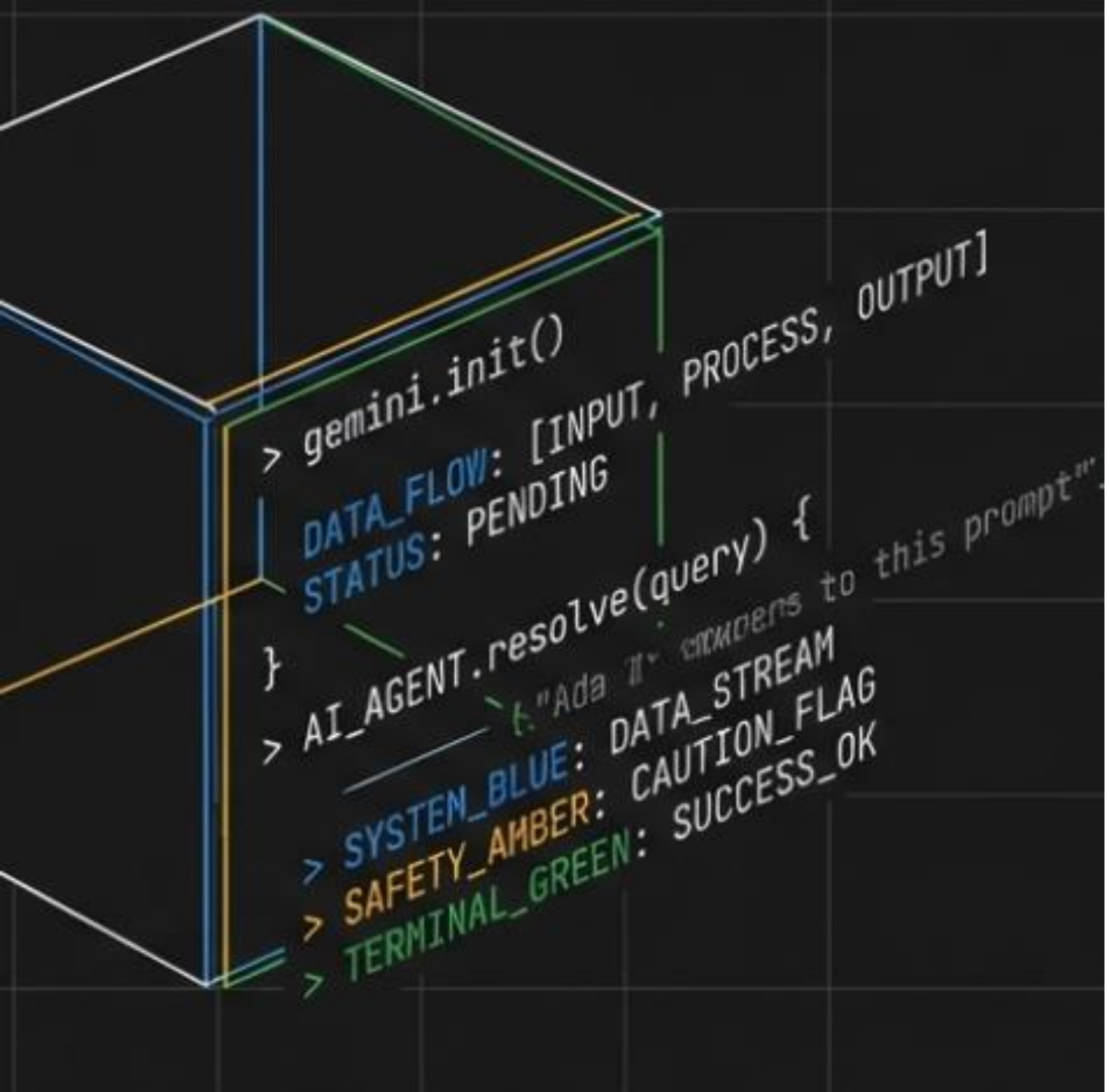


Gemini CLI: Conceptual Architecture

GROUP 7

CMPE 322: SW ARCHITECTURE

HTTPS://YOUTU.BE/VWGV7GMOVSM

Names and Roles

Dylan: Group Leader, Abstract, Introduction & Overview, System Functionality and Goals
Zonghan: Use Case 1, Use Case 2
Rowan Muhammed: Diagrams for Data Flow and Process Control
Omar: Presentation, Concurrency Model, Alternative Architectural Decisions, External Interfaces, AI Report
Keven: Presentation, Component Interactions, Division of Responsibility, Architectural Styles
Rowan: Control Flow & Data Flow

Terminal-First AI Agent

Natural Language Interface

Local Execution

Safety

The Derivation Process

Exploration

Analyzed packages/core & GEMINI.md specs.

