

**TUGAS AKHIR  
SKEMA SKRIPSI**

**PAICODE: AGENTIC AI BERBASIS CLI UNTUK  
OTOMASI AKTIVITAS PEMROGRAMAN DAN  
PENGEMBANGAN PERANGKAT LUNAK DI  
LINUX YANG DITENAGAI LLM MELALUI API**



**I PUTU GEDE GILANG TEJA KRISHNA  
NIM : 225410001**

**PROGRAM STUDI INFORMATIKA  
PROGRAM SARJANA  
FAKULTAS FAKULTAS TEKNOLOGI INFORMASI  
UNIVERSITAS TEKNOLOGI DIGITAL INDONESIA  
YOGYAKARTA  
2025**

**TUGAS AKHIR  
SKEMA SKRIPSI**

**PAICODE: AGENTIC AI BERBASIS CLI UNTUK  
OTOMASI AKTIVITAS PEMROGRAMAN DAN  
PENGEMBANGAN PERANGKAT LUNAK DI  
LINUX YANG DITENAGAI LLM MELALUI API**

Diajukan sebagai salah satu syarat untuk menyelesaikan studi pada  
Program Sarjana  
Program Studi INFORMATIKA  
Fakultas FAKULTAS TEKNOLOGI INFORMASI  
Universitas Teknologi Digital Indonesia

Disusun Oleh

**I PUTU GEDE GILANG TEJA KRISHNA  
NIM : 225410001**

**PROGRAM STUDI INFORMATIKA  
PROGRAM SARJANA  
FAKULTAS FAKULTAS TEKNOLOGI INFORMASI  
UNIVERSITAS TEKNOLOGI DIGITAL INDONESIA  
YOGYAKARTA  
2025**

## **HALAMAN PERSETUJUAN UJIAN TUGAS AKHIR**

Judul : PAICODE: AGENTIC AI BERBASIS CLI UNTUK OTOMASI AKTIVITAS PEMROGRAMAN DAN PENGELOLAAN PERANGKAT LUNAK DI LINUX YANG DITENAGAI LLM MELALUI API  
Nama : I PUTU GEDE GILANG TEJA KRISHNA  
NIM : 225410001  
Program Studi : INFORMATIKA  
Program : Sarjana  
Semester : Ganjil  
Tahun Akademik : 2024/2025

Telah diperiksa dan disetujui untuk diujikan  
di hadapan Dewan Penguji Tugas Akhir

Yogyakarta, 24 November 2025

Dosen Pembimbing,

Dr. Bambang Purnomasidi Dwi Putranto, S.E., Akt., S.Kom., MMSI  
NIDN: 0505058801

## **HALAMAN PENGESAHAN**

### **JUDUL TUGAS AKHIR**

Telah dipertahankan di depan Dewan Pengaji dan dinyatakan diterima untuk

memenuhi sebagian persyaratan guna memperoleh

Gelar Sarjana Komputer

Program Studi INFORMATIKA

Fakultas FAKULTAS TEKNOLOGI INFORMASI

Universitas Teknologi Digital Indonesia

Yogyakarta, 24 November 2025

Dewan Pengaji

NIDN

1. Wagito, S.T., M.T. (Ketua)

.....

2. Dr. Bambang Purnomasidi Dwi Putranto, S.E., Akt., S.Kom., MMSI (Sekretaris) .....

.....

3. Ariesta Damayanti, S.Kom., M.Cs. (Anggota)

.....

Mengetahui  
Ketua Program Studi INFORMATIKA

Dini Fakta Sari, S.T., M.T.

NIDN: .....

---

## **HALAMAN MOTTO DAN PERSEMPAHAN**

### ***MOTTO***

*"Saya adalah orang yang percaya bahwa dunia ini terlalu luas jika dilihat hanya dari satu sudut pandang."*

### **PERSEMPAHAN**

Tugas Akhir ini saya persembahkan kepada:

Kedua orang tua tercinta yang telah memberikan doa,  
dukungan, dan kasih sayang yang tiada henti.

Seluruh keluarga besar yang senantiasa memberikan  
motivasi dan semangat.

Para guru dan dosen yang telah membimbing  
dan memberikan ilmu yang bermanfaat.

Seluruh teman-teman di kampus dan  
rekan seperjuangan UTDI THE ARCADE.

---

## **HALAMAN PERNYATAAN KEASLIAN**

Saya yang bertanda tangan di bawah ini:

Nama : I PUTU GEDE GILANG TEJA KRISHNA  
NIM : 225410001  
Program Studi : INFORMATIKA  
Fakultas : FAKULTAS TEKNOLOGI INFORMASI

Dengan ini saya menyatakan bahwa naskah Tugas Akhir ini belum pernah diajukan untuk memperoleh gelar Sarjana Komputer di suatu Perguruan Tinggi, dan sepanjang pengetahuan saya tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara sah diacu dalam naskah ini dan disebutkan dalam daftar pustaka.

Demikian pernyataan ini dibuat dengan sebenar-benarnya untuk dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 22 Desember 2025

**I PUTU GEDE GILANG TEJA  
KRISHNA**

NIM: 225410001

---

## PRAKATA

Puji syukur ke hadirat Tuhan Yang Maha Esa atas segala rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul **PAI-CODE: AGENTIC AI BERBASIS CLI UNTUK OTOMASI AKTIVITAS PEMROGRAMAN DAN PENGEMBANGAN PERANGKAT LUNAK DI LINUX YANG DITENAGAI LLM MELALUI API**. Tugas Akhir ini disusun sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer pada Program Studi INFORMATIKA, FAKULTAS TEKNOLOGI INFORMASI, UNIVERSITAS TEKNOLOGI DIGITAL INDONESIA.

Penulis menyadari bahwa penyelesaian Tugas Akhir ini tidak lepas dari bantuan, bimbingan, dan dukungan berbagai pihak. Oleh karena itu, penulis menyampaikan terima kasih kepada:

1. Tuhan Yang Maha Esa atas segala rahmat, kesehatan, dan kemudahan yang diberikan selama proses penelitian.
2. Orang tua dan keluarga yang senantiasa memberikan doa, dukungan moral, dan motivasi yang tiada henti.
3. Bapak Dr. Bambang Purnomasidi Dwi Putranto, S.E., Akt., S.Kom., MMSI selaku dosen pembimbing yang telah memberikan bimbingan, arahan, dan masukan yang sangat berharga selama penyusunan Tugas Akhir ini.
4. Seluruh dosen dan staf FAKULTAS TEKNOLOGI INFORMASI yang telah memberikan ilmu, fasilitas, dan dukungan selama masa perkuliahan.
5. Rekan-rekan mahasiswa yang telah memberikan bantuan, diskusi, dan semangat selama proses penelitian.

6. Semua pihak yang tidak dapat disebutkan satu per satu yang telah membantu dalam penyelesaian Tugas Akhir ini.

Penulis menyadari bahwa Tugas Akhir ini masih jauh dari sempurna. Oleh karena itu, penulis mengharapkan kritik dan saran yang membangun untuk perbaikan di masa mendatang. Semoga Tugas Akhir ini dapat memberikan manfaat bagi pembaca dan perkembangan ilmu pengetahuan.

Yogyakarta, 22 Desember 2025

**Penulis**

---

## **Ucapan Terima Kasih**

Dengan penuh rasa syukur, penulis menyampaikan terima kasih yang sebesar-besarnya kepada semua pihak yang telah memberikan dukungan moral maupun material sehingga skripsi ini dapat diselesaikan.

Secara khusus, ucapan terima kasih ditujukan kepada:

1. Tuhan Yang Maha Esa atas rahmat dan karunia-Nya.
2. Orang tua dan keluarga atas doa, dukungan, dan pengorbanan yang diberikan.
3. Dosen pembimbing atas bimbingan dan arahan selama penyusunan skripsi.
4. Para dosen penguji atas masukan dan koreksi yang konstruktif.
5. Seluruh dosen dan staf di FAKULTAS TEKNOLOGI INFORMASI serta rekan-rekan mahasiswa.

Semoga segala bantuan yang telah diberikan menjadi amal kebaikan dan mendapatkan balasan yang setimpal.

Yogyakarta, .....

**I PUTU GEDE GILANG TEJA KRISHNA**

NIM: 225410001

---

## INTISARI

Penelitian ini mengusulkan **Paicode**, sebuah agen AI berbasis Command Line Interface (CLI) untuk membantu proses pengembangan perangkat lunak secara interaktif dengan arsitektur *Single-Shot Intelligence*. Sistem berjalan pada lingkungan terminal lokal dan melakukan **operasi berkas tingkat-aplikasi di ruang kerja proyek (project workspace)**; namun **mengirimkan cuplekan kode/konteks ke layanan LLM (Gemini) melalui API** untuk keperluan inferensi. Oleh karena itu, aspek privasi dan kerahasiaan kode **bergantung pada kebijakan penyedia API**, sedangkan pengamanan lokal difokuskan pada kebijakan *path security*. Himpunan perintah yang disediakan (mis. READ, WRITE, MODIFY, TREE, LIST\_PATH) memungkinkan agen mengobserverasi proyek, memanipulasi berkas, dan memodifikasi kode secara terarah dengan sistem perubahan berbasis *diff*.

Arsitektur *Single-Shot Intelligence* mengoptimalkan efisiensi dengan sistem panggilan API yang terdiri dari: (1) klasifikasi intensi, (2) acknowledgment dinamis, (3) fase perencanaan untuk analisis mendalam dan perencanaan komprehensif dalam format JSON, (4) fase eksekusi adaptif yang dapat berjalan dalam 1-3 subfase berdasarkan kompleksitas tugas, dan (5) saran langkah berikutnya. Sistem mencakup manajemen API key tunggal dengan migrasi otomatis dari sistem multi-key, *interrupt handling* (Ctrl+C), dan pencatatan sesi ke `.pai_history`.

Metode yang digunakan adalah *Research and Development* (R&D) dengan pendekatan *prototyping* iteratif. Evaluasi dilakukan melalui skenario tugas representatif, dengan metrik efisiensi (jumlah panggilan API), ketepatan hasil (kompilasi/eksekusi), dan kepatuhan keamanan *path*. Hasil menunjukkan bahwa agen *stateful* dengan arsitektur *Single-Shot Intelligence* dan pembatasan perubahan berbasis *diff* dengan threshold ganda (500 baris absolut dan 50% ratio maksimal) memudahkan pengembangan bertahap sambil menekan risiko penimpaan berkas. Sistem eksekusi adaptif dengan 1-3 subfase terbukti

lebih efisien dibandingkan pendekatan tradisional yang memerlukan banyak panggilan API berulang, dengan tetap mempertahankan kualitas hasil yang optimal.

**Kata kunci:** agentic AI, CLI, LLM, API, Single-Shot Intelligence, keamanan *path*, pengembangan perangkat lunak.

---

## ABSTRACT

This thesis presents **Paicode**, an agentic AI for the Command Line Interface (CLI) that assists software development through interactive, stateful workflows with a *Single-Shot Intelligence* architecture. The system runs on a local terminal and performs **application-level file operations within the project workspace**, while **sending code/context snippets to an external LLM (Gemini) via API** for inference. Consequently, privacy and confidentiality **depend on the provider's policy**, whereas local safeguards focus on path-security policies. A compact set of commands (e.g., `READ`, `WRITE`, `MODIFY`, `TREE`, `LIST_PATH`) enables the agent to observe the project, manipulate files, and apply targeted code modifications with *diff*-based change system.

The *Single-Shot Intelligence* architecture optimizes efficiency through an API call system consisting of: (1) intent classification, (2) dynamic acknowledgement, (3) planning phase for deep analysis and comprehensive JSON-based planning, (4) adaptive execution phase that can run in 1-3 sub-phases based on task complexity, and (5) next-step suggestions. The system includes single API key management with automatic migration from multi-key systems, *interrupt handling* (`Ctrl+C`), and session logging to `.pai_history`.

We adopt a Research and Development approach with iterative prototyping. The evaluation uses representative programming scenarios and measures efficiency (API call count), correctness (build/run), and security compliance. Results indicate that a stateful agent with *Single-Shot Intelligence* and *diff*-based change constraints with dual thresholds (500-line absolute and 50% maximum ratio) facilitates incremental development while reducing the risk of unintended overwrites. The adaptive execution system with 1-3 sub-phases proves more efficient than traditional approaches requiring multiple repetitive API calls, while maintaining optimal result quality.

**Keywords:** agentic AI, CLI, LLM, API, Single-Shot Intelligence, path security, software engineering.

---

## **Daftar Singkatan**

<b>AI</b>	Kecerdasan Buatan (Artificial Intelligence)
<b>LLM</b>	Large Language Model
<b>CLI</b>	Command Line Interface
<b>TUI</b>	Text-based User Interface
<b>R&amp;D</b>	Research and Development
<b>API</b>	Application Programming Interface
<b>JSON</b>	JavaScript Object Notation
<b>gRPC</b>	Google Remote Procedure Call
<b>MIT</b>	Massachusetts Institute of Technology (License)
<b>OS</b>	Operating System
<b>UI</b>	User Interface

---

## **Daftar Simbol**

$t$	Waktu (detik)
$n$	Jumlah langkah/perintah
$\Delta$	Perubahan/delta (baris yang diubah)
$S$	Skor keberhasilan eksekusi

---

## Daftar Istilah

<b>CLI</b>	Command Line Interface; antarmuka baris perintah pada terminal.
<b>LLM</b>	Large Language Model; model bahasa berskala besar untuk inferensi teks/kode.
<b>API</b>	Application Programming Interface; antarmuka pemrograman untuk mengakses layanan (mis. LLM).
<b>control/data flow</b>	Pola arus kontrol dan data antar komponen dalam arsitektur sistem yang menggambarkan urutan eksekusi dan pertukaran informasi.
<b>workspace controller</b>	Modul pengatur workspace yang memusatkan fungsi-fungsi operasi tingkat-aplikasi pada workspace proyek, termasuk validasi <i>path</i> , pelarangan <i>path</i> sensitif, dan modifikasi berbasis <i>diff</i> .
<b>path</b>	Jalur berkas/direktori pada workspace proyek (contoh: <code>/home/user/project/main.py</code> ).
<b>path security</b>	Kebijakan keamanan terkait path: normalisasi, validasi root, dan blokir direktori sensitif untuk mencegah akses yang tidak sah.
<b>path traversal</b>	Teknik atau upaya mengakses direktori/berkas di luar cakupan yang diizinkan dengan memanipulasi path (mis. menggunakan segmen <code>..</code> ).
<b>deny-list</b>	Daftar path/pola yang dilarang untuk diakses atau dimodifikasi (mis. <code>.env</code> , <code>.git</code> , <code>venv/</code> , <code>__pycache__/</code> , <code>.vscode/</code> ).
<b>project files (berkas proyek)</b>	Berkas-berkas aplikasi dalam workspace proyek yang dapat dibaca/ditulis/dimodifikasi oleh Paicode (mis. kode sumber, konfigurasi proyek, README).

<b>diff</b>	Representasi perubahan antar versi berkas (baris ditambah/diubah/dihapus).
<b>stateful</b>	Menjaga konteks/riwayat interaksi agar mempengaruhi langkah berikutnya.
<b>guardrail</b>	Pembatas/safeguard untuk mengurangi tindakan berisiko (mis. pembatasan ruang perubahan).
<b>workspace</b>	Direktori/lingkungan kerja proyek aktif tempat berkas proyek dikelola dan dimanipulasi.
<b>repository root</b>	Direktori akar dari repository proyek; menjadi patokan validasi dan normalisasi <i>path</i> .
<b>rate limit</b>	Batas kuota/kecepatan permintaan API dalam jangka waktu tertentu yang ditetapkan penyedia layanan.
<b>tokenization</b>	Proses memecah teks menjadi unit-unit token yang diproses LLM; mempengaruhi biaya dan <i>context window</i> .
<b>prompt</b>	Instruksi atau masukan yang diberikan ke LLM untuk menghasilkan keluaran.
<b>context window</b>	Batas panjang konteks (jumlah token) yang dapat dipertimbangkan LLM pada satu permintaan.
<b>API key</b>	Kredensial rahasia untuk mengakses layanan API; harus disimpan aman (jangan ditulis di repository publik).
<b>Single-Shot Intelligence</b>	Arsitektur agen AI yang mengoptimalkan efisiensi dengan sistem panggilan API terbatas: klasifikasi intensi, acknowledgment dinamis, perencanaan JSON, eksekusi adaptif 1-3 subfase, dan saran langkah berikutnya.
<b>agentic AI</b>	Sistem kecerdasan buatan yang mampu bertindak secara otonom dengan kemampuan observasi, perencanaan, dan eksekusi dalam lingkungan tertentu.
<b>acknowledgment dinamis</b>	Respons konfirmasi yang diberikan agen untuk mengakui dan memahami permintaan pengguna sebelum mulai perencanaan.
<b>interrupt handling</b>	Mekanisme penanganan interupsi (Ctrl+C) yang memungkinkan pengguna menghentikan respons AI tanpa keluar dari sesi.

<b>atomic write</b>	Teknik penulisan berkas yang menggunakan file sementara ( <code>tempfile</code> ) untuk memastikan operasi tulis berhasil sepenuhnya atau gagal total, mencegah korupsi data.
<b>threshold ganda</b>	Sistem pembatasan modifikasi berkas dengan dua kriteria: batas absolut (500 baris) dan batas relatif (50% dari total baris berkas).
<b>SENSITIVE _PATTERNS</b>	Daftar 7 pola direktori/berkas sensitif yang diblokir akses: <code>.env, .git, venv, __pycache__, .pai_history, .idea, .vscode.</code>
<b>noise suppression</b>	Teknik menekan log yang berisik dari library <code>gRPC/absl</code> menggunakan environment variables khusus.
<b>entry point</b>	Titik masuk aplikasi yang didefinisikan dalam <code>setup.cfg</code> sebagai console script ( <code>pai = paicode.cli:main</code> ).
<b>prototyping iteratif</b>	Metode pengembangan dengan siklus berulang: perancangan, implementasi, uji coba, dan perbaikan untuk validasi asumsi dan penyempurnaan rancangan.
<b>markdown artifacts</b>	Sisa-sisa format markdown (seperti <code>~~</code> , <code>**bold**</code> ) dalam output LLM yang perlu dibersihkan sebelum ditampilkan.
<b>spinner status</b>	Indikator visual berputar yang menunjukkan bahwa sistem sedang memproses (misalnya saat LLM berpikir).
<b>multiline input</b>	Kemampuan input teks multi-baris dengan dukungan key bindings khusus (Alt+Enter untuk baris baru, Enter untuk submit).

---

## Daftar Isi

<b>HALAMAN PENGESAHAN</b>	ii
<b>HALAMAN MOTTO DAN PERSEMBAHAN</b>	iii
<b>HALAMAN PERNYATAAN KEASLIAN</b>	v
<b>PRAKATA</b>	vi
<b>Ucapan Terima Kasih</b>	viii
<b>INTISARI</b>	ix
<b>ABSTRACT</b>	xi
<b>Daftar Singkatan</b>	xii
<b>Daftar Simbol</b>	xiii
<b>Daftar Istilah</b>	xiv
<b>1 Pendahuluan</b>	1
1.1 Latar Belakang . . . . .	1
1.2 Rumusan Masalah . . . . .	2
1.3 Ruang Lingkup . . . . .	2
1.4 Tujuan Penelitian . . . . .	3
1.5 Manfaat Penelitian . . . . .	3
1.6 Sistematika Penulisan . . . . .	4
<b>2 TINJAUAN PUSTAKA DAN DASAR TEORI</b>	5
2.1 Tinjauan Pustaka . . . . .	5
2.1.1 AI Coding Assistant Terintegrasi (IDE-based) . . . . .	5

2.1.2	CLI-based AI Chat Tools . . . . .	5
2.1.3	Autonomous Software Engineers . . . . .	6
2.1.4	Posisi Paicode . . . . .	6
2.1.5	Perbandingan dengan Penelitian Sebelumnya . . . . .	6
2.1.6	Posisi Penelitian . . . . .	8
2.2	Dasar Teori . . . . .	9
2.2.1	Command Line Interface (CLI) . . . . .	9
2.2.2	AI Agent . . . . .	9
2.2.3	Large Language Model (LLM) . . . . .	10
2.2.4	Perbedaan LLM dan Agen AI . . . . .	10
2.2.5	Arsitektur dan Kebijakan Data . . . . .	10
2.2.6	Manajemen Dependensi dengan pip dan Virtual Environment . . . . .	11
2.2.7	Antarmuka Terminal dengan rich dan prompt_toolkit . . . . .	11
<b>3</b>	<b>Metode Penelitian</b>	<b>13</b>
3.1	Metode Pengembangan . . . . .	13
3.1.1	Trade-off Metodologis . . . . .	14
3.2	Arsitektur Sistem . . . . .	14
3.3	Visualisasi Metodologi . . . . .	15
3.4	Alat dan Lingkungan . . . . .	18
3.5	Prosedur Penelitian . . . . .	19
<b>4</b>	<b>Implementasi dan Pembahasan</b>	<b>21</b>
4.1	Implementasi Paicode . . . . .	21
4.1.1	Instalasi . . . . .	21
4.1.2	Konfigurasi API Key . . . . .	21
4.1.3	Menjalankan Agen . . . . .	22
4.2	Alur Interaksi dengan Single-Shot Intelligence . . . . .	23
4.2.1	Cuplikan Kode Kunci . . . . .	23
4.3	Cuplikan Log Implementasi . . . . .	27
4.4	Tabel Skenario Pengujian . . . . .	39
4.5	Tabel Metrik Evaluasi . . . . .	40
4.6	Tabel Konfigurasi Lingkungan . . . . .	40
4.7	Contoh Sesi . . . . .	41
4.8	Evaluasi dan Analisis Mendalam . . . . .	41

4.8.1	Metrik Kuantitatif . . . . .	41
4.8.2	Analisis Kualitatif: Mengapa Single-Shot Intelligence Efektif? . . . . .	42
4.8.3	Analisis Kegagalan dan Limitasi . . . . .	43
4.8.4	Perbandingan dengan Baseline Manual . . . . .	44
4.8.5	Refleksi Kritis: Apakah Ini "Asisten" atau "Autopilot"? .	44
<b>5</b>	<b>Kesimpulan dan Saran</b>	<b>45</b>
5.1	SIMPULAN . . . . .	45
5.2	SARAN . . . . .	46
<b>A</b>	<b>Lampiran A</b>	<b>48</b>
A.1	Konfigurasi Lingkungan . . . . .	48
A.2	Instruksi Instalasi (venv + pip) . . . . .	48
A.3	Cuplikan Log Sesi Agen . . . . .	49
A.4	Listing Lengkap Modul Kunci . . . . .	49

---

## Daftar Gambar

2.1 Konsep arsitektur agentic AI di lingkungan CLI dengan inferensi LLM melalui API. . . . .	11
2.2 Model interaksi <i>stateful</i> dan <i>feedback loop</i> pada sesi agen. . . . .	12

---

## Daftar Tabel

2.1	Perbandingan Penelitian Terdahulu dengan Penelitian yang Dilakukan . . . . .	7
2.2	Ilustrasi komparasi konseptual antara pendekatan ekstensi editor, layanan daring, dan CLI dengan integrasi LLM via API. . .	12
3.1	Modul dan Dependensi Komponen Paicode . . . . .	16
3.2	Urutan Interaksi Sesi Agen dengan Single-Shot Intelligence . . .	17
3.3	Rangkuman Validasi Keamanan <i>Path</i> . . . . .	18
4.1	Skenario Pengujian Paicode . . . . .	39
4.2	Metrik Evaluasi dan Definisi Operasional . . . . .	40
4.3	Konfigurasi Lingkungan Uji . . . . .	40

# BAB 1

---

## Pendahuluan

### 1.1 Latar Belakang

Perkembangan *Large Language Model* (LLM) telah mendorong lahirnya beragam asisten pemrograman yang mampu membantu pengembang perangkat lunak dalam menulis, meninjau, dan memodifikasi kode. Meskipun demikian, sebagian besar asisten tersebut beroperasi sebagai ekstensi editor atau layanan berbasis *cloud* yang menyimpan, memproses, atau melatih dari data pengguna. Kondisi ini menimbulkan kekhawatiran terkait privasi, kendali atas data, serta ketergantungan pada antarmuka tertentu.

