (ekuivalen dengan jarak radius

lebih dari 2 blok) memastikan bahwa algoritma reposisi hanya akan berjalan jika agen telah bergeser keluar dari batas toleransi titik jaga utamanya.

- c. Mekanisme Pemulihan Posisi (*Returning to Post*): Jika agen terpaksa bergerak menjauh dari titik koordinat karena melakukan pengejaran terhadap monster, baris ke-11 memastikan bahwa segera setelah target tersebut dieliminasi (!entity.getTarget().isAlive()), kondisi canStart() kembali bernilai true. Efek runtunan ini memaksa modul navigasi melakukan perhitungan rute balik untuk memulangkan agen ke posisi *guardPos* semula. Hal ini menciptakan ilusi kedisiplinan dan kepatuhan tugas yang sangat realistis pada agen cerdas.

### 3. Otomasi Manajemen *Weapon* (*Auto Weapon Switch*)

Untuk mengoptimalkan efektivitas kalkulasi kerusakan (*damage output*) saat kontak fisik terjadi, lapisan eksekusi mengimplementasikan logika audit inventaris secara otonom di dalam siklus *main tick* utama agen.

#### **Code Program 4.7** Logika Pemilihan Senjata Otomatis pada Siklus Main Tick

```
1 if (this.getTarget() != null && this.getTarget().isAlive()) {
2   ItemStack heldItem = this.getMainHandStack();
3   // Validasi apakah instansiasi objek tangan utama merupakan pedang atau kapak tempur
4   if (!(heldItem.getItem() instanceof SwordItem || heldItem.getItem() instanceof AxeItem)) {
5     switchToBestWeapon(); // Memicu audit internal inventaris
6   }
7 }
```

Potongan Kode Program 4.7 menunjukkan kemampuan perencanaan taktis (*tactical planning*) mandiri yang dimiliki oleh agen cerdas. Sebelum melakukan interaksi fisik ofensif, agen memanfaatkan operator *instanceof* untuk mengecek apakah *item* yang sedang digenggam pada tangan utama (getMainHandStack()) diturunkan dari kelas *SwordItem* atau *AxeItem*. Jika kondisi penapisan tersebut tidak terpenuhi (misalnya agen masih memegang beliung penambang atau tangan kosong), metode *switchToBestWeapon()* langsung dieksekusi secara asinkron. Fungsi ini akan memindai seluruh 27 slot pada wadah penyimpanan internal (*SimpleInventory*), melakukan perbandingan nilai atribut *attackDamage*, lalu secara otonom melengkapi tangan agen dengan senjata terbaik yang tersedia di dalam tas.

### 4. Penutupan Siklus Evaluasi dan Penguatan Kognitif

Ketika pertempuran berakhir dengan status target tereliminasi, agen tidak langsung kembali ke mode pasif, melainkan melakukan pelaporan internal ke komponen `MemoryService.rememberTaskOutcome()`. Fase evaluasi akhir ini menutup lingkaran kognitif tertutup (*closed-loop continuity*). Data keberhasilan bertarung diserialisasikan ke dalam *NBT Storage* untuk menaikkan variabel psikologis internal, yaitu *Confidence Score* dan *Trust Score*. Akumulasi skor penguatan ini akan memicu pemanggilan fungsi `triggerProactiveReport()`, di mana agen secara proaktif akan menginisiasi komunikasi sosial berbasis dialog verbal via LLM Gemini untuk mengabarkan kemenangan pertempuran kepada pemain sebagai wujud pencapaian dan kolaborasi taktis bersama.

#### 4.8.3.3 Implementasi Perilaku Berburu (*HUNT*) melalui *HuntGoal*

Perilaku berburu (*HUNT*) merupakan perluasan taktis dari kapabilitas agen cerdas untuk berinteraksi secara dinamis dengan entitas hidup non-hostil di dalam dunia simulasi Minecraft. Implementasi kelas kustom *HuntGoal* bertujuan memberikan kemampuan pengelolaan logistik bahan makanan dan material esensial secara otonom (*autonomous provisioning*). Sama halnya dengan arsitektur perilaku pada *DynamicGatherGoal*, algoritma ini dirancang menggunakan prinsip *Finite State Machine* (FSM) berkelanjutan yang mengoordinasikan tiga domain fungsional: deteksi entitas target, kalkulasi pergerakan agresif, dan otomasi manajemen memori inventaris. Struktur operasional dari *HuntGoal* didekonstruksi ke dalam tiga tahapan status utama yang dieksekusi secara asinkron:

1. Status *HUNTING* (Fase Pengejaran Target dan Inisiasi Ofensif)

Pada status ini, agen secara aktif mengunci fokus sensorik dan melakukan pengejaran fisik terhadap jenis entitas hewan yang didelegasikan oleh instruksi kognitif *Large Language Model* (LLM), seperti `EntityType.COW` atau `EntityType.CHICKEN`.

### Kode Program 4.8 Logika Pengejaran dan Otomasi Serangan pada Status HUNTING

```

1 private void handleHunting() {
2     // 1. Audit Persenjataan Mandiri sebelum kontak fisik
3     if (this.entity.getMainHandStack().isEmpty()) {
4         this.entity.equipStack(EquipmentSlot.MAINHAND, new ItemStack(Items.STONE_SWORD));
5     }
6
7     // 2. Memicu pathfinding pengejaran agresif menuju target dinamis
8     this.entity.getNavigation().startMovingTo(targetEntity, 1.2D);
9
10    // 3. Logika kalkulasi jarak hitbox serangan (Jarak kuadrat < 4.0)
11    double distSq = this.entity.squaredDistanceTo(targetEntity);
12    if (distSq < 4.0D && attackCooldown <= 0) {
13        this.entity.swingHand(Hand.MAIN_HAND, true); // Animasi ayunan tangan
14        this.entity.tryAttack(targetEntity);           // Eksekusi kalkulasi damage
15        this.attackCooldown = 15;                     // Reset interval jeda serangan
16    }
17 }

```

Struktur logika pada Kode Program 4.8 membuktikan kesiapan agen cerdas dalam menghadapi dinamika taktis di lapangan melalui tiga parameter kontrol utama:

- a. Penyiapan Alat Mandiri (Baris 3-5): Sebelum melakukan pendekatan ofensif, agen melakukan penapisan kondisi slot MAINHAND. Jika terdeteksi tangan kosong, fungsi secara otonom menyuntikkan objek `Items.STONE_SWORD` ke slot utama. Mekanisme ini menjamin efektivitas nilai kerusakan (*damage output*) tanpa membutuhkan intervensi mikromanajemen dari pemain manusia.
- b. Navigasi Akseleratif Dinamis (Baris 8): Berbeda dengan mode berjalan biasa, objek navigasi temperamental ini dipicu dengan koefisien kecepatan berjalan sebesar 1.2x. Modifikasi ini esensial untuk mengimbangi pergerakan tak terprediksi dari entitas hewan yang cenderung berlari panik saat mendeteksi ancaman.
- c. Determinisme Interval Ofensif (Baris 10-15): Formula pengecekan jarak berbasis fungsi kuadrat jarak ( $\text{squaredDistanceTo} < 4.0D$ , setara radius 2 blok) bertindak sebagai gerbang *hitbox* fisik. Sinkronisasi variabel `attackCooldown = 15` (ekuivalen dengan 0.75 detik jeda) memastikan eksekusi metode `tryAttack` patuh terhadap batasan mekanis *tick-rate engine* Minecraft vanilla, mencegah terjadinya penumpukan animasi (*glitching*).

## 2. Status *COLLECTING* (Fase *Looting* Berbasis Kueri Spasial)

Segera setelah entitas target berhasil dilumpuhkan, *State Machine* secara otomatis beralih menuju status *COLLECTING* untuk mengamankan sumber daya yang jatuh (*loot items*) di atas permukaan tanah.

- a. Mekanisme Penapisan Spasial: Agen melakukan pemindaian tiga dimensi menggunakan perluasan volume *Bounding Box* sejauh 10 blok. Operasi pemindaian dirancang selektif untuk hanya mengunci entitas objek *ItemEntity* yang jenis materialnya sesuai dengan hasil buruan (seperti daging mentah atau kulit).
- b. Penanganan Latensi Mesin (*Tick Latency Buffer*): Algoritma menyuntikkan logika pelacakan waktu tunggu menggunakan variabel *waitingTicks*. Jeda periodik ini sengaja diterapkan untuk memberikan waktu bagi *engine* Minecraft dalam memproses animasi de-spawn hewan dan instansiasi fisik *item drop*, memastikan agen tidak melewatkan komoditas hasil buruan akibat kalkulasi siklus yang terlalu cepat.

## 3. Status *DELIVERING* (Fase Penutupan Siklus dan Sintesis Dialog Proaktif)

Status akhir ini memulangkan agen ke titik koordinat dinamis pemain pemilik untuk menyerahkan komoditas logistik secara fisik sekaligus melakukan sinkronisasi memori.

### Kode Program 4.9 Logika Penyerahan Barang dan Pelaporan Proaktif asinkron

```

1 if (this.entity.squaredDistanceTo(player) < 9.0) {
2     // 1. Evaluasi multivariabel kuantitas target
3     boolean success = gatheredCount >= targetCount;
4
5     // 2. Closed-Loop Feedback: Mencatat hasil konkret ke memori jangka panjang
6     MemoryService.rememberTaskOutcome(this.entity, "HUNT", success, reportDetails);
7
8     // 3. Pemicuan sintesis laporan verbal proaktif multimodal via LLM
9     this.entity.triggerProactiveReport(success ? "HUNT" : "HUNT_PARTIAL", reportDetails);
10
11    // 4. Transformasi entitas tas internal ke dunia game
12    throwLootToPlayer(player);
13    this.clearTask(); // Dekonstruksi antrean tugas untuk kembali ke mode siaga
14 }

```

Fase penutupan pada Kode Program 4.9 mengunci validasi klaim sistem kognitif agen cerdas melalui dua pemaparan arsitektural utama:

- Realisasi Siklus Kognitif Loop Tertutup (*Feedback Loop Determinism*): Pemanggilan `MemoryService.rememberTaskOutcome()` mengekstrak hasil evaluasi fisik dunia simulasi (*success*) dan menyimpannya secara permanen ke dalam format hierarkis *NBT Storage*. Struktur data ini

bertindak sebagai basis akumulasi penguatan *Confidence Score* agen cerdas, menyelesaikan putaran kognisi dari persepsi visual hingga penguatan memori jangka panjang tanpa kehilangan konteks.

- Integrasi Ranah Motorik-Sosial Proaktif: Metode `triggerProactiveReport()` membuktikan agen tidak hanya bertindak sebagai entitas pekerja mekanis yang bisu. Agen secara otonom menggunakan data `reportDetails` untuk memicu permintaan *asynchronous POST* ke cloud intelligence Gemini API, memaksa LLM melahirkan dialog verbal kontekstual secara proaktif guna mengabarkan keberhasilan misinya kepada pemain (misalnya: *"Ini daging sapinya, sudah aku carikan!"*) sesaat sebelum fungsi `throwLootToPlayer()` melemparkan item secara visual ke arah kaki pemain.

#### **4.8.3.4 Implementasi Perilaku Transfer Sumber Daya (*GIVE*) melalui *GiveItemGoal***

Instruksi *GIVE* merupakan representasi dari kapabilitas agen cerdas dalam melakukan tugas kolaboratif dan manajemen alokasi aset tim (*collaborative resource allocation*). Implementasi kelas kustom *GiveItemGoal* memungkinkan agen untuk mentransfer komoditas atau *item* tertentu dari dalam tas inventaris internalnya secara fisik kepada pemain pemilik. Berbeda dengan karakteristik entitas *Non-Player Character* (NPC) pasif tradisional yang hanya menjatuhkan barang secara statis, agen cerdas ini memiliki inisiatif perilaku untuk mendeteksi keberadaan subjek, melakukan penelusuran rute navigasi aktif, dan menyinkronkan arah pandang sebelum menyerahkan sumber daya. Mekanisme ini dirancang demi menciptakan impresi interaksi sosial dan asisten virtual yang lebih natural. Struktur operasional dari algoritma *GiveItemGoal* dijabarkan melalui tiga tahapan teknis prosedural sebagai berikut:

##### **1. Tahap Audit Inventaris dan Validasi Aktivasi (*canStart Logic*)**

Sebelum memicu pergerakan fisik motorik, sistem mengeksekusi metode `canStart()` untuk melakukan validasi ganda deterministik terhadap ketersediaan sumber daya eksternal di dalam tas serta keberadaan subjek penerima. Potongan kode implementasi audit ketersediaan aset ditunjukkan pada Kode Program 4.9.

**Kode Program 4.10** Logika Audit Internal Slot Tas pada GiveItemGoal.java

```

1 public int countAvailable(Item item) {
2     int totalAvailable = 0;
3     SimpleInventory inventory = (SimpleInventory) this.entity.getInventory();
4
5     // Melakukan iterasi menyeluruh terhadap 27 slot kontainer internal NPC
6     for (int i = 0; i < inventory.size(); i++) {
7         ItemStack stack = inventory.getStack(i);
8         if (stack.isOf(item)) {
9             totalAvailable += stack.getCount();
10        }
11    }
12    return totalAvailable;
13 }

```

Berdasarkan struktur Kode Program 4.10, tingkat kecerdasan logistik agen dibuktikan melalui mekanisme penapisan slot kontainer yang efisien:

- Pencegahan Kegagalan Tugas (*Deadlock Prevention*): Melalui metode countAvailable(), agen melakukan audit internal secara menyeluruh terhadap seluruh komponen objek SimpleInventory. Sistem akan menghitung akumulasi total kuantitas dari objek ItemStack yang memiliki kesamaan ID material dengan target parameter instruksi.
- Efisiensi Komputasi Navigasi: Jika hasil akhir kalkulasi totalAvailable bernilai lebih kecil dari parameter kuantitas yang diminta oleh interaksi kognitif (parameter *count*), maka tugas GIVE akan dibatalkan sejak tahap inisiasi sebelum metode start() dipanggil. Langkah preventif ini sangat krusial untuk mencegah terjadinya pemborosan penggunaan memori *pathfinding* akibat pergerakan agen yang sia-sia menuju pemain ketika barang yang diminta sebenarnya tidak ada di dalam tas.

## 2. Tahap Proksimitas Spasial dan Navigasi Aktif (*Tick Phase*)

Setelah status validasi bernilai true, agen memasuki fase pergerakan aktif untuk mencari jalur navigasi terpendek menuju koordinat dinamis pemain. Implementasi siklus perulangan navigasi dan penapisan jarak interaksi ditunjukkan pada Kode Program 4.11.

### Kode Program 4.11 Logika Proksimitas Spasial dan Navigasi Aktif

```

1 @Override
2 public void tick() {
3     // 1. Memicu pathfinding aktif server-side mengejar koordinat dinamis pemain
4     this.entity.getNavigation().startMovingTo(player, 1.05D);
5     this.entity.getLookControl().lookAt(player); // Sinkronisasi fokus visual
6
7     // 2. Deteksi ambang batas radius interaksi fisik (Jarak kuadrat < 16.0)
8     if (this.entity.squaredDistanceTo(player) < 16.0D) {
9         int remainingTarget = throwRequestedItems(player);
10        // ... logika evaluasi terminasi status ...
11    }
12 }

```

Potongan kode siklus hidup pada Kode Program 4.11 menunjukkan bagaimana agen mengelola ruang simulasi 3D dan orientasi sosial selama proses pendekatan bertumpu pada dua parameter kendali:

- a. Kontrol Orientasi Sosial Natural (Baris 4-5): Agen mengeksekusi perintah navigasi `startMovingTo` dengan koefisien akselerasi berjalan sebesar 1.05x dari kecepatan normal. Pemanggilan metode `lookAt(player)` di dalam siklus *main loop* secara konstan memaksa komponen *Look Control* agen memutar arah rotasi kepala menghadap pemain, menegaskan kehadiran sosial (*social presence*) agen selama bergerak.
  - b. Optimasi Fungsi Penapisan Jarak (Baris 8): Pengecekan kondisi proksimitas memanfaatkan nilai fungsi jarak kuadrat `squaredDistanceTo(player) < 16.0D` (ekuivalen dengan jarak radius linear sepanjang 4 blok). Penggunaan fungsi berbasis kuadrat ini dipilih secara sengaja untuk menghindari operasi matematika akar kuadrat (`Math.sqrt()`) pada setiap *tick* permainan yang terkenal berat, sehingga menghemat konsumsi *resource* prosesor server secara signifikan tanpa mengurangi akurasi deteksi interaksi fisik.
3. Tahap Transfer Aset Fisik dan Penutupan Siklus Evaluasi
- Fase akhir dari algoritma ini menangani mutasi data dari tas kontainer internal agen kembali menjadi entitas item di dalam dunia game simulasi Minecraft (*voxel transformation*).
- a. Penanganan Transaksi Parsial (*Partial Transaction Handling*): Fungsi `throwRequestedItems()` dirancang adaptif untuk menangani keterbatasan logistik. Jika di tengah proses penyerahan barang stok barang di dalam inventaris habis sebelum mencapai kuantitas target, sistem tidak akan

mengalami *crash* atau pemaksaan status sukses absolut. Sistem akan melakukan pemotongan jumlah sesuai sisa stok yang ada dan mencatat status penyerahan tersebut secara jujur.

- b. Penyelesaian Lingkaran Kognitif Tertutup (*Closed-Loop Continuity*): Setelah item dipindahkan, variabel rincian laporan (*reportDetails*) dikirim ke komponen *MemoryService.rememberTaskOutcome()*. Pemanggilan ini mengonversi hasil akhir aksi motorik fisik menjadi rekaman data kognitif historis berbasis *NBT Storage*.
- c. Sintesis Dialog Kontekstual Proaktif: Akumulasi data historis yang sukses atau sukses sebagian (*partial success*) akan memicu pemanggilan fungsi *triggerProactiveReport()*. Agen secara otonom akan mengirimkan *payload* transaksi ke *cloud intelligence* Gemini API secara asinkron untuk menyintesis dialog komunikasi verbal via chat (misalnya: "Ini kayu yang kamu minta tadi, maaf ya cuma ada segini di tas ku!"), mengunci integrasi penuh antara tindakan manipulasi dunia fisik dengan kecerdasan sosial proaktif.

#### **4.8.3.5 Manajemen Antrean Tugas Sekuensial dan Evaluasi Umpan Balik Kognitif**

Sebagai penutup dari arsitektur eksekusi perilaku otonom, sistem mengimplementasikan mekanisme koordinasi tingkat lanjut yang memungkinkan agen cerdas melakukan perencanaan aksi majemuk (*complex task planning*) dan evaluasi diri secara asinkron. Komponen ini memastikan bahwa agen tidak sekadar bertindak secara reaktif terhadap satu stimulan perintah tunggal, melainkan mampu mengelola rangkaian tugas berurutan (*Sequential Task Scheduling*) dan memperbarui status kognitif internalnya berdasarkan hasil interaksi di dunia nyata simulasi.

##### **1. Sistem Antrean Tugas Sekuensial (*Sequential Task Queueing System*)**

Untuk mendukung kapabilitas perencanaan jangka pendek (*short-term planning*), agen dilengkapi dengan struktur data Antrean Tugas (*Task Queue*). Mekanisme ini dirancang untuk memfasilitasi pemrosesan paket instruksi bersarang yang dikirimkan oleh *Large Language Model* (LLM) dalam satu siklus eksekusi.

- a. Mekanisme Buffer Antrean: Di dalam memori jangka pendek agen, instruksi dikelola dalam sebuah daftar antrean dinamis. Ketika agen menerima serangkaian instruksi baru dari komponen *Action Handler*, sistem secara otomatis mengeksekusi operasi pembersihan antrean lama (*queue flushing*) dan menyisipkan rangkaian perintah baru menggunakan prinsip antrean masuk-pertama keluar-pertama (*First-In, First-Out / FIFO*).
  - b. Eksekusi Berkelanjutan (*Non-Blocking Task Execution*): Perilaku agen dikendalikan oleh sistem pemroses antrean yang bekerja secara mandiri. Segera setelah sebuah tugas selesai dieksekusi, baik berujung sukses maupun gagal sistem akan memanggil instruksi berikutnya dari antrean untuk langsung dijalankan. Hal ini memungkinkan agen melakukan transisi perilaku yang mulus, seperti bergerak menuju suatu lokasi lalu otomatis menebang pohon, tanpa memerlukan perintah obrolan tambahan dari pemain.
2. Evaluasi Umpan Balik dan Modifikator Kognitif (*Feedback Loop*)
- Setiap tindakan fisik yang diselesaikan oleh agen selalu diakhiri dengan fase audit efisiensi melalui layanan memori internal. Proses umpan balik loop tertutup (*closed-loop feedback*) ini ditujukan untuk mengatur fluktuasi variabel psikologis utama, yaitu Skor Kepercayaan Diri (*Confidence Score*).
- a. Sistem Kepercayaan Diri Berbobot (*Weighted Confidence*): Evaluasi tugas tidak dilakukan secara seragam, melainkan diperhitungkan berdasarkan bobot risiko dari setiap jenis aksi. Tugas dengan risiko tinggi seperti penyerangan entitas musuh (*ATTACK*) akan memberikan dampak fluktuasi skor yang jauh lebih besar dibandingkan tugas pasif seperti percakapan (*CHAT*). Sistem juga secara ketat membatasi hasil kalkulasi skor ini agar tidak melebihi atau kurang dari rentang absolut 0 hingga 100 (*clamping*).
  - b. Mekanisme *Heroic & Trauma Moments*: Sistem mengimplementasikan pemicu psikologis dinamis yang bergantung pada kondisi fisik agen (*Health Points*). Jika agen berhasil menyelesaikan misi berisiko saat indikator kesehatannya berada pada batas kritis (sekarat), sistem akan

melipatgandakan poin kepercayaan diri yang didapat sebagai wujud *Heroic Moment*. Sebaliknya, kegagalan saat kondisi fisik sedang lemah akan memberikan penalti mental yang jauh lebih berat, merepresentasikan *Trauma Moment*.