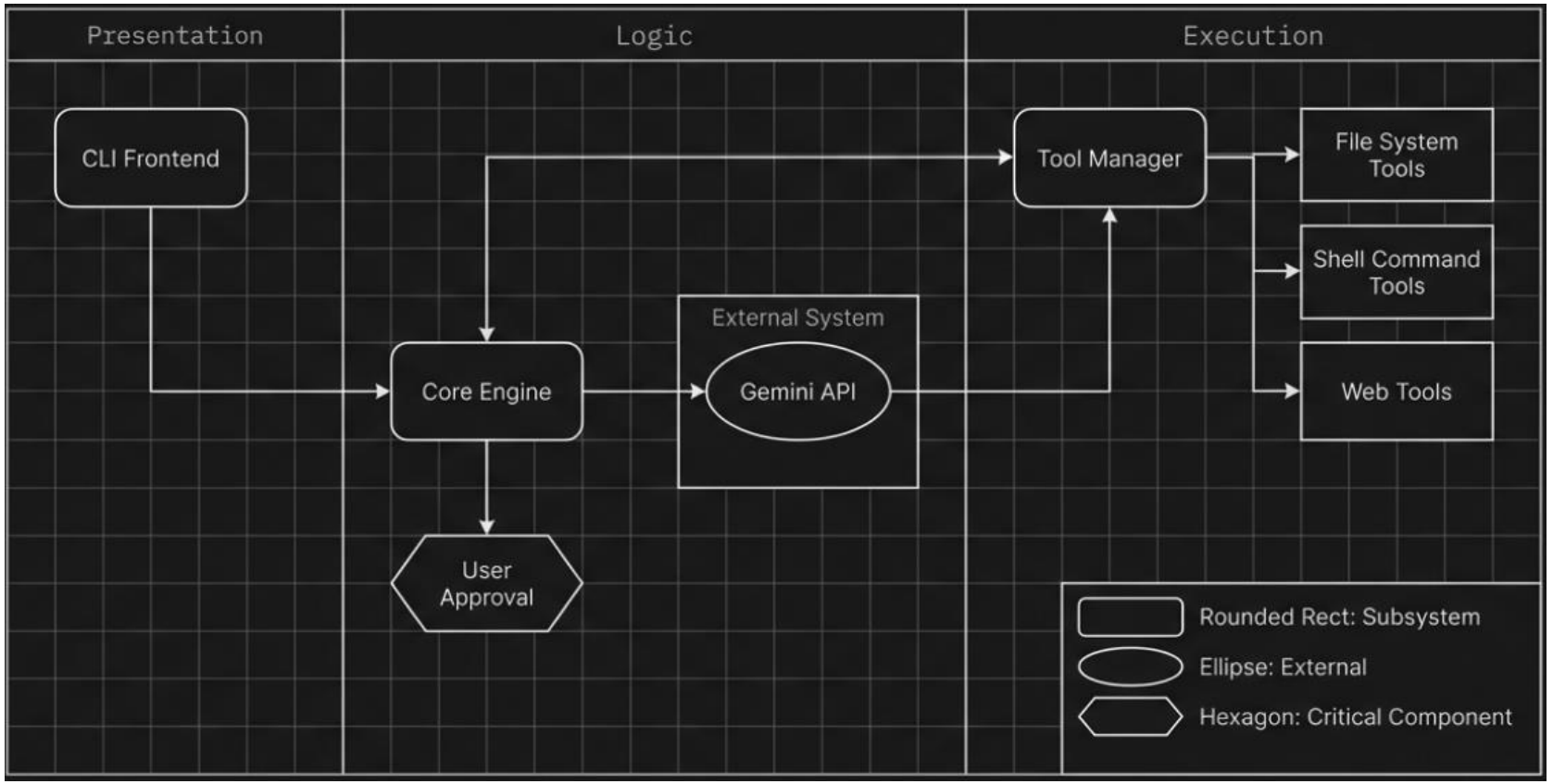
Dynamic Mapping

Created Sequence Diagrams for "Read" vs. "Write" use cases.