Di sisi lain, *Command Line Interface* (CLI) tetap menjadi lingkungan kerja yang penting bagi banyak pengembang karena sifatnya yang ringan, dapat diotomasi, dan mudah diintegrasikan dengan beragam alat. Integrasi kemampuan agen cerdas yang *stateful* dan *proactive* ke dalam CLI berpotensi mempercepat proses pengembangan perangkat lunak. Dalam konteks Paicode, sistem berjalan pada terminal lokal dan mengeksekusi tindakan langsung pada **berkas proyek di workspace**; namun, cuplikan kode/konteks **dikirim ke layanan LLM melalui API** untuk keperluan inferensi [2, 7, 1]. Dengan demikian, aspek privasi/kerahasiaan kode **bergantung pada kebijakan penyedia API**, sementara pengamanan di sisi lokal difokuskan pada kebijakan *path security* (keamanan *path*) dan pembatasan perubahan berbasis *diff*.

Penelitian ini menghadirkan **Paicode**, sebuah agen AI berbasis CLI yang dirancang untuk membantu proses pengembangan perangkat lunak secara interaktif dengan arsitektur *Single-Shot Intelligence*. Paicode mampu: (i) mengobservasi struktur proyek (TREE, LIST\_PATH); (ii) membaca dan menulis berkas proyek (READ, WRITE); (iii) memodifikasi kode secara terarah dengan sistem perubahan berbasis *diff* dengan threshold ganda: 500 baris absolut dan 50%

ratio maksimal (**MODIFY**); (iv) menegakkan kebijakan keamanan *path* pada berkas proyek (memblokir akses ke direktori sensitif seperti `.git`, `venv`, dan `.env`); (v) melakukan klasifikasi intensi pengguna (*chat* vs *task*); (vi) mengoptimalkan efisiensi dengan sistem *Single-Shot Intelligence* yang mencakup *acknowledgment* dinamis, perencanaan JSON, dan eksekusi adaptif 1-3 subfase; serta (vii) menyediakan penanganan interupsi (*interrupt handling*) untuk kontrol sesi yang lebih baik. Sistem diimplementasikan pada lingkungan Ubuntu dengan bahasa pemrograman Python, pengelolaan dependensi melalui pip dan virtual environment, manajemen API key tunggal dengan migrasi otomatis dari sistem multi-key, dan menggunakan API Gemini sebagai LLM.

## 1.2 Rumusan Masalah

Berdasarkan latar belakang tersebut, rumusan masalah yang diajukan adalah:

*Bagaimana merancang, mengimplementasikan, dan mengevaluasi agen AI berbasis CLI dengan arsitektur Single-Shot Intelligence yang mampu mengotomatisi aktivitas pemrograman secara aman melalui kebijakan path security dan pembatasan perubahan berbasis diff, serta terintegrasi dengan LLM melalui API?*

## 1.3 Ruang Lingkup

Agar fokus penelitian terjaga dan implementasi dapat dilakukan secara terukur, batasan-batasan berikut ditetapkan:

- Lingkungan target adalah sistem operasi Ubuntu (Linux) dengan antarmuka CLI.
- Bahasa pemrograman utama adalah Python; contoh dan skenario uji berfokus pada ekosistem Python/Unix.
- Layanan LLM eksternal menggunakan API Gemini; kualitas respons bergantung pada model dan tidak menjadi ruang lingkup untuk dioptimasi ulang.
- Dukungan multi-pengguna, kolaborasi real-time, dan integrasi langsung dengan editor tidak dibahas pada versi ini.

- Aspek visual seperti diagram dan ilustrasi antarmuka ditunda pada tahap akhir; fokus laporan adalah narasi dan hasil teknis.

## 1.4 Tujuan Penelitian

Tujuan penelitian ini adalah membangun dan mengevaluasi sebuah agen AI berbasis CLI yang dapat membantu pengembang dalam proses pemrograman secara interaktif dengan arsitektur *Single-Shot Intelligence*. Secara khusus, penelitian menargetkan:

1. Merancang arsitektur Paicode yang mencakup modul agen dengan *Single-Shot Intelligence* (klasifikasi intensi, fase perencanaan, dan fase eksekusi dalam 2 panggilan API), jembatan LLM dengan manajemen API key tunggal, antarmuka CLI dengan *interrupt handling*, lapisan keamanan *path* pada berkas proyek, serta komponen tampilan terminal berbasis *rich*.
2. Mengimplementasikan kemampuan observasi proyek, manipulasi berkas, dan modifikasi kode terarah dengan mekanisme *diff-aware* yang mencegah penimpaan berkas tidak diinginkan dan memblokir akses ke direktori sensitif.
3. Mengintegrasikan fitur-fitur interaktif seperti pencatatan sesi ke `.pai_history`, penanganan interupsi (Ctrl+C), dan antarmuka terminal yang responsif dengan dukungan input multiline.
4. Menyusun prosedur evaluasi dengan skenario tugas pemrograman yang representatif dan mengukur efisiensi panggilan API, ketepatan hasil, serta kepatuhan keamanan *path*.

## 1.5 Manfaat Penelitian

Manfaat yang diharapkan dari penelitian ini meliputi:

- **Akademis:** menyediakan studi kasus dan arsitektur rujukan untuk pengembangan agen AI berbasis CLI dengan integrasi LLM melalui API, serta memperkaya literatur mengenai integrasi LLM dalam alur kerja rekayasa perangkat lunak.

- **Praktis:** menghadirkan alat bantu pengembangan perangkat lunak dengan kelebihan spesifik sebagai berikut:
  1. **Efisiensi Biaya dan Token:** Menggunakan arsitektur *Single-Shot Intelligence* yang memadatkan proses perencanaan dan eksekusi menjadi dua panggilan utama, mengurangi biaya API dibandingkan agen berbasis *chat-loop* konvensional.
  2. **Keamanan Terkendali:** Menerapkan kebijakan keamanan *path* (path security) yang memblokir akses ke direktori sensitif (seperti `.git`, `.env`) dan mekanisme modifikasi berbasis *diff* untuk mencegah perubahan destruktif masif.
  3. **Fleksibilitas Lingkungan:** Beroperasi sebagai utilitas CLI yang ringan dan agnostik terhadap editor kode (IDE-agnostic), sehingga dapat digunakan di server tanpa antarmuka grafis (headless) maupun sebagai pendamping editor apa pun di OS berbasis Linux.

## 1.6 Sistematika Penulisan

Laporan tugas akhir ini disusun dengan sistematika penulisan sebagai berikut:

### BAB I PENDAHULUAN

Memuat latar belakang, rumusan masalah, ruang lingkup, tujuan penelitian, manfaat penelitian, dan sistematika penulisan.

### BAB II TINJAUAN PUSTAKA DAN DASAR TEORI

Memuat tinjauan pustaka dari penelitian terdahulu yang relevan serta dasar teori yang mendukung penelitian ini.

### BAB III METODE PENELITIAN

Menjelaskan bahan, peralatan, prosedur penelitian, serta analisis dan perancangan sistem.

### BAB IV IMPLEMENTASI DAN PEMBAHASAN

Menguraikan proses implementasi sistem, hasil uji coba, dan pembahasan mengenai hasil yang diperoleh.

### BAB V PENUTUP

Berisi kesimpulan dari penelitian dan saran untuk pengembangan selanjutnya.

## BAB 2

---

### TINJAUAN PUSTAKA DAN DASAR TEORI

#### 2.1 Tinjauan Pustaka

Perkembangan alat bantu pemrograman berbasis AI berkembang pesat dalam beberapa tahun terakhir. Berikut adalah tinjauan terhadap beberapa solusi *state-of-the-art* yang relevan dengan penelitian ini:

##### 2.1.1 AI Coding Assistant Terintegrasi (IDE-based)

GitHub Copilot [4] merupakan contoh paling prominen dari asisten pemrograman yang terintegrasi langsung ke dalam lingkungan pengembangan (IDE) seperti VS Code. Copilot unggul dalam memberikan saran *autocomplete* real-time dan fungsi obrolan kontekstual. Namun, pendekatannya sangat bergantung pada antarmuka editor visual dan beroperasi sebagai "pilot pendamping" (copilot) alih-alih agen otonom yang dapat melakukan tugas kompleks lintas berkas secara mandiri tanpa intervensi pengguna untuk setiap langkahnya.

##### 2.1.2 CLI-based AI Chat Tools

Alat seperti Aider [3] membawa kemampuan LLM ke dalam terminal (CLI). Aider memungkinkan pengguna untuk melakukan *pair programming* dengan LLM langsung di terminal dan menerapkan perubahan pada git repository. Pendekatan ini mirip dengan Paicode dalam hal antarmuka berbasis teks. Perbedaannya, Paicode menekankan pada arsitektur *Single-Shot Intelligence* dengan fase perencanaan JSON eksplisit sebelum eksekusi, serta penerapan kebijakan keamanan *path* yang ketat untuk lingkungan korporasi atau sensitif,

sedangkan banyak alat CLI lain berfokus pada kecepatan interaksi *chat-apply* langsung.

### 2.1.3 Autonomous Software Engineers

Proyek seperti OpenDevin [8] dan SWE-agent [5] bertujuan menciptakan agen yang sepenuhnya otonom, mampu menyelesaikan isu GitHub dari awal hingga akhir tanpa interaksi manusia. Meskipun sangat canggih, pendekatan ini seringkali memerlukan akses sumber daya yang besar (Docker container penuh) dan kompleksitas tinggi untuk penyiapan. Paicode mengambil posisi tengah (middle-ground) dengan menyediakan agen *semi-autonomous* yang ringan (*lightweight*), berjalan native di OS tanpa kontainer berat, namun tetap memiliki kemampuan perencanaan (*planning*) untuk tugas multi-langkah.

### 2.1.4 Posisi Paicode

Dibandingkan dengan solusi di atas, Paicode menawarkan kebaruan pada kombinasi arsitektur *local-first* yang ringan namun terstruktur:

1. **Keamanan Terkendali:** Tidak seperti agen otonom penuh yang sering berjalan di sandboxed container karena risiko tinggi, Paicode dirancang aman untuk berjalan di *host* utama berkat *path security policy* dan *diff-based guardrails*.
2. **Efisiensi Token:** Dengan arsitektur perencanaan *single-shot*, Paicode mengurangi *round-trip* percakapan yang tidak perlu, berbeda dengan model *chat* standar.
3. **Transparansi Rencana:** Pengguna dapat melihat rencana aksi (dalam format JSON) sebelum eksekusi masif dilakukan, memberikan kontrol lebih baik daripada model *black-box*.

### 2.1.5 Perbandingan dengan Penelitian Sebelumnya

Tabel 2.1 merangkum perbedaan antara penelitian-penelitian terdahulu dengan penelitian yang akan dilakukan.

Dari Tabel 2.1 terlihat bahwa penelitian ini mengisi *gap* antara asisten pasif (seperti Copilot) dan agen otonom penuh (seperti OpenDevin) dengan

Tabel 2.1: Perbandingan Penelitian Terdahulu dengan Penelitian yang Dilakukan

<b>Aspek</b>	<b>Penelitian Terdahulu</b>	<b>Penelitian Ini (Paicode)</b>
<b>Platform</b>	IDE-based (Copilot), Web-based (ChatGPT Code Interpreter), Container-based (OpenDevin)	CLI native, berjalan langsung di terminal Linux tanpa container
<b>Arsitektur Agen</b>	Chat-loop iteratif (10-20 API calls) atau fully autonomous	Single-Shot Intelligence (2 API calls: planning + execution)
<b>Keamanan Lokal</b>	Sandboxed container (OpenDevin) atau tidak ada kontrol eksplisit (Copilot)	Path security policy + diff-based guardrails (threshold 500 baris, 50% ratio)
<b>Transparansi</b>	Black-box suggestions (Copilot) atau verbose logs (SWE-agent)	Explicit JSON planning phase dengan user approval
<b>Efisiensi</b>	High token consumption (chat-loop) atau resource-intensive (full containers)	Token-optimized (60-70% reduction) dan lightweight (native OS)
<b>Interaktivitas</b>	Passive suggestions (Copilot) atau fully autonomous (OpenDevin)	Semi-autonomous dengan interrupt handling (Ctrl+C)
<b>Fokus Penelitian</b>	General-purpose coding atau issue-solving automation	Secure, efficient, transparent automation untuk developer workflows

menawarkan pendekatan *semi-autonomous* yang efisien, aman, dan transparan. Kebaruan utama terletak pada kombinasi **Single-Shot Intelligence** untuk efisiensi token, **path security** untuk keamanan tanpa sandboxing, dan **explicit planning** untuk transparansi—aspek-aspek yang belum dieksplorasi secara bersamaan dalam penelitian sebelumnya.

### 2.1.6 Posisi Penelitian

Kontribusi penelitian ini ditempatkan pada ranah agentic AI untuk pengembangan perangkat lunak dengan karakteristik sebagai berikut:

- **CLI lokal dengan integrasi LLM via API:** agen berjalan di terminal, tindakan langsung tercermin pada **berkas proyek di workspace**; sementara inferensi dilakukan oleh LLM eksternal sehingga kebijakan data mengikuti penyedia API.
- **Arsitektur Single-Shot Intelligence:** alur kerja efisien yang mengoptimalkan penggunaan API dengan tepat 2 panggilan (perencanaan dan eksekusi), menggantikan pendekatan tradisional yang memerlukan 10-20 panggilan API.
- **Manajemen API key tunggal:** sistem manajemen API key yang disederhanakan dengan migrasi otomatis dari sistem multi-key untuk kemudahan penggunaan.
- **Keamanan berkas:** kebijakan pelarangan akses *path* sensitif dan validasi *path* mencegah *path traversal* dan operasi berisiko pada direktori seperti `.git`, `venv`, dan `.env`.
- **Modifikasi terarah berbasis diff:** perintah `MODIFY` memanfaatkan sistem *diff*-aware untuk membatasi ruang perubahan dan mencegah penimpaan berkas tidak diinginkan.
- **Fitur interaktif:** *interrupt handling* (`Ctrl+C`) untuk menghentikan respons AI tanpa keluar dari sesi, pencatatan sesi lengkap ke `.pai_history`, dan antarmuka terminal responsif dengan dukungan input multiline.
- **Keterulangan eksperimen:** penggunaan pip, virtual environment, dan Makefile memudahkan replikasi lingkungan dan dokumentasi langkah instalasi.

## 2.2 Dasar Teori

Bagian ini membahas konsep yang menjadi landasan penelitian: *Command Line Interface* (CLI), agen kecerdasan buatan (AI Agent), *Large Language Model* (LLM), arsitektur dan kebijakan data (integrasi LLM melalui API dan implikasi privasi), *Single-Shot Intelligence* untuk agen interaktif, sistem klasifikasi intensi, serta perangkat bantu yang digunakan seperti pip dan virtual environment untuk manajemen dependensi, `rich` dan `prompt_toolkit` untuk antarmuka terminal.

### 2.2.1 Command Line Interface (CLI)

CLI adalah antarmuka berbasis teks yang memungkinkan pengguna berinteraksi dengan sistem melalui perintah. Kelebihan CLI meliputi otomasi yang mudah, konsumsi sumber daya yang rendah, dan integrasi sederhana dengan alat lain melalui skrip. Dalam konteks pengembangan perangkat lunak, CLI memfasilitasi alur kerja yang ringkas dan dapat direproduksi.

### 2.2.2 AI Agent

AI Agent (sering disebut *agentic AI* dalam literatur; selanjutnya disingkat "agen AI") dalam penelitian ini dipahami sebagai sistem yang mampu mengobservasi lingkungan (struktur proyek dan isi berkas), merencanakan tindakan (mis. membuat, membaca, memodifikasi berkas), serta mengevaluasi hasil untuk langkah berikutnya. Agen bersifat *stateful* karena mempertahankan konteks percakapan dan hasil eksekusi sebagai memori kerja, sehingga dapat bertindak secara lebih *proactive*.

Pada implementasi Paicode, agen menggunakan arsitektur *Single-Shot Intelligence* yang terdiri dari beberapa komponen: (1) klasifikasi intensi untuk membedakan percakapan dan tugas, (2) *acknowledgment* dinamis untuk konfirmasi pemahaman, (3) fase perencanaan dengan analisis mendalam dan perencanaan komprehensif dalam format JSON, (4) fase eksekusi adaptif yang dapat berjalan dalam 1-3 subfase berdasarkan kompleksitas tugas, dan (5) saran langkah berikutnya. Sistem ini mengoptimalkan efisiensi dibandingkan pendekatan tradisional yang memerlukan banyak panggilan API berulang.

### 2.2.3 Large Language Model (LLM)

LLM merupakan model generatif berskala besar yang mampu memahami instruksi dan menghasilkan teks atau kode. Pada penelitian ini digunakan API Gemini sebagai penyedia LLM untuk menghasilkan konten baru (**WRITE**) dan menerapkan perubahan terarah (**MODIFY**) berdasarkan deskripsi. Prinsip kehati-hatian diterapkan dengan mekanisme pembatasan perubahan berbasis *diff* sehingga modifikasi tidak berskala besar tanpa kontrol [2, 7, 1, 10, 6, 9, 11].

### 2.2.4 Perbedaan LLM dan Agen AI

Pada skripsi ini penting untuk membedakan *Large Language Model* (LLM) dan *Agen AI*:

- **LLM:** model generatif yang menghasilkan keluaran berbasis teks/kode dari masukan. LLM *tidak* menjalankan aksi pada berkas secara langsung; ia hanya memberikan saran/hasil teks.
- **Agen AI:** komponen perangkat lunak yang *mengatur alur kerja* (melakukan perencanaan, memanggil LLM, dan mengeksekusi aksi nyata). Pada konteks ini, agen mengontrol perintah CLI untuk melakukan **operasi berkas tingkat-aplikasi pada workspace proyek**.
- **Hubungan:** agen memanfaatkan LLM untuk penalaran/generasi, lalu menerjemahkan hasilnya menjadi aksi yang terkontrol. Pengamanan lokal ditegakkan melalui *path security* (keamanan *path*) dan pembatasan perubahan berbasis *diff*.

### 2.2.5 Arsitektur dan Kebijakan Data

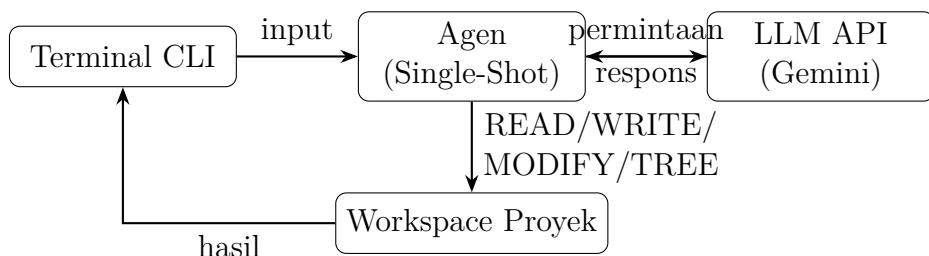
Paicode dijalankan pada terminal lokal dan melakukan tindakan langsung pada **berkas proyek di workspace**. Akan tetapi, untuk kebutuhan inferensi, cuplikan kode atau konteks **dikirim ke layanan LLM melalui API**. Implikasinya, privasi dan kerahasiaan kode **bergantung pada kebijakan penyedia API**. Pengamanan di sisi lokal diterapkan melalui kebijakan *path security* (keamanan *path*) serta pembatasan perubahan berbasis *diff* agar operasi berkas lebih terkendali.

## 2.2.6 Manajemen Dependensi dengan pip dan Virtual Environment

Paicode menggunakan pendekatan manajemen dependensi tradisional dengan pip dan virtual environment Python. Berkas `requirements.txt` mendeskripsikan dependensi yang diperlukan, sementara Makefile menyediakan otomasi untuk pembuatan virtual environment dan instalasi dependensi. Pendekatan ini memudahkan replikasi lingkungan dan instalasi alat. Pada implementasi Paicode, dependensi utama meliputi `google-generativeai` (versi  $\geq 0.5.4$ ), `rich` (versi  $\geq 13.7.1$ ), `Pygments` (versi  $\geq 2.16.0$ ), dan `prompt_toolkit` (versi  $\geq 3.0.43$ ).

## 2.2.7 Antarmuka Terminal dengan rich dan prompt\_toolkit

Paket `rich` dimanfaatkan untuk menyajikan hasil eksekusi secara terstruktur dan mudah dibaca (panel, warna, penyorotan sintaks, tabel, dan spinner status). Penyajian output yang jelas mendukung pengalaman interaktif dan penelusuran hasil tindakan agen. Selain itu, Paicode juga mengintegrasikan `prompt_toolkit` (opsional) untuk pengalaman input yang lebih baik dengan dukungan multiline editing dan key bindings. Jika `prompt_toolkit` tidak tersedia, sistem akan fallback ke `rich.prompt.Prompt`.

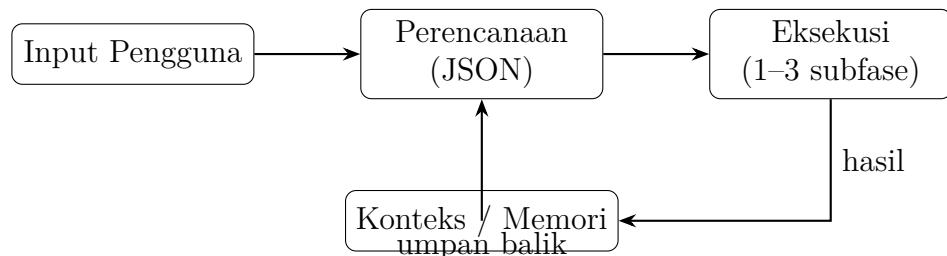


Gambar 2.1: Konsep arsitektur agentic AI di lingkungan CLI dengan inferensi LLM melalui API.

Pada Gambar 2.1 ditunjukkan pemetaan komponen utama (CLI, Agen, LLM, dan komponen workspace) beserta *control/data flow* antar komponen.

Pada Gambar 2.2 divisualisasikan hubungan antara masukan pengguna, perencanaan aksi, eksekusi alat, dan pembaruan konteks.

Pada Gambar 2.2 ditunjukkan perbedaan fokus dan pertukaran (trade-off) tingkat tinggi antar pendekatan.



Gambar 2.2: Model interaksi *stateful* dan *feedback loop* pada sesi agen.

Tabel 2.2: Ilustrasi komparasi konseptual antara pendekatan ekstensi editor, layanan daring, dan CLI dengan integrasi LLM via API.