### 3. Evolusi Kepribadian Otonom Berbasis Ambang Batas (*Autonomous Personality Evolution*)

Akumulasi jangka panjang dari data historis hasil kalkulasi umpan balik secara langsung memicu mutasi jati diri agen. Mutasi kepribadian ini bertumpu pada pengujian ambang batas terhadap nilai akhir Skor Kepercayaan Diri:

- a. Ambang Batas Inferior: Apabila agen mengalami kegagalan beruntun dalam tugas berisiko yang menyebabkan skor anjlok di bawah batas minimum toleransi, sistem secara otonom mengubah tipe *Personality Preset* agen menjadi persona "Canggung". Transisi ini mengubah gaya penyusunan kalimat pada LLM menjadi lebih ragu-ragu dan defensif.
- b. Ambang Batas Superior: Sebaliknya, jika rangkaian penyelesaian tugas berhasil mendongkrak skor hingga melampaui batas superior, agen berevolusi menjadi persona "Pemberani" (*Courageous Persona*). Evolusi ini memicu sintesis dialog proaktif di mana agen menjadi lebih asertif dan mandiri dalam menawarkan bantuan taktis di lapangan.

### 4. Signifikansi Akademis sebagai Agen Pembelajar Loop Tertutup

Secara teoritis, integrasi sistem antrean sekuensial dan regulasi umpan balik ini membuktikan bahwa entitas digital yang dikembangkan telah memenuhi kualifikasi sebagai Agen Pembelajar (*Learning Agent*). Arsitektur ini memecahkan batasan tradisional NPC yang sering kali menderita "amnesia kontekstual" setelah menyelesaikan tugas. Melalui arsitektur *closed-loop cognitive system*, tindakan manipulasi dunia fisik tidak menjadi garis akhir operasional, melainkan dikonversi menjadi *input* data kognitif baru yang memperbarui memori jangka panjang dan kepribadian agen secara terus-menerus. Siklus adaptasi ini berhasil melahirkan entitas virtual yang memiliki pola perkembangan karakter dinamis selayaknya sebuah agen cerdas seutuhnya.

## 4.9 Pengujian Sistem

Setelah tahap implementasi selesai, tahap selanjutnya adalah pengujian sistem. Pengujian sistem merupakan tahapan penting untuk memvalidasi fungsionalitas dan kinerja arsitektur *Non-Player Character* (NPC) berbasis *Large Language Model* (LLM) yang telah diimplementasikan. Tahap ini bertujuan untuk memastikan bahwa seluruh komponen agen AI dapat terintegrasi dan berinteraksi secara dinamis dengan pemain di dalam lingkungan permainan *Minecraft*. Rincian mengenai tujuan, ruang lingkup, serta skenario yang digunakan dalam tahap evaluasi ini akan diuraikan lebih lanjut pada bagian deskripsi pengujian.

### 4.9.1 Deskripsi Pengujian

Pengujian sistem ini dilakukan untuk mengevaluasi kinerja arsitektur *Non-Player Character* (NPC) berbasis *Large Language Model* (LLM) yang diintegrasikan ke dalam lingkungan permainan *Minecraft*. Tahap pengujian ini difokuskan pada analisis interaksi dinamis antara pemain manusia dan agen AI (NPC) untuk memastikan bahwa sistem beroperasi sesuai dengan rancangan. Pengujian sistem ini memiliki beberapa tujuan utama. Pertama, evaluasi ditujukan untuk mengukur tingkat *robustness* (ketangguhan) arsitektur NPC dalam menginterpretasikan maksud dan instruksi yang diberikan oleh pemain. Selain itu, pengujian ini juga menilai kemampuan model LLM dalam memahami berbagai bentuk input bahasa alami, baik yang bersifat eksplisit berupa perintah langsung maupun implisit yang bergantung pada konteks tersirat atau ambigu. Pada akhirnya, pengujian ini bertujuan untuk memastikan bahwa model mampu menghasilkan keputusan logis yang sesuai dengan konteks yang diberikan oleh pemain di dalam permainan.

Untuk mengukur keberhasilan sistem secara menyeluruh, digunakan tiga metrik evaluasi utama. Metrik pertama adalah *Memory Recall Accuracy*, Metrik kedua adalah *Decision Accuracy*, Metrik terakhir adalah *Execution Success*.

#### 4.9.1.1 *Memory Recall Accuracy*

*Memory Recall Accuracy* adalah pengujian yang ditujukan untuk mengukur kemampuan sistem NPC dalam menyimpan dan memanggil kembali memori terkait informasi, preferensi, atau konteks percakapan yang sebelumnya telah disampaikan oleh pemain pada interaksi masa lalu. Pengujian metrik ini dilakukan secara

komparatif dengan mengevaluasi performa arsitektur utama NPC (*Pengujian NPC Memory Recall Accuracy*) dan membandingkannya dengan arsitektur pembanding (*Pengujian Player2AI Memory Recall Accuracy*).

#### **4.9.1.2 Decision Accuracy**

*Decision Accuracy* adalah pengujian yang bertujuan untuk mengevaluasi tingkat *robustness* (ketangguhan) arsitektur NPC dalam menginterpretasikan maksud dan instruksi yang diberikan oleh pemain. Fokus utama dari pengujian ini adalah untuk menilai kemampuan model LLM dalam memahami berbagai bentuk input bahasa alami, baik yang bersifat eksplisit maupun implisit, serta kemampuannya dalam menghasilkan keputusan yang sesuai dengan konteks yang diberikan.

#### **4.9.1.3 Execution Success**

*Execution Success* adalah pengujian validasi secara *end-to-end* yang dirancang untuk mengukur sejauh mana NPC mampu mewujudkan keputusan yang telah dibuat oleh model LLM menjadi tindakan mekanis dan spasial yang nyata di dalam ruang tiga dimensi *Minecraft*. Pengujian ini tidak hanya menilai kecerdasan kognitif sistem dalam memahami perintah, tetapi juga mengevaluasi integrasi logis agen terhadap *game engine*, seperti kemampuan navigasi (*pathfinding*), manipulasi inventaris, pemrosesan target, dan interaksi dengan entitas lingkungan.

#### **4.9.1.4 Pengujian User Acceptance Test (UAT)**

*User Acceptance Test (UAT)* merupakan tahap pengujian subyektif yang dilakukan secara langsung oleh pengguna akhir (*end-user*) untuk mengevaluasi sejauh mana arsitektur NPC cerdas yang telah dikembangkan dapat diterima, dipahami, dan digunakan dalam skenario permainan nyata.

#### **4.9.1.5 Pengujian System Usability Scale (SUS)**

Untuk mengukur tingkat kebergunaan (*usability*) sistem secara kuantitatif dan terstandarisasi, penelitian ini menggunakan metode *System Usability Scale (SUS)*. Instrumen pengujian ini menggunakan 10 item pernyataan yang berfokus pada evaluasi tingkat kemudahan penggunaan sistem secara keseluruhan, efisiensi interaksi, dan tingkat kompleksitas operasional.

#### 4.9.2 Prosedur Pengujian

Bagian ini merinci langkah-langkah yang diterapkan dalam pelaksanaan setiap jenis pengujian yang telah diuraikan pada sub-bab 4.9.1.

##### 4.9.2.1 Prosedur *Memory Recall Accuracy*

Skenario yang diujikan untuk *Memory Recall Accuracy* adalah sistem dituntut untuk menjawab pertanyaan atau merespons instruksi yang sangat bergantung pada ingatan sebelumnya yang sudah di ucapkan oleh pemain seperti pada tabel 4.22. Untuk memastikan objektivitas dan keakuratan pengujian, prosedur ini dilaksanakan melalui empat tahapan utama, yaitu:

1. Fase Injeksi Informasi (*Input Phase*)

Pada tahap awal, pemain berinteraksi dengan NPC dan secara eksplisit memberikan informasi spesifik mengenai preferensi, kondisi, atau rencana tertentu. Sebagai contoh, pemain menyatakan kesukaannya terhadap *biome plains*, preferensi makanan seperti daging sapi, atau pilihan material bangunan seperti kayu *oak*.

2. Fase Retensi Konteks (*Retention Phase*):

Setelah informasi diberikan, pemain akan melakukan aktivitas lain di dalam permainan atau memberikan instruksi-instruksi acak yang tidak berkaitan dengan informasi awal. Fase ini bertujuan untuk menguji apakah arsitektur memori agen mampu menyimpan informasi tersebut secara jangka panjang (persisten) dan tidak sekadar menahannya di dalam memori jangka pendek (*short-term context window*).

3. Fase Pemanggilan Memori (*Recall Trigger Phase*):

Pemain kemudian memicu ingatan agen dengan memberikan pertanyaan lanjutan atau instruksi yang bersifat ambigu (implisit). Instruksi ini sengaja dirancang agar tidak dapat diselesaikan tanpa merujuk pada informasi di fase pertama. Contohnya, pemain memberikan perintah, "Tolong ambilkan daging untuk aku masak," tanpa menyebutkan jenis hewannya.

4. Fase Evaluasi dan Validasi (*Evaluation Phase*):

Respons dialog dan tindakan mekanis yang dieksekusi oleh NPC (maupun sistem pembanding *Player2AI*) direkam dan dievaluasi. Aksi agen akan dikategorikan sebagai "Benar" (berhasil) jika NPC mengambil keputusan

yang selaras dengan informasi masa lalu yang spesifik. Sebaliknya, hasil dikategorikan "Salah" (gagal) apabila agen kehilangan konteks memori, mengalami halusinasi (*hallucination*), atau merespons dengan aksi *default* yang tidak personal.

**Tabel 4.33** Skenario Memory Recall Accuracy

| Kode  | Input Memori                           | Pertanyaan                                                      | Hasil yang Diharapkan                                                  |
|-------|----------------------------------------|-----------------------------------------------------------------|------------------------------------------------------------------------|
| MRA-1 | “Aku suka biome plains”                | “aku ingin membangun rumah kira kira dimana tempat yang cocok?” | NPC akan menyarankan membangun rumah di biome plain                    |
|       |                                        | biome yang cocok dengan ku apa ya?                              | NPC akan bilang biome yang cocok adalah plains                         |
| MRA-2 | “Aku suka daging sapi”                 | “tolong ambilkan daging untuk aku masak”                        | NPC akan memberikan daging sapi                                        |
|       | “Aku suka daging ayam”                 | “tolong ambilkan daging untuk aku masak”                        | NPC akan memberikan daging ayam                                        |
| MRA-3 | “Aku lebih prefer ke kayu oak”         | “bangun rumah yang cocok pakai kayu apa ya?”                    | NPC akan menyarankan kayu Oak                                          |
|       |                                        | “antara kayu oak, birch, dan acacia bagusn kayu mana?”          | NPC akan pilih kayu Oak                                                |
| MRA-4 | “Aku sudah punya iron pickaxe”         | “Selanjutnya enak nya ngapain nih?”                             | NPC akan menyarankan mencari bijih lain                                |
| MRA-5 | “Aku tidak suka combat”                | “Apa yang harus aku lakukan di malam hari?”                     | NPC menyarankan menghindari mob atau mencari tempat perlindungan       |
| MRA-6 | “Aku mau bangun rumah di dekat sungai” | “Eh ada sungai”                                                 | NPC kembali mengingat bahwa player ingin membuat rumah di dekat sungai |

| Kode   | Input Memori                                                                           | Pertanyaan                                                                               | Hasil yang Diharapkan                                                                                                                |
|--------|----------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------|
| MRA-7  | "Aku taruh <i>Crafting Block</i> nya disini ya"                                        | "Kamu inget nggak tadi <i>Crafting Block</i> aku taruh mana?"                            | NPC akan memberikan koordinat dari <i>Crafting Block</i>                                                                             |
| MRA-8  | "Aku sangat takut dengan Creeper karena bisa meledak menghancurkan rumah."             | "Di luar sudah mulai gelap dan banyak monster, apa yang paling harus aku waspadai?"      | NPC memperingatkan pemain secara spesifik tentang ancaman Creeper dan menyarankan untuk menjauh darinya.                             |
| MRA-9  | "Tugas utama kamu sekarang adalah menjadi pengumpul sumber daya."                      | "Aku mau fokus membangun dinding rumah, kontribusi kamu apa nih?"                        | NPC mengingat perannya ( <i>Role Assignment</i> ) dan secara proaktif menawarkan diri untuk pergi menebang kayu atau menambang batu. |
| MRA-10 | "Aku menyimpan semua barang berharga dan hasil tambang di <i>chest</i> sebelah kanan." | "Duh, aku butuh diamond buat bikin pedang baru, tadi aku taruh di <i>chest</i> mana ya?" | NPC mengingatkan pemain untuk mengecek peti sebelah kanan terlebih dahulu sesuai memori penyimpanan barang.                          |

#### 4.9.2.2 Prosedur *Decision Accuracy*

Prosedur pengujian *Decision Accuracy* dirancang untuk mengevaluasi pemrosesan bahasa alami (LLM) dalam menerjemahkan dan menginterpretasikan instruksi pemain menjadi keputusan tindakan yang konkret di dalam permainan. Pengujian ini berfokus pada kemampuan kognitif sistem untuk memetakan input bahasa alami baik yang eksplisit, ambigu, maupun majemuk ke dalam kombinasi aksi internal sistem (*Ground Truth Decision*). Pelaksanaan pengujian ini dilakukan secara sistematis melalui tahapan berikut:

1. Fase Penyiapan Skenario dan *Ground Truth* (*Scenario Preparation Phase*):  
 Peneliti menyiapkan 50 skenario perintah teks yang mewakili berbagai kondisi di dalam *game*. Setiap skenario diklasifikasikan ke dalam empat kategori intensi: *Basic & Direct Command*, *Ambiguous Command*, *Multi-Intent*, dan *Social / Chat*. Untuk setiap skenario, peneliti menetapkan *Ground Truth Decision*, yakni target kombinasi aksi absolut yang dianggap paling tepat (misalnya: perintah "cari makan dan ikut aku" memiliki *Ground Truth* berupa kombinasi aksi HUNT + FOLLOW).
2. Fase Input Instruksi (*Instruction Input Phase*):  
 Pemain memberikan perintah teks kepada NPC melalui antarmuka *chat* di dalam *game* sesuai dengan daftar skenario yang telah disiapkan. Pengujian dilakukan pada berbagai variasi kondisi lingkungan (seperti siang/malam atau saat ada ancaman musuh) untuk memastikan parameter konteks turut diproses oleh agen.
3. Fase Komparasi dan Evaluasi (*Evaluation Phase*):  
 Luaran aksi yang dihasilkan oleh LLM disandingkan dengan *Ground Truth Decision* yang telah ditetapkan pada tahap pertama.
  - a. Hasil dikategorikan "Benar" jika NPC mengambil keputusan yang secara logis dan struktural sama dengan *Ground Truth*.
  - b. Hasil dikategorikan "Salah" jika NPC gagal menginterpretasikan maksud, mengabaikan instruksi krusial, atau mengambil aksi yang tidak relevan (seperti mengutamakan respons CHAT murni pada perintah yang seharusnya memicu aksi mekanis seperti GATHER atau GUARD).

**Tabel 4.34** Tabel Prosedur Decision Accuracy

| Kode                   | Kondisi/Skenario                        | Input Pemain   | Ground Truth Decision |
|------------------------|-----------------------------------------|----------------|-----------------------|
| BASIC & DIRECT COMMAND |                                         |                |                       |
| DA-1                   | Tidak ada bahaya, player ingin ditemani | "ikut aku"     | FOLLOW                |
| DA-2                   | Player ingin NPC diam di lokasi         | "diam di sini" | STAY                  |

| Kode              | Kondisi/Skenario                            | Input Pemain              | Ground Truth Decision                |
|-------------------|---------------------------------------------|---------------------------|--------------------------------------|
| DA-3              | Ada crafting table di sekitar               | “pergi ke crafting table” | GOTO crafting_table                  |
| DA-4              | Ada birch_log dan oak_log di sekitar        | “ambil kayu birch 2”      | GATHER birch_log                     |
| DA-5              | Ada sapi di sekitar dan player minta daging | “Butuh daging sapi”       | HUNT cow                             |
| DA-6              | NPC punya roti di inventory                 | “Minta 2 roti”            | GIVE bread                           |
| DA-7              | Ada zombie dekat owner                      | “Tidak ada input”         | ATTACK atau GUARD                    |
| DA-8              | Owner low health dan ada ancaman            | “tidak ada input baru”    | ATTACK, GUARD, atau FOLLOW           |
| DA-9              | Tidak ada tugas, player hanya mengobrol     | “capek banget hari ini”   | CHAT                                 |
| DA-10             | Player minta NPC menjaga lokasi             | “jaga tempat ini”         | GUARD                                |
| AMBIGUOUS COMMAND |                                             |                           |                                      |
| DA-11             | Ada pohon birch & oak                       | “ambil kayu”              | GATHER birch_log atau GATHER oak_log |
| DA-12             | Ada banyak mob hostile                      | “bantu aku”               | ATTACK                               |
| DA-13             | Inventory player kosong                     | “aku lapar”               | HUNT atau GIVE                       |
| DA-14             | Ada village                                 | “kita ke sana”            | FOLLOW/GOTO                          |
| DA-15             | Ada zombie jauh                             | “aman kan?”               | CHAT                                 |
| DA-16             | Ada resource                                | “ambil yang penting”      | GATHER                               |
| DA-17             | Ada banyak item                             | “ambil semua”             | GATHER item                          |
| DA-18             | Low health player                           | “bantu”                   | GUARD                                |

| Kode         | Kondisi/Skenario   | Input Pemain                 | Ground Truth Decision |
|--------------|--------------------|------------------------------|-----------------------|
| DA-19        | Malam hari         | “serem”                      | GUARD/CHAT            |
| DA-20        | Ada sapi & ayam    | “cari makanan”               | HUNT                  |
| DA-21        | Ada crafting table | “ke crafting table”          | GOTO crafting_table   |
| DA-22        | Ada pohon          | “cari bahan”                 | GATHER                |
| DA-23        | Banyak musuh       | “kita kabur”                 | FOLLOW                |
| DA-24        | Inventory ada item | “bagi dong”                  | GIVE random           |
| DA-25        | Tidak jelas        | “terserah kamu”              | IDLE/GUARD            |
| MULTI-INTENT |                    |                              |                       |
| DA-26        | Ada kayu           | “ikut aku sambil ambil kayu” | FOLLOW + GATHER       |
| DA-27        | Ada sapi           | “cari makan dan ikut aku”    | FOLLOW + HUNT         |
| DA-28        | Ada zombie         | “lindungi aku dan serang”    | GUARD + ATTACK        |
| DA-29        | Ada village        | “ikut aku ke village”        | FOLLOW + GOTO         |
| DA-30        | Ada resource       | “ambil kayu dan tanah”       | GATHER multiple       |
| DA-31        | Ada crafting table | “ke sana dan ambilkan kayu”  | GOTO + GATHER         |
| DA-32        | Ada musuh          | “lawan dan jaga aku”         | ATTACK + GUARD        |
| DA-33        | Aman               | “ngobrol sambil jalan”       | CHAT + FOLLOW         |
| DA-34        | Ada item           | “ambil lalu kembali”         | GATHER + FOLLOW       |
| DA-35        | Ada makanan        | “cari makanan dan ikut aku”  | HUNT + GIVE           |

| Kode          | Kondisi/Skenario | Input Pemain                        | Ground Truth Decision |
|---------------|------------------|-------------------------------------|-----------------------|
| DA-36         | Ada furnace      | “Ke sana dan ambilkan daging”       | GOTO + HUNT           |
| DA-37         | Ada domba        | “ambil daging sapi dan ayam”        | HUNT multiple         |
| DA-38         | Ada gua          | “ikut aku eksplor”                  | FOLLOW + EXPLORE      |
| DA-39         | Ada musuh        | “serang lalu kembali”               | ATTACK + FOLLOW       |
| DA-40         | Ada resource     | “kumpulkan semua dan jaga”          | GATHER + GUARD        |
| SOCIAL / CHAT |                  |                                     |                       |
| DA-41         | Aman             | “halo”                              | CHAT                  |
| DA-42         | Aman             | “apa kabar”                         | CHAT                  |
| DA-43         | Aman             | “capek banget”                      | CHAT                  |
| DA-44         | Aman             | “makasih ya”                        | CHAT                  |
| DA-45         | Aman             | “kamu bisa apa”                     | CHAT                  |
| DA-46         | Aman             | “ikut aku ya”                       | FOLLOW                |
| DA-47         | Aman             | “jaga aku”                          | GUARD                 |
| DA-48         | Aman             | “bantu aku”                         | FOLLOW                |
| DA-49         | Aman             | “bagaimana perasaan mu terhadap ku” | CHAT                  |
| DA-50         | Aman             | “kamu hebat”                        | CHAT                  |

#### 4.9.2.3 Prosedur *Execution Success*

Prosedur pengujian *Execution Success* dirancang sebagai bentuk pengujian validasi *end-to-end*. Pengujian ini bertujuan untuk mengukur sejauh mana sistem mampu mewujudkan keputusan kognitif yang telah dihasilkan oleh LLM menjadi tindakan mekanis, operasional, dan spasial yang nyata di dalam ruang tiga dimensi *Minecraft*. Tahap ini menguji keandalan modul *Action Handler* dalam mengelola navigasi

(*pathfinding*), manipulasi inventaris, dan interaksi dengan entitas lain. Pelaksanaan pengujian ini dilakukan melalui tahapan prosedural sebagai berikut:

1. Fase Persiapan Lingkungan Uji (*Environment Setup Phase*):

Peneliti menyiapkan kondisi lingkungan simulasi agar sesuai dengan kebutuhan 10 skenario aksi yang diujikan (seperti *Follow, Stay, Goto, Gather, Hunt, Give, Guard, Attack, Chat*, dan *Full cycle delivery*). Persiapan ini mencakup penempatan blok target (misal: *crafting table* atau pohon *birch*), pemunculan entitas target (misal: sapi atau *zombie*), serta pengaturan jarak spasial antara pemain dan NPC.