Static Synthesis

Derived the Box-and-Arrow diagram from the components identified in step 2.

High-Level Architecture (Box-and-Arrow)



- CLI (Frontend): Presentation only. Ink/React based.
- Core (Backend): The orchestration brain.
- Tools (MCP): The arms and legs (File system, Web).

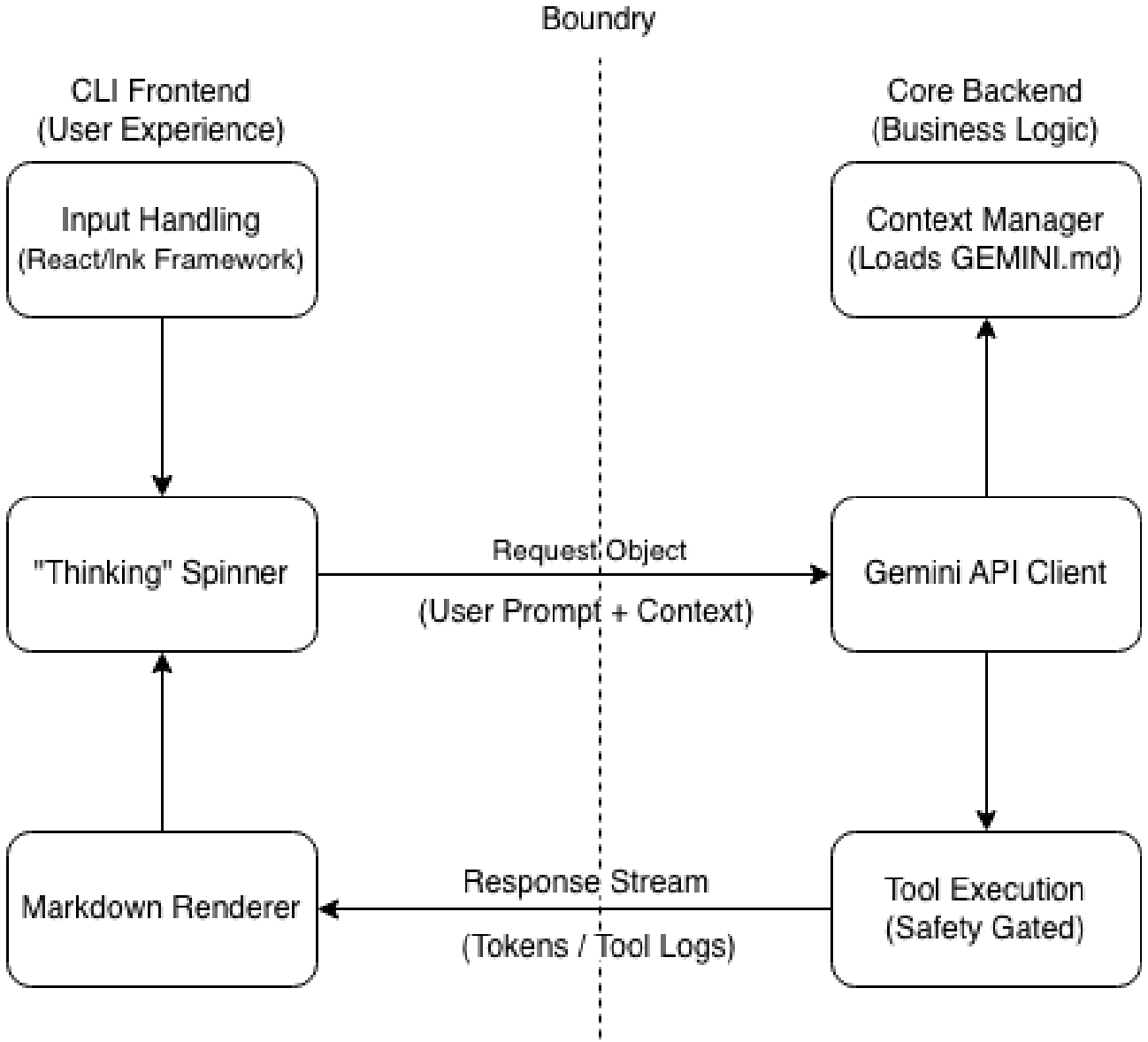
Rationale for Interactions

- Why split CLI/Core?