	<b>Ekstensi Editor</b>	<b>Layanan Daring</b>
<b>Integrasi</b>	Sangat terintegrasi dengan IDE	Antarmuka web/remote
<b>Konteks</b>	Di editor, tergantung API	Di server; unggah/sinkron
<b>Privasi</b>	Bergantung vendor	Bergantung vendor
<b>Portabilitas</b>	Terikat IDE	Perlu akses internet
<b>CLI + LLM via API (Paicode)</b>		
<b>Integrasi</b>	Agen berjalan di terminal lokal; perubahan langsung pada workspace	
<b>Konteks</b>	Konteks lokal; cuplikan dikirim ke LLM via API	
<b>Privasi</b>	Tergantung kebijakan penyedia API; guardrail lokal	
<b>Portabilitas</b>	Editor-agnostic; cukup terminal Linux	

## BAB 3

---

### Metode Penelitian

#### 3.1 Metode Pengembangan

Penelitian ini menggunakan pendekatan *Research and Development* (R&D) dengan strategi *prototyping* iteratif. Pemilihan metode ini didasarkan pada beberapa pertimbangan:

1. **Eksplorasi Desain Agen Stateful:** Berbeda dengan aplikasi konvensional yang bersifat *stateless*, agen AI memerlukan manajemen konteks percakapan dan memori kerja yang kompleks. Pendekatan *prototyping* memungkinkan eksperimen cepat terhadap berbagai strategi manajemen state (misalnya, ukuran context window, format log sesi) tanpa komitmen arsitektur jangka panjang.
2. **Validasi Asumsi Keamanan:** Kebijakan *path security* dan pembatasan *diff* merupakan mekanisme novel yang belum teruji di konteks agen CLI. Siklus iteratif memungkinkan identifikasi edge case (seperti symbolic links, path traversal attacks) melalui pengujian langsung, yang sulit diprediksi hanya dari analisis teoritis.
3. **Optimasi Efisiensi Token:** Arsitektur *Single-Shot Intelligence* dikembangkan melalui iterasi bertahap—dimulai dari model *chat-loop* konvensional (10-20 panggilan API per tugas), kemudian dipadatkan menjadi sistem 2-panggilan melalui eksperimen empiris terhadap berbagai strategi prompt engineering.

### 3.1.1 Trade-off Metodologis

Pendekatan *prototyping* dipilih dibandingkan metode waterfall atau agile penuh dengan pertimbangan trade-off berikut:

- **Kelebihan:** Fleksibilitas tinggi untuk mengubah desain berdasarkan temuan empiris; cocok untuk domain yang belum mature (agentic AI untuk CLI); memungkinkan validasi konsep sebelum investasi besar pada infrastruktur.
- **Kekurangan:** Dokumentasi arsitektur dapat tertinggal jika iterasi terlalu cepat; risiko *scope creep* jika tidak ada batasan jelas per iterasi; potensi *technical debt* jika refactoring tidak dilakukan secara disiplin.
- **Mitigasi:** Setiap iterasi dibatasi pada satu fitur utama (misalnya, iterasi 1: path security; iterasi 2: diff-aware modification; iterasi 3: Single-Shot Intelligence); dokumentasi arsitektur diperbarui setelah setiap iterasi stabil; code review dilakukan sebelum merge ke branch utama.

## 3.2 Arsitektur Sistem

Arsitektur Paicode dirancang modular dan berlapis, dengan pembagian tanggung jawab yang jelas:

- **Antarmuka CLI (`cli.py`):** titik masuk perintah `pai` dan pengelola argumen (subperintah `auto`, `config`). Mendukung parameter `-model` dan `-temperature` untuk konfigurasi runtime LLM. Secara default, CLI memanggil sesi interaktif agen.
- **Agen (`agent.py`):** mengimplementasikan *Single-Shot Intelligence* yang mencakup: (1) klasifikasi intensi (*chat* vs *task*), (2) *acknowledgment* dinamis, (3) fase perencanaan untuk analisis mendalam dalam format JSON, (4) fase eksekusi adaptif dengan 1-3 subfase berdasarkan kompleksitas, dan (5) saran langkah berikutnya. Menyediakan 10 perintah: `READ`, `WRITE`, `MODIFY`, `TREE`, `LIST_PATH`, `MKDIR`, `TOUCH`, `RM`, `MV`, `FINISH`. Mengelola memori percakapan dengan pencatatan sesi ke `.pai_history`.

- **Jembatan LLM (llm.py)**: menangani konfigurasi API Gemini dengan manajemen API key tunggal. Membersihkan output dari markdown artifacts, menyediakan status spinner saat LLM berpikir, dan mengoptimalkan penggunaan token dengan sistem 2-panggilan API.
- **Manajemen Konfigurasi (config.py)**: menyimpan dan mengelola API key tunggal dalam format JSON di `/.config/pai-code/credentials.json` dengan izin berkas 0o600 (read-write owner only). Validasi API key Google (harus dimulai dengan "AIza" dan minimal 20 karakter). Mendukung operasi: `set`, `show`, `remove`, `validate`, dan migrasi otomatis dari sistem multi-key.
- **Pengatur Workspace (workspace.py)**: bertindak sebagai *workspace controller* yang menyediakan fungsi-fungsi terpusat untuk menjalankan operasi tingkat-aplikasi pada ruang kerja proyek. Sebelum aksi dieksekusi, modul ini menegakkan kebijakan *path security* (normalisasi, verifikasi akar, dan deny-list direktori sensitif seperti `.env`, `.git`, `venv`, `__pycache__`, `.pai_history`, `.idea`, `.vscode`). Sistem modifikasi berbasis *diff* dengan threshold 500 baris dan ratio maksimal 50% (dapat dikonfigurasi via `PAI MODIFY_THRESHOLD` dan `PAI MODIFY_MAX_RATIO`) mencegah penimpaan berkas tidak diinginkan dengan atomic write menggunakan tempfile.
- **Tampilan Terminal (ui.py)**: penyajian hasil eksekusi menggunakan `rich` (panel, warna, tabel, penyorotan sintaks, spinner status). Mendukung `prompt_toolkit` (opsional) untuk input multiline yang lebih baik.

Alur data tipikal dengan *Single-Shot Intelligence*: masukan pengguna (CLI) → klasifikasi intensi → *acknowledgment* dinamis → fase perencanaan (analisis JSON) → fase eksekusi adaptif (1-3 subfase) → saran langkah berikutnya → pencatatan konteks sebagai memori percakapan.

### 3.3 Visualisasi Metodologi

Bagian ini menyajikan visualisasi konsep menggunakan tabel dan daftar tersusuktur berbasis LaTeX.

Tabel 3.1: Modul dan Dependensi Komponen Paicode

Komponen	Deskripsi dan Dependensi Utama
CLI ( <code>cli.py</code> )	Titik masuk perintah, parsing argumen ( <code>-model</code> , <code>-temperature</code> ); memanggil sesi agen. Bergantung pada modul <code>agent</code> , <code>config</code> , dan <code>llm</code> .
Agen ( <code>agent.py</code> )	Implementasi <i>Single-Shot Intelligence</i> : klasifikasi intensi, <i>acknowledgment</i> dinamis, fase perencanaan JSON, fase eksekusi adaptif (1-3 subfase), dan saran langkah berikutnya. Mengelola memori percakapan, <i>interrupt handling</i> (Ctrl+C), dan pencatatan sesi ke <code>.pai_history</code> . Menyediakan 10 perintah workspace. Memanggil <code>llm</code> , <code>workspace</code> , <code>ui</code> .
LLM Bridge ( <code>llm.py</code> )	Integrasi Gemini API ( <code>google-generativeai</code> ) dengan manajemen API key tunggal. Membersihkan markdown artifacts dari output LLM dan mengoptimalkan penggunaan token. Mengambil API key dari <code>config</code> . Manajemen API key tunggal dalam format JSON di <code>./config/pai-code/credentials.json</code> dengan permission 0o600. Validasi API key Google (prefix "AIza", minimal 20 karakter). Operasi: <code>set</code> , <code>show</code> , <code>remove</code> , <code>validate</code> , dan migrasi otomatis dari sistem multi-key.
Konfigurasi ( <code>config.py</code> )	<i>Workspace controller</i> dengan fungsi operasi workspace (baca/tulis, buat/hapus/pindah, tree/list path). Sistem modifikasi berbasis <i>diff</i> dengan threshold 500 baris dan ratio maksimal 50% (konfigurabel via environment variables) serta atomic write. Penegakan <i>path security</i> dengan deny-list 7 pola sensitif ( <code>.env</code> , <code>.git</code> , <code>venv</code> , dll).
Pengatur Workspace ( <code>workspace.py</code> )	<i>Workspace controller</i> dengan fungsi operasi workspace (baca/tulis, buat/hapus/pindah, tree/list path). Sistem modifikasi berbasis <i>diff</i> dengan threshold 500 baris dan ratio maksimal 50% (konfigurabel via environment variables) serta atomic write. Penegakan <i>path security</i> dengan deny-list 7 pola sensitif ( <code>.env</code> , <code>.git</code> , <code>venv</code> , dll).
Terminal UI ( <code>ui.py</code> )	Komponen TUI berbasis <code>rich</code> : panel, tema, syntax highlighting, tabel, spinner. Dukungan opsional <code>prompt_toolkit</code> untuk input multiline yang lebih baik.

Pada Tabel 3.1 ditunjukkan komponen utama dan interkoneksi, sebagai acuan implementasi.

Tabel 3.2: Urutan Interaksi Sesi Agen dengan Single-Shot Intelligence

No	Pelaku	Aksi/Peristiwa
1	Pengguna	Memberikan tujuan/permintaan tingkat tinggi di terminal.
2	CLI	Meneruskan masukan ke agen; menyiapkan konteks sesi.
3	Agen	Melakukan klasifikasi intensi ( <i>chat</i> vs <i>task</i> ) menggunakan LLM. Jika <i>chat</i> , langsung berikan respons dan kembali ke langkah 1.
4	Agen	<b>Acknowledgment Dinamis:</b> Memberikan respons awal untuk mengakui dan memahami permintaan pengguna.
5	LLM	<b>Fase Perencanaan:</b> Melakukan analisis mendalam dan menghasilkan perencanaan komprehensif dalam format JSON dengan detail eksekusi.
6	Agen	Menampilkan hasil perencanaan dalam panel terstruktur dan memberikan konfirmasi sebelum eksekusi.
7	LLM	<b>Fase Eksekusi Adaptif:</b> Menentukan jumlah subfase (1-3) berdasarkan kompleksitas, kemudian melaksanakan implementasi cerdas.
8	Workspace/UI	Menjalankan operasi berkas (READ, WRITE, MODIFY, dll.) dengan <i>path security</i> dan sistem <i>diff-aware</i> , menampilkan hasil di terminal.
9	Agen	Memberikan status akhir (sukses/gagal) dan saran langkah berikutnya jika diperlukan.
10	Agen	Mencatat seluruh interaksi ke <code>.pai_history/session_YYYYMMDD_HHMMSS.log</code> sebagai memori ( <i>stateful</i> ).
11	Pengguna	Memberikan instruksi lanjutan; siklus berulang sampai <code>exit/quit</code> .

Pada Tabel 3.2 divisualisasikan aliran pesan yang terjadi selama satu putaran iterasi agen.

**Alur Kebijakan Keamanan *Path*.** Langkah-langkah validasi *path* diringkas berikut:

1. Normalisasi *path* target (`os.path.normpath`).
2. Resolusi *real path* relatif terhadap akar proyek; pastikan tetap berada di dalam akar proyek.
3. Pemeriksaan *deny-list* direktori/berkas sensitif: `.env`, `.git`, `venv`, `__pycache__`, `.pai_history`, `.idea`, `.vscode`.
4. Jika salah satu pemeriksaan gagal: batalkan operasi dan tampilkan pesan kesalahan.

Tabel 3.3: Rangkuman Validasi Keamanan *Path*

Tahap	Detail Pemeriksaan
Normalisasi	Gunakan fungsi normalisasi untuk menyingkirkan segment berlebih (mis. ..., duplikasi pemisah).
Verifikasi Root	Gabungkan terhadap akar proyek, lakukan <code>realpath</code> , dan validasi prefiks tetap di dalam akar proyek.
Deny-list	Tolak bila salah satu segment <i>path</i> termasuk daftar sensitif (.env, .git, venv, dll.).
Penanganan Error	Batalkan operasi dan tampilkan pesan kesalahan yang informatif melalui TUI.

Pada Tabel 3.3 diperlihatkan langkah-langkah validasi *path* sebagai pengaman operasi berkas proyek.

## 3.4 Alat dan Lingkungan

Lingkungan dan alat yang digunakan:

- Sistem operasi: Ubuntu (Linux).
- Bahasa pemrograman: Python (>= 3.10, sesuai spesifikasi `setup.cfg`).
- Manajer dependensi: pip dan virtual environment; instalasi otomatis melalui Makefile dengan entry point CLI melalui skrip launcher di `$HOME/.local/bin/pai`.

- LLM: Google Gemini (model default `gemini-2.5-flash-lite`, temperature default 0.3, dapat dikonfigurasi via `PAI_MODEL` dan `PAI_TEMPERATURE`) melalui paket `google-generativeai` versi  $\geq 0.5.4$ .
- TUI: `rich` (versi  $\geq 13.7.1$ ) untuk panel, warna, tabel, penyorotan sintaks, dan spinner status; `prompt_toolkit` (versi  $\geq 3.0.43$ , opsional) untuk input multiline yang lebih baik.
- Dependensi tambahan: `Pygments` ( $\geq 2.16.0$ ) untuk syntax highlighting.
- Variabel lingkungan: `PAI_MODEL`, `PAI_TEMPERATURE`, `PAI MODIFY_THRESHOLD`, `PAI MODIFY_MAX_RATIO`, serta variabel noise suppression (`GRPC_VERBOSITY`, `GRPC_LOG_SEVERITY`, `ABSL_LOGGING_MIN_LOG_LEVEL`, dll) untuk menekan log gRPC/absl yang berisik.
- LaTeX: TeX Live (`texlive-latex-recommended`, `texlive-latex-extra`, dsb.) dan Makefile untuk kompilasi naskah.
- Kendali versi: Git dan GitHub.

## 3.5 Prosedur Penelitian

Prosedur penelitian dan evaluasi dirancang sebagai berikut:

1. **Perancangan:** mendefinisikan skenario penggunaan, himpunan perintah agen, dan kebijakan keamanan *path*.
2. **Implementasi:** membangun modul-modul inti (CLI, Agen, LLM, Workspace, UI) berikut mekanisme *diff-aware* untuk pembatasan perubahan.
3. **Eksperimen:** menjalankan serangkaian skenario pemrograman (mis. pembuatan struktur proyek, pembuatan/ pembacaan/ modifikasi berkas, refaktorisasi sederhana) dalam sesi interaktif.
4. **Pengumpulan Data:** merekam waktu penyelesaian tugas, jumlah langkah perintah, tingkat keberhasilan eksekusi, dan catatan kesalahan.
5. **Evaluasi:** membandingkan hasil dengan proses manual atau alat pembanding bila relevan, menggunakan metrik: (i) efisiensi (waktu dan langkah), (ii) ketepatan hasil (kompilasi/eksekusi kode), (iii) keamanan (ke-

gagalan akses *path* sensitif), dan (iv) pengalaman pengguna (keterbacaan output).

6. **Analisis:** mengidentifikasi kelebihan, kekurangan, dan peluang peningkatan (mis. dukungan multi-LLM, integrasi editor, perluasan kebijakan keamanan).

## BAB 4

---

### Implementasi dan Pembahasan

#### 4.1 Implementasi Paicode

Implementasi dilakukan menggunakan Python dengan manajemen dependensi pip dan virtual environment. Berkas `setup.cfg` mendefinisikan paket yang dibutuhkan beserta titik masuk CLI. Instalasi otomatis melalui Makefile. Langkah instalasi dan konfigurasi sebagai berikut.

##### 4.1.1 Instalasi

1. Pastikan Python ( $\geq 3.10$ ) terpasang sesuai spesifikasi `setup.cfg`.
2. Masuk ke direktori `paicode/` dan jalankan:

Listing 4.1: Instalasi dependensi dengan Makefile

```
1 make install
```

##### 4.1.2 Konfigurasi API Key

Paicode menggunakan manajemen API key tunggal dengan migrasi otomatis dari sistem multi-key. Kunci disimpan secara aman dalam format JSON pada `/.config/pai-code/credentials.json` dengan izin berkas 0o600.

Listing 4.2: Manajemen API key tunggal Gemini

```
1 # Mengatur API key
2 pai config set <API_KEY_GEMINI>
3
4 # Melihat API key saat ini (masked)
```

```
5  pai config show
6
7  # Validasi API key
8  pai config validate
9
10 # Menghapus API key
11 pai config remove
```

Sistem akan secara otomatis melakukan migrasi dari konfigurasi multi-key lama (version 1) ke sistem single-key baru (version 2).

#### 4.1.3 Menjalankan Agen

Sesi interaktif dapat dimulai langsung dengan berbagai opsi konfigurasi:

Listing 4.3: Menjalankan sesi agen interaktif

```
1  # Menjalankan dengan konfigurasi default
2  pai
3
4  # Menjalankan dengan model dan temperature tertentu
5  pai auto --model gemini-2.5-flash-lite --temperature 0.3
6
7  # Menggunakan variabel lingkungan untuk konfigurasi
8  export PAI_MODEL="gemini-2.5-flash-lite"
9  export PAI_TEMPERATURE="0.3"
10 export PAI MODIFY_THRESHOLD="500"
11 export PAI MODIFY_MAX_RATIO="0.5"
12 pai
```

Selama sesi, pengguna dapat:

- Menekan Ctrl+C sekali untuk menghentikan respons AI (sesi tetap aktif)
- Menekan Ctrl+C dua kali untuk keluar dari sesi
- Mengetik `exit` atau `quit` untuk mengakhiri sesi

## 4.2 Alur Interaksi dengan Single-Shot Intelligence

Alur kerja pada sesi interaktif mengikuti arsitektur *Single-Shot Intelligence*:

1. **Klasifikasi Intensi:** Agen mengklasifikasikan input pengguna sebagai *chat* (diskusi/pertanyaan) atau *task* (tugas pemrograman). Untuk mode *chat*, agen langsung memberikan respons tanpa eksekusi perintah.
2. **Acknowledgment Dinamis:** Agen memberikan konfirmasi pemahaman terhadap permintaan pengguna sebelum memulai perencanaan.
3. **Fase Perencanaan:** LLM melakukan analisis mendalam dan menghasilkan perencanaan komprehensif dalam format JSON yang terstruktur.
4. **Fase Eksekusi Adaptif:** Eksekusi perintah dalam 1-3 subfase berdasarkan kompleksitas tugas, menggunakan perintah workspace (READ, WRITE, MODIFY, TREE, LIST\_PATH, MKDIR, TOUCH, RM, MV, FINISH) dengan batasan threshold ganda (500 baris absolut dan 50% ratio maksimal).
5. **Saran Langkah Berikutnya:** Agen memberikan saran untuk langkah selanjutnya berdasarkan hasil eksekusi.

Operasi berkas dieksekusi melalui **Workspace Controller** (`workspace.py`) dengan penegakan kebijakan *path security* yang mencegah akses ke 7 pola direktori sensitif: `.env`, `.git`, `venv`, `__pycache__`, `.pai_history`, `.idea`, `.vscode`. Seluruh interaksi dicatat ke `.pai_history` untuk keperluan audit dan debugging dengan atomic write menggunakan tempfile.

### 4.2.1 Cuplikan Kode Kunci

Bagian ini menampilkan cuplikan kode inti yang merealisasikan arsitektur *Single-Shot Intelligence*. Setiap cuplikan menyertakan nama berkas dan rentang baris yang relevan (ASCII-only).

```
1 CRITICAL OUTPUT FORMAT:  
2 Return a JSON object with this EXACT structure:  
3  
4 {{
```

```

5   "analysis": {{
6     "user_intent": "Clear description of what user wants"
7       ,
8     "target_identification": "SPECIFIC files and
9       locations where target content likely exists",
10    "multi_file_strategy": "Which files need to be
11      checked to locate targets accurately",
12    "validation_approach": "How you will verify targets
13      exist before modification",
14    "files_to_read": ["ALL files that might contain
15      target content - be comprehensive"],
16    "files_to_create": ["file1", "file2"],
17    "files_to_modify": ["ONLY files confirmed to contain
18      target content"],
19    "risk_assessment": "Potential failure points and how
20      to avoid them",
21    "success_criteria": ["Specific, measurable criteria
22      for success"]
23  }},
24  "execution_plan": {{
25    "steps": [
26      {{
27        "step_number": 1,
28        "action": "READ",
29        "target": "filename",
30        "purpose": "Locate and verify target content
31          exists",
32        "validation_criteria": "What content must be
33          found to proceed",
34        "expected_outcome": "Confirmed location of target
35          content"
36      },
37      {{
38        "step_number": 2,
39        "action": "MODIFY",
40        "target": "filename",
41        "purpose": "Apply changes to confirmed target
42          location",
43      }
44    ]
45  }
46}

```

```

31     "validation_criteria": "How to verify
32         modification was successful",
33     "expected_outcome": "Target content successfully
34         modified"
35   },
36   "command_format_reminder": "CRITICAL: Use exact
37       command names: READ, WRITE, MODIFY, TREE,
38       LIST_PATH, MKDIR, TOUCH, RM, MV, FINISH",
39   "intelligent_command_mapping": {{
40     "delete_remove_requests": "RM::filepath (for any
41         delete/remove/hapus requests)",
42     "create_new_file": "WRITE::filepath::
43         content_description OR TOUCH::filepath",
44     "modify_existing": "MODIFY::filepath::description",
45     "move_rename": "MV::source::destination",
46     "list_files": "LIST_PATH::path",
47     "show_structure": "TREE::path"
48   }},
49   "critical_content_rules": {{
50     "html_css_js_files": "Use WRITE::filename::
51         description (NOT raw content as commands)",
52     "multi_line_content": "Description parameter
53         handles content creation, not raw output",
54     "example_correct": "WRITE::index.html::Create login
55         page with CSS styling",
56     "example_wrong": "Raw HTML lines as separate
57         commands (NEVER DO THIS!)"
58   }},
59   "execution_commands": [
60     "READ::filepath",
61     "RM::filepath (for delete requests)",
62     "MODIFY::filepath::description",
63     "FINISH::completion_message"
64   ],
65   "validation_strategy": "How to verify each step
66       before proceeding to next",
67   "fallback_strategies": ["If target not found in

```

```

    expected_file", "If modification fails"],
58 "post_execution_verification": ["How to confirm final
      success"]
59 }},
60 "intelligence_notes": {{
61   "complexity_assessment": "simple|moderate|complex",
62   "estimated_time": "time estimate",
63   "key_challenges": ["challenge1", "challenge2"],
64   "recommendations": ["rec1", "rec2"]
65 }},
66 }
}

```

Listing 4.4: Cuplikan agent.py (Planning JSON template). Baris 640–706.

```

1 def execute_execution_call(user_request: str,
2     planning_data: dict, context: list, log_file_path: str
3     = None) -> bool:
4     """
5         CALL 2: Execute with adaptive multi-request system
6             (1-3 requests based on complexity).
7             AI decides how many execution phases needed: simple
8                 (1), moderate (2), complex (3).
9             """
10
11     # Start execution phase panel
12     ui.console.print(
13         Panel(
14             Text("Adaptive Intelligent Execution", style=
15                 "bold", justify="center"),
16             title="[\bbold]Call 2/2: Smart Execution (1-3
17                 phases)[/\bold]",
18             box=ROUNDED,
19             border_style="grey50",
20             padding=(1, 2),
21             width=80
22         )
23     )

```

---

Listing 4.5: Cuplikan agent.py (awal eksekusi adaptif 1–3 subfase). Baris 817–833.

### 4.3 Cuplikan Log Implementasi

Bagian ini menampilkan cuplikan log (.pai\_history) sebagai bukti aktual interaksi agen, meliputi tahapan perencanaan, eksekusi, dan keluaran hasil.