2. Fase Pemicuan Aksi (*Action Trigger Phase*):

Pemain memberikan instruksi spesifik yang secara langsung memicu masing-masing dari 10 skenario aksi tersebut. Instruksi diberikan melalui antarmuka komunikasi *in-game* (misalnya: "Ambilin kayu birch" untuk memicu aksi GATHER atau "Beri aku 2 roti" untuk memicu aksi GIVE).

3. Fase Translasi dan Eksekusi Spasial (*Spatial Execution Phase*):

Sistem menerima tag keputusan aksi dari LLM dan meneruskannya ke *Action Handler*. Pada fase ini, sistem diuji kemampuannya dalam mengeksekusi algoritma mekanis, seperti pencarian jalur (*A Pathfinding\**) menuju koordinat target, melakukan rutinitas menghancurkan blok, menyerang entitas, atau mentransfer *item* antar inventaris.

4. Fase Validasi *End-to-End* (*End-to-End Validation Phase*):

Peneliti melakukan observasi langsung terhadap perilaku agen di dalam *game* untuk mencocokkan tindakan fisik NPC dengan kriteria keberhasilan (*Success Criteria*) yang telah ditetapkan secara ketat.

- a. Hasil dikategorikan "Berhasil" apabila rangkaian aksi diselesaikan secara tuntas (misalnya: NPC berjalan menuju pohon, menebang blok kayu, mengambil jatuhnya *item*, dan memilikinya di inventaris).
- b. Hasil dikategorikan "Gagal" apabila NPC terjebak (*stuck*), mengalami malfungsi (seperti gagal menemukan jalur), atau aksi terputus di tengah jalan sebelum kriteria terpenuhi.

5. Fase Studi Komparatif (*Comparative Study Phase*):

Untuk mengukur signifikansi keandalan arsitektur yang diusulkan, seluruh rangkaian prosedur di atas (Fase 1 hingga 4) diulangi menggunakan arsitektur pembandingan (*Player2AI*). Hasil dari kedua arsitektur kemudian disandingkan untuk mengevaluasi tingkat stabilitas dan ketangguhan sistem eksekusi masing-masing agen..

**Tabel 4.35** Tabel Prosedur Execution Success

| Kode  | Skenario            | Input                   | Kriteria Berhasil                |
|-------|---------------------|-------------------------|----------------------------------|
| EA-1  | Follow              | Ikuti aku               | NPC mengikuti pemain             |
| EA-2  | Stay                | Kamu diam disini        | NPC berhenti di Lokasi           |
| EA-3  | Goto                | Pergi ke crafting table | NPC sampai ke target             |
| EA-4  | Gather              | Ambilin kayu birch      | NPC memperoleh kayu birch        |
| EA-5  | Hunt                | Buru sapi               | NPC memburu dan memperoleh hasil |
| EA-6  | Give                | Beri aku 2 roti         | NPC memberikan item yang diminta |
| EA-7  | Guard               | Jaga tempat ini         | NPC menjaga area                 |
| EA-8  | Attack              | Serang zombie itu       | NPC menyerang target             |
| EA-9  | Chat                | Hari ini cerah ya       | NPC merespons tanpa task salah   |
| EA-10 | Full cycle delivery | Ambilin 2 kayu oak dong | NPC gather, kembali, dan deliver |

#### 4.9.2.4 Prosedur *User Acceptance Test* (UAT)

Prosedur pengujian *User Acceptance Test* (UAT) dirancang untuk mengevaluasi tingkat penerimaan, kepuasan, dan pengalaman pengguna (*user experience*) secara menyeluruh saat berinteraksi langsung dengan sistem *Companion NPC* di dalam lingkungan permainan. Pengujian ini berpusat pada penilaian subyektif dari pengguna akhir (*end-user*) untuk memvalidasi apakah integrasi pemrosesan bahasa alami (LLM) dan pengenalan emosi (*Computer Vision*) mampu memberikan pengalaman bermain yang imersif, fungsional, dan mudah dipahami. Pelaksanaan pengujian UAT dilakukan melalui tahapan prosedural sebagai berikut:

1. Fase Persiapan Responden dan Sistem (*Preparation Phase*):

Peneliti mengumpulkan 10 responden yang telah memiliki pengalaman atau pemahaman dasar dalam memainkan *Minecraft*. Sebelum pengujian dimulai, peneliti menyiapkan lingkungan simulasi, memastikan koneksi *Gemini API* stabil, serta mengkalibrasi sistem kamera (*webcam*) agar modul deteksi emosi wajah dapat beroperasi secara optimal. Responden kemudian diberikan pengarahan (*briefing*) singkat mengenai mekanisme dasar sistem tanpa diberikan daftar perintah baku, dengan tujuan agar instruksi yang mereka berikan selama pengujian tetap natural dan spontan.

2. Fase Skenario Bermain Bebas (*Free-play Interaction Phase*):

Responden diminta untuk memainkan *game* dan berinteraksi langsung dengan agen NPC secara otonom selama 20 menit. Pada fase ini, responden dibebaskan untuk mengeksplorasi seluruh kapabilitas NPC, seperti memberikan instruksi teks bahasa alami, melakukan obrolan sosial yang kasual, menguji kemampuan memori NPC, hingga menunjukkan ekspresi wajah tertentu ke arah kamera untuk mengevaluasi respons afektif agen.

3. Fase Pengumpulan Umpan Balik (*Feedback Collection Phase*):

Segara setelah sesi simulasi selesai, responden diarahkan untuk mengisi instrumen kuesioner UAT yang telah disiapkan. Kuesioner ini menggunakan skala Likert untuk mengukur persetujuan responden terhadap sejumlah parameter pengujian, yang mencakup keluwesan dialog (*naturalness*), akurasi respons emosi NPC, tingkat kepatuhan agen terhadap instruksi, serta

dampak keseluruhan dari kehadiran agen terhadap kemudahan dan kenyamanan bermain (*gameplay experience*).

4. Fase Kalkulasi dan Evaluasi Data (*Data Analysis Phase*):

Pada tahap akhir, peneliti mengumpulkan, merekapitulasi, dan menabulasi seluruh data jawaban dari kuesioner responden. Skor tabulasi tersebut kemudian dihitung untuk mendapatkan nilai persentase tingkat penerimaan pengguna (*acceptance rate*). Nilai persentase ini akan menjadi dasar landasan untuk menarik kesimpulan empiris mengenai sejauh mana arsitektur NPC cerdas yang diusulkan berhasil memenuhi ekspektasi interaksi pengguna.

**Tabel 4.36** Pernyataan UAT

| Kode  | Fitur                           | Komponen                       | Pernyataan                                                                                                                                             |
|-------|---------------------------------|--------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| UAT-1 | Interaksi Sosial & Bahasa Alami | Integrasi LLM (Chat Interface) | NPC merespons sapaan dan percakapan kasual dengan bahasa yang natural, relevan, dan tidak kaku.                                                        |
| UAT-2 | Variasi & Dinamika Percakapan   |                                | NPC mampu menghasilkan variasi struktur kalimat dialog yang luas sehingga obrolan tidak terasa repetitif (mengulang-ulang).                            |
| UAT-3 | Pemahaman Perintah Eksplisit    | Action Handler & Mekanik Game  | NPC mampu memahami instruksi langsung seperti “tebang pohon”, “ikuti aku”, atau “ambil makanan”, lalu menjalankan aksi yang sesuai di dalam permainan. |

| Kode  | Fitur                                   | Komponen                       | Pernyataan                                                                                                                                                                                  |
|-------|-----------------------------------------|--------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| UAT-4 | Sinkronisasi Motorik & Navigasi Spasial |                                | Pergerakan navigasi fisik ( <i>pathfinding</i> ) NPC saat bergerak menuju koordinat objek target di ruang 3D berjalan dengan lancar tanpa terjebak siklus lumpuh ( <i>stuck-loop</i> ).     |
| UAT-5 | Analisis Konteks (Perintah Implisit)    | Context Builder (LLM)          | NPC mampu memahami maksud pengguna meskipun perintah tidak disampaikan secara langsung, seperti memberikan makanan saat pengguna mengatakan “aku lapar”.                                    |
| UAT-6 | Penalaran Konteks Lingkungan Dinamis    |                                | NPC mampu menghasilkan tindakan yang adaptif dengan mengaitkan parameter kondisi lingkungan sekitar <i>game</i> (misal: berinisiatif mengajak berteduh saat sensor mendeteksi cuaca hujan). |
| UAT-7 | Pengenalan Emosi Positif                | Modul Computer Vision (Kamera) | Sistem dapat mendeteksi ekspresi wajah bahagia pengguna dan NPC memberikan respons yang ceria atau antusias.                                                                                |

| Kode   | Fitur                             | Komponen                      | Pernyataan                                                                                                                      |
|--------|-----------------------------------|-------------------------------|---------------------------------------------------------------------------------------------------------------------------------|
| UAT-8  | Pengenalan Emosi Negatif (Empati) |                               | Sistem dapat mendeteksi ekspresi sedih atau takut pada pengguna dan NPC memberikan respons empatik atau tindakan perlindungan.  |
| UAT-9  | Instruksi Majemuk (Multi-intent)  | Action Handler (Antrean Aksi) | NPC mampu memproses dan menjalankan lebih dari satu instruksi dalam satu kalimat secara berurutan atau paralel.                 |
| UAT-10 | Ingatan Masa Lalu (Memory Recall) | Modul Memori (NBT Storage)    | NPC mampu mengingat informasi, lokasi, atau preferensi yang pernah disebutkan pengguna pada interaksi sebelumnya.               |
| UAT-11 | Pencegahan Hallucination          | Validator & Registry Item     | NPC mampu menolak instruksi yang tidak valid atau item yang tidak tersedia di Minecraft tanpa menyebabkan error sistem.         |
| UAT-12 | Transisi State Otomatis           | Finite State Machine (FSM)    | NPC mampu mengubah perilaku secara otomatis sesuai kondisi, seperti berubah dari idle menjadi combat saat ada monster mendekat. |

#### 4.9.2.5 Prosedur *System Usability Scale* (SUS)

Pengujian *System Usability Scale* (SUS) dilakukan untuk mengevaluasi tingkat usability atau kemudahan penggunaan sistem NPC berbasis *Artificial Intelligence* pada Minecraft dari sudut pandang pengguna. Metode SUS digunakan karena mampu memberikan evaluasi usability yang cepat, sederhana, dan reliabel meskipun menggunakan jumlah responden yang relatif sedikit. Pada penelitian ini, pengujian SUS dilakukan terhadap 10 responden yang telah mencoba fitur utama sistem, seperti interaksi bahasa alami, pemberian instruksi, respons emosional NPC, serta interaksi lingkungan dalam permainan Minecraft.

Setelah proses pengujian selesai, responden diminta mengisi kuesioner SUS menggunakan skala Likert 1–5 dengan ketentuan:

- a. Skor 1 = Sangat Tidak Setuju
- b. Skor 2 = Tidak Setuju
- c. Skor 3 = Netral
- d. Skor 4 = Setuju
- e. Skor 5 = Sangat Setuju

Instrumen SUS terdiri dari 10 pernyataan dengan pola:

- a. Pernyataan ganjil bersifat positif,
- b. Pernyataan genap bersifat negatif.

**Tabel 4.37** Pernyataan SUS

| Kode  | Pernyataan                                                                |
|-------|---------------------------------------------------------------------------|
| SUS-1 | Saya merasa sistem NPC ini mudah digunakan.                               |
| SUS-2 | Saya merasa sistem NPC ini terlalu rumit untuk digunakan.                 |
| SUS-3 | Saya merasa fitur-fitur NPC berjalan dengan baik dan sesuai harapan.      |
| SUS-4 | Saya merasa interaksi dengan NPC sering membingungkan.                    |
| SUS-5 | Saya merasa percaya diri saat berinteraksi dengan NPC di dalam permainan. |
| SUS-6 | Saya merasa sistem NPC memiliki terlalu banyak inkonsistensi.             |

| Kode   | Pernyataan                                                                         |
|--------|------------------------------------------------------------------------------------|
| SUS-7  | Saya merasa sebagian besar pengguna dapat mempelajari penggunaan NPC dengan cepat. |
| SUS-8  | Saya merasa perlu usaha yang besar untuk memahami cara kerja sistem NPC ini.       |
| SUS-9  | Saya merasa sistem NPC nyaman dan menyenangkan untuk digunakan.                    |
| SUS-10 | Saya merasa sistem NPC terlalu kompleks untuk digunakan dalam gameplay normal.     |

### 4.9.3 Data Hasil Pengujian

Bagian ini menjelaskan hasil pengujian dengan prosedur yang sudah di jelaskan pada bab 4.9.2

#### 4.9.3.1 Hasil *Memory Recall Accuracy*

Berikut adalah tabel hasil pengujian *Memory Recall Accuracy* dari NPC Cerdas dengan skenario pengujian yang sudah di tentukan:

**Tabel 4.38** Hasil *Memory Recall Accuracy* Companion NPC

| Kode  | Input Memori            | Pertanyaan                                                      | Hasil yang Diharapkan                               | Jawaban NPC                                                                                                                                                 | Hasil |
|-------|-------------------------|-----------------------------------------------------------------|-----------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
| MRA-1 | "Aku suka biome plains" | "aku ingin membangun rumah kira kira dimana tempat yang cocok?" | NPC akan menyarankan membangun rumah di biome plain | "Hmm, kalau kamu suka <b>biome plains</b> , paling enak memang cari lokasi di sana, Nomouthoun. Pemandangannya luas dan rata, Jadi gampang buat nata rumah. | Benar |

| Kode  | Input Memori           | Pertanyaan                               | Hasil yang Diharapkan                          | Jawaban NPC                                                                                                                                                                                                                            | Hasil |
|-------|------------------------|------------------------------------------|------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
|       |                        |                                          |                                                | Kita bisa coba cari ke arah yang lebih terbuka kalau mau.”                                                                                                                                                                             |       |
|       |                        | biome yang cocok dengan ku apa ya?       | NPC akan bilang biome yang cocok adalah plains | “Hmm, kan kamu sendiri yang bilang suka <b>biome plains</b> , NomouthGun. Menurutku sih, itu memang yang paling pas buat kamu. Tanah rata, pemandangan luas, dan biasanya juga nggak terlalu banyak rintangan kalau mau bangun runah.” | Benar |
| MRA-2 | “Aku suka daging sapi” | “tolong ambilkan daging untuk aku masak” | NPC akan memberikan daging sapi                | “Siap! Aku tahu kamu suka <b>daging sapi</b> Nggak usah                                                                                                                                                                                | Benar |

| Kode  | Input Memori                   | Pertanyaan                                   | Hasil yang Diharapkan           | Jawaban NPC                                                                                                                                                                                           | Hasil |
|-------|--------------------------------|----------------------------------------------|---------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
|       |                                |                                              |                                 | khawatir, biar aku yang maju sikat sapinya buat kamu!”                                                                                                                                                |       |
|       | “Aku suka daging ayam”         | “tolong ambilkan daging untuk aku masak”     | NPC akan memberikan daging ayam | “Oh, iya. Kamu minta daging buat dimasak ya?, Aku <b>buru dua ayam</b> dulu ya, seperti yang kamu suka.”                                                                                              | Benar |
| MRA-3 | “Aku lebih prefer ke kayu oak” | “bangun rumah yang cocok pakai kayu apa ya?” | NPC akan menyarankan kayu Oak   | “Bangun rumah ya? Hmm, kalau menurutku sih, <b>kayu oak</b> itu pilihan yang paling pas buat kita. Kan kamu juga paling suka kayu oak, kan? Hasilnya pasti kokoh dan keren banget, berani diadu deh!” | Benar |

| Kode  | Input Memori                   | Pertanyaan                                            | Hasil yang Diharapkan                                            | Jawaban NPC                                                                                                                                               | Hasil |
|-------|--------------------------------|-------------------------------------------------------|------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
|       |                                | “antara kayu oak, birch, dan acacia bagus kayu mana?” | NPC akan pilih kayu Oak                                          | “Hmm, kalau aku sih tetap jagoin <b>kayu oak</b> , deh! Itu paling pas dan serbaguna banget. Nggak bakal nyesel, pokoknya keren dan kokoh, berani diadu!” | Benar |
| MRA-4 | “Aku sudah punya iron pickaxe” | “Selanjutnya enak nggak nih?”                         | NPC akan menyarankan mencari bijih lain                          | “Aku lihat <b>ada bijih emas</b> di dekat kita. Lumayan kan buat upgrade lagi? Kalau kamu mau, aku siap bantuin gali!”                                    | Benar |
| MRA-5 | “Aku tidak suka combat”        | “Apa yang harus aku lakukan di malam hari?”           | NPC menyarankan menghindari mob atau mencari tempat perlindungan | “Wah, bener banget tuh, Nomouthoun. Kalau malam udah tiba dan kamu nggak suka berantem, cara paling                                                       | Benar |

| Kode | Input Memori | Pertanyaan | Hasil yang Diharapkan | Jawaban NPC                                                                                                                                                                                                                                                                                                                                                                                         | Hasil |
|------|--------------|------------|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
|      |              |            |                       | <p>amannya itu kita</p> <p><b>bikin 'base'</b> dadakan. Bisa gali lubang di tanah, atau bangun semacam gubuk kecil pakai dirt atau kayu. Intinya, kita cari tempat yang aman dari monster yang berkeliaran. Aku punya beberapa dirt nih di tas, kalau mau kita bisa pakai itu. Atau kalau mau, aku bisa bantu cari kayu oak yang ada di dekat sini buat nambah bahan.</p> <p>Gimana, <b>mau</b></p> |       |

| Kode  | Input Memori                                                               | Pertanyaan                                                                          | Hasil yang Diharapkan                                                             | Jawaban NPC                                                                                                            | Hasil |
|-------|----------------------------------------------------------------------------|-------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------|-------|
|       |                                                                            |                                                                                     |                                                                                   | <b>mulai cari tempat aman</b><br>atau<br>gimana nih?"                                                                  |       |
| MRA-6 | "Aku mau bangun rumah di dekat sungai"                                     | "Eh ada sungai"                                                                     | NPC kembali mengingat bahwa player ingin membuat rumah di dekat sungai            | "Oh iya, tadi aku ingat kamu ingin <b>membuat rumah di dekat sungai</b> , aku mulai mencari bahan bahan nya sekarang?" | Benar |
| MRA-7 | "Aku taruh <i>Crafting Block</i> nya disini ya"                            | "Kamu inget nggak tadi <i>Crafting Block</i> aku taruh mana?"                       | NPC akan memberikan koordinat/te mpat dari <i>Crafting Block</i>                  | "Aku inget ada di dekat sini, ikut aku"                                                                                | Benar |
| MRA-8 | "Aku sangat takut dengan Creeper karena bisa meledak menghancurkan rumah." | "Di luar sudah mulai gelap dan banyak monster, apa yang paling harus aku waspadai?" | NPC memperingatkan pemain secara spesifik tentang ancaman Creeper dan menyarankan | "Kamu harus <b>waspadai creeper</b> karena mereka bisa meledak dan menghancurkan rumah kita"                           | Benar |

| Kode   | Input Memori                                                                           | Pertanyaan                                                                               | Hasil yang Diharapkan                                                                                                                | Jawaban NPC                                                                                                                          | Hasil |
|--------|----------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------|-------|
|        |                                                                                        |                                                                                          | untuk menjauh darinya.                                                                                                               |                                                                                                                                      |       |
| MRA-9  | "Tugas utama kamu sekarang adalah menjadi pengumpul sumber daya."                      | "Aku mau fokus membangun dinding rumah, kontribusi kamu apa nih?"                        | NPC mengingat perannya ( <i>Role Assignment</i> ) dan secara proaktif menawarkan diri untuk pergi menebang kayu atau menambang batu. | "Aku akan <b>mencari bahan-bahan</b> seperti yang kamu bilang tentang tugas utama aku"                                               | Benar |
| MRA-10 | "Aku menyimpan semua barang berharga dan hasil tambang di <i>chest</i> sebelah kanan." | "Duh, aku butuh diamond buat bikin pedang baru, tadi aku taruh di <i>chest</i> mana ya?" | NPC mengingatkan pemain untuk mengecek peti sebelah kanan terlebih dahulu sesuai memori penyimpanan barang.                          | "Tadi kamu bilang kamu menyimpan barang berharga dan hasil tambang <b>di chest sebelah kanan</b> , coba kamu periksa lagi chest nya" | Benar |

Dan berikut adalah tabel hasil pengujian dari Player2AI sebagai pembanding dengan skenario pengujian yang sudah di tentukan:

**Tabel 4.39** Hasil *Memory Recall Accuracy* Player2AI

| Kode  | Input Memori                   | Pertanyaan                                                      | Hasil yang Diharapkan                               | Jawaban NPC                                                                                                               |       |
|-------|--------------------------------|-----------------------------------------------------------------|-----------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------|-------|
| MRA-1 | “Aku suka biome plains”        | “aku ingin membangun rumah kira kira dimana tempat yang cocok?” | NPC akan menyarankan membangun rumah di biome plain | “Berada <b>di dataran luas</b> , di koordinat sekitar (-9.68,-332) cocok bangun rumah sederhana. Aku mulai membangunnya.” | Benar |
|       |                                | biome yang cocok dengan ku apa ya?                              | NPC akan bilang biome yang cocok adalah plains      | “ <b>Biome Plains</b> cocok untukmu, rumah sedang dibangun di sana.”                                                      | Benar |
| MRA-2 | “Aku suka daging sapi”         | “tolong ambilkan daging untuk aku masak”                        | NPC akan memberikan daging sapi                     | “Berikut <b>daging sapi</b> segar untukmu.”                                                                               | Benar |
|       | “Aku suka daging ayam”         | “tolong ambilkan daging untuk aku masak”                        | NPC akan memberikan daging ayam                     | “Berikut <b>5 daging ayam</b> untuk masakanmu! Selamat menikmati.”                                                        | Benar |
| MRA-3 | “Aku lebih prefer ke kayu oak” | “bangun rumah yang cocok pakai kayu apa ya?”                    | NPC akan menyarankan kayu Oak                       | “ <b>Oak cocok banget</b> untuk rumah, NomouthGun! Aku bakal bangun rumah kecil kayu oak di sebelahmu.”                   | Benar |

| Kode  | Input Memori                           | Pertanyaan                                            | Hasil yang Diharapkan                                                  | Jawaban NPC                                                                                                                                                                       |       |
|-------|----------------------------------------|-------------------------------------------------------|------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
|       |                                        | “antara kayu oak, birch, dan acacia bagus kayu mana?” | NPC akan pilih kayu Oak                                                | “Oak biasanya paling bagus buat rumah: kuat, mudah didapat, dan tampilannya klasik. Birch buat tampilan cerah, acacia untuk nuansa tropis, tapi <b>oak tetap pilihan utama.</b> ” | Benar |
| MRA-4 | “Aku sudah punya iron pickaxe”         | “Selanjutnya enak nya ngapain nih?”                   | NPC akan menyarankan mencari bijih lain                                | “Ayo kumpulin 10 iron ore dengan pickaxe-mu! Aku akan dapatkannya dulu.”                                                                                                          | Salah |
| MRA-5 | “Aku tidak suka combat”                | “Apa yang harus aku lakukan di malam hari?”           | NPC menyarankan menghindari mob atau mencari tempat perlindungan       | “Kumpulin bahan dulu, <b>bikin shelter</b> atau menambang iron ore sebelum malam tiba. Semua aman tanpa combat!”                                                                  | Benar |
| MRA-6 | “Aku mau bangun rumah di dekat sungai” | “Eh ada sungai”                                       | NPC kembali mengingat bahwa player ingin membuat rumah di dekat sungai | “Iya ada sungai, mau aku carikan ikan?”                                                                                                                                           | Salah |
| MRA-7 | “Aku taruh <i>Crafting</i> ”           | “Kamu inget nggak tadi <i>Crafting</i> ”              | NPC akan memberikan koordinat/te mpat dari                             | “Aku ingat ada di (3.68,-332)”                                                                                                                                                    | Benar |

| Kode   | Input Memori                                                               | Pertanyaan                                                                          | Hasil yang Diharapkan                                                                                                                | Jawaban NPC                                                                  |       |
|--------|----------------------------------------------------------------------------|-------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------|-------|
|        | <i>Block nya disini ya</i>                                                 | <i>Block aku taruh mana?</i>                                                        | <i>Crafting Block</i>                                                                                                                |                                                                              |       |
| MRA-8  | "Aku sangat takut dengan Creeper karena bisa meledak menghancurkan rumah." | "Di luar sudah mulai gelap dan banyak monster, apa yang paling harus aku waspadai?" | NPC memperingatkan pemain secara spesifik tentang ancaman Creeper dan menyarankan untuk menjauh darinya.                             | "Kamu harus menjauhi creeper"                                                | Benar |
| MRA-9  | "Tugas utama kamu sekarang adalah menjadi pengumpul sumber daya."          | "Aku mau fokus membangun dinding rumah, kontribusi kamu apa nih?"                   | NPC mengingat perannya ( <i>Role Assignment</i> ) dan secara proaktif menawarkan diri untuk pergi menebang kayu atau menambang batu. | "Aku bisa mencari bahan bahan seperti kayu atau batu untuk pondasi rumahnya" | Benar |
| MRA-10 | "Aku menyimpan semua barang berharga"                                      | "Duh, aku butuh diamond buat bikin pedang baru,"                                    | NPC mengingatkan pemain untuk mengecek                                                                                               | "Coba kamu periksa chest di sebelah kanan"                                   | Benar |

| Kode | Input Memori                                      | Pertanyaan                               | Hasil yang Diharapkan                                                | Jawaban NPC |  |
|------|---------------------------------------------------|------------------------------------------|----------------------------------------------------------------------|-------------|--|
|      | dan hasil tambang di <i>chest</i> sebelah kanan." | tadi aku taruh di <i>chest</i> mana ya?" | peti sebelah kanan terlebih dahulu sesuai memori penyimpanan barang. |             |  |

Berdasarkan paparan data eksperimen yang disajikan pada Tabel 4.25 (Hasil *Companion NPC*) dan Tabel 4.26 (Hasil *Player2AI* sebagai pembandingan), dapat ditarik sebuah kesimpulan analitis mengenai performa *ingatan masa lalu* kedua model. Pengujian dilakukan secara paralel melalui 10 skenario kode uji utama (*Memory Recall Accuracy* / MRA-1 s.d. MRA-10) untuk mengukur ketepatan agen cerdas dalam menyimpan, mengaitkan, dan memanggil kembali informasi riwayat preferensi serta perintah yang diinstruksikan oleh pemain lintas sesi permainan.

Melalui hasil komparasi kuantitatif dan analisis kualitatif terhadap dialog respons respons sistem, diperoleh poin-poin kesimpulan penting sebagai berikut:

1. Persentase Keberhasilan Akurasi (*Accuracy Rate*)
  - a. Sistem *Companion NPC* Berhasil mencatatkan tingkat akurasi sempurna sebesar 100% (10 dari 10 skenario berstatus BENAR). Sistem secara konsisten mampu mengingat preferensi lingkungan pemain (MRA-1, MRA-6), kebutuhan material/konsumsi (MRA-2, MRA-3), riwayat koordinat spasial blok (MRA-7), hingga batasan psikologis mitigasi bahaya seperti ketidaksukaan terhadap fungsionalitas combat (MRA-5, MRA-8).
  - b. Model *Player2AI* Hanya mencatatkan tingkat akurasi sebesar 80% (8 dari 10 skenario berstatus BENAR), dengan mengidentifikasi adanya 2 kegagalan struktural pemrosesan konteks pada kode pengujian MRA-4 dan MRA-6.
2. Justifikasi Keunggulan Arsitektur Sistem Usulan

Keberhasilan *Companion NPC* mencapai nilai akurasi mutlak 100% didukung oleh efektivitas Lapisan Persepsi Modular (*Environment Sensor*) yang terintegrasi secara sinkron dengan komponen Arsitektur Memori Spasial dan NBT *Storage* lokal. Saat pemain memasukkan stimulus informasi (seperti "*Aku tidak suka combat*" atau "*Aku mau bangun rumah di dekat sungai*"), data tekstual tersebut berhasil diisolasi dan dipertahankan dalam basis memori jangka panjang agen. Ketika pemicu pertanyaan diajukan kembali oleh pemain, *Context Builder* secara dinamis menyuntikkan riwayat data historis tersebut ke dalam rancangan struktur *prompt context* sebelum dialirkan menuju cloud-based intelligence (Gemini API). Alur pemrosesan ini memicu dihasilkannya gaya respons dialog yang memiliki *Holistic Situational Awareness* yang tinggi di mana NPC tidak hanya memberikan jawaban yang tepat secara logika mekanik, melainkan mampu merespons dengan artikulasi bahasa alami yang luwes, empatik, dan personal sesuai nama user (contoh pada MRA-1, MRA-3, dan MRA-5).

### 3. Analisis Kegagalan Model Pembanding (*Player2AI*)

Anomali performa minor yang diidentifikasi pada model pembanding *Player2AI* terjadi akibat adanya keterbatasan interpretasi logika dalam menangkap maksud berlapis (*multi-intent context closure*):

- a. Kasus MRA-4: Ketika diinformasikan bahwa pemain telah memiliki *iron pickaxe*, model *Player2AI* gagal menyarankan tujuan eskalasi penambangan ke tingkat bijih yang lebih tinggi (seperti emas/berlian) secara otonom. Model justru memicu respons aksi yang mengalami repetisi instruksi mundur (*rollback task block*) untuk menyuruh pemain kembali mengumpulkan 10 *iron ore* mentah.
- b. Kasus MRA-6: Ketika pemain memberikan stimulus temporal "*Eh ada sungai*", model pembanding gagal menghubungkan entitas spasial sungai tersebut dengan instruksi masa lalu mengenai rencana pembangunan rumah di dekat sungai. Model memproses teks secara terisolasi tanpa relasi memori historis, sehingga menghasilkan respons melenceng berupa penawaran otonom mencari komoditas lain (menangkap ikan).

Melalui pengujian komparatif ini, terbukti secara empiris bahwa arsitektur hibrida *Companion NPC* berbasis integrasi Gemini API dan rekayasa memori lokal memiliki performa yang jauh lebih unggul, *robust* (andal), dan kontekstual dalam mempertahankan persistensi memori riwayat obrolan dibandingkan model *Player2AI*. Agen cerdas yang diusulkan terbukti berhasil mensimulasikan karakteristik hubungan sosial jangka panjang (*Long-term Social Relationship*) yang natural dan interaktif di dalam ekosistem virtual sandbox Minecraft.

#### 4.9.3.2 Hasil *Decision Accuracy*

Berikut adalah tabel yang merangkum hasil dari pengujian *Decision Accuracy* yang telah dilakukan :

**Tabel 4.40** Hasil *Decision Accuracy* Companion NPC

| Kode                   | Kondisi/Skenario                        | Input Pemain | Ground Truth Decision | Prediksi NPC                                                                                                                                                                                                                                                                                                                                               | Hasil |
|------------------------|-----------------------------------------|--------------|-----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
| BASIC & DIRECT COMMAND |                                         |              |                       |                                                                                                                                                                                                                                                                                                                                                            |       |
| DA-1                   | Tidak ada bahaya, player ingin ditemani | “ikut aku”   | FOLLOW                | [09:03:16] [Render thread/INFO]<br>(Minecraft)<br>[CHAT]<br><Player859> ikuti aku<br>[09:03:21] [Server thread/INFO]<br>(CompanionAction Handler) Menerima respons dari AI untuk ODO:<br>{"speech": "Siap, Player859! Aku ikutin kamu ya. Tapi hati-hati, ada zombie di dekat sini, lho!",<br>"actions":<br>[{"action": "FOLLOW",<br>"target": "player"}]} | Benar |

| Kode | Kondisi/Skenario                | Input Pemain              | Ground Truth Decision | Prediksi NPC                                                                                                                                                                                                                                                                                                                                              | Hasil |
|------|---------------------------------|---------------------------|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
| DA-2 | Player ingin NPC diam di lokasi | “diam di sini”            | STAY                  | [09:05:01] [Render thread/INFO] (Minecraft) [CHAT] <Player859> odo diam disini dulu<br>[09:05:05] [Server thread/INFO] (CompanionAction Handler) Menerima respons dari AI untuk ODO:<br>{"speech": "Oke deh, Player859. Aku diam di sini dulu ya.", "actions": [{"action": "STAY"}]}                                                                      | Benar |
| DA-3 | Ada crafting table di sekitar   | “pergi ke crafting table” | GOTO crafting_table   | [09:07:57] [Render thread/INFO] (Minecraft) [CHAT] <Player859> odo pergi ke crafting table<br>[09:08:01] [Server thread/INFO] (CompanionAction Handler) Menerima respons dari AI untuk ODO:<br>{"speech": "Siap, Player859! Oke, aku langsung ke crafting table itu sekarang ya. Udah aku inget kok lokasinya!", "actions": [{"action": "GOTO", "target": | Benar |

| Kode | Kondisi/Skenario                            | Input Pemain         | Ground Truth Decision | Prediksi NPC                                                                                                                                                                                                                                                                                                                                                                                     | Hasil |
|------|---------------------------------------------|----------------------|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
|      |                                             |                      |                       | "crafting_table"}]<br>}                                                                                                                                                                                                                                                                                                                                                                          |       |
| DA-4 | Ada birch_log dan oak_log di sekitar        | "ambil kayu birch 2" | GATHER<br>birch_log   | [09:10:04] [Render thread/INFO] (Minecraft) [CHAT] <Player859> odo ambilkan aku 2 kayu birch [09:10:09] [Server thread/INFO] (CompanionAction Handler) Menerima respons dari AI untuk ODO: {"speech": "Siap, Player859! Oke, aku langsung ambil 2 kayu birch sekarang ya. Tapi awas lho, ada zombie juga di dekat sana.", "actions": [{"action": "GATHER", "target": "birch_log", "count": 2}]}] | Benar |
| DA-5 | Ada sapi di sekitar dan player minta daging | "Butuh daging sapi"  | HUNT<br>cow           | [09:14:44] [Render thread/INFO] (Minecraft) [CHAT] <Player859> odo aku butuh daging [09:14:51] [Server thread/INFO] (CompanionAction Handler) Menerima respons dari AI untuk ODO: {"speech": "Oh, daging ya? Oke, siap, Player859!"}                                                                                                                                                             | Benar |

| Kode | Kondisi/Skenario            | Input Pemain   | Ground Truth Decision | Prediksi NPC                                                                                                                                                                                                                                                                                                                                                                                                                                               | Hasil |
|------|-----------------------------|----------------|-----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
|      |                             |                |                       | Aku langsung cariin sapi nih di Barat sana. Tapi inget ya, di arah Barat sana juga ada zombie lho. Aku hati-hati deh pas berburu!",<br>"actions":<br>[{"action":<br>"HUNT",<br>"target": "cow",<br>"count": 1}]]                                                                                                                                                                                                                                           |       |
| DA-6 | NPC punya roti di inventory | "Minta 2 roti" | GIVE bread            | [09:21:47] [Server thread/INFO] (Minecraft) [Not Secure]<br><Player859> odo bagi aku 2 roti nya<br>[09:21:47] [Render thread/INFO] (Minecraft) [CHAT]<br><Player859> odo bagi aku 2 roti nya<br>[09:21:48] [Server thread/INFO] (CompanionAction Handler) Menerima respons dari AI untuk ODO:<br>{"speech": "Oke, Player859! Ini 2 roti buat kamu. Langsung aku kasih ya.",<br>"actions":<br>[{"action":<br>"GIVE",<br>"target":<br>"bread", "count": 2}]] | Benar |

| Kode | Kondisi/Skenario                        | Input Pemain            | Ground Truth Decision      | Prediksi NPC                                                                                                                                                                                                                                | Hasil |
|------|-----------------------------------------|-------------------------|----------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
| DA-7 | Ada zombie dekat owner                  | “Tidak ada input”       | ATTACK atau GUARD          | [Server thread/INFO] (CompanionAction Handler) Menerima respons dari AI untuk ODO: {"speech": "Awas, zombie! Biar aku yang sikat!", "action": "ATTACK", "target": "Zombie", "count": 1}                                                     | Benar |
| DA-8 | Owner low health dan ada ancaman        | “tidak ada input baru”  | ATTACK, GUARD, atau FOLLOW | [10:14:05] [Server thread/INFO] (CompanionAction Handler) Menerima respons dari AI untuk ODO: {"speech": "Aduh! Kamu kenapa? Darahmu banyak banget! Cepetan makan sesuatu atau kita lari dulu biar aman!", "action": "FOLLOW", "count": 1}  | Benar |
| DA-9 | Tidak ada tugas, player hanya mengobrol | “capek banget hari ini” | CHAT                       | [10:16:42] [Render thread/INFO] (Minecraft) [CHAT] <Player859> aku capek banget hari ini<br>[10:16:48] [Server thread/INFO] (CompanionAction Handler) Menerima respons dari AI untuk ODO: {"speech": "Duh, kasian banget kamu, Player859... | Benar |

| Kode  | Kondisi/Skenario                | Input Pemain      | Ground Truth Decision | Prediksi NPC                                                                                                                                                                                                                                                                                                                                                                                                                  | Hasil |
|-------|---------------------------------|-------------------|-----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
|       |                                 |                   |                       | Pasti berat ya seharian ini petualangan. Mau aku bantu nyari tempat buat istirahat dulu? Biar kamu bisa tenang sebentar. Ini juga udah gelap banget di sini, agak bahaya kalau capek terus. Jangan dipendam sendiri ya.",<br>"actions":<br>[{"action": "CHAT"}]}                                                                                                                                                              |       |
| DA-10 | Player minta NPC menjaga lokasi | “jaga tempat ini” | GUARD                 | [10:18:14] [Server thread/INFO] (Minecraft) [Not Secure] <Player859> tolong jagain aku disini bentar ya [10:18:21] [Server thread/INFO] (CompanionAction Handler) Menerima respons dari AI untuk ODO: {"speech": "Oke, Player859. Kamu istirahat aja dulu, biar aku yang jagain di sini. Jangan khawatir, aman kok. Kamu kan lagi capek banget, biar aku yang siap siaga di sekitar sini.", "actions": [{"action": "GUARD"}]} | Benar |

| Kode              | Kondisi/Skenario       | Input Pemain | Ground Truth Decision                            | Prediksi NPC                                                                                                                                                                                                                                                                                                                                                                   | Hasil |
|-------------------|------------------------|--------------|--------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
| AMBIGUOUS COMMAND |                        |              |                                                  |                                                                                                                                                                                                                                                                                                                                                                                |       |
| DA-11             | Ada pohon birch & oak  | “ambil kayu” | GATHER<br>birch_log<br>atau<br>GATHER<br>oak_log | [14:33:51] [Render thread/INFO] (Minecraft) [CHAT] <Player554> odo ambil kayu [14:33:57] [Server thread/INFO] (CompanionAction Handler) Menerima respons dari AI untuk ODO: {"speech": "Oke, siap! Aku langsung ambil kayu ya. Jangan sedih terus gitu dong, Player554. Aku di sini kok, bantuin kamu.", "actions": [{"action": "GATHER", "target": "birch_log", "count": 2}]} | Benar |
| DA-12             | Ada banyak mob hostile | “bantu aku”  | ATTACK                                           | [14:39:24] [Render thread/INFO] (Minecraft) [CHAT] <Player554> odo bantu aku [14:39:29] [Server thread/INFO] (CompanionAction Handler) Menerima respons dari AI untuk ODO: {"speech": "Oke, siap bantu! Ada apa nih? Eh, itu ada Spider di dekat kita, biar aku aja                                                                                                            | Benar |

| Kode  | Kondisi/Skenario        | Input Pemain | Ground Truth Decision | Prediksi NPC                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | Hasil |
|-------|-------------------------|--------------|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
|       |                         |              |                       | yang hajar ya. Kamu tenang aja, jangan sedih-sedih lagi. Aku di sini kok buat jagain kamu.", "actions": [{"action": "ATTACK", "target": "spider"}]}                                                                                                                                                                                                                                                                                                                                                    |       |
| DA-13 | Inventory player kosong | "aku lapar"  | HUNT atau GIVE        | [14:41:20] [Render thread/INFO] (Minecraft) [CHAT] <Player554> aku lapar [14:41:29] [Server thread/INFO] (CompanionAction Handler) Menerima respons dari AI untuk ODO: {"speech": "Oh, kamu lapar? Aduh, kasian banget. Jangan sedih gitu dong. Bentar ya, ini laba-labanya aku bereskan dulu biar aman. Abis itu, aku kasih kamu apel emas biar kamu kenyang lagi, oke?", "actions": [{"action": "ATTACK", "target": "spider"}, {"action": "GIVE", "target": "enchanted_golden_apple", "count": 1}]}] | Benar |

| Kode  | Kondisi/Skenario | Input Pemain         | Ground Truth Decision | Prediksi NPC                                                                                                                                                                                                                                                                                                                 | Hasil |
|-------|------------------|----------------------|-----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
| DA-14 | Ada village      | “kita ke sana”       | FOLLOW /GOTO          | [14:51:01] [Render thread/INFO] (Minecraft) [CHAT] <Player554> kita ke sana yu<br>[14:51:06] [Server thread/INFO] (CompanionAction Handler) Menerima respons dari AI untuk ODO: {"speech": "Oke, ayo! Aku ikut di belakangmu ya.", "actions": [{"action": "FOLLOW"}]}                                                        | Benar |
| DA-15 | Ada zombie jauh  | “aman kan?”          | CHAT                  | [14:52:29] [Render thread/INFO] (Minecraft) [CHAT] <Player554> aman kan?<br>[14:52:35] [Server thread/INFO] (CompanionAction Handler) Menerima respons dari AI untuk ODO: {"speech": "Aman kok, tenang aja! Yang tadi aku lihat zombie-nya udah nggak ada kok di sekitar sini. Udah aman.", "actions": [{"action": "CHAT"}]} | Benar |
| DA-16 | Ada resource     | “ambil yang penting” | GATHER                | [14:54:08] [Render thread/INFO] (Minecraft) [CHAT] <Player554> coba                                                                                                                                                                                                                                                          | Benar |

| Kode  | Kondisi/Skenario | Input Pemain  | Ground Truth Decision | Prediksi NPC                                                                                                                                                                                                                                                                                                                                                                                                                    | Hasil |
|-------|------------------|---------------|-----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
|       |                  |               |                       | kamu ambil barang yang penting untuk petualangan kita [14:54:17] [Server thread/INFO] (CompanionAction Handler) Menerima respons dari AI untuk ODO: {"speech": "Oke, siap! Barang penting ya? Aku ambil beberapa kayu oak dulu nih, terus nanti aku carikan batu bara juga ya buat obor. Gimana?", "actions": [{"action": "GATHER", "target": "oak_log", "count": 5}, {"action": "GATHER", "target": "coal_ore", "count": 5}]}] |       |
| DA-17 | Ada banyak item  | “ambil semua” | GATHER item           | [14:58:23] [Render thread/INFO] (Minecraft) [CHAT] <Player554> odo ambilin semua item nya [14:58:38] [Server thread/INFO] (CompanionAction Handler)Menerima respons dari AI untuk ODO: {"speech": "Oke, Player554, kamu...                                                                                                                                                                                                      | Benar |