Responsiveness. The CLI renders spinners/streaming text while the Core waits for the slow API.

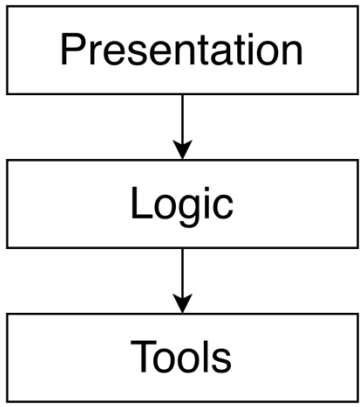
- Why separate Tools?

Safety. By abstracting tools behind a registry, we can inject a "Safety Gate" before execution.



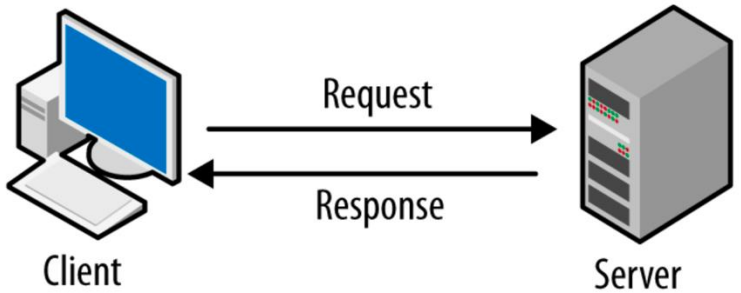
Architectural Styles

LAYERED ARCHITECTURE



Strict separation of concerns.
Frontend never touches Tools directly

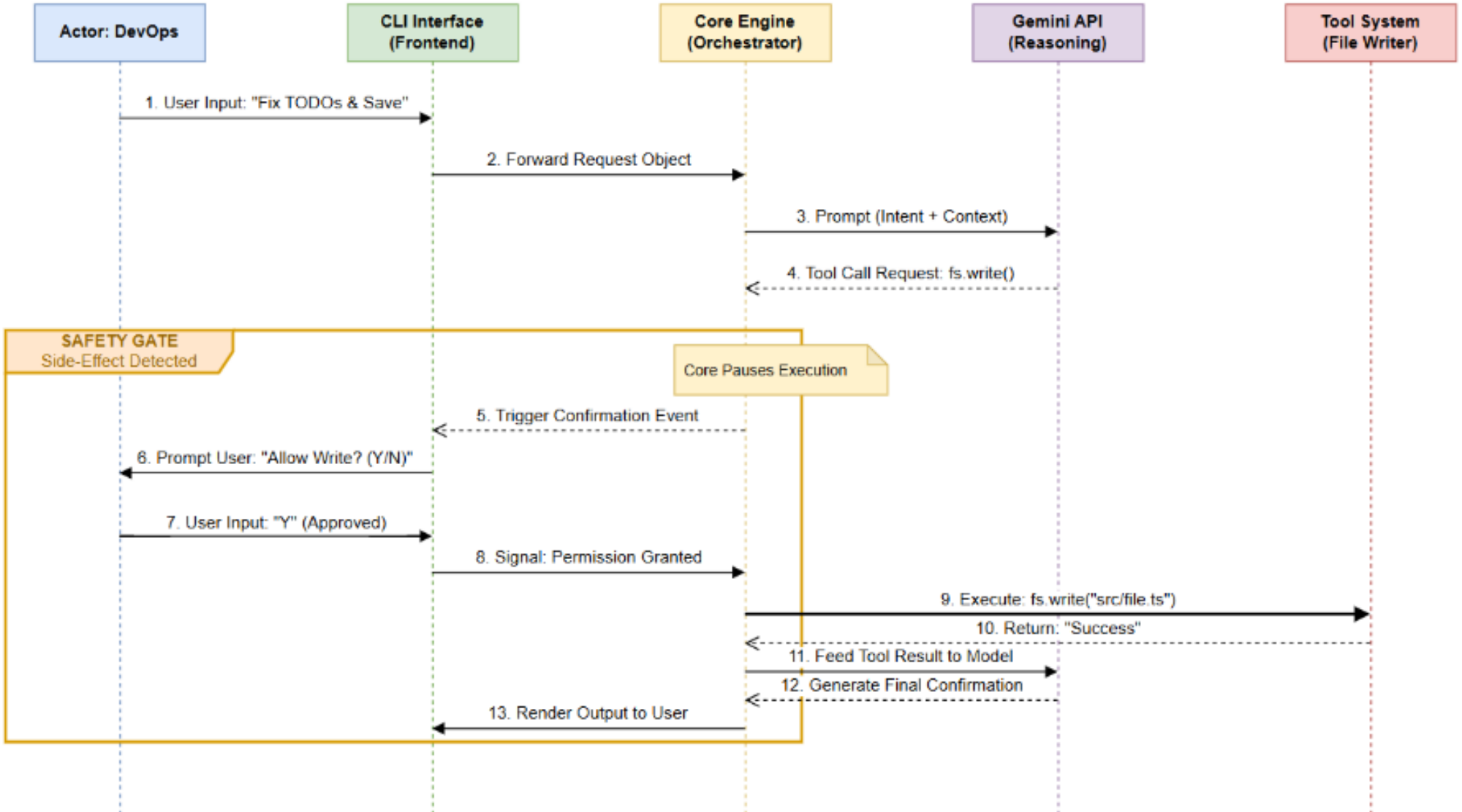
CLIENT-SERVER STYLE



Local application (CLI + Core) act as Client.

External entities (Gemini API, MCP tools, etc.) act as Server

Sequence Diagram



Alternatives Consideration

❌ Alternative 1: Sandbox (Claude Code)	✅ Alternative 2: Host-Based (Gemini CLI)
Higher friction when using and can't access local git configurations and linters easily.	Data sovereignty protected. It is suitable for enterprise users, as they require code to stay local. Only the the inference tokens are transmitted.