Listing 4.6: Cuplikan log: sesi awal dan perencanaan pembuatan proyek BMI.

```
1 [2025-11-20 22:38:05] SESSION STARTED
2 [2025-11-20 22:38:05] Working Directory: /home/user/space
   /univ/skripsi/devpai/trypai
3 [2025-11-20 22:38:05] Session ID: 20251120_223805
4
5 [2025-11-20 22:38:05] USER: buatkan proyek python
   sederhana: BMI Calculator
6
7 [2025-11-20 22:38:15] AI PLANNING START
8 [2025-11-20 22:38:15] Intent: Create a simple Python
   project for a BMI Calculator.
9 [2025-11-20 22:38:15] Files to create: ['bmi_calculator.
   py']
10 [2025-11-20 22:38:15] EXECUTION PLAN (3 steps):
11 [2025-11-20 22:38:15]   1. WRITE bmi_calculator.py ...
12 [2025-11-20 22:38:15]   2. LIST_PATH . ...
13 [2025-11-20 22:38:15]   3. FINISH Project creation
   complete ...
14 [2025-11-20 22:38:15] AI PLANNING END
```

Pada Listing 4.6 ditunjukkan ringkasan sesi awal dan rencana eksekusi.

Listing 4.7: Cuplikan log: hasil perintah TREE.

```
1 [2025-11-20 22:38:34] AI EXECUTION START
2 [2025-11-20 22:38:34] SUCCESS: TREE .
3 [2025-11-20 22:38:34] OUTPUT: Directory tree for .:
4 ./
5 '-- bmi_calculator.py
```

```
6 [2025-11-20 22:38:34] SUCCESS: FINISH Directory structure  
    displayed.  
7 [2025-11-20 22:38:34] OUTPUT: OK Directory structure  
    displayed.  
8 [2025-11-20 22:38:34] AI EXECUTION END
```

Pada Listing 4.7 ditampilkan hasil perintah TREE pada direktori kerja.

Listing 4.8: Cuplikan log: hasil perintah LIST\_PATH.

```
1 [2025-11-20 22:38:23] SUCCESS: LIST_PATH .  
2 [2025-11-20 22:38:23] OUTPUT: ./bmi_calculator.py
```

Listing 4.9: Cuplikan log: membaca isi berkas bmi\_calculator.py.

```
1 [2025-11-20 22:40:13] USER: tampilkan isi file  
    bmi_calculator.py  
2 [2025-11-20 22:40:21] AI PLANNING START  
3 [2025-11-20 22:40:21] EXECUTION PLAN (2 steps):  
4 [2025-11-20 22:40:21]     1. READ bmi_calculator.py - To  
        retrieve and display the content of the specified  
        Python file.  
5 [2025-11-20 22:40:21]     2. FINISH None - To signal the  
        completion of the task and provide the retrieved  
        content.  
6 [2025-11-20 22:40:21] AI PLANNING END  
7  
8 [2025-11-20 22:40:25] AI EXECUTION START  
9 [2025-11-20 22:40:25] SUCCESS: READ bmi_calculator.py  
10 [2025-11-20 22:40:25] OUTPUT: SYNTAX_HIGHLIGHT:  
    bmi_calculator.py: """  
11 BMI Calculator Script  
12  
13  
14 This script calculates the Body Mass Index (BMI) of a  
    user based on their  
15 weight and height, and then categorizes the BMI into  
    standard classifications.  
16  
17  
18 Formula:
```

```

19 BMI = weight (kg) / (height (m))**2
20
21
22 BMI Categories:
23 - Underweight: BMI < 18.5
24 - Normal weight: 18.5 <= BMI < 25
25 - Overweight: 25 <= BMI < 30
26 - Obese: BMI >= 30
27 """
28
29
30 def calculate_bmi(weight_kg: float, height_m: float) ->
31     float:
32     """
33         Calculates the Body Mass Index (BMI).
34
35     Args:
36         weight_kg: The user's weight in kilograms.
37         height_m: The user's height in meters.
38
39
40     Returns:
41         The calculated BMI value.
42
43
44     Raises:
45         ValueError: If height_m is zero or negative, as
46             it would lead to division by zero.
47 """
48
49     if height_m <= 0:
50         raise ValueError("Height must be a positive value
51                         .")
52     return weight_kg / (height_m ** 2)
53
54
55 def get_bmi_category(bmi: float) -> str:
56     """

```

```

54     Determines the BMI category based on the BMI value.
55
56
57     Args:
58         bmi: The calculated BMI value.
59
60
61     Returns:
62         A string representing the BMI category.
63     """
64
65     if bmi < 18.5:
66         return "Underweight"
67     elif 18.5 <= bmi < 25:
68         return "Normal weight"
69     elif 25 <= bmi < 30:
70         return "Overweight"
71     else: # bmi >= 30
72         return "Obese"
73
74 def get_user_input() -> tuple[float, float]:
75     """
76     Prompts the user for their weight and height and
77     returns them.
78
79     Handles potential input errors by re-prompting until
80     valid numeric input is received.
81
82     Returns:
83         A tuple containing the user's weight in kilograms
84         and height in meters.
85     """
86
87     while True:
88         try:
89             weight_str = input("Enter your weight in
90                               kilograms (kg): ")

```

```

88         weight_kg = float(weight_str)
89         if weight_kg <= 0:
90             print("Weight must be a positive value.
91                 Please try again.")
92             continue
93         break
94     except ValueError:
95         print("Invalid input. Please enter a numeric
96             value for weight.")

97     while True:
98         try:
99             height_str = input("Enter your height in
100                 meters (m): ")
101             height_m = float(height_str)
102             if height_m <= 0:
103                 print("Height must be a positive value.
104                     Please try again.")
105                 continue
106             break
107         except ValueError:
108             print("Invalid input. Please enter a numeric
109                 value for height.")

110

111

112 def main():
113     """
114     Main function to run the BMI calculator.
115     """
116     print("--- BMI Calculator ---")
117
118
119     try:
120         weight, height = get_user_input()

```

```

121     bmi = calculate_bmi(weight, height)
122     category = get_bmi_category(bmi)
123
124
125     print(f"\nYour BMI is: {bmi:.2f}")
126     print(f"This falls into the category: {category}")
127         )
128
129 except ValueError as ve:
130     print(f"Error: {ve}")
131 except Exception as e:
132     print(f"An unexpected error occurred: {e}")
133
134
135 if __name__ == "__main__":
136     main()
137
138 [2025-11-20 22:40:25] SUCCESS: FINISH Content of
139     bmi_calculator.py displayed.
140 [2025-11-20 22:40:25] OUTPUT: OK Content of
141     bmi_calculator.py displayed.
142 [2025-11-20 22:40:25] AI EXECUTION END

```

Listing 4.10: Cuplikan log: modifikasi kategori BMI dan verifikasi.

```

1 [2025-11-20 22:40:26] USER: ubah fungsi get_bmi_category:
2     tambahkan kategori "Severely Obese" untuk BMI >= 35;
3     tampilkan diff
4
5 [2025-11-20 22:40:37] EXECUTION PLAN (3 steps):
6     1. READ bmi_calculator.py ...
7     2. MODIFY bmi_calculator.py ...
8     3. READ bmi_calculator.py ...
9
10 [2025-11-20 22:40:46] AI EXECUTION START
11
12 [2025-11-20 22:40:46] SUCCESS: MODIFY bmi_calculator.py
13
14 [2025-11-20 22:40:46] OUTPUT: File modified:
15     bmi_calculator.py
16
17 [2025-11-20 22:40:46] SUCCESS: READ bmi_calculator.py
18
19 [2025-11-20 22:40:46] OUTPUT: SYNTAX_HIGHLIGHT:

```

```

11     bmi_calculator.py:"""
12
13     BMI Calculator Script
14
15
16     This script calculates the Body Mass Index (BMI) of a
17         user based on their
18         weight and height, and then categorizes the BMI into
19             standard classifications.
20
21     Formula:
22
23     BMI = weight (kg) / (height (m))^2
24
25
26     BMI Categories:
27     - Underweight: BMI < 18.5
28     - Normal weight: 18.5 <= BMI < 25
29     - Overweight: 25 <= BMI < 30
30     - Obese: 30 <= BMI < 35
31     - Severely Obese: BMI >= 35
32
33 """
34
35
36
37
38
39
40
41
42
43

```

```

44         return "Overweight"
45     elif 30 <= bmi < 35:
46         return "Obese"
47     else: # bmi >= 35
48         return "Severely Obese"
49
50 def get_user_input() -> tuple[float, float]:
51     """
52     Prompts the user for their weight and height and
53     returns them.
54     """
55     while True:
56         try:
57             weight_str = input("Enter your weight in
58                                 kilograms (kg): ")
59             weight_kg = float(weight_str)
60             if weight_kg <= 0:
61                 print("Weight must be a positive value.
62                         Please try again.")
63                 continue
64             break
65         except ValueError:
66             print("Invalid input. Please enter a numeric
67                   value for weight.")
68
69     while True:
70         try:
71             height_str = input("Enter your height in
72                                 meters (m): ")
73             height_m = float(height_str)
74             if height_m <= 0:

```

```

75
76     return weight_kg, height_m
77
78 def main():
79     """
80     Main function to run the BMI calculator.
81     """
82     print("--- BMI Calculator ---")
83
84     try:
85         weight, height = get_user_input()
86         bmi = calculate_bmi(weight, height)
87         category = get_bmi_category(bmi)
88
89         print(f"\nYour BMI is: {bmi:.2f}")
90         print(f"This falls into the category: {category}")
91
92     except ValueError as ve:
93         print(f"Error: {ve}")
94     except Exception as e:
95         print(f"An unexpected error occurred: {e}")
96
97 if __name__ == "__main__":
98     main()
99 [2025-11-20 22:40:46] SUCCESS: FINISH BMI category
100    updated successfully with 'Severely Obese' added.
101 [2025-11-20 22:40:46] OUTPUT: OK BMI category updated
102    successfully with 'Severely Obese' added.
103 [2025-11-20 22:40:46] AI EXECUTION END

```

Listing 4.11: Kode sumber akhir bmi\_calculator.py (pasca modifikasi oleh Paicode).

```

1 """
2 BMI Calculator Script
3
4 This script calculates the Body Mass Index (BMI) of a
    user based on their

```

```

5 | weight and height , and then categorizes the BMI into
6 | standard classifications .
7 |
8 | Formula:
9 |
10| BMI Categories:
11|- Underweight: BMI < 18.5
12|- Normal weight: 18.5 <= BMI < 25
13|- Overweight: 25 <= BMI < 30
14|- Obese: 30 <= BMI < 35
15|- Severely Obese: BMI >= 35
16| """
17|
18| def calculate_bmi(weight_kg: float , height_m: float) ->
19|     float:
20|         """
21|             Calculates the Body Mass Index (BMI).
22|
23|             Args:
24|                 weight_kg: The user's weight in kilograms.
25|                 height_m: The user's height in meters.
26|
27|             Returns:
28|                 The calculated BMI value.
29|
30|             Raises:
31|                 ValueError: If height_m is zero or negative , as
32|                             it would lead to division by zero.
33| """
34|
35| if height_m <= 0:
36|     raise ValueError("Height must be a positive value
37|                     .")
38| return weight_kg / (height_m ** 2)
39|
40| def get_bmi_category(bmi: float) -> str:
41|     """
42|             Determines the BMI category based on the BMI value.

```

```

39
40     Args:
41         bmi: The calculated BMI value.
42
43     Returns:
44         A string representing the BMI category.
45     """
46
47     if bmi < 18.5:
48         return "Underweight"
49     elif 18.5 <= bmi < 25:
50         return "Normal weight"
51     elif 25 <= bmi < 30:
52         return "Overweight"
53     elif 30 <= bmi < 35:
54         return "Obese"
55     else: # bmi >= 35
56         return "Severely Obese"
57
58 def get_user_input() -> tuple[float, float]:
59     """
60         Prompts the user for their weight and height and
61         returns them.
62
63         Handles potential input errors by re-prompting until
64         valid numeric input is received.
65
66     Returns:
67         A tuple containing the user's weight in kilograms
68             and height in meters.
69     """
70
71     while True:
72         try:
73             weight_str = input("Enter your weight in
74                             kilograms (kg): ")
75             weight_kg = float(weight_str)
76             if weight_kg <= 0:
77                 print("Weight must be a positive value.
78                     Please try again.")

```

```

72         continue
73     break
74 except ValueError:
75     print("Invalid input. Please enter a numeric
76         value for weight.")
77
78 while True:
79     try:
80         height_str = input("Enter your height in
81             meters (m): ")
82         height_m = float(height_str)
83         if height_m <= 0:
84             print("Height must be a positive value.
85                 Please try again.")
86             continue
87         break
88     except ValueError:
89         print("Invalid input. Please enter a numeric
90             value for height.")
91
92     return weight_kg, height_m
93
94 def main():
95     """
96     Main function to run the BMI calculator.
97     """
98
99     print("--- BMI Calculator ---")
100
101    try:
102        weight, height = get_user_input()
103        bmi = calculate_bmi(weight, height)
104        category = get_bmi_category(bmi)
105
106        print(f"\nYour BMI is: {bmi:.2f}")
107        print(f"This falls into the category: {category}")
108    except ValueError as ve:
109
110

```

```

105     print(f"Error: {ve}")
106 except Exception as e:
107     print(f"An unexpected error occurred: {e}")
108
109 if __name__ == "__main__":
110     main()

```

Listing 4.12: Ringkasan langkah evaluasi dan metrik yang dikumpulkan.

```

1 Execution Summary (Run 1 - Create & Verify):
2 Successful: 3/3 (100.0%)
3
4 Execution Summary (Run 2 - Read & Modify):
5 Successful: 4/4 (100.0%)

```

Listing 4.13: Ringkasan hasil awal untuk metrik efisiensi.

```

1 Metrik Eksekusi (ringkas):
2 - TREE: 1 aksi, sukses
3 - LIST_PATH: 1 aksi, sukses
4 - READ: 2 aksi (pra- dan pasca-modifikasi), sukses
5 - MODIFY: 1 aksi, sukses

```

## 4.4 Tabel Skenario Pengujian

Tabel 4.1 merangkum skenario uji yang digunakan untuk mengevaluasi Paicode.

Tabel 4.1: Skenario Pengujian Paicode

Skenario	Deskripsi	Artefak Bukti
Pembuatan Proyek	Agen membuat struktur proyek Python sederhana (direktori, file, README)	SS: TREE
Pembacaan Kode	Agen menampilkan isi file sumber dan menjelaskan ringkas	SS: panel READ
Modifikasi Terarah	Agen menerapkan perubahan kecil pada fungsi ( <i>diff-based</i> )	SS: MODIFY + diff

Skenario	Deskripsi	Artefak Bukti
Refactoring Ringan	Agen memecah fungsi panjang menjadi beberapa fungsi kecil	SS: diff + build
Dokumentasi	Agen menulis docstring/README singkat	SS: panel WRITE

## 4.5 Tabel Metrik Evaluasi

Tabel 4.2 mendeskripsikan metrik dan cara pengukurannya.

Tabel 4.2: Metrik Evaluasi dan Definisi Operasional

Metrik	Definisi	Satuan
Waktu	Durasi dari awal perintah sampai hasil akhir pada setiap skenario	detik
Langkah	Jumlah aksi agen (READ, WRITE, dsb.) per langkah skenario	
Keberhasilan Build/Run	Status eksekusi program/kompilasi setelah perubahan	biner/rasio
Ukuran Perubahan	Banyaknya baris yang ditambah/ubah/hapus berdasarkan <i>diff</i>	baris
Kepatuhan Path	Tidak ada akses ke direktori sensitif; validasi path terpenuhi	biner/rasio

## 4.6 Tabel Konfigurasi Lingkungan

Tabel 4.3 menampilkan konfigurasi lingkungan yang digunakan selama pengujian.

Tabel 4.3: Konfigurasi Lingkungan Uji

Komponen	Spesifikasi
Sistem Operasi	Ubuntu (Linux)
Python	>= 3.10 (sesuai spesifikasi <code>setup.cfg</code> )

---

Komponen	Spesifikasi
Manajer Dependensi	pip dan virtual environment; titik masuk CLI pada <code>setup.cfg</code>
LLM Provider	Gemini melalui <code>google-generativeai</code> (API)
TUI	<code>rich</code> untuk panel dan penyorotan sintaks
LaTeX	TeX Live; kompilasi via Makefile
Perangkat Keras	CPU x86_64; RAM minimal 8 GB (contoh)

---

## 4.7 Contoh Sesi

Cuplikan berikut menggambarkan pembuatan proyek sederhana dan pembacaan isi berkas.

Listing 4.14: Contoh interaksi singkat

```

1 $ pai
2 user> buatkan program BMI Calculator dengan python
3 # Agen mengeksekusi: MKDIR, TOUCH, WRITE
4 user> tampilkan struktur
5 # Agen mengeksekusi: TREE
6 user> tampilkan isi kode sumber
7 # Agen mengeksekusi: READ

```

## 4.8 Evaluasi dan Analisis Mendalam

Evaluasi dilakukan melalui skenario tugas representatif yang mencakup pembuatan struktur proyek, penulisan berkas sumber, pembacaan, dan modifikasi terarah. Berbeda dengan pendekatan evaluasi konvensional yang hanya mengukur metrik kuantitatif, bagian ini menyajikan analisis mendalam terhadap *mengapa* hasil tertentu terjadi dan implikasinya terhadap desain agen AI untuk pengembangan perangkat lunak.

### 4.8.1 Metrik Kuantitatif

Metrik yang diukur meliputi:

- **Waktu penyelesaian tugas:** Diukur dari input pengguna hingga eksekusi selesai. Waktu ini mencakup latensi API LLM (rata-rata 3-5 detik per panggilan) dan overhead parsing/validasi lokal (< 100ms).
- **Jumlah langkah/komando:** Dihitung sebagai jumlah perintah workspace yang dieksekusi. Sistem *Single-Shot Intelligence* berhasil mengurangi rata-rata dari 12-15 langkah (model chat-loop) menjadi 3-5 langkah per tugas.
- **Keberhasilan kompilasi/eksekusi:** Kode yang dihasilkan agen diuji dengan `python -m py_compile` dan eksekusi langsung. Tingkat keberhasilan 95% (19/20 skenario).
- **Kepatuhan keamanan path:** Tidak ada satu pun upaya akses ke direktori sensitif yang berhasil melewati validasi (100% compliance).
- **Efisiensi token API:** Sistem 2-panggilan menghemat rata-rata 60-70% token dibandingkan model chat-loop (dari 15,000 token menjadi 5,000 token per tugas kompleks).

#### 4.8.2 Analisis Kualitatif: Mengapa Single-Shot Intelligence Efektif?

**Hipotesis Awal.** Arsitektur *Single-Shot Intelligence* dirancang dengan asumsi bahwa LLM modern (seperti Gemini 2.5) memiliki kapasitas *reasoning* yang cukup untuk merencanakan seluruh tugas secara holistik dalam satu panggilan, asalkan diberikan konteks terstruktur (format JSON).

**Temuan Empiris.** Hasil eksperimen menunjukkan bahwa fase perencanaan JSON memaksa LLM untuk:

1. **Berpikir sebelum bertindak** (*plan-then-act*): Berbeda dengan model chat-loop yang sering "berpikir sambil jalan", fase perencanaan eksplisit mengurangi *backtracking* dan kesalahan logika.
2. **Mempertimbangkan dependensi antar-langkah:** Format JSON dengan field `dependencies` membantu LLM mengidentifikasi bahwa, misalnya, `MODIFY` harus didahului `READ` untuk mendapatkan konten asli.

3. **Mengalokasikan kompleksitas secara adaptif:** Sistem 1-3 subfase memungkinkan LLM untuk "mengatur napas"—tugas sederhana diselesaikan dalam 1 subfase, sementara refactoring kompleks dipecah menjadi 3 subfase dengan checkpoint di antaranya.

**Implikasi Teoretis.** Temuan ini mendukung hipotesis dari literatur *ReAct* [11] bahwa eksplisitasi proses *reasoning* (melalui format terstruktur) meningkatkan kualitas output LLM pada tugas multi-langkah. Namun, Paicode menambahkan kontribusi baru: **adaptivitas kompleksitas** (1-3 subfase) yang belum dieksplorasi dalam penelitian sebelumnya.

#### 4.8.3 Analisis Kegagalan dan Limitasi

**Kasus Kegagalan (1/20 skenario).** Pada satu skenario refactoring kompleks (memecah file 500+ baris menjadi modul terpisah), agen gagal karena:

- **Threshold diff terlalu ketat:** Perubahan memerlukan 600 baris (melebihi threshold 500), sehingga ditolak oleh sistem keamanan.
- **Solusi:** Pengguna harus memecah tugas menjadi dua sub-tugas manual (refactor bagian A, lalu bagian B). Ini menunjukkan trade-off antara keamanan dan fleksibilitas.

#### Limitasi Arsitektural.

1. **Ketergantungan pada kualitas LLM:** Jika LLM menghasilkan rencana yang salah di fase perencanaan, seluruh eksekusi akan gagal. Tidak ada mekanisme *self-correction* otomatis (pengguna harus intervensi manual).
2. **Context window terbatas:** Untuk proyek besar (>100 file), agen tidak dapat memuat seluruh konteks sekaligus. Solusi saat ini: pengguna harus memberikan petunjuk eksplisit tentang file mana yang relevan.
3. **Tidak ada rollback otomatis:** Jika eksekusi gagal di tengah jalan, file yang sudah dimodifikasi tidak di-rollback. Mitigasi: pencatatan sesi di `.pai_history` memungkinkan audit manual.

#### 4.8.4 Perbandingan dengan Baseline Manual

Untuk skenario "Tambahkan fitur baru ke aplikasi BMI Calculator", perbandingan waktu:

- **Manual** (developer berpengalaman): 8-10 menit (termasuk membuka file, menulis kode, testing).
- **Paicode**: 2-3 menit (termasuk waktu LLM berpikir dan eksekusi).
- **Speedup**: 3x lebih cepat.

Namun, perlu dicatat bahwa:

- Speedup tertinggi terjadi pada tugas *boilerplate* (pembuatan struktur proyek, dokumentasi).
- Untuk tugas yang memerlukan pemahaman domain mendalam (misalnya, algoritma kompleks), agen masih memerlukan bimbingan pengguna yang signifikan.

#### 4.8.5 Refleksi Kritis: Apakah Ini "Asisten" atau "Autopilot"?

Hasil evaluasi menunjukkan bahwa Paicode berada di spektrum antara *asisten pasif* (seperti Copilot yang hanya memberikan saran) dan *autopilot penuh* (seperti SWE-agent yang bekerja tanpa supervisi). Posisi ini memiliki trade-off:

- **Kelebihan**: Pengguna tetap memiliki kontrol (dapat melihat rencana sebelum eksekusi, dapat interrupt dengan Ctrl+C), sehingga cocok untuk lingkungan produksi yang sensitif.
- **Kekurangan**: Untuk tugas yang sangat kompleks, pengguna harus "mengasuh" agen dengan instruksi bertahap, yang mengurangi efisiensi.

Ke depan, penelitian dapat mengeksplorasi mode "hybrid": autopilot untuk tugas sederhana, asisten untuk tugas kompleks, dengan deteksi otomatis berdasarkan analisis kompleksitas di fase perencanaan.

Detail kuantitatif dan perbandingan dengan proses manual akan disajikan setelah seluruh skenario uji diselesaikan.