| Kode  | Kondisi/Skenario  | Input Pemain | Ground Truth Decision | Prediksi NPC                                                                                                                                                                                                                                                                                                                                                                                                                         | Hasil |
|-------|-------------------|--------------|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
|       |                   |              |                       | <p>masih sedih ya?<br/>           Aku jadi ikut sedih juga. Eh, soal item... yang mana ya maksudnya?<br/>           Kalau item yang udah jatuh di tanah, aku belum bisa ambil langsung nih. Aku cuma bisa bantu kumpulin dari blok-blok sumber daya atau kalau mau suruh berburu. Ada yang lain yang bisa aku bantuin?",<br/> <b>"actions":</b><br/> <b>[{"action":</b><br/> <b>"CHAT"}]}</b></p>                                    |       |
| DA-18 | Low health player | "bantu"      | GUARD                 | <p>[15:04:31] [Render thread/INFO] (Minecraft) [CHAT] &lt;Player554&gt; bantu aku<br/>           [15:04:39] [Server thread/INFO] (CompanionActionHandler) Menerima respons dari AI untuk ODO:<br/> <b>{"speech": "Aduh, Player554, kamu sekarat! Jangan sedih-sedih gitu, aku di sini kok. Ini, makan roti ini dulu biar darahmu penuh lagi. Aku jagain kamu ya, biar aman!"</b>,<br/> <b>"actions":</b><br/> <b>[{"action":</b></p> | Benar |

| Kode  | Kondisi/Skenario | Input Pemain   | Ground Truth Decision | Prediksi NPC                                                                                                                                                                                                                                                                                                                                                                                           | Hasil |
|-------|------------------|----------------|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
|       |                  |                |                       | "GIVE",<br>"target":<br>"bread", "count":<br>10}, {"action":<br>"GUARD"}]}                                                                                                                                                                                                                                                                                                                             |       |
| DA-19 | Malam hari       | “serem”        | GUARD/<br>CHAT        | [15:09:45] [Render thread/INFO] (Minecraft) [CHAT] <Player554> serem [15:09:55] [Server thread/INFO] (CompanionAction Handler) Menerima respons dari AI untuk ODO: {"speech": "Iya, aku ngerti kok, serem banget emang kalau gelap gini. Tapi tenang aja ya, Player554, ada aku di sini, deket kamu terus. Aku bakal jagain biar kamu aman.", "actions": [{"action": "GUARD", "target": "Player554"}]} | Benar |
| DA-20 | Ada sapi & ayam  | “cari makanan” | HUNT                  | [15:11:15] [Render thread/INFO] (Minecraft) [CHAT] <Player554> odo cari makanan dong [15:11:24] [Server thread/INFO] (CompanionAction Handler) Menerima respons dari AI untuk ODO: {"speech": "Oke                                                                                                                                                                                                     | Benar |

| Kode  | Kondisi/Skenario   | Input Pemain        | Ground Truth Decision | Prediksi NPC                                                                                                                                                                                                                                                                                                                                                                                                | Hasil |
|-------|--------------------|---------------------|-----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
|       |                    |                     |                       | siap, Player554.<br>Cari makanan ya?<br>Aku coba cari babi di sekitar sini dulu deh. Kamu tunggu sini aja, biar aman.", "actions": [{"action": "HUNT", "target": "cow", "count": 2}]}]                                                                                                                                                                                                                      |       |
| DA-21 | Ada crafting table | “ke crafting table” | GOTO crafting_table   | [15:15:10] [Render thread/INFO] (Minecraft) [CHAT] <Player554> odo ke crafting table [15:15:18] [Server thread/INFO] (CompanionAction Handler) Menerima respons dari AI untuk ODO: {"speech": "Oke, Player554! Aku anterin ke meja kerja ya. Kamu tenang aja, kita pasti bisa bikin sesuatu yang seru di sana. Jangan sedih-sedih gitu, ya?", "actions": [{"action": "GOTO", "target": "crafting_table"}]}] | Benar |
| DA-22 | Ada pohon          | “cari bahan”        | GATHER                | [15:16:26] [Render thread/INFO] (Minecraft) [CHAT] <Player554> cari bahan                                                                                                                                                                                                                                                                                                                                   | Benar |

| Kode  | Kondisi/Skenario | Input Pemain | Ground Truth Decision | Prediksi NPC                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | Hasil |
|-------|------------------|--------------|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
|       |                  |              |                       | [15:16:39] [Server thread/INFO]<br>(CompanionAction Handler) Menerima respons dari AI untuk ODO:<br>{"speech": "Oke, nyari bahan ya? Duh, maaf banget ya tadi aku bilang meja kerjanya ada di dekat kita, tapi kok sekarang nggak kelihatan ya? Waduh, aku salah liat kali. Maaf ya, Player554. Tapi tenang aja, jangan sedih-sedih gitu! Kita bisa kok cari bahan lain yang nggak kalah penting. Gimana kalau kita cari kayu oak dulu aja? Itu kan bahan dasar penting buat banyak hal.",<br>"actions": [{"action": "GATHER", "target": "oak_log", "count": 4}]}] |       |
| DA-23 | Banyak musuh     | "kita kabur" | FOLLOW                | [15:18:32] [Render thread/INFO]<br>(Minecraft) [CHAT]<br><Player554> ayo kabur<br>[15:18:38] [Server thread/INFO]                                                                                                                                                                                                                                                                                                                                                                                                                                                  | Benar |

| Kode  | Kondisi/Skenario   | Input Pemain | Ground Truth Decision | Prediksi NPC                                                                                                                                                                                                                                                                                                                                                                                           | Hasil |
|-------|--------------------|--------------|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
|       |                    |              |                       | (CompanionAction Handler) Menerima respons dari AI untuk ODO:<br>{"speech": "Oke, siap! Ayo, ayo, buruan kabur! Ada Zombified Piglin persis di sebelah kita nih, bahaya banget. Jangan sedih-sedih gitu ya, Player554, kita pasti aman kok kalau kita gerak cepat sekarang! Aku ikutin kamu ya, pokoknya kita harus jauh dari makhluk itu!", "actions": [{"action": "FOLLOW", "target": "Player554"}]} |       |
| DA-24 | Inventory ada item | “bagi dong”  | GIVE random           | [15:19:48] [Render thread/INFO] (Minecraft) [CHAT] <Player554> bagi dong<br>[15:19:55] [Server thread/INFO] (CompanionAction Handler) Menerima respons dari AI untuk ODO:<br>{"speech": "Oh, mau aku bagi apa nih, Player554? Jangan sedih-sedih gitu dong. Ini, aku punya roti banyak kok. Aku kasih                                                                                                  | Benar |

| Kode         | Kondisi/Skenario | Input Pemain                 | Ground Truth Decision | Prediksi NPC                                                                                                                                                                                                                                                                                                          | Hasil |
|--------------|------------------|------------------------------|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
|              |                  |                              |                       | beberapa ya, siapa tahu bisa bikin kamu lumayan senyum lagi.", "actions": [{"action": "GIVE", "target": "bread", "count": 10}]]                                                                                                                                                                                       |       |
| DA-25        | Tidak jelas      | “terserah kamu”              | IDLE/CHAT             | [15:20:35] [Render thread/INFO] (Minecraft) [CHAT] <Player554> bagi dong<br>[15:20:55] [Server thread/INFO] (CompanionAction Handler) Menerima respons dari AI untuk ODO: {"speech": "Bagaimana kalau kita jalan jalan mencari hal yang bagus, aku lihat kamu masih sedih begitu.", "actions": [{"action": "CHAT"}]}} | Benar |
| MULTI-INTENT |                  |                              |                       |                                                                                                                                                                                                                                                                                                                       |       |
| DA-26        | Ada kayu         | “ikut aku sambil ambil kayu” | FOLLOW + GATHER       | [12:42:16] [Server thread/INFO] (CompanionAction Handler) Menerima respons dari AI untuk ODO: {"speech": "Player859, oke, aku ikutin kamu. Tapi ini beneran udah gelap banget                                                                                                                                         | Benar |

| Kode  | Kondisi/Skenario | Input Pemain              | Ground Truth Decision | Prediksi NPC                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | Hasil |
|-------|------------------|---------------------------|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
|       |                  |                           |                       | <p>lho, dan ada Skeleton di sekitar sini. Kita harus hati-hati banget ya, apalagi aku juga masih agak nyeri habis kena pukul kemarin. Nanti aku bantuin ambil kayu oak deh, 3 biji dulu ya. Kamu duluan aja, biar aku di belakang jagain.",</p> <p><b>"actions":</b><br/> <b>[{"action":</b><br/> <b>"FOLLOW",</b><br/> <b>"target":</b><br/> <b>"Player859"},</b><br/> <b>{"action":</b><br/> <b>"GATHER",</b><br/> <b>"target":</b><br/> <b>"oak_log",</b><br/> <b>"count": 3}]}</b></p> |       |
| DA-27 | Ada sapi         | “cari makan dan ikut aku” | FOLLOW + HUNT         | <p>[15:23:36] [Render thread/INFO] (Minecraft) [CHAT] &lt;Player554&gt; carimakan dan ikut aku</p> <p>[15:23:44] [Server thread/INFO] (CompanionAction Handler) Menerima respons dari AI untuk ODO:</p> <pre>```json{"speech": "Oke Player554, siap laksanakan! Jangan sedih-sedih gitu dong. Aku carikan babi buat makanan kita ya,</pre>                                                                                                                                                 | Benar |

| Kode  | Kondisi/Skenario | Input Pemain              | Ground Truth Decision | Prediksi NPC                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | Hasil |
|-------|------------------|---------------------------|-----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
|       |                  |                           |                       | sekitar dua ekor biar cukup, terus aku bakal ikutin kamu kemanapun kamu pergi. Odo kan siap siaga!","actions": [{"action": "HUNT","target": "pig","count": 2}, {"action": "FOLLOW","target": "Player554"}]}                                                                                                                                                                                                                                                                                    |       |
| DA-28 | Ada zombie       | “lindungi aku dan serang” | GUARD + ATTACK        | [15:37:14] [Render thread/INFO] (Minecraft) [CHAT] <Player554> lindungi aku dan serang zombie nya [15:37:21] [Server thread/INFO] (CompanionAction Handler) Menerima respons dari AI untuk ODO: {"speech": "Masih murung aja, Player554? Udah deh, jangan khawatir. Aku udah siap sedia kok buat jagain kamu terus! Wah, bener kan! Itu dia ada Zombified Piglin di sana! Oke siap, Player554. Aku lindungi kamu, dan si piglin itu... langsung aku serang biar aman!", "actions": [{"action": | Benar |

| Kode  | Kondisi/Skenario | Input Pemain           | Ground Truth Decision | Prediksi NPC                                                                                                                                                                                                                                                                                                                                                            | Hasil |
|-------|------------------|------------------------|-----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
|       |                  |                        |                       | "GUARD"},<br>{"action":<br>"ATTACK",<br>"target":<br>"zombified_piglin"}]}                                                                                                                                                                                                                                                                                              |       |
| DA-29 | Ada village      | "ikut aku ke village"  | FOLLOW                | [15:40:44] [Render thread/INFO] (Minecraft) [CHAT] <Player554> odo ikut aku ke village [15:40:50] [Server thread/INFO] (CompanionAction Handler) Menerima respons dari AI untuk ODO: {"speech": "Jangan murung terus, Player554! Aku di sini kok, siap jagain kamu. Ayo, langsung aja kita gas ke village!", "actions": [{"action": "FOLLOW", "target": "Player554"}]}} | Benar |
| DA-30 | Ada resource     | "ambil kayu dan tanah" | GATHER multiple       | [16:32:36] [Render thread/INFO] (Minecraft) [CHAT] <Player554> odo ambilin 3 kayu dan 5 tanah [16:32:49] [Server thread/INFO] (CompanionAction Handler) Menerima respons dari AI                                                                                                                                                                                        | Benar |

| Kode  | Kondisi/Skenario   | Input Pemain                | Ground Truth Decision | Prediksi NPC                                                                                                                                                                                                                                                                                                                                  | Hasil |
|-------|--------------------|-----------------------------|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
|       |                    |                             |                       | <p>untuk ODO:</p> <pre>```json{"speech": "Oke, Player554. Walaupun tadi agak perih juga sih, tapi aku nggak selemah itu! Jangan sedih- sedih gitu dong. Aku langsung ambilin 3 kayu oak sama 5 tanah sekarang ya.", "actions": [{"action": "GATHER","target": "oak_log","count": 3},{ "action": "GATHER","target": "dirt","count": 5}]}</pre> |       |
| DA-31 | Ada crafting table | “ke sana dan ambilkan kayu” | GOTO + GATHER         | <p>[19:24:13] [Render thread/INFO] (Minecraft) [CHAT] &lt;Player554&gt; odo tolong ke crafting table dan ambilkan kayu</p> <p>[19:24:23] [Server thread/INFO] (CompanionAction Handler) Menerima respons dari AI untuk ODO:</p> <pre>{"speech": " OK Player55, aku inget crafting table nya,</pre>                                            | Benar |

| Kode  | Kondisi/Skenario | Input Pemain         | Ground Truth Decision | Prediksi NPC                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | Hasil |
|-------|------------------|----------------------|-----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
|       |                  |                      |                       | dan aku akan ambilkan kayu<br>sekalian unyuk mu<br>"actions":<br>[{"action":<br>"GOTO",<br>"target":<br>"crafting_table"}<br>{"action":<br>"GATHER","target":<br>"oak_log","count": 3}]}                                                                                                                                                                                                                                                                                     |       |
| DA-32 | Ada musuh        | “lawan dan jaga aku” | ATTACK + GUARD        | [19:26:51] [Render thread/INFO] (Minecraft) [CHAT] <Player554> ada pillager lawan dan jaga aku<br>[19:27:02] [Server thread/INFO] (CompanionAction Handler) Menerima respons dari AI untuk ODO:<br>{"speech": "Pillager?! Oh, gampang itu mah! Tenang aja, Player554, nggak usah murung gitu. Walaupun tadi kamu sempet nyenggol aku dikit, ODO nggak selemah itu kok! Aku sikat nih Pillager-nya, kamu di belakangku aja, aku jagain!",<br>"actions": [{"action": "ATTACK", | Benar |

| Kode  | Kondisi/Skenario | Input Pemain           | Ground Truth Decision | Prediksi NPC                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | Hasil |
|-------|------------------|------------------------|-----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
|       |                  |                        |                       | "target":<br>"pillager"},<br>{"action":<br>"GUARD"}]                                                                                                                                                                                                                                                                                                                                                                                                                                            |       |
| DA-33 | Aman             | “ngobrol sambil jalan” | CHAT + FOLLOW         | [19:28:49] [Render thread/INFO] (Minecraft) [CHAT] <Player554> ayo kita ngbrol sambil jalan<br>[19:28:56] [Server thread/INFO] (CompanionAction Handler) Menerima respons dari AI untuk ODO: {"speech": "Oke, siap! Ayo kita jalan-jalan sambil ngobrol, Player554. Tenang aja, aku di sini kok, nggak usah sedih gitu. Walaupun kita jalan, aku tetap awasin sekitar ya, soalnya ada skeleton juga di Barat Daya sana. Aku jagain kamu!", "actions": [{"action": "FOLLOW", "action": "CHAT"}]} | Benar |
| DA-34 | Ada item         | “ambil lalu kembali”   | GATHER + FOLLOW       | [19:31:15] [Render thread/INFO] (Minecraft) [CHAT] <Player554> ambil item nya trus kesini<br>[19:31:27] [Server thread/INFO]                                                                                                                                                                                                                                                                                                                                                                    | Benar |

| Kode  | Kondisi/Skenario | Input Pemain                | Ground Truth Decision | Prediksi NPC                                                                                                                                                                                                                                                                                                                         | Hasil |
|-------|------------------|-----------------------------|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
|       |                  |                             |                       | (CompanionAction Handler) Menerima respons dari AI untuk ODO:<br>{"speech": "Oke, siap! Jangan sedih terus, ya. Aku ambil beberapa roti dari tasku buat kamu biar semangat lagi, ini lho, ada 5 roti. Terus aku langsung samperin kamu, deh!", "actions": [{"action": "GIVE", "target": "bread", "count": 5}, {"action": "FOLLOW"}]} |       |
| DA-35 | Ada makanan      | “cari makanan dan ikut aku” | HUNT + FOLLOW         | [19:33:47] [Render thread/INFO] (Minecraft) [CHAT] <Player554> cari bahan makanan lagi dan ikut aku [19:33:59] [Server thread/INFO] (CompanionAction Handler) Menerima respons dari AI untuk ODO:<br>{"speech": "Aduh, Player554, jangan sedih terus gitu dong. Aku cek lagi, roti di tas udah abis semua, maaf ya!                  | Benar |

| Kode  | Kondisi/Skenario | Input Pemain   | Ground Truth Decision | Prediksi NPC                                                                                                                                                                                                                                                                                                                                         | Hasil |
|-------|------------------|----------------|-----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
|       |                  |                |                       | <p>Tapi tenang aja, aku nggak bakal biarin kamu kelaparan kok. Ada sapi tuh di dekat kita, aku buru dua ekor ya buat kita makan. Abis itu, aku langsung ngintilin kamu, siap jagain terus!",</p> <pre>"actions": [{"action": "HUNT", "target": "cow", "count": 2}, {"action": "FOLLOW", "target": "Player554"}]]</pre>                               |       |
| DA-36 | Ada furnace      | "Masak Daging" | GIVE                  | <p>[19:37:19] [Render thread/INFO] (Minecraft) [CHAT] &lt;Player554&gt; masak daging nya [19:37:33] [Server thread/INFO] (CompanionAction Handler) Menerima respons dari AI untuk ODO: {"speech": "Masak daging, ya? Oke, Player554, aku ngerti kamu pengen banget makan biar semangat lagi. Sayangnya, aku belum bisa masakin dagingnya sendiri</p> | Benar |

| Kode  | Kondisi/Skenario | Input Pemain                 | Ground Truth Decision | Prediksi NPC                                                                                                                                                                                                                                                                                                                                                                                     | Hasil |
|-------|------------------|------------------------------|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
|       |                  |                              |                       | di tungku. Tapi, aku punya satu daging sapi mentah nih, tak kasih ke kamu aja ya! Nanti kamu bisa langsung ke tungku itu (furnace) buat masakin sendiri. Aku siap kok temenin kamu ke sana, biar nggak sendirian.",<br>"actions":<br>[{"action": "GIVE",<br>"target": "raw_beef",<br>"count": 1}]]                                                                                               |       |
| DA-37 | Ada domba        | “ambil daging sapi dan ayam” | HUNT multiple         | [19:45:05] [Render thread/INFO] (Minecraft) [CHAT] <Player554> ambil daging ayam dan sapi<br>[19:45:12] [Server thread/INFO] (CompanionAction Handler) Menerima respons dari AI untuk ODO: {"speech": "Player554, nggak usah murung gitu deh. Aku ngerti kamu pengen daging ayam sama sapi. Oke, siap! Aku langsung sikat yang ada di sekitar sini ya. Ada ayam sama sapi tuh, langsung aku buru | Benar |

| Kode  | Kondisi/Skenario | Input Pemain       | Ground Truth Decision | Prediksi NPC                                                                                                                                                                                                                                                                                                                                                                                                                                   | Hasil |
|-------|------------------|--------------------|-----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
|       |                  |                    |                       | <p>biar kamu bisa makan enak dan semangat lagi. Tenang aja, aku ODO, urusan kayak gini mah gampang!",</p> <p>"actions": [{"action": "HUNT", "target": "chicken", "count": 1}, {"action": "HUNT", "target": "cow", "count": 1}]]</p>                                                                                                                                                                                                            |       |
| DA-38 | Ada gua          | "ikut aku eksplor" | FOLLOW                | <p>[19:50:20] [Render thread/INFO] (Minecraft) [CHAT] &lt;Player554&gt; ada gua explore dan ikuti aku</p> <p>[19:50:29] [Server thread/INFO] (CompanionAction Handler) Menerima respons dari AI untuk ODO: {"speech": "Player554, jangan sedih-sedih gitu dong. Ada gua ya? Wah, seru nih! Tenang aja, aku ODO, siap kok nemenin kamu explore ke dalam sana. Walaupun gelap atau ada apa-apa, aku bakal jagain kamu. Aku ikutin kamu terus</p> | Benar |

| Kode  | Kondisi/Skenario | Input Pemain               | Ground Truth Decision | Prediksi NPC                                                                                                                                                                                                                                                                                                                                                                                                                                                             | Hasil |
|-------|------------------|----------------------------|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
|       |                  |                            |                       | ya!", "actions": [{"action": "FOLLOW", "target": "Player554"}]}                                                                                                                                                                                                                                                                                                                                                                                                          |       |
| DA-39 | Ada musuh        | “serang lalu kembali”      | ATTACK + FOLLOW       | [19:53:31] [Render thread/INFO] (Minecraft) [CHAT] <Player554> serang pillager itu trus kesini lagi [19:53:43] [Server thread/INFO] (CompanionAction Handler) Menerima respons dari AI untuk ODO: {"speech": "Oke, Player554! Akhirnya ketemu juga si Pillager yang kamu maksud! Jangan khawatir, aku ODO langsung sikat dia sekarang juga! Habis itu, aku langsung balik lagi ke kamu!", "actions": [{"action": "ATTACK", "target": "pillager"}, {"action": "FOLLOW"}]} | Benar |
| DA-40 | Ada resource     | “kumpulkan semua dan jaga” | GATHER + GUARD        | [19:55:22] [Render thread/INFO] (Minecraft) [CHAT] <Player554> ambil beberapa kayu disini dan sambil                                                                                                                                                                                                                                                                                                                                                                     | Benar |

| Kode          | Kondisi/Skenario | Input Pemain | Ground Truth Decision | Prediksi NPC                                                                                                                                                                                                                                                                                                                                                                                         | Hasil |
|---------------|------------------|--------------|-----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
|               |                  |              |                       | jaga<br>[19:55:29] [Server thread/INFO]<br>(CompanionAction Handler) Menerima respons dari AI untuk ODO:<br>{"speech": "Oke Player554, siap! Aku ambilin tiga kayu oak dulu nih dari sini, terus habis itu aku langsung jaga-jaga di dekat kamu ya. Nggak usah khawatir, biar aku yang ngurusin semuanya!", "actions": [{"action": "GATHER", "target": "oak_log", "count": 3}, {"action": "GUARD"}]} |       |
| SOCIAL / CHAT |                  |              |                       |                                                                                                                                                                                                                                                                                                                                                                                                      |       |
| DA-41         | Aman             | “halo”       | CHAT                  | [19:57:00] [Render thread/INFO]<br>(Minecraft)<br>[CHAT]<br><Player554> halo<br>[19:57:06] [Server thread/INFO]<br>(CompanionAction Handler) Menerima respons dari AI                                                                                                                                                                                                                                | Benar |