Lessons Learned

The Context Fallacy

Infinite context window will never be enough, need Hierarchical Loading to solve the issue

To avoid “Lost-In-The-Middle” issue

The Alert Fatigue

Frequent approval alert is annoying, and users might choose yes blindly

Use tiered rules to reduce friction

Limitation of Discovery

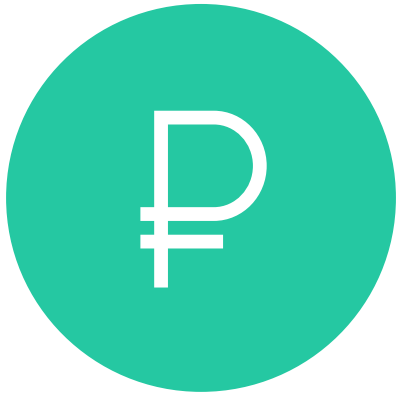
Static Analysis Only

- The discoveries are only from documentations and code files, no runtime tests conducted

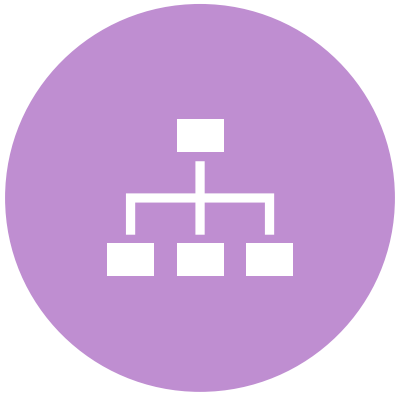
Conceptual & Concrete

- Concrete structure analysis might reveal more hidden dependencies that we missed

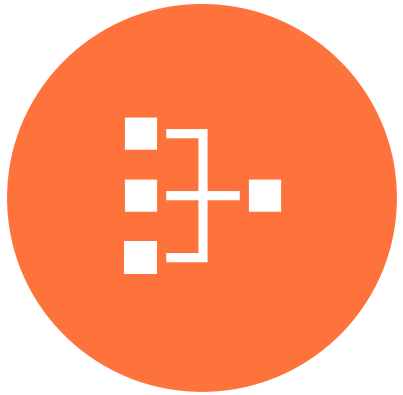
AI Teammates: Selection & Process



Exploration: analyzed packages/core & GEMINI.Md specs.