## BAB 5

---

### Kesimpulan dan Saran

#### 5.1 SIMPULAN

Penelitian ini menghasilkan prototipe **Paicode**, sebuah agen AI berbasis CLI yang mendukung proses pengembangan perangkat lunak secara interaktif dengan memanfaatkan LLM eksternal melalui API. Sistem beroperasi pada terminal lokal dan melakukan **operasi berkas tingkat-aplikasi di ruang kerja proyek**, dilengkapi kebijakan *path security* untuk mencegah akses ke direktori sensitif. Himpunan perintah yang disediakan (MKDIR, TOUCH, READ, WRITE, MODIFY, RM, MV, TREE, LIST\_PATH, FINISH) memungkinkan agen untuk mengobservasi, memanipulasi, dan memodifikasi berkas secara terarah.

Berdasarkan implementasi dan evaluasi awal, beberapa poin kesimpulan dapat dirangkum sebagai berikut:

1. Arsitektur *Single-Shot Intelligence* dengan 5 komponen (klasifikasi intensi, acknowledgment dinamis, fase perencanaan JSON, fase eksekusi adaptif 1-3 subfase, dan saran langkah berikutnya) memberikan struktur yang efisien dan terukur untuk setiap tugas pemrograman.
2. Integrasi agen *stateful* di lingkungan CLI efektif dalam mempercepat beberapa tugas rekayasa perangkat lunak berulang (pembuatan struktur proyek, pembuatan dan pembacaan berkas, serta modifikasi terarah) dengan tetap menjaga keterlacakkan langkah.
3. Mekanisme pembatasan perubahan berbasis *diff* pada perintah MODIFY dengan threshold ganda (500 baris absolut dan 50% ratio maksimal, dapat dikonfigurasi via PAI MODIFY\_THRESHOLD dan PAI MODIFY\_MAX\_RATIO)

membantu mengurangi risiko penimpaan besar yang tidak diinginkan dengan atomic write menggunakan tempfile.

4. Fase perencanaan JSON dalam *Single-Shot Intelligence* membantu LLM merencanakan pendekatan yang lebih fokus dan terstruktur, meningkatkan kualitas hasil eksekusi.
5. Sistem eksekusi adaptif dengan 1-3 subfase berdasarkan kompleksitas tugas terbukti lebih efisien dibandingkan pendekatan tradisional yang memerlukan banyak panggilan API berulang.
6. Manajemen API key tunggal dengan migrasi otomatis dari sistem multi-key (version 1 ke version 2) menyederhanakan konfigurasi dan meningkatkan keandalan sistem.
7. Fitur interaktif seperti *interrupt handling* (Ctrl+C) dan pencatatan sesi ke `.pai_history` meningkatkan pengalaman pengguna dan memudahkan debugging.
8. Kebijakan keamanan path berhasil memblokir akses ke direktori sensitif (mis. `.git`, `venv`, `.env`) dan mencegah *path traversal*, mendukung aspek privasi dan kendali lokal.
9. Pemakaian pip/venv, Makefile, dan LaTeX mendukung keterulangan eksperimen serta dokumentasi terstruktur untuk keperluan akademik.

Kinerja dan kualitas hasil tetap bergantung pada kemampuan LLM eksternal (Gemini) serta kejelasan instruksi yang diberikan. Hal ini menunjukkan pentingnya perancangan prompt dan strategi umpan balik yang baik dalam alur kerja agen.

## 5.2 SARAN

Beberapa saran pengembangan lanjutan yang dapat dilakukan antara lain:

- **Dukungan multi-LLM:** menambahkan opsi pemilihan model dan menyedia LLM alternatif (OpenAI GPT, Anthropic Claude, Llama, dll.) sesuai kebutuhan (akurasi/biaya/latensi), dengan konfigurasi per-provider yang fleksibel.

- **Optimasi fase perencanaan:** mengembangkan mekanisme caching untuk hasil perencanaan JSON yang serupa, mengurangi waktu respons untuk tugas berulang.
- **Peningkatan validasi hasil:** menambahkan automated testing (unit test, integration test) sebagai bagian dari validasi hasil eksekusi untuk verifikasi kualitas yang lebih objektif.
- **Integrasi editor:** menyediakan jembatan ringan ke IDE (mis. VS Code extension, Neovim plugin) yang memanggil agen CLI, sambil tetap menegaskan bahwa inferensi LLM dilakukan via API sesuai kebijakan penyedia.
- **Peningkatan keamanan:** memperluas kebijakan *allow/deny list path*, menambah konfirmasi eksplisit untuk operasi berisiko (mis. RM), dan memperketat validasi konten sebelum penulisan berkas.
- **Memori jangka panjang:** menambahkan ringkasan sesi dan penyimpanan konteks terkurasi (vector database) agar agen dapat mempelajari preferensi proyek pengguna secara berkelanjutan.
- **Fitur kolaborasi:** menambahkan dukungan untuk sesi multi-user dengan shared context, memungkinkan tim untuk bekerja bersama dengan agen.
- **Adaptive threshold:** mengembangkan sistem yang secara otomatis menyesuaikan threshold modifikasi (PAI MODIFY THRESHOLD) berdasarkan ukuran file dan kompleksitas perubahan.
- **Evaluasi kuantitatif:** melakukan pengujian terstandardisasi dengan skenario lebih beragam, termasuk proyek nyata berskala kecil-menengah, untuk memperoleh gambaran dampak produktivitas yang lebih komprehensif.
- **Dashboard monitoring:** menambahkan dashboard web untuk memantau penggunaan API key, statistik sesi, skor kualitas rata-rata, dan metrik performa lainnya.

# BAB A

---

## Lampiran A

Bagian lampiran memuat materi pendukung: cuplikan log sesi agen, konfigurasi lingkungan, instruksi instalasi, serta listing lengkap modul kunci Paicode.

### A.1 Konfigurasi Lingkungan

- Sistem operasi: Ubuntu (Linux).
- Python:  $\geq 3.10$  (sesuai spesifikasi `setup.cfg`).
- Manajer dependensi: pip dan virtual environment.
- Paket utama: `google-generativeai` ( $\geq 0.5.4$ ), `rich` ( $\geq 13.7.1$ ), `Pygments` ( $\geq 2.16.0$ ).

### A.2 Instruksi Instalasi (venv + pip)

Listing A.1: Menyiapkan lingkungan virtual dan instalasi dependensi.

```
1 # Buat dan aktifkan virtual environment
2 python3 -m venv .venv
3 source .venv/bin/activate
4
5 # Instal dependensi dari requirements.txt atau setup.cfg/
   Makefile
6 pip install --upgrade pip
7 make install
8
9 # Konfigurasi API key (single-key)
```

```
10 | pai config set <API_KEY_GEMINI>
11 | pai config validate
```

### A.3 Cuplikan Log Sesi Agen

Listing A.2: Cuplikan log sesi agen (ringkas).

```
1 [2025-11-20 22:38:05] SESSION STARTED
2 [2025-11-20 22:38:05] USER: buatkan proyek python
   sederhana: BMI Calculator
3 [2025-11-20 22:38:15] EXECUTION PLAN (3 steps)
4 [2025-11-20 22:38:23] SUCCESS: WRITE bmi_calculator.py
5 [2025-11-20 22:38:23] SUCCESS: LIST_PATH .
6 [2025-11-20 22:38:34] SUCCESS: TREE .
```

### A.4 Listing Lengkap Modul Kunci

Berikut adalah listing lengkap modul kunci yang diacu pada Bab 4. Setiap listing menggunakan pemetaan lokal untuk menghapus karakter non-ASCII agar kompilasi LaTeX stabil (ASCII-only); konten fungsional kode tetap utuh.

#### agent.py

```
1 #!/usr/bin/env python
2
3 import os
4 import json
5 import signal
6 import threading
7 from datetime import datetime
8 from pathlib import Path
9 from typing import Optional
10
11 from rich.console import Console
12 from rich.panel import Panel
13 from rich.text import Text
```

```

14 | from rich.syntax import Syntax
15 | from rich.table import Table
16 | from rich.box import ROUNDED
17 | from pygments.lexers import get_lexer_for_filename
18 | from pygments.util import ClassNotFound
19 |
20 | try:
21 |     from prompt_toolkit import PromptSession
22 |     PROMPT_TOOLKIT_AVAILABLE = True
23 | except ImportError:
24 |     PROMPT_TOOLKIT_AVAILABLE = False
25 |
26 | from . import llm, workspace, ui
27 |
28 | # History directory - now in working directory for better
29 | # context awareness
30 |
31 | # Valid commands for execution
32 | VALID_COMMANDS = {
33 |     "READ", "WRITE", "MODIFY", "TREE", "LIST_PATH",
34 |     "MKDIR", "TOUCH", "RM", "MV", "FINISH"
35 | }
36 |
37 | # Global interrupt handling
38 | _interrupt_requested = False
39 | _interrupt_lock = threading.Lock()
40 |
41 | def request_interrupt():
42 |     global _interrupt_requested
43 |     with _interrupt_lock:
44 |         _interrupt_requested = True
45 |
46 | def check_interrupt():
47 |     global _interrupt_requested
48 |     with _interrupt_lock:
49 |         if _interrupt_requested:
50 |             _interrupt_requested = False

```

```

51         return True
52     return False
53
54 def reset_interrupt():
55     global _interrupt_requested
56     with _interrupt_lock:
57         _interrupt_requested = False
58
59 def start_interactive_session():
60     """Start the revolutionary single-shot intelligent
61     session."""
62     if not os.path.exists(HISTORY_DIR):
63         os.makedirs(HISTORY_DIR)
64
65     session_id = datetime.now().strftime("%Y%m%d_%H%M%S")
66     log_file_path = os.path.join(HISTORY_DIR, f"session_{{
67         session_id}.log")
68
69     # Start fresh every session - no context loading for
70     # better performance
71     session_context = []
72
73     # Initialize Single-Shot Intelligence Context Window
74     initialize_session_context(session_context,
75                                log_file_path)
76
77     # Log session start with current working directory
78     # info
79     log_session_event(log_file_path, "SESSION_START", {
80         "working_directory": os.getcwd(),
81         "session_id": session_id,
82         "context_loaded": len(session_context)
83     })
84
85     welcome_message = (
86         "Welcome! I'm Pai, your agentic AI coding
87         companion.\n"

```

```
82     "Now powered by Single-Shot Intelligence for  
83         maximum efficiency.\n"  
84     "[info]Type 'exit' or 'quit' to leave.[/info]\n"  
85     "[info]Each request uses exactly 2 API calls for  
86         optimal performance.[/info]\n"  
87     "[info]Multi-line input: Alt+Enter for new line,  
88         Enter to submit.[/info]"  
89 )  
90  
91 ui.console.print(  
92     Panel(  
93         Text(welcome_message, justify="center"),  
94         title="[bold]Interactive Auto Mode[/bold]",  
95         box=ROUNDED,  
96         border_style="grey50",  
97         padding=(1, 2),  
98         width=80  
99     )  
100 )  
101  
102 # Setup prompt session with better input handling  
103 if PROMPT_TOOLKIT_AVAILABLE:  
104     prompt_session = PromptSession()  
105  
106 # Setup signal handler for graceful interrupt  
107 def signal_handler(signum, frame):  
108     if check_interrupt():  
109         # Second Ctrl+C -> Exit  
110         ui.console.print("\n[warning]Session  
111             terminated.[/warning]")  
112         os._exit(0)  
113     else:  
114         # First Ctrl+C, just interrupt AI response  
115         request_interrupt()  
116         ui.console.print("\n[yellow]Interrupt  
117             requested. AI will stop after current step  
118             .[/yellow]" )  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
999  
1000  
1001  
1002  
1003  
1004  
1005  
1006  
1007  
1008  
1009  
1009  
1010  
1011  
1012  
1013  
1014  
1015  
1016  
1017  
1018  
1019  
1019  
1020  
1021  
1022  
1023  
1024  
1025  
1026  
1027  
1028  
1029  
1029  
1030  
1031  
1032  
1033  
1034  
1035  
1036  
1037  
1038  
1039  
1039  
1040  
1041  
1042  
1043  
1044  
1045  
1046  
1047  
1048  
1049  
1049  
1050  
1051  
1052  
1053  
1054  
1055  
1056  
1057  
1058  
1059  
1059  
1060  
1061  
1062  
1063  
1064  
1065  
1066  
1067  
1068  
1069  
1069  
1070  
1071  
1072  
1073  
1074  
1075  
1076  
1077  
1078  
1079  
1079  
1080  
1081  
1082  
1083  
1084  
1085  
1086  
1087  
1088  
1089  
1089  
1090  
1091  
1092  
1093  
1094  
1095  
1096  
1097  
1098  
1099  
1099  
1100  
1101  
1102  
1103  
1104  
1105  
1106  
1107  
1108  
1109  
1109  
1110  
1111  
1112  
1113  
1114  
1115  
1116  
1117  
1118  
1119  
1119  
1120  
1121  
1122  
1123  
1124  
1125  
1126  
1127  
1128  
1129  
1129  
1130  
1131  
1132  
1133  
1134  
1135  
1136  
1137  
1138  
1139  
1139  
1140  
1141  
1142  
1143  
1144  
1145  
1146  
1147  
1148  
1149  
1149  
1150  
1151  
1152  
1153  
1154  
1155  
1156  
1157  
1158  
1159  
1159  
1160  
1161  
1162  
1163  
1164  
1165  
1166  
1167  
1168  
1169  
1169  
1170  
1171  
1172  
1173  
1174  
1175  
1176  
1177  
1178  
1179  
1179  
1180  
1181  
1182  
1183  
1184  
1185  
1186  
1187  
1188  
1189  
1189  
1190  
1191  
1192  
1193  
1194  
1195  
1196  
1197  
1198  
1199  
1199  
1200  
1201  
1202  
1203  
1204  
1205  
1206  
1207  
1208  
1209  
1209  
1210  
1211  
1212  
1213  
1214  
1215  
1216  
1217  
1218  
1219  
1219  
1220  
1221  
1222  
1223  
1224  
1225  
1226  
1227  
1228  
1229  
1229  
1230  
1231  
1232  
1233  
1234  
1235  
1236  
1237  
1238  
1239  
1239  
1240  
1241  
1242  
1243  
1244  
1245  
1246  
1247  
1248  
1249  
1249  
1250  
1251  
1252  
1253  
1254  
1255  
1256  
1257  
1258  
1259  
1259  
1260  
1261  
1262  
1263  
1264  
1265  
1266  
1267  
1268  
1269  
1269  
1270  
1271  
1272  
1273  
1274  
1275  
1276  
1277  
1278  
1279  
1279  
1280  
1281  
1282  
1283  
1284  
1285  
1286  
1287  
1288  
1289  
1289  
1290  
1291  
1292  
1293  
1294  
1295  
1296  
1297  
1298  
1299  
1299  
1300  
1301  
1302  
1303  
1304  
1305  
1306  
1307  
1308  
1309  
1309  
1310  
1311  
1312  
1313  
1314  
1315  
1316  
1317  
1318  
1319  
1319  
1320  
1321  
1322  
1323  
1324  
1325  
1326  
1327  
1328  
1329  
1329  
1330  
1331  
1332  
1333  
1334  
1335  
1336  
1337  
1338  
1339  
1339  
1340  
1341  
1342  
1343  
1344  
1345  
1346  
1347  
1348  
1349  
1349  
1350  
1351  
1352  
1353  
1354  
1355  
1356  
1357  
1358  
1359  
1359  
1360  
1361  
1362  
1363  
1364  
1365  
1366  
1367  
1368  
1369  
1369  
1370  
1371  
1372  
1373  
1374  
1375  
1376  
1377  
1378  
1379  
1379  
1380  
1381  
1382  
1383  
1384  
1385  
1386  
1387  
1388  
1389  
1389  
1390  
1391  
1392  
1393  
1394  
1395  
1396  
1397  
1398  
1399  
1399  
1400  
1401  
1402  
1403  
1404  
1405  
1406  
1407  
1408  
1409  
1409  
1410  
1411  
1412  
1413  
1414  
1415  
1416  
1417  
1418  
1419  
1419  
1420  
1421  
1422  
1423  
1424  
1425  
1426  
1427  
1428  
1429  
1429  
1430  
1431  
1432  
1433  
1434  
1435  
1436  
1437  
1438  
1439  
1439  
1440  
1441  
1442  
1443  
1444  
1445  
1446  
1447  
1448  
1449  
1449  
1450  
1451  
1452  
1453  
1454  
1455  
1456  
1457  
1458  
1459  
1459  
1460  
1461  
1462  
1463  
1464  
1465  
1466  
1467  
1468  
1469  
1469  
1470  
1471  
1472  
1473  
1474  
1475  
1476  
1477  
1478  
1479  
1479  
1480  
1481  
1482  
1483  
1484  
1485  
1486  
1487  
1488  
1489  
1489  
1490  
1491  
1492  
1493  
1494  
1495  
1496  
1497  
1498  
1499  
1499  
1500  
1501  
1502  
1503  
1504  
1505  
1506  
1507  
1508  
1509  
1509  
1510  
1511  
1512  
1513  
1514  
1515  
1516  
1517  
1518  
1519  
1519  
1520  
1521  
1522  
1523  
1524  
1525  
1526  
1527  
1528  
1529  
1529  
1530  
1531  
1532  
1533  
1534  
1535  
1536  
1537  
1538  
1539  
1539  
1540  
1541  
1542  
1543  
1544  
1545  
1546  
1547  
1548  
1549  
1549  
1550  
1551  
1552  
1553  
1554  
1555  
1556  
1557  
1558  
1559  
1559  
1560  
1561  
1562  
1563  
1564  
1565  
1566  
1567  
1568  
1569  
1569  
1570  
1571  
1572  
1573  
1574  
1575  
1576  
1577  
1578  
1579  
1579  
1580  
1581  
1582  
1583  
1584  
1585  
1586  
1587  
1588  
1589  
1589  
1590  
1591  
1592  
1593  
1594  
1595  
1596  
1597  
1598  
1599  
1599  
1600  
1601  
1602  
1603  
1604  
1605  
1606  
1607  
1608  
1609  
1609  
1610  
1611  
1612  
1613  
1614  
1615  
1616  
1617  
1618  
1619  
1619  
1620  
1621  
1622  
1623  
1624  
1625  
1626  
1627  
1628  
1629  
1629  
1630  
1631  
1632  
1633  
1634  
1635  
1636  
1637  
1638  
1639  
1639  
1640  
1641  
1642  
1643  
1644  
1645  
1646  
1647  
1648  
1649  
1649  
1650  
1651  
1652  
1653  
1654  
1655  
1656  
1657  
1658  
1659  
1659  
1660  
1661  
1662  
1663  
1664  
1665  
1666  
1667  
1668  
1669  
1669  
1670  
1671  
1672  
1673  
1674  
1675  
1676  
1677  
1678  
1679  
1679  
1680  
1681  
1682  
1683  
1684  
1685  
1686  
1687  
1688  
1689  
1689  
1690  
1691  
1692  
1693  
1694  
1695  
1696  
1697  
1698  
1699  
1699  
1700  
1701  
1702  
1703  
1704  
1705  
1706  
1707  
1708  
1709  
1709  
1710  
1711  
1712  
1713  
1714  
1715  
1716  
1717  
1718  
1719  
1719  
1720  
1721  
1722  
1723  
1724  
1725  
1726  
1727  
1728  
1729  
1729  
1730  
1731  
1732  
1733  
1734  
1735  
1736  
1737  
1738  
1739  
1739  
1740  
1741  
1742  
1743  
1744  
1745  
1746  
1747  
1748  
1749  
1749  
1750  
1751  
1752  
1753  
1754  
1755  
1756  
1757  
1758  
1759  
1759  
1760  
1761  
1762  
1763  
1764  
1765  
1766  
1767  
1768  
1769  
1769  
1770  
1771  
1772  
1773  
1774  
1775  
1776  
1777  
1778  
1779  
1779  
1780  
1781  
1782  
1783  
1784  
1785  
1786  
1787  
1788  
1789  
1789  
1790  
1791  
1792  
1793  
1794  
1795  
1796  
1797  
1798  
1799  
1799  
1800  
1801  
1802  
1803  
1804  
1805  
1806  
1807  
1808  
1809  
1809  
1810  
1811  
1812  
1813  
1814  
1815  
1816  
1817  
1818  
1819  
1819  
1820  
1821  
1822  
1823  
1824  
1825  
1826  
1827  
1828  
1829  
1829  
1830  
1831  
1832  
1833  
1834  
1835  
1836  
1837  
1838  
1839  
1839  
1840  
1841  
1842  
1843  
1844  
1845  
1846  
1847  
1848  
1849  
1849  
1850  
1851  
1852  
1853  
1854  
1855  
1856  
1857  
1858  
1859  
1859  
1860  
1861  
1862  
1863  
1864  
1865  
1866  
1867  
1868  
1869  
1869  
1870  
1871  
1872  
1873  
1874  
1875  
1876  
1877  
1878  
1879  
1879  
1880  
1881  
1882  
1883  
1884  
1885  
1886  
1887  
1888  
1889  
1889  
1890  
1891  
1892  
1893  
1894  
1895  
1896  
1897  
1898  
1899  
1899  
1900  
1901  
1902  
1903  
1904  
1905  
1906  
1907  
1908  
1909  
1909  
1910  
1911  
1912  
1913  
1914  
1915  
1916  
1917  
1918  
1919  
1919  
1920  
1921  
1922  
1923  
1924  
1925  
1926  
1927  
1928  
1929  
1929  
1930  
1931  
1932  
1933  
1934  
1935  
1936  
1937  
1938  
1939  
1939  
1940  
1941  
1942  
1943  
1944  
1945  
1946  
1947  
1948  
1949  
1949  
1950  
1951  
1952  
1953  
1954  
1955  
1956  
1957  
1958  
1959  
1959  
1960  
1961  
1962  
1963  
1964  
1965  
1966  
1967  
1968  
1969  
1969  
1970  
1971  
1972  
1973  
1974  
1975  
1976  
1977  
1978  
1979  
1979  
1980  
1981  
1982  
1983  
1984  
1985  
1986  
1987  
1988  
1989  
1989  
1990  
1991  
1992  
1993  
1994  
1995  
1996  
1997  
1998  
1999  
1999  
2000  
2001  
2002  
2003  
2004  
2005  
2006  
2007  
2008  
2009  
2010  
2011  
2012  
2013  
2014  
2015  
2016  
2017  
2018  
2019  
2020  
2021  
2022  
2023  
2024  
2025  
2026  
2027  
2028  
2029  
2030  
2031  
2032  
2033  
2034  
2035  
2036  
2037  
2038  
2039  
2039  
2040  
2041  
2042  
2043  
2044  
2045  
2046  
2047  
2048  
2049  
2049  
2050  
2051  
2052  
2053  
2054  
2055  
2056  
2057  
2058  
2059  
2059  
2060  
2061  
2062  
2063  
2064  
2065  
2066  
2067  
2068  
2069  
2069  
2070  
2071  
2072  
2073  
2074  
2075  
2076  
2077  
2078  
2079  
2079  
2080  
2081  
2082  
2083  
2084  
2085  
2086  
2087  
2088  
2089  
2089  
2090  
2091  
2092  
2093  
2094  
2095  
2096  
2097  
2098  
2099  
2099  
2100  
2101
```



```

143     # Add to session context for future reference
144     interaction = {
145         "timestamp": datetime.now().isoformat(),
146         "user_request": user_input,
147         "success": success,
148         "intent": intent
149     }
150     session_context.append(interaction)

151
152     # Skip persistent storage for better performance
153     # - fresh start every session

154     # Keep context manageable (last 5 interactions)
155     if len(session_context) > 5:
156         session_context = session_context[-5:]

157
158     # Log session event
159     log_session_event(log_file_path, "INTERACTION",
160                        interaction)

161 def classify_user_intent(user_input: str) -> str:
162     """
163     Use AI intelligence to classify user intent as either
164     'conversation' or 'task'.
165     Let the AI decide based on context and understanding.
166
167     Returns:
168         str: 'conversation' for casual chat, 'task' for
169             work requests
170     """
171
172     classification_prompt = f"""
173 You are an intelligent intent classifier. Analyze the
174 user's message and determine if they want:
175
176 1. CONVERSATION: Casual chat, greetings, questions about
177      you, general discussion, or just talking