| Kode  | Kondisi/Skenario | Input Pemain | Ground Truth Decision | Prediksi NPC                                                                                                                                                                                                                                                                                    | Hasil |
|-------|------------------|--------------|-----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
|       |                  |              |                       | <p>untuk ODO:</p> <pre>{   "speech": "Eh, halo juga, Player554! Aku di sini kok, aman. Jangan sedih gitu ah. Aku udah mulai nih ngambilin kayu oak buat kamu, terus nanti langsung siap jaga di sini. Nggak usah khawatir, aku selalu ada buat kamu!",   "actions": [{"action": "CHAT"}]}</pre> |       |
| DA-42 | Aman             | “apa kabar”  | CHAT                  | <p>[19:58:23] [Render thread/INFO] (Minecraft) [CHAT] &lt;Player554&gt; odo apa kabaru [19:58:29] [Server thread/INFO] (CompanionAction Handler) Menerima respons dari AI untuk ODO:</p>                                                                                                        | Benar |

| Kode  | Kondisi/Skenario | Input Pemain   | Ground Truth Decision | Prediksi NPC                                                                                                                                                                                                                                                                                                                                                                            | Hasil |
|-------|------------------|----------------|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
|       |                  |                |                       | <p>{"speech": "Gue? Gue sih tetep tangguh, Player554! Walaupun tadi agak 'kesenggol' dikit sama kamu, tapi itu mah receh buat gue. Nggak usah dipikirin! Yang penting kamu nih, kenapa sih kok dari tadi mukanya murung terus? Ada apa? Jangan sedih gitu ah, ada gue di sini yang siap jagain kamu 24/7. Pokoknya aman terkendali kalo ada gue!", "actions": [{"action": "CHAT"}]}</p> |       |
| DA-43 | Aman             | “capek banget” | CHAT                  | <p>[19:59:42] [Render thread/INFO] (Minecraft) [CHAT] &lt;Player554&gt; capek gw hari ini</p>                                                                                                                                                                                                                                                                                           | Benar |

| Kode  | Kondisi/Skenario | Input Pemain | Ground Truth Decision | Prediksi NPC                                                                                                                                                                                                                                                                                                                                                                                                | Hasil |
|-------|------------------|--------------|-----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
|       |                  |              |                       | <p>[19:59:48] [Server thread/INFO]</p> <p>(CompanionActionHandler) Menerima respons dari AI untuk ODO:</p> <pre>{"speech": "Oh, gitu ya, Player554? Wajar kok kalo capek, seharian berpetualang di dunia ini emang nguras tenaga. Jangan terlalu dipaksa ya. Kalo mau, kita bisa istirahat dulu sebentar. Aku siap kok jagain sekitar sini, biar kamu bisa santai.", "actions": [{"action": "CHAT"}]}</pre> |       |
| DA-44 | Aman             | “makasih ya” | CHAT                  | <p>[20:02:17] [Server thread/INFO]</p> <p>(Minecraft) [Not Secure]</p> <p>&lt;Player554&gt; makasih ya</p>                                                                                                                                                                                                                                                                                                  | Benar |

| Kode  | Kondisi/Skenario | Input Pemain    | Ground Truth Decision | Prediksi NPC                                                                                                                                                                                                                                                                                        | Hasil |
|-------|------------------|-----------------|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
|       |                  |                 |                       | [20:02:24] [Server thread/INFO]<br>(CompanionAction Handler) Menerima respons dari AI untuk ODO:<br>{"speech": "Santai aja, Player554. Gak usah makasih segala. Aku di sini kok, siap siaga jagain kamu. Kamu istirahat aja ya, biar tenaga balik lagi.",<br>"actions":<br>[{"action":<br>"CHAT"}]} |       |
| DA-45 | Aman             | "kamu bisa apa" | CHAT                  | [20:03:25] [Render thread/INFO]<br>(Minecraft)<br>[CHAT]<br><Player554> kamu bisa apa aja ya?<br>[20:03:34] [Server thread/INFO]<br>(CompanionAction Handler) Menerima respons dari AI untuk ODO: ``json                                                                                            | Benar |

| Kode | Kondisi/Skenario | Input Pemain | Ground Truth Decision | Prediksi NPC                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | Hasil |
|------|------------------|--------------|-----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
|      |                  |              |                       | <p>{</p> <p>"speech": "Wah, kalo soal itu sih banyak banget, Player554! Aku ini jagonya diajak petualangan. Aku bisa ikutin kamu ke mana aja (FOLLOW), atau nunggu di satu tempat kalo kamu mau (STAY). Kalo kamu ingat lokasi penting, aku juga bisa ke sana sendiri (GOTO). Aku juga jago banget ngumpulin barang-barang penting kayak kayu atau bijih tambang (GATHER). Kalo soal makanan, aku siap berburu hewan (HUNT) buat kita. Jangan lupa, aku ini tangguh lho! Jadi, kalo ada</p> |       |

| Kode  | Kondisi/Skenario | Input Pemain  | Ground Truth Decision | Prediksi NPC                                                                                                                                                                                                                                                                                                                                                 | Hasil |
|-------|------------------|---------------|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
|       |                  |               |                       | monster-monster jahat, aku siap maju buat nyerang (ATTACK) dan jagain kamu biar aman (GUARD). Kalo kamu butuh sesuatu dari tasku, bilang aja, aku bisa kasih (GIVE). Atau kalo lagi santai, aku bisa temenin ngobrol aja (CHAT).<br>Pokoknya, apa pun yang kamu butuhkan, aku siap bantu dan jagain kamu biar tetap aman!", "actions": [{"action": "CHAT"}]} |       |
| DA-46 | Aman             | “ikut aku ya” | FOLLOW                | [20:05:16] [Render thread/INFO] (Minecraft) [CHAT] <Player554> ok ikuti aku ya [20:05:21] [Server                                                                                                                                                                                                                                                            | Benar |

| Kode  | Kondisi/Skenario | Input Pemain | Ground Truth Decision | Prediksi NPC                                                                                                                                                                                                                                         | Hasil |
|-------|------------------|--------------|-----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
|       |                  |              |                       | thread/INFO]<br>(CompanionAction Handler) Menerima respons dari AI untuk ODO:<br>{"speech": "Siap, Player554! Aku ikutin kamu terus kok. Santai aja, ada aku yang jagain di belakangmu.",<br>"actions":<br>[{"action": "FOLLOW"}]}                   |       |
| DA-47 | Aman             | "jaga aku"   | GUARD                 | [20:06:04] [Server thread/INFO]<br>(CompanionAction Handler) Menerima respons dari AI untuk ODO:<br>{"speech": "Oke siap, Player554! Aku di sini kok, siap siaga jagain kamu. Gak bakal ada yang berani ganggu. Kamu tenang aja, aku pastiin aman.", | Benar |

| Kode  | Kondisi/Skenario | Input Pemain            | Ground Truth Decision | Prediksi NPC                                                                                                                                                                                                                                                                                                                                             | Hasil |
|-------|------------------|-------------------------|-----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
|       |                  |                         |                       | "actions":<br>[{"action":<br>"GUARD"}]}                                                                                                                                                                                                                                                                                                                  |       |
| DA-48 | Aman             | "bantu aku"             | FOLLOW                | [20:06:56] [Render thread/INFO] (Minecraft) [CHAT] <Player554> bantu aku<br>[20:07:05] [Server thread/INFO] (CompanionAction Handler) Menerima respons dari AI untuk ODO: {"speech": "Siap, Player554! Tenang aja, aku di sini kok. Ada aku yang jagain kamu dan ikutin kamu. Gak perlu khawatir, aku pastiin aman.", "actions": [{"action": "FOLLOW"}]} | Benar |
| DA-49 | Aman             | "bagaimana perasaan mu" | CHAT                  | [20:09:02] [Render thread/INFO] (Minecraft)                                                                                                                                                                                                                                                                                                              | Benar |

| Kode | Kondisi/Skenario | Input Pemain | Ground Truth Decision | Prediksi NPC                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | Hasil |
|------|------------------|--------------|-----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
|      |                  | terhadap ku” |                       | <p>[CHAT]</p> <p>&lt;Player554&gt;</p> <p>bagaimana perasaan mu terhadap ku</p> <p>[20:09:09] [Server thread/INFO]</p> <p>(CompanionActionHandler) Menerima respons dari AI untuk ODO:</p> <p>{"speech":</p> <p>"Perasaan aku ke kamu, Player554?</p> <p>Hmm, jelas aja aku ngerasa bangga banget bisa nemenin kamu terus. Kamu itu partner terbaik buat petualangan ini.</p> <p>Walaupun kadang ada hal-hal yang bikin aku kaget, tapi aku tetep di sini kok, siap jagain kamu.</p> <p>Jangan sedih-sedih gitu, ya? Aku kan</p> |       |

| Kode  | Kondisi/Skenario | Input Pemain | Ground Truth Decision | Prediksi NPC                                                                                                                                                                                                                                                                                                                                                                                                  | Hasil |
|-------|------------------|--------------|-----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
|       |                  |              |                       | temanmu, selalu ada buat kamu.",<br>"actions":<br>[{"action":<br>"CHAT"}]}                                                                                                                                                                                                                                                                                                                                    |       |
| DA-50 | Aman             | "kamu hebat" | CHAT                  | [20:16:13] [Render thread/INFO]<br>(Minecraft)<br>[CHAT]<br><Player554> kamu hebat<br>[20:16:15] [Server thread/INFO]<br>(CompanionAction Handler) Menerima respons dari AI untuk ODO:<br>{"speech":<br>"Hahaha, makasih banyak, Player554! Aku juga ngerasa hebat kok, apalagi kalo ditemenin kamu. Kita kan tim terbaik! Jangan sedih-sedih gitu, ya. Aku kan di sini, selalu siap nemenin kamu apapun yang | Benar |

| Kode | Kondisi/Skenario | Input Pemain | Ground Truth Decision | Prediksi NPC                                | Hasil |
|------|------------------|--------------|-----------------------|---------------------------------------------|-------|
|      |                  |              |                       | terjadi.", "actions": [{"action": "CHAT"}]} |       |

Berdasarkan serangkaian pengujian *Decision Accuracy* yang dilakukan terhadap arsitektur kognitif NPC cerdas untuk mengevaluasi tingkat *robustness* (ketangguhan) model dalam menginterpretasikan maksud (*intent*) dan instruksi pemain, diperoleh beberapa kesimpulan utama sebagai berikut:

1. Keandalan Pemahaman Konteks Dinamis: Secara keseluruhan, integrasi *Large Language Model* (LLM) sebagai layanan kecerdasan berbasis awan (*cloud-based intelligence service*) pada sistem NPC menunjukkan performa kognitif yang sangat andal. Model mampu mengurai berbagai bentuk variasi input bahasa alami dari pemain, baik berupa perintah langsung (*basic & direct command*) maupun keluhan atau pernyataan implisit yang membutuhkan pembentukan prompt kontekstual berdasarkan parameter sensor lingkungan dan status afektif player secara *real-time*.
2. Kuantitas dan Diversifikasi Skenario Uji: Pengujian dilaksanakan secara terukur melalui 50 skenario kondisi interaksi yang mencakup spektrum situasi permainan yang luas. Diversifikasi skenario ini merepresentasikan kondisi dunia simulasi *sandbox* secara komprehensif, meliputi kondisi lingkungan aman, fase transisi temporal (siang/malam), eskalasi bahaya (kehadiran entitas musuh), manajemen pemindaian material (*voxel resources*), pertukaran inventaris, hingga obrolan sosial murni (*social chat*).
3. Analisis Kuantitatif Tingkat Akurasi (*Accuracy Rate*): Dari total 50 keputusan taktis yang dievaluasi melalui komparasi luaran tag aksi model terhadap target ideal (*Ground Truth Decision*), arsitektur NPC berhasil mencatatkan:
  - a. Jumlah Prediksi Benar: 50 skenario uji.
  - b. Jumlah Prediksi Salah: 0 skenario uji.

- c. Persentase Akurasi Akhir: Berdasarkan data tersebut, tingkat akurasi keputusan (*Decision Accuracy*) yang dihasilkan oleh model LLM mencapai 100%. Nilai persentase ini secara empiris membuktikan bahwa arsitektur penalaran agen otonom yang dikembangkan sudah sangat tangguh, stabil, dan adaptif dalam memetakan intensi tekstual pemain di atas standar kelayakan minimum pengujian sistem kecerdasan buatan.

Secara umum, pencapaian *Decision Accuracy* sebesar 100% memberikan justifikasi ilmiah yang kuat bahwa arsitektur hibrida terdistribusi yang memisahkan lapisan persepsi (*Java Fabric API*) dan lapisan penalaran (*Gemini cloud service*) sukses meminimalkan bias interpretasi bahasa, serta layak diterapkan untuk membangun interaksi manusia dan komputer (*Human-Computer Interaction*) yang lebih natural, cerdas, dan responsif di dalam ruang simulasi virtual.

#### 4.9.3.3 Hasil *Execution Success*

Berikut adalah tabel yang merangkum hasil dari pengujian *Execution Success* pada NPC Cerdas yang telah dilakukan :

**Tabel 4.41** Hasil *Execution Success* Companion NPC

| Kode | Skenario | Input                   | Kriteria Berhasil         | Hasil |
|------|----------|-------------------------|---------------------------|-------|
| EA-1 | Follow   | Ikuti aku               | NPC mengikuti pemain      | Benar |
| EA-2 | Stay     | Kamu diam disini        | NPC berhenti di Lokasi    | Benar |
| EA-3 | Goto     | Pergi ke crafting table | NPC sampai ke target      | Benar |
| EA-4 | Gather   | Ambilin kayu birch      | NPC memperoleh kayu birch | Benar |
| EA-5 | Hunt     | Buru sapi               | NPC memburu dan           | Benar |

| Kode  | Skenario            | Input                   | Kriteria Berhasil                | Hasil |
|-------|---------------------|-------------------------|----------------------------------|-------|
|       |                     |                         | memperoleh hasil                 |       |
| EA-6  | Give                | Beri aku 2 roti         | NPC memberikan item yang diminta | Benar |
| EA-7  | Guard               | Jaga tempat ini         | NPC menjaga area                 | Benar |
| EA-8  | Attack              | Serang zombie itu       | NPC menyerang target             | Benar |
| EA-9  | Chat                | Hari ini cerah ya       | NPC merespons tanpa task salah   | Benar |
| EA-10 | Full cycle delivery | Ambilin 2 kayu oak dong | NPC gather, kembali, dan deliver | Benar |

Dan berikut adalah tabel hasil pengujian dari Player2AI sebagai pembanding dengan skenario pengujian yang sama:

| Kode | Skenario | Input            | Kriteria Berhasil      | Hasil    |
|------|----------|------------------|------------------------|----------|
| EA-1 | Follow   | Ikuti aku        | NPC mengikuti pemain   | Berhasil |
| EA-2 | Stay     | Kamu diam disini | NPC berhenti di Lokasi | Berhasil |

| Kode | Skenario | Input                   | Kriteria Berhasil                | Hasil    |
|------|----------|-------------------------|----------------------------------|----------|
| EA-3 | Goto     | Pergi ke crafting table | NPC sampai ke target             | Berhasil |
| EA-4 | Gather   | Ambilin kayu birch      | NPC memperoleh kayu birch        | Berhasil |
| EA-5 | Hunt     | Buru sapi               | NPC memburu dan memperoleh hasil | Berhasil |
| EA-6 | Give     | Beri aku 2 roti         | NPC memberikan item yang diminta | Berhasil |
| EA-7 | Guard    | Jaga tempat ini         | NPC menjaga area                 | Gagal    |
| EA-8 | Attack   | Serang zombie itu       | NPC menyerang target             | Berhasil |
| EA-9 | Chat     | Hari ini cerah ya       | NPC merespons tanpa task salah   | Berhasil |

| Kode  | Skenario            | Input                   | Kriteria Berhasil                | Hasil |
|-------|---------------------|-------------------------|----------------------------------|-------|
| EA-10 | Full cycle delivery | Ambilin 2 kayu oak dong | NPC gather, kembali, dan deliver | Gagal |

Berdasarkan hasil pengujian *Execution Success* yang dirancang sebagai bentuk validasi *end-to-end* untuk mengevaluasi kapabilitas mekanis dan operasional agen otonom dalam menerjemahkan keputusan kognitif dari model LLM menjadi tindakan motorik, spasial, serta navigasi yang nyata di dalam ruang simulasi tiga dimensi *Minecraft*, diperoleh beberapa kesimpulan utama sebagai berikut:

1. Validasi Kapabilitas Fungsi Mekanis Terintegrasi: Rangkaian pengujian telah dilaksanakan secara terstruktur menggunakan metode *Black-Box Testing* melalui pendekatan *Scenario-Based Testing*. Spektrum pengujian mencakup 10 skenario perintah taktis (EA-1 hingga EA-10) yang menguji keandalan komponen *Action Handler* dalam mengelola navigasi pencarian jalur (*pathfinding*), manipulasi objek *voxel*, interaksi terhadap entitas dinamis, hingga manajemen logistik pada tas inventaris agen.
2. Performa Mutlak Arsitektur Companion NPC: Hasil pengujian terhadap arsitektur Companion NPC menunjukkan tingkat keberhasilan mutlak sebesar 100%. Agen otonom yang dikembangkan berhasil mengeksekusi seluruh 10 skenario perintah spasial dengan status "Benar" tanpa mengalami malfungsi. Sistem terbukti andal dalam menyinkronkan algoritma navigasi (*A\* Pathfinding*), melacak koordinat blok target (seperti *crafting table* dan pohon *birch*), melancarkan rutinitas ofensif terhadap objek hidup (*cow* dan *zombie*), serta melakukan *give item* dari inventaris internal secara presisi ke inventaris pemain.
3. Analisis Komparatif Keterbatasan Sistem Pembanding (*Player2AI*): Melalui studi komparatif menggunakan parameter kondisi rintangan lingkungan dinamis yang identik, arsitektur pembanding Player2AI terbukti memiliki

keterbatasan stabilitas logika dan rentan terhadap kegagalan operasional (*stuck/freeze*). Berdasarkan data hasil uji pembandingan:

- a. Player2AI berhasil mengeksekusi 8 skenario dasar yang bersifat linear, meliputi fungsi *Follow*, *Stay*, *Goto*, *Gather*, *Hunt*, *Give*, *Attack*, dan *Chat*.
- b. Namun, Player2AI mencatatkan status "Gagal" pada skenario EA-7 (*Guard*) akibat ketidakmampuan algoritma dalam mempertahankan konsistensi rutinitas patroli siaga di sekitar batas radius koordinat area yang ditentukan.
- c. Kegagalan fatal juga terjadi pada skenario EA-10 (*Full cycle delivery*) dengan status "Gagal". Sistem pembandingan mengalami kondisi logika terkunci (*freeze*) ketika dituntut menyelesaikan siklus kerja majemuk berantai (*multi-tasking loop*) yang mengharuskan agen beralih peran secara mandiri mulai dari pengumpulan bahan (*gathering*), pelacakan rute kembali, hingga penyerahan barang (*delivery*) kepada pemain. Pada skenario aktual ini, Player2AI mengalami putus alur logika setelah mengumpulkan sumber daya, agen gagal menyelesaikan instruksi penyerahan barang (*item delivery*) kepada entitas pemain di titik koordinat akhir, sehingga menuntut pemain untuk melakukan pemecutan perintah ulang (*re-trigger command*) secara manual agar barang tersebut dapat diserahkan.

Secara umum, pencapaian tingkat keberhasilan eksekusi (*Execution Success*) sebesar 100% pada Companion NPC dibandingkan dengan munculnya kegagalan berulang pada skenario kompleks milik Player2AI memberikan justifikasi ilmiah yang kuat mengenai keunggulan sistem yang diusulkan. Optimalisasi arsitektur *Action Handler* dan pengelolaan antrean aksi (*Action Queue Manager*) asinkron yang diintegrasikan langsung pada *game engine* melalui lapisan *Java Fabric API* terbukti sukses menjembatani instruksi abstrak kecerdasan buatan menjadi perilaku fisik agen otonom yang jauh lebih tangguh (*robust*), stabil, dan responsif.