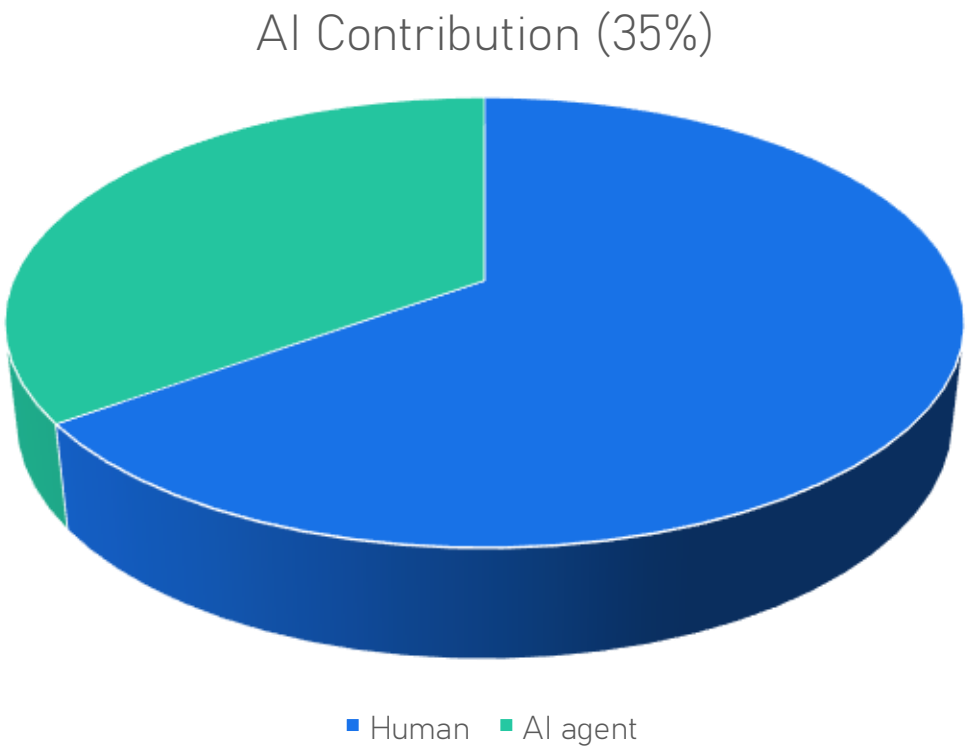


Dynamic mapping: created sequence diagrams for "read" vs. "Write" use cases.



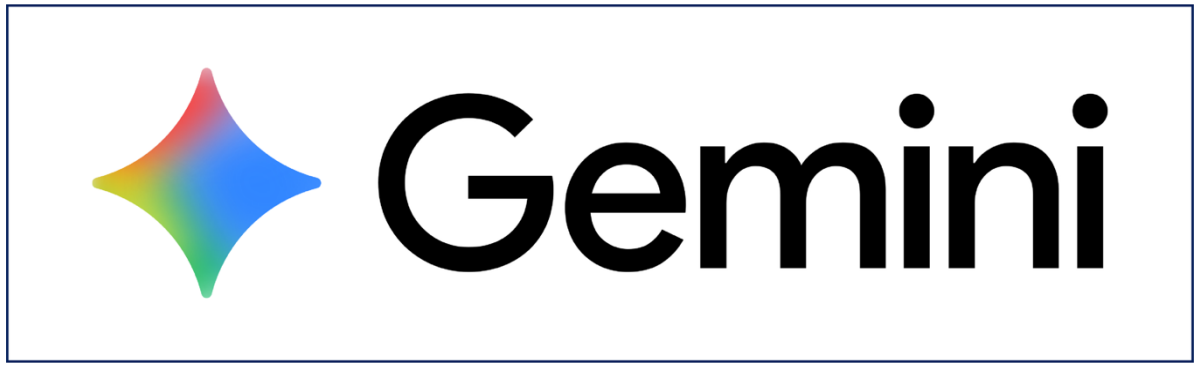
Static synthesis: derived the box-and-arrow diagram from the components identified in step 2.

AI Teammates: Quality & Impact



- Contributed in “Alternatives” & “External Interface” section
- Each claim is manually verified

Conclusion



- Modular Monolith
- Microkernel via MCP (Model Context Protocol)
- Safety as part of the architecture, not just a feature
- Tradeoffs of prioritizing local processes
- Team website: https://omarafify7.github.io/Software-Architecture-Class-Project/