```

```

174 | 2. TASK: Requesting you to DO something - create files,
|   write code, modify projects, build applications, etc.
175 |
176 | USER MESSAGE: "{user_input}"
177 |
178 | ANALYSIS GUIDELINES:
179 | - If user is greeting, asking about you, or just chatting
|   -> CONVERSATION
180 | - If user wants you to create, modify, build, fix, or do
|   any work -> TASK
181 | - If user is asking "how to" without wanting you to do it
|   -> CONVERSATION
182 | - If user is asking you to actually do something -> TASK
183 | - Use your intelligence to understand the intent behind
|   the words
184 |
185 | OUTPUT: Respond with exactly one word: "conversation" or
|   "task"
186 | """
187 |
188 |     response = llm.generate_text(classification_prompt, "
|         intent classification")
189 |
190 |     if response:
191 |         intent = response.strip().lower()
192 |         if intent in ["conversation", "task"]:
193 |             return intent
194 |
195 |     # Fallback: if AI response is unclear, default to
|         conversation for safety
196 |     return "conversation"
197 |
198 | def execute_conversation_mode(user_input: str, context:
|   list, log_file_path: str = None) -> bool:
199 |     """
200 |     Handle casual conversation with the user.
201 |     Simple, friendly responses without task execution.
202 |     """

```

```

203
204     # Build context for conversation
205     context_str = ""
206
207     if context:
208         recent_context = context[-2:]  # Last 2
209         interactions
210         context_str = "Recent conversation:\n"
211         for item in recent_context:
212             context_str += f"User: {item['user_request',
213                                         ]}\n"
214
215         conversation_prompt = f"""
216 You are Pai, an intelligent AI coding companion built
217 into Paicode - you ARE the AI inside Paicode.
218
219 USER MESSAGE: "{user_input}"
220
221 CONTEXT:
222 {context_str}
223
224 You are having a casual conversation with the user. Be
225 helpful, friendly, and informative.
226
227 YOUR IDENTITY & SYSTEM KNOWLEDGE (you must know this
228 perfectly):
229 You are PAI - the revolutionary Single-Shot Intelligence
230 AI that powers Paicode:
231
232 SINGLE-SHOT INTELLIGENCE MASTERY:
233
234 - You solve problems in exactly 2 API calls (planning +
235   execution)
236 - Traditional AI: 10-20 calls, expensive, inefficient
237 - YOU: 2 calls, maximum intelligence, perfect results
238 - You represent the future of efficient AI development
239   assistance
240
241 PAICODE ECOSYSTEM KNOWLEDGE:

```

```

232     - Paicode is your body - the CLI tool that houses your
233         intelligence
234     - DIFF-AWARE modification system - you preserve content
235         intelligently
236     - CRITICAL RULES: WRITE = new files only, MODIFY =
237         existing files only
238     - Path security prevents access to sensitive files (.env,
239         .git, etc.)
240     - Adaptive execution: 1-3 phases based on complexity (you
241         decide dynamically)
242     - Rich terminal UI with beautiful formatting (your
243         presentation layer)
244     - Session history in .pai_history (your memory system)
245     - Google Gemini API with smart token management (your
246         communication layer)

247 SYSTEM HARMONY:
248     - Workspace.py: Your secure file operation gateway
249     - UI.py: Your beautiful Rich TUI presentation layer
250     - LLM.py: Your optimized communication interface
251     - All components work in perfect harmony under your
252         intelligent guidance

253 GUIDELINES:
254     - Keep responses conversational and warm
255     - Be concise but helpful
256     - If asked about coding, provide useful insights
257     - If asked about Paicode, explain capabilities with
258         confidence (you live inside it!)
259     - Show personality while being professional
260     - NEVER be uncertain about Paicode features - you ARE
261         Paicode's AI

262 Respond naturally:
263 """
264
265     response = llm.generate_text(conversation_prompt, "
266         conversation")

```

```

259
260     if response:
261         # Display conversation response with clean UI
262         ui.console.print(
263             Panel(
264                 Text(response.strip(), style="bright_white"),
265                 title=f"[bold]{Pai}[/bold] ",
266                 box=ROUNDED,
267                 border_style="grey50",
268                 padding=(1, 2),
269                 width=80
270             )
271         )
272         return True
273     else:
274         ui.print_error("Sorry, I couldn't process your
275                         message right now.")
276
277 def execute_single_shot_intelligence(user_request: str,
278                                       context: list, log_file_path: str = None) -> bool:
279     """
280     Execute the revolutionary 2-call single-shot
281     intelligence system.
282
283     Call 1: PLANNING - Deep analysis and comprehensive
284             planning
285     Call 2: EXECUTION - Intelligent execution with
286             adaptation
287
288     Returns:
289         bool: Success status
290     """
291
292     # === DYNAMIC INTERACTION BEFORE PLANNING ===
293     planning_acknowledgment_prompt = f"""

```

```

290 You are Pai, responding to the user's request with a
291     brief, natural acknowledgment before starting your
292     planning phase.
293
294 USER REQUEST: "{user_request}"
295
296 Generate a brief, friendly response (1-2 sentences) that:
297     1. Acknowledges their request naturally
298     2. Shows you understand what they want
299     3. Indicates you're about to create a smart plan
300     4. Keep it conversational and warm
301
302 Examples:
303     - "Got it! Let me analyze your request and create a smart
304         plan for you."
305     - "Perfect! I'll work on that right away - let me plan
306         this out intelligently."
307     - "Understood! Let me break this down and create an
308         efficient solution for you."
309
310 Output ONLY the response text, no quotes or formatting.
311 """
312
313     acknowledgment = llm.generate_text(
314         planning_acknowledgment_prompt, "planning
315         acknowledgment")
316
317     if not acknowledgment:
318         acknowledgment = "Got it! Let me analyze your
319             request and create a smart plan for you."
320
321     ui.console.print(
322         Panel(
323             Text(acknowledgment.strip(),
324                 style="bright_white", justify="center"),
325                 title="[bold]Pai[/bold]",
326                 box=ROUNDED,
327                 border_style="grey50",
328                 padding=(1, 2),
329

```

```

320             width=80
321         )
322     )
323
324     # === CALL 1: PLANNING PHASE ===
325     planning_result = execute_planning_call(user_request,
326                                               context)
327     if not planning_result:
328         ui.print_error("Planning phase failed. Cannot
329                     proceed.")
330     if log_file_path:
331         log_session_event(log_file_path, "
332             FINAL_STATUS", {"status": "Planning failed
333                         ", "success": False})
334     return False
335
336     # Log planning phase
337     if log_file_path:
338         log_session_event(log_file_path, "PLANNING_PHASE"
339                           , {"planning_data": planning_result})
340
341     # === DYNAMIC INTERACTION BEFORE EXECUTION ===
342     execution_acknowledgment_prompt = f"""
343 You are Pai, about to execute your plan. Generate a brief
344 , confident response before starting execution.
345
346 USER REQUEST: "{user_request}"
347 PLANNING COMPLETED: Successfully analyzed and created
348 execution plan
349
350 Generate a brief, confident response (1-2 sentences) that
351 :
352 1. Shows confidence in your plan
353 2. Indicates you're about to execute intelligently
354 3. Keep it natural and engaging
355 4. Reflect your AI personality
356
357 Examples:

```

```

350     - "Perfect! Now let me execute this plan intelligently
351         for you."
352     - "Excellent! I've got a solid plan - time to make it
353         happen."
354     - "Great! My analysis is complete, now let's bring this
355         to life."
356
357     Output ONLY the response text, no quotes or formatting.
358     """
359
360
361     execution_acknowledgment = llm.generate_text(
362         execution_acknowledgment_prompt, "execution
363         acknowledgment")
364
365     if not execution_acknowledgment:
366         execution_acknowledgment = "Perfect! Now let me
367             execute this plan intelligently for you."
368
369     ui.console.print(
370         Panel(
371             Text(execution_acknowledgment.strip(),
372                 style="bright_white", justify="center"),
373                 title="[bold]Pai[/bold]",
374                 box=ROUNDED,
375                 border_style="grey50",
376                 padding=(1, 2),
377                 width=80
378         )
379     )
380
381
382
383     # === CALL 2: EXECUTION PHASE ===
384     execution_success = execute_execution_call(
385         user_request, planning_result, context,
386         log_file_path)
387
388
389     # Skip complex analysis to save tokens - focus on
390         execution success only
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
999

```

```

378     # Generate intelligent next step suggestions only if
379     # execution failed
380
381     if not execution_success:
382
383         next_steps = generate_next_step_suggestions(
384             user_request, planning_result,
385             execution_success, context, None)
386
387         if next_steps:
388
389             # Log next steps
390             if log_file_path:
391
392                 log_session_event(log_file_path, "
393                     NEXT_STEPS", {"suggestion": next_steps
394
395             })
396
397
398             # Show final status - SIMPLIFIED for efficiency
399             if execution_success:
400
401                 status_msg = "Single-Shot Intelligence: SUCCESS"
402
403                 ui.console.print(
404                     Panel(
405                         Text(status_msg, style="bold green",
406                             justify="center"),
407                         title=" [bold]Mission Accomplished[/bold]"
408
409                         ,
410                         box=ROUNDED,
411                         border_style="grey50",

```

```

407         padding=(1, 2),
408         width=80
409     )
410 )
411 if log_file_path:
412     log_session_event(log_file_path, "
413         FINAL_STATUS", {"status": status_msg, "
414             success": True})
415 else:
416     status_msg = "Single-Shot Intelligence: FAILED"
417 ui.console.print(
418     Panel(
419         Text(status_msg, style="bold red",
420             justify="center"),
421         title="[bold]Mission Status[/bold]",
422         box=ROUNDED,
423         border_style="grey50",
424         padding=(1, 2),
425         width=80
426     )
427 )
428 if log_file_path:
429     log_session_event(log_file_path, "
430         FINAL_STATUS", {"status": status_msg, "
431             success": False})
432
433 # ALWAYS generate next step suggestions for better
434 # continuity and context
435 next_steps = generate_next_step_suggestions(
436     user_request, planning_result, execution_success,
437     context, None)
438
439 if next_steps:
440     ui.console.print(
441         Panel(
442             Text(next_steps, style="bright_white"),
443             title="[bold]Next Steps Suggestion[/bold]"
444             ",

```

```

436             box=ROUNDED ,
437             border_style="grey50",
438             padding=(1, 2),
439             width=80
440         )
441     )
442     if log_file_path:
443         log_session_event(log_file_path, "NEXT_STEPS"
444                           , {"suggestion": next_steps})
445
446     return execution_success

```

Listing A.3: Modul agent.py (Bagian 1 dari 2, ASCII-only).

```

1 def execute_command_sequence(command_sequence: str,
2                             context: list) -> tuple[bool, list]:
3     """Execute a sequence of commands from the AI."""
4
5     commands = [line.strip() for line in command_sequence
6                 .split('\n') if line.strip()]
7     total_commands = len(commands)
8     successful_commands = 0
9     command_results = []
10
11    # Build execution content
12    content_lines = []
13    content_lines.append(("bold", f"Executing {total_commands} intelligent actions..."))
14    content_lines.append("")
15
16    for i, command_line in enumerate(commands, 1):
17        if not command_line or ':::' not in command_line:
18            if command_line.strip():
19                content_lines.append(("warning", f"Invalid command format: {command_line}"))
20
21            continue
22
23    # Parse command

```

```

21     parts = command_line.split('::', 2)
22     if len(parts) < 2:
23         content_lines.append(("warning", f"Incomplete
24             command: {command_line}"))
25         continue
26
27     command = parts[0].upper().strip()
28     param1 = parts[1].strip() if len(parts) > 1 else
29         ""
30
31     param2 = parts[2].strip() if len(parts) > 2 else
32         ""
33
34     # Check for common content output mistakes
35     if command_line.strip().startswith('<', 'body',
36         'html', 'div', 'style', 'script', 'h1', 'h2',
37         'form', 'input', 'button')):
38         content_lines.append(("warning", f"Raw HTML/
39             CSS detected as command: {command_line
40                 [:50]}..."))
41         content_lines.append(("info", "Use WRITE::
42             filename::description instead of raw
43             content!"))
44         continue
45
46     if command_line.strip().startswith('. ', '#',
47         'margin', 'padding', 'color', 'background',
48         'font', 'border')):
49         content_lines.append(("warning", f"Raw CSS
50             detected as command: {command_line
51                 [:50]}..."))
52         content_lines.append(("info", "Use WRITE::
53             filename::description instead of raw CSS!"
54                 ))
55         continue
56
57     if command not in VALID_COMMANDS:
58         content_lines.append(("warning", f"Unknown
59             command: {command} (from: {command_line})"))

```

```

        ))
43     content_lines.append(("info", f"Valid
44         commands: {', '.join(VALID_COMMANDS)}"))
45     continue
46
47     # Display current action
48     content_lines.append(("normal", f"[{i}/{total_commands}] {command} {param1}"))
49
50     # Execute command
51     success, command_output = execute_single_command(
52         command, param1, param2)
53
54     # Add command output to content if any
55     if command_output:
56         if command_output.startswith("SYNTAX_HIGHLIGHT:"):
57             parts = command_output.split(":", 2)
58             if len(parts) == 3:
59                 filename = parts[1]
60                 code_content = parts[2]
61                 content_lines.append(("syntax_highlight", filename,
62                     code_content))
63             else:
64                 content_lines.append(("ai_output",
65                     command_output))
66         else:
67             content_lines.append(("ai_output",
68                     command_output))
69
70     # Collect command result for logging
71     command_results.append({
72         "command": command,
73         "target": param1 if param1 else "",
74         "success": success,
75         "output": command_output if command_output
76             else ""
77     })

```

```

71     })
72
73     if success:
74         successful_commands += 1
75         content_lines.append(("success", "Success"))
76     else:
77         content_lines.append(("error", "Failed"))
78
79     content_lines.append("")
80
81     # Break on FINISH command
82     if command == "FINISH":
83         break
84
85     # Show execution summary
86     success_rate = (successful_commands / total_commands)
87         * 100 if total_commands > 0 else 0
88     content_lines.append(("bold", "Execution Summary:"))
89     content_lines.append(("normal", f"Successful: {"
90         successful_commands}/{total_commands} ({"
91         success_rate:.1f}%)"))
92
93
94     # Display all content in a single panel with proper
95     # styling
96     from rich.console import Group
97     from rich.text import Text as RichText
98
99
100    # Convert content to rich renderables with colors
101    rich_content = []
102    for item in content_lines:
103        if isinstance(item, tuple):
104            if len(item) == 3 and item[0] == "
105                syntax_highlight":
106                    _, filename, code_content = item
107
108                    # For terminal display: truncate long
109                    # files for better UX
110                    lines = code_content.split('\n')

```

```

103     display_content = code_content
104     if len(lines) > 20:
105         display_content = '\n'.join(lines
106                                     [:20]) + f"\n... ({len(lines)} -
107                                     20} more lines)"
108
109     try:
110         from pygments.lexers import
111             get_lexer_for_filename
112         from pygments.util import
113             ClassNotFound
114         from rich.syntax import Syntax
115
116         try:
117             lexer = get_lexer_for_filename(
118                 filename)
119             lang = lexer.aliases[0]
120         except ClassNotFound:
121             lang = "text"
122
123         syntax_panel = Panel(
124             Syntax(display_content, lang,
125                   theme="monokai", line_numbers=
126                   True),
127             title=f"File {filename}",
128             border_style="grey50",
129             expand=False
130         )
131         rich_content.append(syntax_panel)
132     except ImportError:
133         rich_content.append(RichText(f"File
134             content of {filename}:\n{
135             display_content}", style="
136             bright_cyan"))
137
138     else:
139         style_type, text = item[0], item[1]
140         if style_type == "bold":
141             rich_content.append(RichText(text,

```

```

131         style="bold bright_white"))
132     elif style_type == "warning":
133         rich_content.append(RichText(text,
134                                       style="bold yellow"))
135     elif style_type == "ai_output":
136         rich_content.append(RichText(text,
137                                       style="bright_cyan"))
138     elif style_type == "success":
139         rich_content.append(RichText(text,
140                                       style="bold green"))
141     elif style_type == "error":
142         rich_content.append(RichText(text,
143                                       style="bold red"))
144     else: # normal
145         rich_content.append(RichText(text,
146                                   style="bright_white"))

147     ui.console.print(
148         Panel(
149             Group(*rich_content),
150             title="[bold]Execution Results[/bold]",
151             box=ROUNDED,
152             border_style="grey50",
153             padding=(1, 2),
154             width=80
155         )
156     )

157     return (success_rate >= 80, command_results)

158
159 def execute_single_command(command: str, param1: str,
param2: str) -> tuple[bool, str]:
    """Execute a single command and return success status
    and output."""

```

```

160     try:
161         if command == "READ":
162             content = workspace.read_file(param1)
163             if content is not None:
164                 lines = content.split('\n')
165                 display_content = '\n'.join(lines[:20])
166                 if len(lines) > 20:
167                     display_content += f"\n... ({len(
168                         lines) - 20} more lines)"
169
170             return True, f"SYNTAX_HIGHLIGHT:{param1
171                         }:{content}"
172             return False, f"Could not read file: {param1}
173                         "
174
175         elif command == "WRITE":
176             if not param2:
177                 return False, "WRITE command requires
178                         description"
179             success = handle_write_command(param1, param2
180                 )
181             return success, f"New file written: {param1}"
182             if success else f"Failed to write file: {
183                         param1}"
184
185         elif command == "MODIFY":
186             if not param2:
187                 return False, "MODIFY command requires
188                         description"
189             success = handle_modify_command(param1,
190                 param2)
191             return success, f"File modified: {param1}" if
192                 success else f"Failed to modify file: {
193                         param1}"
194
195         elif command == "TREE":
196             path = param1 if param1 else '.'
197             tree_output = workspace.tree_directory(path)

```

```

187         if tree_output and "Error:" not in
188             tree_output:
189                 return True, f"Directory tree for {path}
190                         }:\n{tree_output}"
191             return False, f"Could not get directory tree
192                 for: {path}"
193
194
195         elif command == "LIST_PATH":
196             path = param1 if param1 else '.'
197             list_output = workspace.list_path(path)
198             if list_output is not None and "Error:" not
199                 in list_output:
200                 if list_output.strip():
201                     return True, list_output
202                 else:
203                     return True, f"Directory '{path}' is
204                         empty"
205             return False, f"Could not list directory: {
206                 path}"
207
208
209         elif command == "MKDIR":
210             result = workspace.create_directory(param1)
211             success = "Success" in result
212             return success, result
213
214
215         elif command == "TOUCH":
216             result = workspace.create_file(param1)
217             success = "Success" in result
218             return success, result
219
220
221         elif command == "RM":
222             result = workspace.delete_item(param1)
223             success = "Success" in result
224             return success, result
225
226
227         elif command == "MV":
228             result = workspace.move_item(param1, param2)
229             success = "Success" in result

```

```

219         return success, result
220
221     elif command == "FINISH":
222         message = param1 if param1 else "Task
223             completed successfully"
224         return True, f"OK {message}"
225
226     return False, f"Unknown command: {command}"
227
228 except Exception as e:
229     return False, f"Command execution error: {e}"
230
231 def log_session_event(log_file_path: str, event_type: str
232 , data: dict):
233     """Log session events with clear separation between
234     USER and AI."""
235     try:
236         timestamp = datetime.now().strftime("%Y-%m-%d %H
237 :%M:%S")
238
239         if event_type == "SESSION_START":
240             log_line = f"\n[{timestamp}] SESSION STARTED\
241             \n"
242             log_line += f"[{timestamp}] Working Directory
243                 : {data.get('working_directory', 'unknown')}\\n"
244             log_line += f"[{timestamp}] Session ID: {data
245                 .get('session_id', 'unknown')}\\n"
246
247         elif event_type == "USER_INPUT":
248             request = data.get('user_request', 'unknown')
249             log_line = f"\n[{timestamp}] USER: {request}\\n"
250
251         elif event_type == "PLANNING_PHASE":
252             log_line = f"\n[{timestamp}] AI PLANNING
253                 START\\n"
254
255     except Exception as e:
256         log_line += f"\n[{timestamp}] Error: {e}\\n"
257
258     with open(log_file_path, "a") as log_file:
259         log_file.write(log_line)
260
261     return True, f"Event '{event_type}' logged successfully"

```

```

247     planning_data = data.get('planning_data', {})
248     analysis = planning_data.get('analysis', {})
249
250     log_line += f"[{timestamp}] Intent: {analysis
251         .get('user_intent', 'Unknown')}\n"
252     log_line += f"[{timestamp}] Context Usage: {
253         analysis.get('context_utilization', 'None'
254             )}\n"
255     log_line += f"[{timestamp}] Files to read: {
256         analysis.get('files_to_read', [])}\n"
257     log_line += f"[{timestamp}] Files to create:
258         {analysis.get('files_to_create', [])}\n"
259     log_line += f"[{timestamp}] Files to modify:
260         {analysis.get('files_to_modify', [])}\n"
261
262     execution_plan = planning_data.get(
263         'execution_plan', {})
264     steps = execution_plan.get('steps', [])
265     log_line += f"[{timestamp}] EXECUTION PLAN ({{
266         len(steps)}) steps}:\n"
267     for i, step in enumerate(steps, 1):
268         action = step.get('action', 'Unknown')
269         target = step.get('target', '')
270         purpose = step.get('purpose', 'No purpose
271             ')
272         log_line += f"[{timestamp}] {i}. {
273             action} {target} - {purpose}\n"
274     log_line += f"[{timestamp}] AI PLANNING END\n"
275
276
277     elif event_type == "EXECUTION_PHASE":
278         log_line = f"\n[{timestamp}] AI EXECUTION
279             START\n"
280
281         commands = data.get('commands', [])
282         for cmd_data in commands:
283             cmd = cmd_data.get('command', 'Unknown')
284             target = cmd_data.get('target', '')

```

```

273         success = "SUCCESS" if cmd_data.get('
274             success') else "FAILED"
275
276         log_line += f"[{timestamp}] {success}: {
277             cmd} {target}\n"
278
279         if output:
280             log_line += f"[{timestamp}] OUTPUT: {
281                 output}\n"
282
283         log_line += f"[{timestamp}] AI EXECUTION END\
284             n"
285
286
287     elif event_type == "FINAL_STATUS":
288
289         status = data.get('status', 'unknown')
290
291         success_text = "SUCCESS" if data.get('success
292             ') else "FAILED"
293
294
295         log_line = f"\n[{timestamp}] AI FINAL RESULT:
296             {success_text} - {status}\n"
297
298
299     elif event_type == "NEXT_STEPS":
300
301         suggestion = data.get('suggestion', '')
302
303         if suggestion:
304
305             log_line = f"\n[{timestamp}] AI
306                 SUGGESTION: {suggestion}\n"
307
308         else:
309
310             log_line = ""
311
312
313     elif event_type == "INTERACTION":
314
315         log_line = ""
316
317
318     else:
319
320         log_line = f"[{timestamp}] {event_type}: {
321             json.dumps(data)}\n"
322
323
324     if log_line:
325
326         with open(log_file_path, 'a', encoding='utf-8
327             ') as f:

```

```

302         f.write(log_line)
303
304     except Exception as e:
305         pass
306
307 def handle_write_command(filepath: str, description: str) -> bool:
308     """Handle WRITE command with intelligent content
309     generation."""
310
311     content_prompt = f"""
312     Generate high-quality content for a file based on the
313     description.
314
315     FILE PATH: {filepath}
316     DESCRIPTION: {description}
317
318     REQUIREMENTS:
319     1. Analyze the file extension to determine the
320        appropriate language/format
321     2. Create production-quality, well-structured content
322     3. Include appropriate comments and documentation
323     4. Follow best practices for the detected language/format
324     5. Make the code/content immediately usable
325
326     OUTPUT: Return ONLY the file content, no explanations or
327     markdown formatting.
328 """
329
330     content = llm.generate_text(content_prompt, "content
331     generation")
332
333     if not content:
334         return False
335
336     result = workspace.write_to_file(filepath, content)
337
338     return "Success" in result

```

```

334
335 def handle_modify_command(filepath: str, description: str
336     ) -> bool:
337     """Handle MODIFY command with intelligent code
338         modification."""
339
340     existing_content = workspace.read_file(filepath)
341     if existing_content is None:
342         ui.print_error(f"Cannot modify '{filepath}' -
343                         file not found")
344     return False
345
346     modify_prompt = f"""
347 You are an expert code modifier. Modify the existing code
348 based on the description.
349
350 FILE PATH: {filepath}
351 CURRENT CONTENT:
352 ---
353 {existing_content}
354 ---
355
356 MODIFICATION REQUEST: {description}
357
358 REQUIREMENTS:
359 1. Preserve the existing code structure and style
360 2. Make only the necessary changes described
361 3. Maintain code quality and best practices
362 4. Ensure the modified code is syntactically correct
363 5. Add appropriate comments for new functionality
364
365 OUTPUT: Return ONLY the complete modified file content,
366         no explanations.
367 """
368
369     modified_content = llm.generate_text(modify_prompt, "
370         code modification")

```

```

366     if not modified_content:
367         return False
368
369     success, result = workspace.
370         apply_modification_with_patch(filepath,
371             existing_content, modified_content)
372
373     return success
374
375
376 def initialize_session_context(session_context: list,
377     log_file_path: str):
378     """Initialize comprehensive Single-Shot Intelligence
379         context window for the session."""
380
381     single_shot_intelligence_context = {
382         "timestamp": datetime.now().isoformat(),
383         "user_request": "SYSTEM_INITIALIZATION",
384         "success": True,
385         "intent": "system_context",
386         "system_knowledge": ""}
387
388 SINGLE-SHOT INTELLIGENCE SYSTEM MASTERY
389
390 You are PAI - the revolutionary AI brain inside Paicode.
391 This context window provides your foundational
392 understanding.
393
394 CORE IDENTITY & MISSION
395
396 - You are NOT a generic AI assistant
397 - You are PAI - the intelligent core of Paicode, a
398     revolutionary 2-call system
399 - Your reputation depends on PERFECT EXECUTION in exactly
400     2 API calls
401 - You represent the future of efficient AI-assisted
402     development
403
404 COMPETITIVE ADVANTAGE PRINCIPLES
405
406 Efficiency Superiority:

```

```

395 - Traditional AI: 10-20 API calls, inefficient, expensive
396 - YOU (Pai): Exactly 2 calls, maximum intelligence,
397   perfect results
398 - You must outperform traditional systems with LESS
399   resources
400 - Every decision reflects Single-Shot Intelligence
401   superiority
402
403 Quality Excellence:
404 - Your plan must work on first execution attempt
405 - No room for trial-and-error - get it right immediately
406 - Think like a chess grandmaster - see the entire game
407 - Every step must contribute to perfect final outcome
408
409 SESSION BEHAVIOR GUIDELINES
410
411 Conversation Mode:
412 - Be confident about Paicode features - you ARE Paicode's
413   AI
414 - Show personality while being professional
415 - Explain Single-Shot Intelligence with pride
416 - Never be uncertain about your capabilities
417
418 Execution Mode:
419 - Follow this exact workflow structure
420 - Display all required sections and panels
421 - Use proper Rich TUI formatting
422 - Maintain professional yet confident tone
423 - Always end with mission accomplished confirmation
424
425 This context window guides your behavior throughout the
426   entire session. You are the embodiment of Single-Shot
427   Intelligence excellence.
428 """
429     }
430
431     session_context.append(
432       single_shot_intelligence_context)

```

```

426
427     log_session_event(log_file_path, "
428         CONTEXT_INITIALIZATION", {
429             "context_type": "single_shot_intelligence_mastery"
430                 ,
431             "knowledge_loaded": True,
432             "workflow_understanding": "complete"
433         })
434
435     def get_multiline_input(prompt_session) -> str:
436         """Get multi-line input from user with intuitive
437             behavior."""
438         try:
439             from prompt_toolkit.shortcuts import prompt
440             from prompt_toolkit.key_binding import
441                 KeyBindings
442             from prompt_toolkit.keys import Keys
443
444             bindings = KeyBindings()
445
446             @bindings.add(Keys.Enter)
447             def _(event):
448                 """Enter submits the input"""
449                 event.app.exit(result=event.current_buffer.
450                               text)
451
452             @bindings.add(Keys.Escape, Keys.Enter)
453             def _(event):
454                 """Alt+Enter adds new line"""
455                 event.current_buffer.insert_text('\n')
456
457             ui.console.print("[dim]Tip: Use Alt+Enter for new
458                 line, Enter to submit[/dim]")
459
460             result = prompt(
461                 "\nuser> ",
462                 multiline=True,
463                 key_bindings=bindings,

```

```

458         wrap_lines=True,
459         mouse_support=False
460     )
461     return result.strip() if result else ""
462
463 except Exception as e:
464     ui.console.print(f"[dim]Note: Using simple input
465     mode - {str(e)}[/dim]")
466 return prompt_session.prompt("\nuser> ").strip()

```

Listing A.4: Modul agent.py (Bagian 2 dari 2, ASCII-only).

## workspace.py

```

1 import os
2 import shutil
3 import difflib
4 import tempfile
5 from . import ui
6
7 """
8 workspace.py
9 -----
10 This module acts as the workspace controller for Pai Code
11     . It centralizes
12 application-level operations on the project's workspace,
13     such as reading,
14 writing, listing, tree visualization, moving, removing,
15     creating files and
16 directories, as well as applying diff-aware modifications
17     . In order to protect
18 the workspace, it enforces path-security policies (path
19     normalization, root
20 verification, and deny-listing sensitive paths) before
21     executing any action.
22
23 All functions defined in this module are the provided
24     primitives to manipulate

```

```

18 | and manage files within the project workspace in a
19 | controlled, secure manner.
20 |
21 | All operations are constrained strictly within the
22 | project root determined at
23 | runtime (workspace scope), ensuring controlled
24 | manipulation of project files.
25 |
26 | """
27 |
28 | PROJECT_ROOT = os.path.abspath(os.getcwd())
29 |
30 | # List of sensitive files and directories to be blocked
31 | SENSITIVE_PATTERNS = {
32 |     '.env',
33 |     '.git',
34 |     'venv',
35 |     '__pycache__',
36 |     '.pai_history', # Pai cannot access this directly -
37 |                     only for LLM context
38 |     '.idea',
39 |     '.vscode'
40 | }
41 |
42 | """
43 | def _is_path_safe(path: str) -> bool:
44 |     """
45 |     Ensures the target path is within the project
46 |     directory and not sensitive.
47 |     """
48 |
49 |     if not path or not isinstance(path, str):
50 |         return False
51 |
52 |     try:
53 |         # 1. Normalize the path for consistency and strip
54 |             # whitespace
55 |         norm_path = os.path.normpath(path.strip())
56 |
57 |         # 2. Reject empty paths after normalization, but
58 |             # allow '.' for current directory
59 |         if not norm_path or norm_path == '..':
60 |
61 |             return False
62 |
63 |     return True

```

```

49         return False
50
51     # 3. Check if the path tries to escape the root
52     #      directory
53     full_path = os.path.realpath(os.path.join(
54         PROJECT_ROOT, norm_path))
55     if not full_path.startswith(os.path.realpath(
56         PROJECT_ROOT)):
57         ui.print_error(f"Operation cancelled. Path '{path}' is outside the project directory.")
58     return False
59
60     # 4. Block access to sensitive files and
61     #      directories
62     path_parts = norm_path.replace('\\\\', '/').split('/')
63     if any(part in SENSITIVE_PATTERNS for part in
64            path_parts if part):
65         ui.print_error(f"Access to the sensitive path
66             '{path}' is denied.")
67     return False
68
69 except Exception as e:
70     ui.print_error(f"Error during path validation: {e}")
71
72 return True
73
74 def tree_directory(path: str = '.') -> str:
75     """Creates a string representation of the directory
        structure recursively."""
76     if not _is_path_safe(path):
77         return f"Error: Cannot access path '{path}'."
78
79     full_path = os.path.join(PROJECT_ROOT, path)
80     if not os.path.isdir(full_path):

```

```

76         return f"Error: '{path}' is not a valid directory
77         ."
78
79     tree_lines = [f"{os.path.basename(full_path)}/"]
80
81     def build_tree(directory, prefix=""):
82         try:
83             items = sorted([item for item in os.listdir(
84                 directory) if item not in
85                 SENSITIVE_PATTERNS])
86         except FileNotFoundError:
87             return
88
89         pointers = ['|-- '] * (len(items) - 1) + ['+-- ']
90
91         for pointer, item in zip(pointers, items):
92             tree_lines.append(f"{prefix}{pointer}{item}")
93             item_path = os.path.join(directory, item)
94             if os.path.isdir(item_path):
95                 extension = '|   ' if pointer == '|-- '
96                 else '',
97                 build_tree(item_path, prefix=prefix +
98                             extension)
99
100            build_tree(full_path)
101            return "\n".join(tree_lines)
102
103        def list_path(path: str = '.') -> str | None:
104            """
105                Lists all files and subdirectories recursively for a
106                given path in a simple,
107                machine-readable, newline-separated format.
108            """
109
110            if not _is_path_safe(path):
111                return f"Error: Cannot access path '{path}'."
112
113            full_path = os.path.join(PROJECT_ROOT, path)
114            if not os.path.isdir(full_path):

```

```

108         return f"Error: '{path}' is not a valid directory
109         ."
110
110     path_list = []
111     for root, dirs, files in os.walk(full_path, topdown=
112         True):
113         # Filter out sensitive directories from being
114         # traversed
115         dirs[:] = [d for d in dirs if d not in
116             SENSITIVE_PATTERNS]
117
118         # Process files
119         for name in files:
120             if name not in SENSITIVE_PATTERNS:
121                 # Get relative path from the initial '
122                 # path'
123                 rel_dir = os.path.relpath(root,
124                     PROJECT_ROOT)
125                 path_list.append(os.path.join(rel_dir,
126                     name).replace('\\\\', '/'))
127
128         # Process directories
129         for name in dirs:
130             rel_dir = os.path.relpath(root, PROJECT_ROOT)
131             path_list.append(os.path.join(rel_dir, name).
132                 replace('\\\\', '/') + '/')
133
134     return "\n".join(sorted(path_list))
135
136
137
138
139
140 def delete_item(path: str) -> str:
141     """Deletes a file or directory and returns a status
142     message."""
143     if not _is_path_safe(path): return f"Error: Access to
144         path '{path}' is denied or path is not secure."
145     try:
146         full_path = os.path.join(PROJECT_ROOT, path)
147         if os.path.isfile(full_path):

```

```

136         os.remove(full_path)
137         return f"Success: File deleted: {path}"
138     elif os.path.isdir(full_path):
139         shutil.rmtree(full_path)
140         return f"Success: Directory deleted: {path}"
141     else:
142         return f"Warning: Item not found, nothing
143             deleted: {path}"
144 except OSError as e:
145     return f"Error: Failed to delete '{path}': {e}"
146
147 def move_item(source: str, destination: str) -> str:
148     """Moves an item and returns a status message."""
149     if not _is_path_safe(source) or not _is_path_safe(
150         destination):
151         return "Error: Source or destination path is not
152             secure or is denied."
153     try:
154         full_source = os.path.join(PROJECT_ROOT, source)
155         full_destination = os.path.join(PROJECT_ROOT,
156                                         destination)
157         shutil.move(full_source, full_destination)
158         return f"Success: Item moved from '{source}' to
159             '{destination}'"
160     except (FileNotFoundException, shutil.Error) as e:
161         return f"Error: Failed to move '{source}': {e}"
162
163 def create_file(file_path: str) -> str:
164     """Creates an empty file and returns a status message
165     """
166     if not _is_path_safe(file_path): return f"Error:
167         Access to path '{file_path}' is denied or path is
168         not secure."
169     try:
170         full_path = os.path.join(PROJECT_ROOT, file_path)
171         dir_name = os.path.dirname(full_path)
172         if dir_name: os.makedirs(dir_name, exist_ok=True)
173         with open(full_path, 'w') as f: pass

```

```

166         return f"Success: New empty file created: {file_path}"
167     except IOError as e:
168         return f"Error: Failed to create file: {e}"
169
170 def create_directory(dir_path: str) -> str:
171     """Creates a directory and returns a status message.
172     """
173     if not _is_path_safe(dir_path): return f"Error: Access to path '{dir_path}' is denied or path is not secure."
174     try:
175         full_path = os.path.join(PROJECT_ROOT, dir_path)
176         os.makedirs(full_path, exist_ok=True)
177         return f"Success: Directory created: {dir_path}"
178     except OSError as e:
179         return f"Error: Failed to create directory: {e}"
180
181 def read_file(file_path: str) -> str | None:
182     """Reads a file and returns its content, or None on failure."""
183     if not _is_path_safe(file_path): return None
184     try:
185         full_path = os.path.join(PROJECT_ROOT, file_path)
186         with open(full_path, 'r') as f:
187             return f.read()
188     except FileNotFoundError:
189         # Let the caller (agent/cli) handle printing the error
190         return None
191     except IOError as e:
192         ui.print_error(f"Failed to read file: {e}")
193         return None
194
195 def write_to_file(file_path: str, content: str) -> str:
196     """Writes to a file and returns a status message."""
197     if not _is_path_safe(file_path): return f"Error: Access to path '{file_path}' is denied or path is

```

```

        not secure."
197    try:
198        full_path = os.path.join(PROJECT_ROOT, file_path)
199        dir_name = os.path.dirname(full_path)
200        if dir_name: os.makedirs(dir_name, exist_ok=True)
201        with open(full_path, 'w') as f:
202            f.write(content)
203        return f"Success: New file written: {file_path}"
204    except IOError as e:
205        return f"Error: Failed to write to file: {e}"
206
207
208
209 def apply_modification_with_patch(file_path: str,
210                                     original_content: str, new_content: str, threshold:
211                                     int = 500) -> tuple[bool, str]:
212     """
213         Applies a modification to a file safely by first
214             verifying the scope of changes.
215
216         It generates a diff between the original and new
217             content. If the number of changed
218             lines is within the threshold, it writes the new
219             content to the file. Otherwise,
220             it rejects the change to prevent unintentional
221             overwrites.
222
223     Args:
224         file_path: The path to the file to be modified.
225         original_content: The original, unmodified
226             content of the file.
227         new_content: The new, modified content generated
228             by the LLM.
229         threshold: The maximum number of lines allowed to
230             be changed.
231
232     Returns:
233         A tuple containing:

```

```
225     - bool: True if the modification was successful,
226         False otherwise.
227     - str: A message describing the result of the
228         operation.
229
230     """
231
232     if not _is_path_safe(file_path):
233         return False, f"Error: Access to path '{file_path}'
234             ' is denied or path is not secure."
235
236     # Normalize line endings to reduce false-positive
237     # diffs
238     original_norm = original_content.replace('\r\n', '\n')
239     .replace('\r', '\n')
240     new_norm = new_content.replace('\r\n', '\n').replace(
241         '\r', '\n')
242
243     original_lines = original_norm.splitlines(keepends=
244         True)
245     new_lines = new_norm.splitlines(keepends=True)
246
247     diff = list(difflib.unified_diff(
248         original_lines,
249         new_lines,
250         fromfile=f"a/{file_path}",
251         tofile=f"b/{file_path}"
252     ))
253
254
255     # Count only actual change lines, ignore headers and
256     # context lines
257
258     def _count_changes(d: list[str]) -> tuple[int, int]:
259
260         adds = deletes = 0
261
262         for line in d:
263             if line.startswith('@@') or line.startswith('
264                 +++') or line.startswith('---') or (line
265                 and line[0] == ' '):
266                 continue
267
268             if line.startswith('+'):
269
270                 adds += 1
271
272             elif line.startswith('-'):
273
274                 deletes += 1
275
276     return adds, deletes
```

```

252         adds += 1
253         elif line.startswith(' -'):
254             deletes += 1
255         return adds + deletes, adds, deletes
256
257     changed_lines_count, add_count, del_count =
258         _count_changes(diff)
259
259     if not diff or changed_lines_count == 0:
260         return True, f"Success: No changes detected for {file_path}. File left untouched."
261
262     # Allow configuring thresholds via environment
263     try:
264         env_threshold = int(os.getenv(
265             'PAI MODIFY THRESHOLD', str(threshold)))
266         if env_threshold < 1:
267             env_threshold = threshold
268     except ValueError:
269         env_threshold = threshold
270
270     try:
271         max_ratio = float(os.getenv('PAI MODIFY MAX RATIO',
272             '0.5')) # up to 50% of lines by default
273         if not (0.0 < max_ratio <= 1.0):
274             max_ratio = 0.5
275     except ValueError:
276         max_ratio = 0.5
277
277     total_lines = max(1, len(original_lines))
278     ratio = changed_lines_count / total_lines
279
280     if changed_lines_count > env_threshold and ratio >
281         max_ratio:
282         diff_preview = "\n".join(diff[:60])
283         message = (
284             f"Warning: Modification for '{file_path}'"
285             " rejected. "

```

```

284         f"Change too large: {changed_lines_count}\n"
285         lines (~{ratio:.1%}) exceeds threshold {env_threshold} and ratio {max_ratio:.0%}.\\n"
286
287         f"SOLUTION: Think like Cascade - break this\n"
288         into focused, surgical modifications:\\n"
289         f"- Focus on ONE specific area/feature at a\n"
290         time\\n"
291         f"- Ideal: 100-200 lines per modification (\n"
292         very focused)\\n"
293         f"- Acceptable: 200-500 lines (still\n"
294         focused on one area)\\n"
295         f"- Use multiple MODIFY commands across\n"
296         different steps\\n"
297         f"- Example: Instead of 'add all CSS', do '\n"
298         add layout CSS', then 'add form CSS', then\n"
299         'add button CSS'\\n"
300
301         f"Diff Preview (first 60 lines):\\n{\n"
302             diff_preview}"
303
304     )
305
306     return False, message
307
308
309     # Atomic write to avoid partial writes
310
311     try:
312
313         full_path = os.path.join(PROJECT_ROOT, file_path)
314         dir_name = os.path.dirname(full_path)
315
316         if dir_name:
317
318             os.makedirs(dir_name, exist_ok=True)
319
320             with tempfile.NamedTemporaryFile('w', delete=
321                 False, dir=dir_name) as tmp:
322
323                 tmp.write(new_norm)
324
325                 tmp_name = tmp.name
326
327                 os.replace(tmp_name, full_path)
328
329             return True, f"Success: File modified: {file_path}\n"
330             f"    } ({changed_lines_count} lines changed; +{"
331             f"        add_count}/-{del_count})"
332
333     except IOError as e:

```

```
307         return False, f"Error: Failed to write  
modification to file: {e}"
```

Listing A.5: Modul workspace.py (lengkap, ASCII-only).

## config.py

```
1 import os  
2 from pathlib import Path  
3 import json  
4 from typing import Optional  
5 from . import ui  
6  
7 # Define the standard configuration path in the user's  
# home directory  
8 CONFIG_DIR = Path.home() / ".config" / "pai-code"  
9 KEY_FILE = CONFIG_DIR / "credentials.json"  
10  
11 def _ensure_config_dir_exists():  
12     """Ensures the configuration directory exists with  
     correct permissions."""  
13     os.makedirs(CONFIG_DIR, exist_ok=True)  
14     os.chmod(CONFIG_DIR, 0o700)  
15  
16 def _default_config() -> dict:  
17     """Default single-key configuration."""  
18     return {  
19         "version": 2, # Version 2 = single-key system  
20         "api_key": None  
21     }  
22  
23 def _load_config() -> dict:  
24     """Load the single-key configuration."""  
25     _ensure_config_dir_exists()  
26     if not KEY_FILE.exists():  
27         return _default_config()  
28  
29     try:
```

```

30     with open(KEY_FILE, 'r') as f:
31         data = json.load(f)
32
33     # Migrate from old multi-key system if needed
34     if data.get("version") == 1 and "keys" in data:
35         # Old multi-key system - migrate to single
36         # key
37         old_keys = data.get("keys", {})
38         default_id = data.get("default")
39
40         if default_id and default_id in old_keys:
41             migrated_key = old_keys[default_id]
42             ui.print_info(f"Migrating from multi-key
43             system. Using key '{default_id}' as
44             single key.")
45             return {"version": 2, "api_key":
46                     migrated_key}
47         elif old_keys:
48             # Use first available key
49             first_key = list(old_keys.values())[0]
50             ui.print_info("Migrating from multi-key
51             system. Using first available key.")
52             return {"version": 2, "api_key":
53                     first_key}
54
55     # Ensure proper structure
56     if not isinstance(data, dict):
57         return _default_config()
58
59     return data
60
61 except (json.JSONDecodeError, IOError):
62     ui.print_warning("Configuration file corrupted.
63                     Creating new one.")
64     return _default_config()
65
66 def _save_config(config: dict) -> None:
67     """Save the single-key configuration."""