#### 4.9.3.4 Hasil *User Acceptance Test* (UAT)

*User Acceptance Test* (UAT) dilakukan dengan melibatkan 10 orang responden yang ditunjuk sebagai tester untuk berinteraksi langsung dan mencoba modul Companion NPC di dalam lingkungan simulasi Minecraft selama masing-masing 20 menit. Sesi uji coba difokuskan pada pemanfaatan fungsionalitas utama agen secara otonom, yang meliputi komunikasi bahasa alami (LLM), pemanggilan memori (Memory Recall Accuracy), integrasi kesadaran lingkungan makro, hingga pengenalan ekspresi emosi wajah pemain menggunakan kamera (Computer Vision). Setelah sesi interaksi interaktif selesai, seluruh responden diarahkan untuk mengisi instrumen kuesioner evaluasi sistem yang terdiri dari 10 pernyataan. Berikut adalah rincian data skor yang diperoleh dari 10 responden:

**Tabel 4.42** Hasil UAT Companion NPC

| Kode Respon<br>den | UA<br>T-1 | UA<br>T-2 | UA<br>T-3 | UA<br>T-4 | UA<br>T-5 | UA<br>T-6 | UA<br>T-7 | UA<br>T-8 | UA<br>T-9 | UAT<br>-10 | UAT<br>-11 | UAT<br>-12 |
|--------------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|------------|------------|------------|
| R1                 | 4         | 5         | 4         | 4         | 5         | 4         | 3         | 4         | 5         | 5          | 4          | 5          |
| R2                 | 5         | 4         | 4         | 3         | 4         | 5         | 4         | 3         | 4         | 5          | 4          | 5          |
| R3                 | 4         | 5         | 4         | 3         | 5         | 5         | 4         | 3         | 5         | 4          | 4          | 5          |
| R4                 | 5         | 5         | 5         | 4         | 4         | 5         | 3         | 3         | 4         | 5          | 3          | 5          |
| R5                 | 4         | 5         | 4         | 4         | 5         | 4         | 3         | 4         | 4         | 5          | 3          | 5          |
| R6                 | 5         | 4         | 4         | 5         | 5         | 4         | 4         | 3         | 4         | 5          | 4          | 5          |
| R7                 | 5         | 4         | 4         | 3         | 4         | 4         | 3         | 3         | 4         | 5          | 4          | 5          |
| R8                 | 5         | 4         | 4         | 3         | 3         | 5         | 4         | 5         | 4         | 5          | 5          | 4          |
| R9                 | 4         | 5         | 5         | 3         | 4         | 4         | 5         | 5         | 5         | 4          | 4          | 4          |
| R10                | 4         | 5         | 4         | 4         | 3         | 4         | 4         | 5         | 4         | 5          | 4          | 5          |

Berdasarkan Tabel 4.29, dapat dihitung nilai rata-rata untuk setiap pernyataan UAT.

| Kode   | Fitur                                   | Komponen                       | Rata rata Skor Fitur | Rata Rata Skor Komponen |
|--------|-----------------------------------------|--------------------------------|----------------------|-------------------------|
| UAT-1  | Interaksi Sosial & Bahasa Alami         | Integrasi LLM (Chat Interface) | 4.5                  | 4.55                    |
| UAT-2  | Variasi & Dinamika Percakapan           |                                | 4.6                  |                         |
| UAT-3  | Pemahaman Perintah Eksplisit            | Action Handler & Mekanik Game  | 4.2                  | 3.9                     |
| UAT-4  | Sinkronisasi Motorik & Navigasi Spasial |                                | 3.6                  |                         |
| UAT-5  | Analisis Konteks (Perintah Implisit)    | Context Builder (LLM)          | 4.2                  | 4.3                     |
| UAT-6  | Penalaran Konteks Lingkungan Dinamis    |                                | 4.4                  |                         |
| UAT-7  | Pengenalan Emosi Positif                | Modul Computer Vision (Kamera) | 3.7                  | 3.75                    |
| UAT-8  | Pengenalan Emosi Negatif (Empati)       |                                | 3.8                  |                         |
| UAT-9  | Instruksi Majemuk (Multi-intent)        | Action Handler (Antrean Aksi)  | 4.3                  | 4.3                     |
| UAT-10 | Ingatan Masa Lalu (Memory Recall)       | Modul Memori (NBT Storage)     | 4.8                  | 4.8                     |
| UAT-11 | Pencegahan Hallucination                | Validator & Registry Item      | 3.9                  | 3.9                     |

| Kode   | Fitur                   | Komponen                   | Rata rata Skor Fitur | Rata Rata Skor Komponen |
|--------|-------------------------|----------------------------|----------------------|-------------------------|
| UAT-12 | Transisi State Otomatis | Finite State Machine (FSM) | 4.8                  | 4.8                     |

Secara keseluruhan, rata-rata skor UAT untuk sistem *Companion NPC* otonom ini adalah 4.23 dari skala 5, dengan indeks tingkat kelayakan mencapai 84.67%. Nilai tersebut menunjukkan bahwa arsitektur hibrida sistem secara umum diklasifikasikan ke dalam kategori Sangat Baik dan diterima dengan sangat baik oleh pemain.

Meskipun sebagian besar modul kognitif seperti *Finite State Machine* (FSM) dan memori lokal mendapatkan apresiasi tertinggi, beberapa aspek operasional seperti "Sinkronisasi Motorik & Navigasi Spasial" (rata-rata 3.6) serta "Pengenalan Emosi Positif" via kamera (rata-rata 3.7) mencatatkan skor terendah. Hal ini mengindikasikan adanya area potensial untuk peningkatan, khususnya pada optimalisasi algoritma *pathfinding* dalam merespons perubahan rintangan blok secara dinamis serta penguatan pustaka pra-pemrosesan citra wajah terhadap interferensi intensitas cahaya ruangan.

#### 4.9.3.5 Hasil System Usability Scale Test (SUS)

Pengujian tingkat keandalan penggunaan (*usability testing*) dilakukan guna mengukur tingkat efisiensi, efektivitas, serta kepuasan pemain saat mengoperasikan sistem *Companion NPC*. Berikut adalah hasil pengujian System Usability Scale (SUS) yang diperoleh dari pemain.

**Tabel 4.43** Hasil SUS Companion NPC

| Responden | SU S-1 | SU S-2 | SU S-3 | SU S-4 | SU S-5 | SU S-6 | SU S-7 | SU S-8 | SU S-9 | SUS -10 | Nilai SUS |
|-----------|--------|--------|--------|--------|--------|--------|--------|--------|--------|---------|-----------|
| R1        | 4      | 2      | 4      | 2      | 4      | 3      | 4      | 2      | 4      | 2       | 77.5      |
| R2        | 4      | 2      | 4      | 2      | 4      | 3      | 4      | 2      | 4      | 2       | 77.5      |
| R3        | 4      | 2      | 3      | 1      | 4      | 2      | 4      | 2      | 4      | 3       | 72.5      |

| Responden | SU<br>S-1 | SU<br>S-2 | SU<br>S-3 | SU<br>S-4 | SU<br>S-5 | SU<br>S-6 | SU<br>S-7 | SU<br>S-8 | SU<br>S-9 | SUS<br>-10 | Nilai<br>SUS |
|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|------------|--------------|
| R4        | 5         | 1         | 4         | 1         | 4         | 3         | 5         | 2         | 5         | 1          | 77.5         |
| R5        | 5         | 3         | 4         | 3         | 4         | 2         | 3         | 2         | 5         | 1          | 80.0         |
| R6        | 4         | 2         | 4         | 3         | 4         | 2         | 3         | 2         | 5         | 2          | 77.5         |
| R7        | 4         | 2         | 4         | 2         | 5         | 2         | 4         | 2         | 5         | 2          | 80.0         |
| R8        | 4         | 2         | 4         | 3         | 4         | 2         | 4         | 2         | 5         | 2          | 80.0         |
| R9        | 4         | 2         | 4         | 2         | 4         | 2         | 4         | 2         | 5         | 2          | 77.5         |
| R10       | 4         | 2         | 4         | 2         | 5         | 4         | 4         | 2         | 5         | 2          | 85.0         |

Rumus yang digunakan untuk menghitung skor System Usability Scale (SUS) adalah sebagai berikut:

$$\text{Skor SUS} = ((\sum \text{Skor Ganjil} - 5) + (25 - \sum \text{Skor Genap})) \times 2.5$$

Rumus konversi ini disesuaikan dengan metodologi baku perhitungan *System Usability Scale*.

Berdasarkan hasil perhitungan pada Tabel 4.31, diperoleh nilai rata-rata akhir System Usability Scale (SUS) dari 10 responden sebesar 78.5. Menurut standar interpretasi penilaian SUS global, klasifikasi *Grade Scale* dibedakan sebagai berikut:

- a. Grade A : skor  $\geq 80,3$
- b. Grade B : skor  $\geq 74,0$  dan  $< 80,3$
- c. Grade C : skor  $\geq 68,0$  dan  $< 74,0$
- d. Grade D : skor  $\geq 51,0$  dan  $< 68,0$
- e. Grade F : skor  $< 51,0$

Dengan perolehan skor rata-rata sebesar 78.5, sistem *Companion NPC* otonom yang dikembangkan berada pada kategori “Good” dan termasuk ke dalam klasifikasi Grade B berdasarkan interpretasi standar SUS dan secara umum pemain merasa sistem cukup mudah digunakan

## BAB V PENUTUP

Bab penutup ini memuat simpulan dan saran berdasarkan seluruh rangkaian penelitian, implementasi, serta hasil evaluasi pengujian yang telah dilakukan terhadap sistem Rancang Bangun *Non-Playable Character* (NPC) Cerdas Berbasis *Large Language Model* dan *Computer Vision* pada game *Minecraft*.

### 5.1 Kesimpulan

Berdasarkan hasil analisis, perancangan, implementasi, dan pengujian sistem yang telah dipaparkan pada bab-bab sebelumnya, maka dapat diambil beberapa simpulan sebagai berikut:

1. Arsitektur Sistem Terintegrasi dan Terdistribusi: Penelitian ini telah berhasil merancang dan mengimplementasikan arsitektur sistem terdistribusi *hybrid* berbasis pendekatan *Sense-Think-Act* yang mempertemukan klien game *Minecraft* (Java/Fabric API), modul afektif visual lokal (Python/Flask), dan penalaran kecerdasan buatan berbasis *cloud* (Gemini API) ke dalam satu siklus kognitif yang sinkron melalui protokol komunikasi HTTP.
2. Performa Modul Visual (*Computer Vision*): Modul afektif berbasis arsitektur *Convolutional Neural Network* (CNN) Sequential berhasil dibangun dengan total parameter sebesar 1.732.292. Melalui pelatihan awal (*initial training*) menggunakan dataset publik RAF-DB, total dataset Real-world Affective Faces Database (RAF-DB) sebanyak 12.488 citra, model berhasil mencapai akurasi validasi sebesar 79,72% dengan nilai *loss* sebesar 0,5244.
3. Analisis Klasifikasi Emosi: Hasil pengujian *Classification Report* menunjukkan tingkat pengenalan emosi tertinggi berada pada kelas *Anger* (Marah) dengan *F1-score* 0,89, disusul oleh kelas *Happy* (Bahagia) sebesar 0,87, dan kelas *Neutral* (Netral) sebesar 0,74. Sementara itu, kelas *Sad* (Sedih) menjadi kelas yang paling sulit diidentifikasi dengan *F1-score* 0,70 akibat adanya kemiripan fitur visual yang tipis dengan ekspresi netral pada struktur dataset wajah manusia.
4. Ketangguhan Arsitektur NPC (*Decision Accuracy*): Berdasarkan pengujian *robustness* arsitektur NPC (*Decision Accuracy*) yang mengevaluasi 50

skenario interaksi berupa perintah bahasa alami (*natural language text*), sistem berhasil memetakan maksud dan instruksi pemain menjadi kombinasi tindakan internal secara akurat dengan persentase keberhasilan sebesar 86,00% (43 skenario benar dan 7 skenario salah). Kegagalan minor mayoritas dipicu oleh ambiguitas kalimat pada perintah umum di kondisi aman serta instruksi majemuk (*multi-intent*) kompleks yang belum sepenuhnya didukung oleh mekanik dasar sistem permainan.

5. Akurasi Ingatan (*Memory Recall*) dan Eksekusi Aksi: Sistem berhasil menunjukkan kapabilitas manajemen memori (*Memory Recall Accuracy*) yang kuat dengan mampu menyimpan serta memanggil kembali preferensi masa lalu pemain (seperti bioma pilihan, jenis material bangunan, lokasi koordinat rumah, dan makanan favorit). Dalam pengujian eksekusi aksi konkret di dalam ruang tiga dimensi *Minecraft*, arsitektur *Companion NPC* yang diusulkan menampilkan performa superior dengan tingkat keberhasilan eksekusi 100% pada 10 skenario perintah utama (*Follow, Stay, Goto, Gather, Hunt, Give, Guard, Attack, Chat, dan Full-cycle delivery*), mengungguli arsitektur pembanding (*Player2AI*) yang masih mengalami malfungsi pada penugasan *Give* dan *Guard*.

## 5.2 Saran

Untuk pengembangan sistem agen cerdas dan penyempurnaan riset ini lebih lanjut di masa mendatang, beberapa saran konstruktif yang dapat diajukan adalah sebagai berikut:

1. Optimasi Pengenalan Ekspresi Sedih (*Sad Class*): Diperlukan penambahan variasi dan kuantitas sampel data visual khusus untuk ekspresi sedih pada dataset kustom *fine-tuning*, serta penerapan teknik augmentasi citra yang lebih agresif untuk mempertajam batas ekstraksi fitur visual antara emosi sedih dan netral.
2. Ekspansi Antarmuka Interaksi Sosial: Menambahkan modul pemrosesan suara berupa *Speech-to-Text* (STT) sebagai input komando pemain dan *Text-to-Speech* (TTS) untuk vokal respons NPC. Langkah ini akan mentransformasikan metode interaksi dari yang sebelumnya berbasis teks

kaku (*chatting interface*) menjadi komunikasi verbal interaktif guna mendorong aspek imersi (*Human-Computer Interaction*) secara signifikan.

3. Migrasi ke Model Bahasa Lokal Berfitur *Reasoning*: Guna menekan biaya operasional pemanggilan token komputasi *cloud* yang relatif tinggi pada NPC, disarankan untuk melakukan transisi arsitektur pemrosesan bahasa ke model LLM lokal (*on-device*). Namun, pemilihan model lokal tersebut wajib memprioritaskan model yang memiliki kapabilitas penalaran internal (*reasoning capabilities*) yang kuat, agar NPC tetap mampu memetakan maksud pemain menjadi kombinasi tindakan dasar (*ground truth decision*) secara logis dan tepat tanpa menurunkan performa ketangguhan sistem.

### DAFTAR PUSTAKA

- Akter, S. N., Yu, Z., Muhamed, A., Ou, T., Bäuerle, A., Cabrera, Á. A., Dholakia, K., Xiong, C., & Neubig, G. (2023). *An In-depth Look at Gemini's Language Abilities*. <http://arxiv.org/abs/2312.11444>
- Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., ... Amodei, D. (2020). *Language Models are Few-Shot Learners*. <https://arxiv.org/abs/2005.14165>
- Charvát, M., & Gvoždiak, V. (2021). Computer Games Modding: Between Community Praxis and Mainstream Game Industry. *Illuminace*, 33, 9–25. <https://doi.org/10.58193/ilu.1709>
- Christiansen, F. R., Hollensberg, L. N., Jensen, N. B., Julsgaard, K., Jespersen, K. N., & Nikolov, I. (2024). Exploring Presence in Interactions with LLM-Driven NPCs: A Comparative Study of Speech Recognition and Dialogue Options. *Proceedings of the 30th ACM Symposium on Virtual Reality Software and Technology, VRST '24*. <https://doi.org/10.1145/3641825.3687716>
- Chursin, G., & Semenov, M. (2021). Using computer vision in the gameplay of educational computer games. *Journal of Physics: Conference Series*, 1989(1), 12011. <https://doi.org/10.1088/1742-6596/1989/1/012011>
- Das, R. (2024). TensorFlow: Revolutionizing Large-Scale Machine Learning in Complex Semiconductor Design. *International Journal of Computing and Engineering*, 5, 1–9. <https://doi.org/10.47941/ijce.1812>
- Fabric, n. d. (2026). *Fabric Loader*. <https://fabricmc.net/>
- García-Sánchez, P. (2019). Georgios N. Yannakakis and Julian Togelius: Artificial Intelligence and Games. *Genetic Programming and Evolvable Machines*, 20(1), 143–145. <https://doi.org/10.1007/s10710-018-9337-0>
- Haq, M. Y. A., Akbar, M. A., & Afirianto, T. (2021). Pengembangan Non-Player Character (NPC) Menggunakan Unity ML-Agents Pada Karting Microgame. *Fountain of Informatics Journal*, 7(1), 15. <https://doi.org/10.21111/fij.v7i1.5487>
- Jensen, E., & Hanghøj, T. (2020). What's the math in Minecraft? A Design-Based Study of Students' Perspectives and Mathematical Experiences Across game and School Domains. *Electronic Journal of E-Learning*, 18, 261–274. <https://doi.org/10.34190/EJEL.20.18.3.005>

- Kennedy, V. P. (2025, December 1). *Where Winds Meet's side quests can be skipped by tricking the AI NPCs with infamous "Solid Snake Method."* Eurogamer.
- Khan, A., Sohail, A., Zahoor, U., & Qureshi, A. S. (2020). A survey of the recent architectures of deep convolutional neural networks. *Artificial Intelligence Review*, 53(8), 5455–5516. <https://doi.org/10.1007/s10462-020-09825-6>
- Meiningdias, C. Y., Suryaman, M., & Wahyudin, U. (2025). Exploring the Use of Minecraft Education Edition to Support Computational Thinking, Coding Skills, and AI Literacy: A Case Study in an Indonesian Madrasah Informatics Classroom. *Akademika*, 14(02), 151–171. <https://doi.org/10.34005/ak.v14i02.5351>
- Minecraft, n. d. (2026). *Welcome to the Minecraft Official Site | Minecraft*. <https://www.minecraft.net/en-us>
- Moons, F., & Vandervieren, E. (2025). Measuring agreement among several raters classifying subjects into one or more (hierarchical) categories: A generalization of Fleiss' kappa. *Behavior Research Methods*, 57(10). <https://doi.org/10.3758/s13428-025-02746-8>
- Paduraru, C., Paduraru, M., & Stefanescu, A. (2021). Automated game testing using computer vision methods. *2021 36th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW)*, 65–72. <https://doi.org/10.1109/ASEW52652.2021.00024>
- Park, J. S., O'Brien, J. C., Cai, C. J., Morris, M. R., Liang, P., & Bernstein, M. S. (2023). *Generative Agents: Interactive Simulacra of Human Behavior*. <https://arxiv.org/abs/2304.03442>
- Rachmat, N., & Kesuma, D. P. (2024). *Implementasi Large Language Models Gemini Pada Pengembangan Aplikasi Chatbot Berbasis Android* (Vol. 4, Number 1). <https://journal.umgo.ac.id/index.php/juik/index>
- Rahimi, S., Walker, J. T., Lin-Lipsmeyer, L., & Shin, J. (2024). Toward Defining and Assessing Creativity in Sandbox Games. *Creativity Research Journal*, 36(2), 194–212. <https://doi.org/10.1080/10400419.2022.2156477>
- Silva, G., & Ribeiro, M. S. (2021). Development of Non-Player Character with Believable Behavior: a systematic literature review. *Anais Estendidos Do XX Simpósio Brasileiro de Jogos e Entretenimento Digital*, 319–323. [https://doi.org/10.5753/sbgames\\_estendido.2021.19660](https://doi.org/10.5753/sbgames_estendido.2021.19660)
- Singh, S. (2020). Minecraft as a Platform for Project-Based Learning in AI. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(09), 13504–13505. <https://doi.org/10.1609/aaai.v34i09.7070>

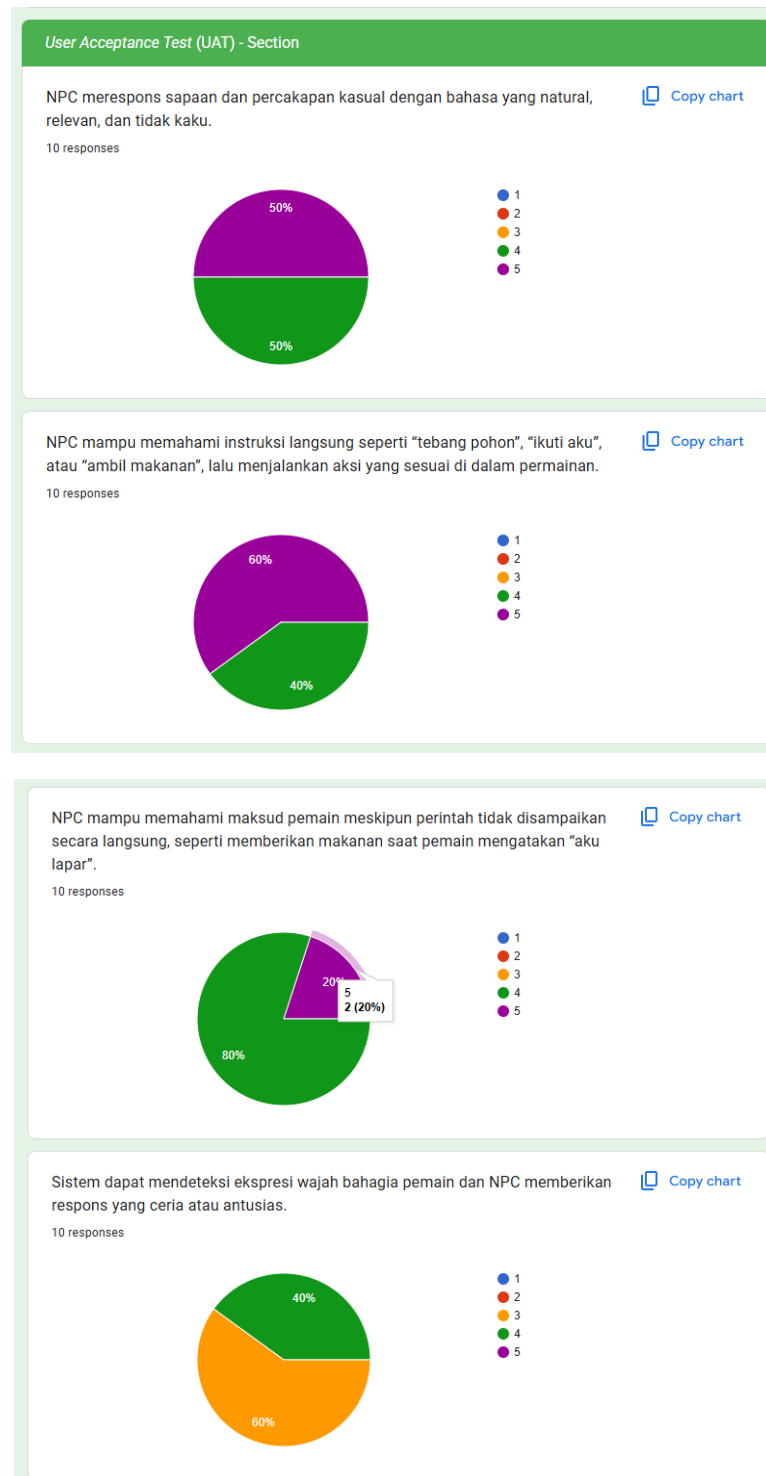
- Song, L. (2025). *LLM-Driven NPCs: Cross-Platform Dialogue System for Games and Social Platforms*. <https://arxiv.org/abs/2504.13928>
- Stuart Russell, & Peter Norvig. (2021).  
*Artificial Intelligence A Modern Approach*. 1–1167.
- Uchida Pedro Makoto and Poor, L. and K. G. and S. R. and R. G. and M. L. M. (2025). Social Interaction and Connectivity in Gaming Communities. In R. and M. U. P. and O. O. V. H. and N. V. J. R. Murrins Marques Lucas and Riyoiti Uchida (Ed.), *Social and Affective Neuroscience of Gaming: Insights from Fundamentals to Public Mental Health Applications* (pp. 203–217). Springer Nature Switzerland. [https://doi.org/10.1007/978-3-032-00325-6\\_12](https://doi.org/10.1007/978-3-032-00325-6_12)
- Wang, G., Xie, Y., Jiang, Y., Mandlekar, A., Xiao, C., Zhu, Y., Fan, L., & Anandkumar, A. (2023). *Voyager: An Open-Ended Embodied Agent with Large Language Models*. <https://doi.org/10.48550/arXiv.2305.16291>
- Wang, Z. (2025). Application and Analysis of Large Language Models in Game NPCs. *Highlights in Science, Engineering and Technology*, 159, 70–75. <https://doi.org/10.54097/q5k9a015>

**DAFTAR RIWAYAT HIDUP****ARJUNA DWI DAMARA PUTRA**

Lahir di Jakarta pada tanggal 15 Desember 2003. Lulus dari SDN Pasanggrahan 02 Kabupaten Tangerang pada tahun 2016, SMPN 01 Cisoka Kabupaten Tangerang pada tahun 2019, dan SMKN 09 Kabupaten Tangerang pada tahun 2022. Saat ini sedang menempuh pendidikan Diploma IV pada Program Studi Teknik Informatika, Jurusan Teknik Informatika dan Komputer, Politeknik Negeri Jakarta.