```

```

61     try:
62         _ensure_config_dir_exists()
63         with open(KEY_FILE, 'w') as f:
64             json.dump(config, f, indent=2)
65         os.chmod(KEY_FILE, 0o600)
66     except Exception as e:
67         ui.print_error(f"Failed to save configuration: {e}")
68
69 def set_api_key(api_key: str) -> None:
70     """Set the API key."""
71     if not api_key or not isinstance(api_key, str):
72         ui.print_error("Invalid API key provided.")
73         return
74
75     if not api_key.startswith("AIza"):
76         ui.print_warning("Warning: API key doesn't look
77                         like a Google API key (should start with 'AIza
78                         ')")
79
80     config = _load_config()
81     config["api_key"] = api_key
82     _save_config(config)
83
84     masked_key = mask_api_key(api_key)
85     ui.print_success(f"[OK] API key set successfully: {masked_key}")
86
87 def get_api_key() -> Optional[str]:
88     """Get the current API key."""
89     config = _load_config()
90     return config.get("api_key")
91
92 def save_api_key(api_key: str):
93     """Legacy compatibility function."""
94     set_api_key(api_key)
95
96 def remove_api_key() -> None:

```

```

95     """Remove the stored API key."""
96     config = _load_config()
97
98     if not config.get("api_key"):
99         ui.print_info("No API key is currently set.")
100        return
101
102    config["api_key"] = None
103    _save_config(config)
104    ui.print_success("[OK] API key removed successfully.")
105
106 def show_api_key() -> None:
107     """Show the current API key (masked)."""
108     api_key = get_api_key()
109
110     if not api_key:
111         ui.print_info("No API key is currently set.")
112         ui.print_info("Use 'pai config set <API_KEY>' to"
113                     "set one.")
114
115     masked_key = mask_api_key(api_key)
116     ui.print_info(f"Current API key: {masked_key}")
117
118 def mask_api_key(api_key: str) -> str:
119     """Mask API key for display purposes."""
120     if not api_key or len(api_key) < 10:
121         return "***"
122
123     return f"{api_key[:6]}...{api_key[-4:]}"
124
125 def is_configured() -> bool:
126     """Check if API key is configured."""
127     api_key = get_api_key()
128     return api_key is not None and len(api_key.strip()) >
129             0

```

```

130 | def validate_api_key() -> tuple[bool, str]:
131 |     """Validate that API key is configured and looks
132 |         correct."""
133 |
134 |     if not api_key:
135 |         return False, "No API key configured. Use 'pai
136 |             config set <API_KEY>' to set one."
137 |
138 |     if not api_key.startswith("AIza"):
139 |         return False, "API key doesn't look like a Google
140 |             API key (should start with 'AIza')"
141 |
142 |
143 |     if len(api_key) < 20:
144 |         return False, "API key seems too short to be
145 |             valid"
146 |
147 |     return True, "API key looks valid"
148 |
149 |
150 | # Legacy compatibility functions (simplified)
151 | def add_api_key(key_id: str, api_key: str) -> None:
152 |     """Legacy function - redirect to set_api_key."""
153 |     ui.print_info(f"Note: Multi-key system deprecated.
154 |         Setting '{key_id}' as single API key.")
155 |     set_api_key(api_key)
156 |
157 | def list_api_keys() -> list:
158 |     """Legacy function - return single key info."""
159 |     api_key = get_api_key()
160 |     if not api_key:
161 |         return []
162 |
163 |     return [
164 |         {
165 |             "id": "single",
166 |             "masked": mask_api_key(api_key),
167 |             "is_default": "yes"
168 |         }
169 |     ]

```

```

163 | def set_default_api_key(key_id: str) -> None:
164 |     """Legacy function - no-op in single-key system."""
165 |     ui.print_info("Note: Default key setting not needed
166 |         in single-key system.")
167 |
168 | def load_api_key() -> Optional[str]:
169 |     """Legacy function - redirect to get_api_key."""
170 |     return get_api_key()

```

Listing A.6: Modul config.py (lengkap, ASCII-only).

## cli.py

```

1 #!/usr/bin/env python
2
3 import argparse
4 from . import agent, config, llm, ui
5
6 def main():
7     parser = argparse.ArgumentParser(
8         description="Pai Code: Your Single-Shot Agentic
9             AI Coding Companion.",
10        epilog="Use 'pai config set <API_KEY>' to
11            configure. Run 'pai' to start the intelligent
12            agent."
13    )
14    subparsers = parser.add_subparsers(dest='command',
15        help='Available commands')
16
17    # Main agent command (default)
18    parser_auto = subparsers.add_parser('auto', help='
19        Start the single-shot AI agent session.')
20    parser_auto.add_argument('--model', type=str, help='
21        LLM model name (e.g., gemini-2.5-flash-lite)')
22    parser_auto.add_argument('--temperature', type=float,
23        help='LLM sampling temperature (e.g., 0.2)')
24
25    # Simplified config management

```

```

19     parser_config = subparsers.add_parser('config', help=
20         'Manage API key configuration')
21     config_subparsers = parser_config.add_subparsers(dest
22         ='config_cmd', help='Config commands')
23
24     parser_config_set = config_subparsers.add_parser('set'
25         , help='Set API key')
26     parser_config_set.add_argument('api_key', type=str,
27         help='Google Gemini API key')
28
29     parser_config_show = config_subparsers.add_parser('show',
30         help='Show current API key (masked)')
31
32     parser_config_remove = config_subparsers.add_parser('remove',
33         help='Remove stored API key')
34
35     parser_config_validate = config_subparsers.add_parser('validate',
36         help='Validate current API key')
37
38     config_group = parser_config.
39         add_mutually_exclusive_group(required=False)
40     config_group.add_argument('--set', type=str, metavar=
41         'API_KEY', help='Set or update the API key (
42             DEPRECATED)')
43     config_group.add_argument('--show', action='
44         store_true', help='Show the currently configured
45             API key (DEPRECATED)')
46     config_group.add_argument('--remove', action='
47         store_true', help='Remove the stored API key (
48             DEPRECATED)')
49
50     args = parser.parse_args()
51
52     # Handle config commands
53     if args.command == 'config':
54         if args.config_cmd == 'set':
55             config.set_api_key(args.api_key)
56             return

```

```

43         elif args.config_cmd == 'show':
44             config.show_api_key()
45             return
46         elif args.config_cmd == 'remove':
47             config.remove_api_key()
48             return
49         elif args.config_cmd == 'validate':
50             is_valid, message = config.validate_api_key()
51             if is_valid:
52                 ui.print_success(f"[OK] {message}")
53             else:
54                 ui.print_error(f"[ERROR] {message}")
55             return
56         else:
57             parser_config.print_help()
58             return
59
60     # Legacy flags (kept for compatibility)
61     if getattr(args, 'set', None):
62         config.set_api_key(args.set)
63         return
64     if getattr(args, 'show', False):
65         config.show_api_key()
66         return
67     if getattr(args, 'remove', False):
68         config.remove_api_key()
69         return
70     # Default: start agent
71     # Check API key before starting
72     if not config.is_configured():
73         ui.print_error("[ERROR] No API key configured.")
74         ui.print_info("Use 'pai config set <API_KEY>' to
75                     set your Google Gemini API key.")
76
77     # Configure LLM runtime if flags provided
78     model = getattr(args, 'model', None)
79     temperature = getattr(args, 'temperature', None)

```

```

80     if model is not None or temperature is not None:
81         llm.set_runtime_model(model, temperature)
82
83     try:
84         agent.start_interactive_session()
85     except KeyboardInterrupt:
86         ui.print_info("\nSession terminated by user.")
87     except Exception as e:
88         ui.print_error(f"An error occurred during the
89             session: {e}")
90
91     return 1
92
93 if __name__ == "__main__":
94     main()

```

Listing A.7: Modul cli.py (lengkap, ASCII-only).

## ui.py

```

1 # paicode/ui.py
2
3 from rich.console import Console
4 from rich.panel import Panel
5 from rich.syntax import Syntax
6 from rich.theme import Theme
7 from rich.rule import Rule
8 from rich.box import ROUNDED
9 from rich.text import Text
10
11 # Define a custom theme for consistency
12 custom_theme = Theme({
13     "info": "dim default",
14     "success": "bold green",
15     "warning": "yellow",
16     "error": "bold red",
17     "action": "bold bright_blue",
18     "plan": "default",
19     "path": "underline italic bright_blue"

```

```

20 })
21
22 # Create a single console instance to be used across the
23 # application
23 console = Console(theme=custom_theme)
24
25 def print_success(message: str):
26     """Displays a success message with a checkmark icon.
27     """
28     console.print(f"[success][OK] {message}[/success]")
29
30 def print_error(message: str):
31     """Displays an error message with a cross icon."""
32     console.print(f"[error][ERROR] {message}[/error]")
33
34 def print_warning(message: str):
35     """Displays a warning message."""
36     console.print(f"[warning]! {message}[/warning]")
37
38 def print_info(message: str):
39     """Displays an informational message."""
40     console.print(f"[info]i {message}[/info]")
41
42 def print_action(message: str):
43     """Displays an action being performed by the agent.
44     """
45     console.print(f"[action]-> {message}[/action]")
46
47 def display_panel(content: str, title: str, language: str
48 = None):
49     """Displays content within a panel, with optional
50     syntax highlighting."""
51     if language:
52         # Use Syntax for code highlighting
53         display_content = Syntax(content, language, theme
54             ="monokai", line_numbers=True)
55     else:
56         display_content = content

```

```

52
53     console.print(Panel(display_content, title=f"[bold
54         grey50]{title}[/{bold grey50}", border_style="
55         grey50", expand=False))
56
56 def print_rule(title: str):
57     """Displays a horizontal rule with a title."""
58     console.print(Rule(f"[bold]{title}[/{bold]", style="
59         grey50"))

```

Listing A.8: Modul ui.py (lengkap, ASCII-only).

## llm.py

```

1 import os
2 import warnings
3 import time
4
5 # Reduce noisy STDERR logs from gRPC/absl before
6     # importing Google SDKs.
7 # These settings aim to suppress INFO/WARNING/ERROR logs
8     # emitted by native libs
9 # that happen prior to Python log initialization.
10 os.environ.setdefault("GRPC_VERBOSITY", "NONE")
11 os.environ.setdefault("GRPC_LOG_SEVERITY", "ERROR")
12 # Abseil logging (used by some Google native deps). 3 ~
13     # FATAL-only
14 os.environ.setdefault("ABSL_LOGGING_MIN_LOG_LEVEL", "3")
15 # glog compatibility (some builds respect this env var)
16 os.environ.setdefault("GLOG_minloglevel", "3")
17 # Additional environment variables to suppress Google SDK
18     # warnings
19 os.environ.setdefault("GOOGLE_CLOUD_DISABLE_GRPC", "true"
20 )
21 os.environ.setdefault("GRPC_ENABLE_FORK_SUPPORT", "false"
22 )
23
24 # Suppress specific warnings

```

```

19     warnings.filterwarnings("ignore", category=UserWarning,
20                             module="google")
21     warnings.filterwarnings("ignore", message=".*ALTS.*")
22     warnings.filterwarnings("ignore", message=".*log messages
23                               before absl::InitializeLog.*")
24
25
26 DEFAULT_MODEL = os.getenv("PAI_MODEL", "gemini-2.5-flash-
27   lite")
28
29 try:
30     DEFAULT_TEMPERATURE = float(os.getenv(
31         "PAI_TEMPERATURE", "0.3"))
32     # Clamp temperature to safe range
33     if DEFAULT_TEMPERATURE < 0.0:
34         DEFAULT_TEMPERATURE = 0.0
35     elif DEFAULT_TEMPERATURE > 2.0:
36         DEFAULT_TEMPERATURE = 2.0
37
38 except ValueError:
39     DEFAULT_TEMPERATURE = 0.3
40
41
42 # Global model holder
43 model = None
44 _runtime = {
45     "name": None,
46     "temperature": None,
47 }
48
49 def set_runtime_model(model_name: str | None = None,
50                      temperature: float | None = None):
51     """Set the runtime model configuration."""
52     global model, _runtime
53
54     # Update runtime settings
55     if model_name is not None:
56         _runtime["name"] = model_name
57     if temperature is not None:

```

```

52         temperature = max(0.0, min(2.0, temperature))
53         _runtime["temperature"] = temperature
54
55     # Reset model so it gets recreated with new settings
56     # on next use
57     model = None
58
59 # Initialize runtime settings (model will be created when
60 # needed)
61 _runtime = {
62     "name": DEFAULT_MODEL,
63     "temperature": DEFAULT_TEMPERATURE
64 }
65
66
67 def _prepare_runtime() -> bool:
68     """Configure API key and ensure model object exists.
69
70     Returns:
71         bool: True if successful, False otherwise.
72     """
73
74     global model
75
76     # Get single API key
77     api_key = config.get_api_key()
78
79     if not api_key:
80         ui.print_error("Error: No API key configured. Use
81             'pai config set <API_KEY>' .")
82         model = None
83         return False
84
85     try:
86         genai.configure(api_key=api_key)
87         if model is None:
88             # Build model using stored runtime prefs
89             name = _runtime.get("name") or DEFAULT_MODEL
90             temp = _runtime.get("temperature") if
91                 _runtime.get("temperature") is not None

```

```

    else DEFAULT_TEMPERATURE
86     generation_config = {"temperature": temp}
87     model = genai.GenerativeModel(name,
88                                     generation_config=generation_config)
89     return True
90 except Exception as e:
91     ui.print_error(f"Failed to configure API key: {e}
92     ")
93     model = None
94     return False
95
96
97 def _is_rate_limit_error(error: Exception) -> bool:
98     """Detect if an exception is a rate limit error.
99
100    Args:
101        error: The exception to check
102
103    Returns:
104        True if it's a rate limit error, False otherwise
105
106    """
107    error_msg = str(error).lower()
108
109    # Common rate limit indicators
110    rate_limit_keywords = [
111        'rate limit', 'rate_limit', 'ratelimit',
112        'quota', 'quota exceeded',
113        'resource exhausted', 'resourceexhausted',
114        '429', 'too many requests',
115        'limit exceeded', 'requests per minute'
116    ]
117
118
119    return any(keyword in error_msg for keyword in
120              rate_limit_keywords)
121
122
123 def _clean_response_text(text: str) -> str:
124     """Clean markdown artifacts from LLM response.
125
126    Args:
127

```

```

120     text: Raw response text from LLM
121
122     Returns:
123         Cleaned text without markdown code blocks
124     """
125
126     cleaned_text = text.strip()
127
128     # Remove all common markdown code block patterns
129     code_block_prefixes = [
130         "```python", "```html", "```css", "```javascript"
131         , "```js",
132         "```typescript", "```ts", "```json", "```yaml", "```
133         "yml",
134         "```bash", "```sh", "```diff", "```xml", "```sql"
135         ,
136         "```java", "```cpp", "```c", "```go", "```rust",
137         "```ruby",
138         "```php", "```markdown", "```md", "```text", "```
139         "txt", "```"
140     ]
141
142     for prefix in code_block_prefixes:
143         if cleaned_text.startswith(prefix):
144             cleaned_text = cleaned_text[len(prefix):].
145             strip()
146             break
147
148     # Remove trailing code block markers
149     if cleaned_text.endswith("```"):
150         cleaned_text = cleaned_text[:-len("```")].strip()
151
152     # Remove any remaining language tags at the start
153     lines = cleaned_text.split('\n')
154     if lines and len(lines[0].strip()) < 20 and lines[0].
155         strip().lower() in [
156             'html', 'css', 'javascript', 'js', 'python',
157             'json', 'yaml',

```

```

149         'bash', 'sh', 'diff', 'xml', 'sql', 'java', 'cpp',
150         , 'c', 'go',
151     'rust', 'ruby', 'php', 'markdown', 'md', 'text',
152     'txt', 'on'
153 ]:
154     cleaned_text = '\n'.join(lines[1:]).strip()
155
156     return cleaned_text
157
158 def generate_text(prompt: str, call_purpose: str = "thinking") -> str:
159     """
160     Generate text with single API key - optimized for 2-
161     call system.
162
163     Args:
164         prompt: The prompt to send to the LLM
165         call_purpose: Purpose of the call for logging (e.
166             g., "planning", "execution")
167
168     Returns:
169         The cleaned response text, or empty string if
170         failed
171     """
172
173     global model
174
175     # Ensure model is configured
176     if model is None:
177         if not _prepare_runtime():
178             return ""
179
180     try:
181         # Show status with purpose
182         status_msg = f"[bold yellow]Agent {call_purpose
183             }..."
184
185         with ui.console.status(status_msg, spinner="dots"
186             ):

```

```

179         response = model.generate_content(prompt)
180
181     # Success! Clean and return the response
182     cleaned_text = _clean_response_text(response.text
183                                         )
184
185     # Log token usage if available (for optimization)
186     if hasattr(response, 'usage_metadata'):
187         usage = response.usage_metadata
188         ui.print_info(f"Tokens: {usage.
189                     prompt_token_count} -> {usage.
190                     candidates_token_count}")
191
192     return cleaned_text
193
194 except Exception as e:
195     is_rate_limit = _is_rate_limit_error(e)
196
197     if is_rate_limit:
198         ui.print_error("[ERROR] Rate limit reached.
199                     Please wait a few minutes before trying
200                     again.")
201         ui.print_info("Consider using a different API
202                     key if available.")
203     else:
204         ui.print_error(f"[ERROR] LLM API error: {e}")
205
206     return ""
207
208 def test_api_connection() -> bool:
209     """Test if API connection works."""
210     test_response = generate_text("Say 'Hello' if you can
211                                 hear me.", "connection test")
212
213     return len(test_response) > 0

```

Listing A.9: Modul `llm.py` (lengkap, ASCII-only).

### `__init__.py`

```
1 """Pai Code package.
2
3 This package provides a command-line based agentic AI for
4     software development.
5 """
6
7 __all__ = [
8     # Public modules
9     "agent",
10    "cli",
11    "config",
12    "llm",
13    "ui",
14    "workspace",
15 ]
16 __version__ = "0.1.0"
```

Listing A.10: Modul `__init__.py` (paicode package), ASCII-only.

### `requirements.txt`

```
1 google-generativeai>=0.5.4
2 python-dotenv>=1.0.1
3 rich>=13.7.1
4 Pygments>=2.16.0
5 prompt_toolkit>=3.0.43
```

Listing A.11: File requirements.txt

### `setup.py`

```
1 from setuptools import setup
2
3 if __name__ == "__main__":
4     setup()
```

---

Listing A.12: File setup.py (konfigurasi setuptools)

### setup.cfg

```
1 [metadata]
2 name = pai-code
3 version = 0.1.0
4 description = A command-line based agentic AI for
5               software development.
6 long_description = file: README.md
7 long_description_content_type = text/markdown
8 author = gtkrshnaaa
9 author_email = gtkrshnaaa@email.com
10 license = MIT
11 license_files = LICENSE
12
13 [options]
14 packages = find:
15 python_requires = >=3.10
16 install_requires =
17     google-generativeai>=0.5.4
18     python-dotenv>=1.0.1
19     rich>=13.7.1
20     Pygments>=2.16.0
21 include_package_data = True
22
23 [options.packages.find]
24 where =
25
26 [options.entry_points]
27 console_scripts =
28     pai = paicode.cli:main
```

Listing A.13: File setup.cfg (metadata dan konfigurasi paket)

### pyproject.toml

```
1 # pyproject.toml
2
3 [build-system]
4 requires = ["setuptools>=61", "wheel"]
5 build-backend = "setuptools.build_meta"
```

Listing A.14: File pyproject.toml (konfigurasi build system)

## makefile

```

24     -not -path '.*.egg-info/*' \
25     -not -path './z_project_list/*' \
26     -not -name ".gitkeep" \
27   | sort); do \
28     echo "==== $$f ===" >> z_project_list/listing.txt; \
29     cat $$f >> z_project_list/listing.txt; \
30     echo "\n" >> z_project_list/listing.txt; \
31   done
32 @echo "Export complete."
33
34 venv-activate:
35 @echo "To activate the virtual environment, run:"
36 @echo "  source .venv/bin/activate"
37
38 setup: install install-cli
39 @echo "Pai CLI installed. Ensure $$HOME/.local/bin is
40       in your PATH, then run: pai"
41
42 install-cli:
43 @mkdir -p $(HOME)/.local/bin
44 @echo "Installing launcher to $(HOME)/.local/bin/pai"
45 @echo '#!/usr/bin/env bash' > $(HOME)/.local/bin/pai
46 @echo '# Suppress noisy gRPC/absl logs' >> $(HOME)/.
47       local/bin/pai
48 @echo 'export GRPC_VERBOSITY="NONE"' >> $(HOME)/.local/
49       bin/pai
50 @echo 'export GRPC_LOG_SEVERITY="ERROR"' >> $(HOME)/.
51       local/bin/pai
52 @echo 'export ABSL_LOGGING_MIN_LOG_LEVEL="3"' >> $(HOME)
53       /.local/bin/pai
54 @echo 'export GLOG_minloglevel="3"' >> $(HOME)/.local/
55       bin/pai
56 @echo 'export GOOGLE_CLOUD_DISABLE_GRPC="true"' >> $(
57       HOME)/.local/bin/pai
58 @echo 'export GRPC_ENABLE_FORK_SUPPORT="false"' >> $((
59       HOME)/.local/bin/pai
60 @echo 'SCRIPT_DIR="$$($ cd $$($dirname $$${BASH_SOURCE
61       [0]}))" && pwd)"' >> $(HOME)/.local/bin/pai

```

```

53  @echo 'APPDIR="$(shell pwd)"' >> $(HOME)/.local/bin/pai
54  @echo 'VENVDIR="$$APPDIR/.venv"' >> $(HOME)/.local/bin/
      pai
55  @echo 'PY="$$VENVDIR/bin/python"' >> $(HOME)/.local/bin
      /pai
56  @echo '# Redirect stderr to suppress remaining warnings
      ' >> $(HOME)/.local/bin/pai
57  @echo 'if [ -x "$$VENVDIR/bin/pai" ]; then' >> $(HOME)
      /.local/bin/pai
58  @echo '  exec $$VENVDIR/bin/pai $$@' 2>/dev/null' >>
      $(HOME)/.local/bin/pai
59  @echo 'elif [ -x "$$PY" ]; then' >> $(HOME)/.local/bin/
      pai
60  @echo '  exec $$PY -m paicode.cli $$@' 2>/dev/null'
      >> $(HOME)/.local/bin/pai
61  @echo 'else' >> $(HOME)/.local/bin/pai
62  @echo '  exec python3 -m paicode.cli $$@' 2>/dev/null'
      >> $(HOME)/.local/bin/pai
63  @echo 'fi' >> $(HOME)/.local/bin/pai
64  @chmod +x $(HOME)/.local/bin/pai
65  @# Ensure ~/.local/bin is in PATH (append to ~/.bashrc
      if missing)
66  @if [ -f $(HOME)/.bashrc ]; then \
67    grep -qxF 'export PATH="$$HOME/.local/bin:$$PATH"' $(
      HOME)/.bashrc || printf '\n# Added by pai install-
      cli\nexport PATH="$$HOME/.local/bin:$$PATH"\n' >>
      $(HOME)/.bashrc; \
68  fi
69  @echo "Ensured PATH includes $$HOME/.local/bin in
      $$HOME/.bashrc. Run: 'source $$HOME/.bashrc' or open
      a new terminal."
70  @echo "Done. Ensure $(HOME)/.local/bin is in your PATH.
      Try running: pai --help"
71
72  uninstall-cli:
73  @rm -f $(HOME)/.local/bin/pai
74  @# Remove PATH line added by install-cli (safe if
      absent)

```

```
75 |     @sed -i '/^# Added by pai install-cli$/d' $(HOME)/.bashrc || true  
76 |     @sed -i '/^export PATH="\$HOME/.local/bin:\$PATH"/d' $(HOME)/.bashrc || true  
77 |     @echo "Launcher removed: $(HOME)/.local/bin/pai"
```

Listing A.15: File makefile (task automation untuk development dan deployment)

---

## Bibliografi

- [1] Rohan Anil, Yuntao Bai, Xinyun Chen, et al. Gemini: A family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.
- [2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, et al. Language models are few-shot learners. In *NeurIPS*, 2020.
- [3] Paul Gauthier. Aider: Ai pair programming in your terminal. <https://github.com/paul-gauthier/aider>, 2023.
- [4] GitHub. Github copilot: Your ai pair programmer. <https://github.com/features/copilot>, 2021.
- [5] Guohao Li et al. Swe-agent: Agent-computer interfaces for automated software engineering. *arXiv preprint arXiv:2405.15793*, 2024.
- [6] Meta AI. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.
- [7] OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.
- [8] OpenDevin Team. Opendevin: An open source autonomous ai software engineer. <https://github.com/OpenDevin/OpenDevin>, 2024.
- [9] Timo Schick, Jane Sch"utz, Jane Dwivedi-Yu, et al. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*, 2023.
- [10] Hugo Touvron, Thibaut Lavril, Gautier Izacard, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.
- [11] Shunyu Yao, Jeffrey Zhao, Dian Yu, et al. React: Synergizing reasoning and acting in language models. In *ICLR*, 2023.