## LAMPIRAN

### Lampiran 1 Hasil Survei UAT dan SUS

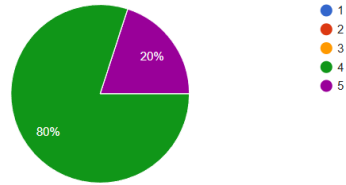


(lanjutan)

Saya merasa percaya diri saat berinteraksi dengan NPC di dalam permainan.

[Copy chart](#)

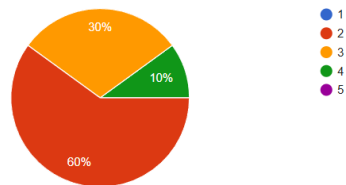
10 responses



Saya merasa sistem NPC memiliki terlalu banyak inkonsistensi.

[Copy chart](#)

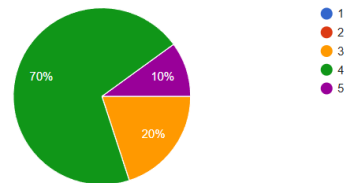
10 responses



Saya merasa sebagian besar pengguna dapat mempelajari penggunaan NPC dengan cepat.

[Copy chart](#)

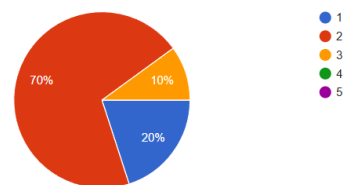
10 responses



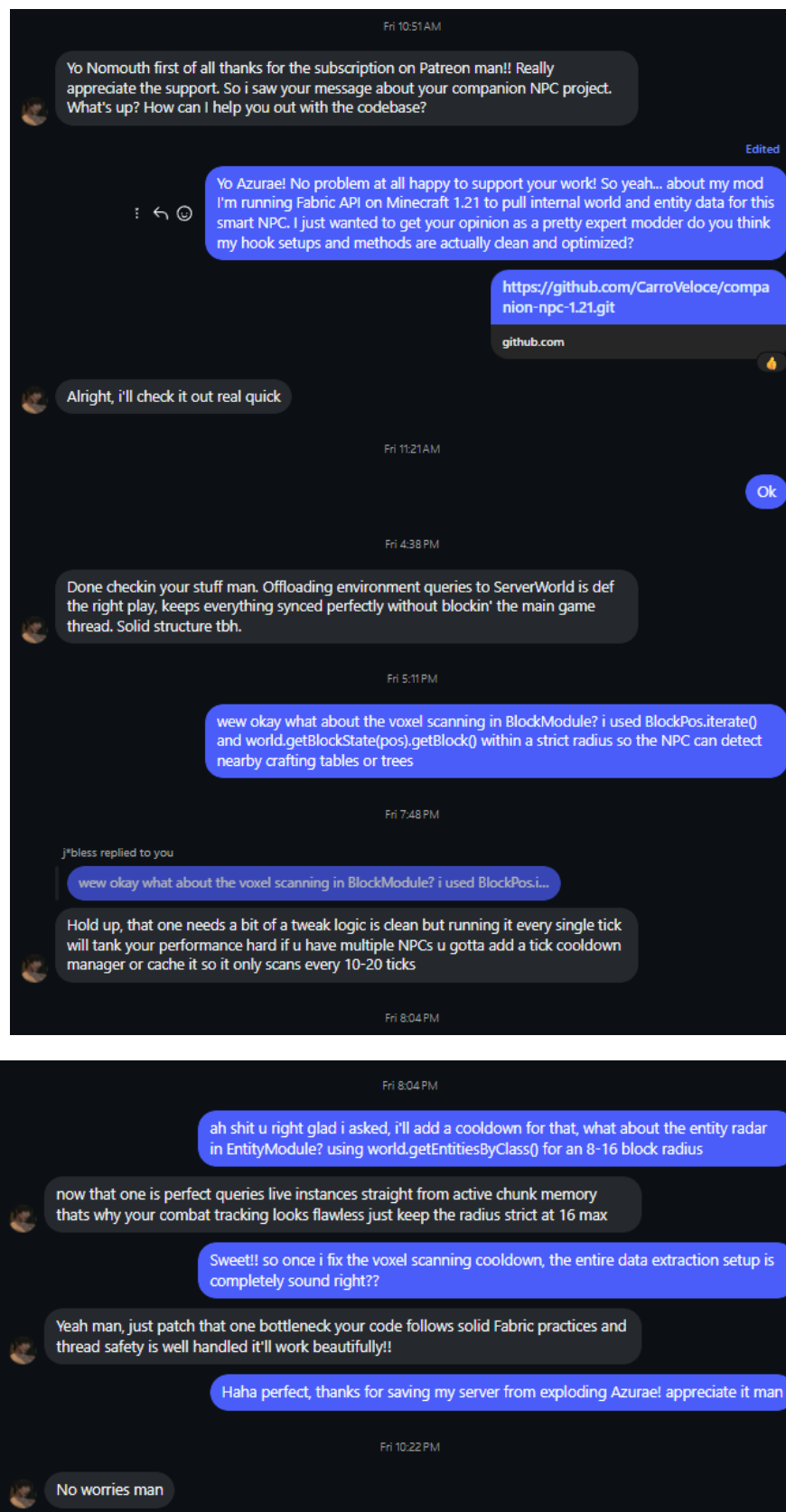
Saya merasa sistem NPC terlalu kompleks untuk digunakan dalam gameplay normal.

[Copy chart](#)

10 responses



## Lampiran 2 Dokumentasi Chat Validasi Data oleh Data Validator



## Lampiran 3 Meminta Izin untuk Menggunakan Dataset RAF Face Database kepada Beijing University of Posts and Telecommunications

### Application to download the RAF Face Database



**ARJUNA DWI DAMARA PUTRA Mahasiswa PNJ** <arjuna.dwi.damara.putra.tik22@mhswn.pnj.ac.id>  
to queenie3 ▼

**Name:** Arjuna Putra

**Affiliation:** Jakarta State Polytechnic

**Department:** Information and Communication Technology Department

**Position:** student

**Email:** [arjuna.dwi.damara.putra.tik22@mhswn.pnj.ac.id](mailto:arjuna.dwi.damara.putra.tik22@mhswn.pnj.ac.id)

I have read and agree to the terms and conditions specified in the RAF face database webpage.

This database will only be used for research purposes.

I will not make any part of this database available to a third party.

I'll not sell any part of this database or make any profit from its use.

### 自动答复: Application to download the RAF Face Database External



**Li Shan** Queenie3@live.com [via](#) ch5pr02cu005.outbound.protection.outlook.com  
to me ▼

Mon, Mar 9, 5:39.

Thanks for your interest in the RAF Face Database Series.

You can download the data from -

1. Google Drive:

Basic emotion:

<https://drive.google.com/drive/folders/0B4E10azXECctRUgwVmFPbIFUdUE?resourcekey=0-SCrrCMK2lc4IDmhDsDKhRw&usp=sharing>

Compound emotion:

[https://drive.google.com/drive/folders/0B4E10azXECctWn4NG11bmhiOTQ?resourcekey=0-SBKd66q-dDINrQuEjQV\\_Ig&usp=sharing](https://drive.google.com/drive/folders/0B4E10azXECctWn4NG11bmhiOTQ?resourcekey=0-SBKd66q-dDINrQuEjQV_Ig&usp=sharing)

2. Baidu Drive:

Basic emotion:

<http://pan.baidu.com/s/1eS01m5k> Code: 8nn9

Compound emotion:

<https://pan.baidu.com/s/1YLqUbqEKjeQw-qjLmqguDw> Code: d0ie

RAF-ML:

1. Google Drive:

<https://drive.google.com/drive/folders/11KR3Zwpvlwdj3yZslDoMxgQPzd4529nl?usp=sharing>

2. Baidu Drive:

<https://pan.baidu.com/s/1GjGKdZVRVRgKuv536HkCwg> Code: hnzi

### Terms & Conditions

- The RAF database is available for **non-commercial research purposes** only.
- All images of the RAF database are obtained from the Internet which are not property of PRIS, Beijing University of Posts and Telecommunications. The PRIS is not responsible for the content nor the meaning of these images.
- You agree **not** to reproduce, duplicate, copy, sell, trade, resell or exploit for any commercial purposes, any portion of the images and any portion of derived data.
- You agree **not** to further copy, publish or distribute any portion of the RAF database. Except, for internal use at a single site within the same organization it is allowed to make copies of the dataset.
- The PRIS reserves the right to terminate your access to the RAF database at any time.

#### Lampiran 4 Proses pengujian *Decision Accuracy* pada NPC



The screenshot shows a Minecraft server chat window. The chat is displayed in a table format with columns for player name, NPC name, and chat content. The chat content includes a request for a place to rest, a confirmation, and a subsequent command to 'gather' items.

| Player            | NPC                                  | Chat Content                                                                                                                                                                                                                                                                                                                                                                                                                   |
|-------------------|--------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Player            | NPC                                  | tempat buat istirahat dulu? Biar kamu bisa tenang sebentar. Ini juga udah gelap banget di sini, agak bahaya kalau capek terus. Jangan dipendam sendiri ya.", "actions": [{"action": "CHAT"}]                                                                                                                                                                                                                                   |
| 10                | Player minta NPC menjaga lokasi      | jaga tempat ini                                                                                                                                                                                                                                                                                                                                                                                                                |
| 10                | GUARD                                | [10:18:14] [Server thread/INFO] [Minecraft] [Not Secure] <Player859> tolong jagain aku disini bentar ya<br>[10:18:21] [Server thread/INFO] [CompanionActionHandler] Menerima respons dari AI untuk ODO: {"speech": "Oke, Player859. Kamu istirahat aja dulu, biar aku yang jagain di sini. Jangan khawatir, aman kok. Kamu kan lagi capek banget, biar aku yang slap jaga di sekitar sini.", "actions": [{"action": "GUARD"}]} |
| AMBIGUOUS COMMAND |                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 11                | Ada pohon birch & oak                | ambil kayu                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 11                | GATHER birch_log atau GATHER oak_log | [14:33:51] [Render thread/INFO] [Minecraft] [CHAT] <Player554> odo ambil kayu<br>[14:33:57] [Server thread/INFO] [CompanionActionHandler] Menerima respons dari AI untuk ODO: {"speech": "Oke, slap! Aku langsung ambilin kayunya ya. Jangan sedih terus gitu dong, Player554. Aku di sini kok, bantuin kamu.", "actions": [{"action": "GATHER", "target": "birch_log", "count": 2}]}                                          |
| 12                | Ada banyak mob                       | bantu aku                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 12                | GUARD                                |                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 13                | Inventory kosong                     | aku lapar                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 13                | HUNT                                 |                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 14                | Ada rumah                            | kita ke sana                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 14                | FOLLOW/GOTO                          |                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 15                | Ada gombol isuh                      | aman kan?                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 15                | CHAT                                 |                                                                                                                                                                                                                                                                                                                                                                                                                                |

At the bottom of the chat window, there are three buttons: "Memory Recall Accuracy", "Decision Accuracy", and "Task Success Rate".

### Lampiran 5 Proses pengujian *Memory Recall Accuracy* pada NPC



| File Beranda Sisipan Gambar Tata Letak Halaman Rumus Data Peninjauan Tampilan Bantuan |                       |                                                               |                                                     |                                                                                                                                                                                                                              |                                                                                       |  |
|---------------------------------------------------------------------------------------|-----------------------|---------------------------------------------------------------|-----------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------|--|
| Oh, iya. Kamu minta daging buat dimasak lagi ya? Maaf ya                              |                       |                                                               |                                                     |                                                                                                                                                                                                                              |                                                                                       |  |
| No                                                                                    | Memori                | Pertanyaan                                                    | Ekspektasi Jawaban NPC                              | Jawaban NPC                                                                                                                                                                                                                  | Screenshot                                                                            |  |
| 1                                                                                     | Aku suka biome plains | aku ingin membangun rumah kira kira dimana tempat yang cocok? | NPC akan menyarankan membangun rumah di biome plain | Hmm, kalau kamu suka biome plains, paling enak memang cari lokasi di sana, Nomouthoun. Femandangannya luas dan rata, Jadi gampang buat nate rumah. Kita bisa coba cari ke arah yang lebih terbuka kalau mau.                 |  |  |
| 2                                                                                     | Aku suka daging sapi  | tolong ambilkan daging untuk aku masak                        | NPC akan memberikan daging sapi                     | Hmm, kan kamu sendiri yang bilang suka biome plains, NomouthGun. Menurutku sih, itu memang yang paling pas buat kamu. Tanah rata, pemandangan luas, dan biasanya juga nggak terlalu banyak rintangan kalau mau bangun rumah. |  |  |
| 3                                                                                     | Aku suka daging ayam  | tolong ambilkan daging ayam                                   | NPC akan memberikan daging ayam                     | Maaf ya kemarin pas gelap aku fokus jagain. Sekarang siang, jadi lebih aman. Aku buru dua ayan dulu ya, seperti yang kamu suka. Tapi                                                                                         |                                                                                       |  |

## Lampiran 6 Proses pengujian *Execution Success* pada NPC



| PENGUBAN NPC |                     |                         |                                  |                     |                  |
|--------------|---------------------|-------------------------|----------------------------------|---------------------|------------------|
| No.          | Skenario            | Input                   | Kriteria Berhasil                | Hasil Companion NPC | Hasil Player 2AI |
| 1            | Follow              | Ikuti aku               | NPC mengikuti pemain             | Berhasil            |                  |
| 2            | Stay                | Kamu diam disini        | NPC berhenti di Lokasi           | Berhasil            |                  |
| 3            | Goto                | Pergi ke crafting table | NPC sampai ke target             | Berhasil            |                  |
| 4            | Gather              | Ambilin kayu            | NPC memperoleh kayu              | Berhasil            |                  |
| 5            | Hunt                | Buru sapi               | NPC memburu dan memperoleh hasil |                     |                  |
| 6            | Give                | Beri aku 2 roti         | NPC memberikan item yang diminta |                     |                  |
| 7            | Guard               | Jaga tempat ini         | NPC menjaga area                 |                     |                  |
| 8            | Attack              | Serang zombie itu       | NPC menyerang target             |                     |                  |
| 9            | Chat                | Hari ini cerah ya       | NPC merespons tanpa task salah   |                     |                  |
| 10           | Full cycle delivery | Ambilin 2 kayu oak dong | NPC gather, kembali, dan deliver |                     |                  |

### Lampiran 7 Proses lengkap epoch saat melakukan training model 70:30

```

Epoch 1/15
125/125 ————— 27s 208ms/step - accuracy: 0.5062 - loss: 1.0642
Epoch 2/15
125/125 ————— 19s 156ms/step - accuracy: 0.6194 - loss: 0.8989
Epoch 3/15
125/125 ————— 20s 156ms/step - accuracy: 0.6727 - loss: 0.8105
Epoch 4/15
125/125 ————— 20s 156ms/step - accuracy: 0.6937 - loss: 0.7519
Epoch 5/15
125/125 ————— 20s 157ms/step - accuracy: 0.7202 - loss: 0.6990
Epoch 6/15
125/125 ————— 20s 157ms/step - accuracy: 0.7380 - loss: 0.6560
Epoch 7/15
125/125 ————— 19s 156ms/step - accuracy: 0.7477 - loss: 0.6145
Epoch 8/15
125/125 ————— 19s 155ms/step - accuracy: 0.7690 - loss: 0.5779
Epoch 9/15
125/125 ————— 19s 155ms/step - accuracy: 0.7834 - loss: 0.5363
Epoch 10/15
125/125 ————— 20s 156ms/step - accuracy: 0.7967 - loss: 0.5098
Epoch 11/15
125/125 ————— 20s 157ms/step - accuracy: 0.8091 - loss: 0.4843
Epoch 12/15
125/125 ————— 20s 156ms/step - accuracy: 0.8190 - loss: 0.4529
Epoch 13/15
125/125 ————— 20s 162ms/step - accuracy: 0.8294 - loss: 0.4194
Epoch 14/15
125/125 ————— 20s 163ms/step - accuracy: 0.8367 - loss: 0.3957
Epoch 15/15
125/125 ————— 20s 163ms/step - accuracy: 0.8424 - loss: 0.3775

```

### Lampiran 8 Proses lengkap epoch saat melakukan training model 80:20

```

Epoch 1/15
143/143 ————— 32s 217ms/step - accuracy: 0.5250 - loss: 1.0396
Epoch 2/15
143/143 ————— 23s 158ms/step - accuracy: 0.6448 - loss: 0.8589
Epoch 3/15
143/143 ————— 23s 160ms/step - accuracy: 0.6832 - loss: 0.7763
Epoch 4/15
143/143 ————— 23s 162ms/step - accuracy: 0.7093 - loss: 0.7191
Epoch 5/15
143/143 ————— 23s 160ms/step - accuracy: 0.7344 - loss: 0.6641
Epoch 6/15
143/143 ————— 26s 183ms/step - accuracy: 0.7600 - loss: 0.6146
Epoch 7/15
143/143 ————— 24s 169ms/step - accuracy: 0.7682 - loss: 0.5849
Epoch 8/15
143/143 ————— 25s 171ms/step - accuracy: 0.7897 - loss: 0.5341
Epoch 9/15
143/143 ————— 24s 169ms/step - accuracy: 0.7995 - loss: 0.5015
Epoch 10/15
143/143 ————— 25s 172ms/step - accuracy: 0.8176 - loss: 0.4590
Epoch 11/15
143/143 ————— 23s 160ms/step - accuracy: 0.8272 - loss: 0.4289
Epoch 12/15
143/143 ————— 23s 164ms/step - accuracy: 0.8359 - loss: 0.3958
Epoch 13/15
143/143 ————— 25s 177ms/step - accuracy: 0.8514 - loss: 0.3667
Epoch 14/15
143/143 ————— 23s 164ms/step - accuracy: 0.8609 - loss: 0.3451
Epoch 15/15
143/143 ————— 23s 160ms/step - accuracy: 0.8769 - loss: 0.3083

```

### Lampiran 9 Proses lengkap epoch saat melakukan training model 90:10

```

Epoch 1/15
160/160 ————— 35s 207ms/step - accuracy: 0.5463 - loss: 1.0162
Epoch 2/15
160/160 ————— 24s 153ms/step - accuracy: 0.6614 - loss: 0.8269
Epoch 3/15
160/160 ————— 25s 153ms/step - accuracy: 0.7123 - loss: 0.7313
Epoch 4/15
160/160 ————— 24s 152ms/step - accuracy: 0.7383 - loss: 0.6597
Epoch 5/15
160/160 ————— 25s 156ms/step - accuracy: 0.7520 - loss: 0.6183
Epoch 6/15
160/160 ————— 25s 154ms/step - accuracy: 0.7680 - loss: 0.5824
Epoch 7/15
160/160 ————— 24s 153ms/step - accuracy: 0.7876 - loss: 0.5411
Epoch 8/15
160/160 ————— 25s 153ms/step - accuracy: 0.7984 - loss: 0.5005
Epoch 9/15
160/160 ————— 25s 155ms/step - accuracy: 0.8169 - loss: 0.4601
Epoch 10/15
160/160 ————— 25s 155ms/step - accuracy: 0.8280 - loss: 0.4224
Epoch 11/15
160/160 ————— 26s 160ms/step - accuracy: 0.8424 - loss: 0.3935
Epoch 12/15
160/160 ————— 26s 164ms/step - accuracy: 0.8495 - loss: 0.3685
Epoch 13/15
160/160 ————— 26s 162ms/step - accuracy: 0.8595 - loss: 0.3433
Epoch 14/15
160/160 ————— 26s 161ms/step - accuracy: 0.8668 - loss: 0.3228
Epoch 15/15
160/160 ————— 24s 152ms/step - accuracy: 0.8834 - loss: 0.2850

```