Conceptual Architecture Analysis:

Gemini CLI

Group 9
The Unemployed

Video Link

Team Members

Kesiya Jacob
Group Leader - Abstract, Intro/ Overview, control flow expert

Gabrielle Garey
Presenter - Overall architecture, system functionality, subsystems

Sophia Bushareb
Symptom evolution, use cases

Grisha Tyukin
Concurrency, AI collaboration, subsystems

Ryan Walsh
Presenter - Sequence diagrams, conclusions, lessons learnt, AI collaboration

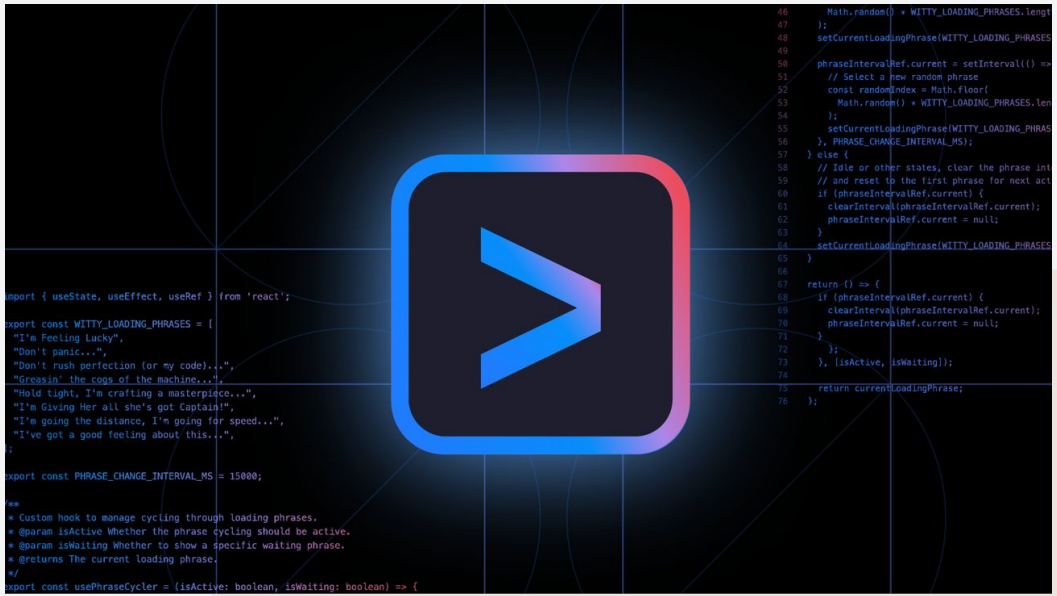
Calvin Wong
Data dictionary, naming conventions

Gemini CLI

- Open source AI agent
- Reason and act (ReAct loop)
- Access Gemini in your coding workflow

Key Features:

- Local system integration,
- automation / scripting
- Agentic capabilities



Derivation Process

Research: class notes, online documentation

Identify key features of Gemini CLI and design use cases

Create separate parts within pairs.

Research the subsystems that make up Gemini CLI

Sequence Diagrams

Create Box-and-Arrows Diagram

Dictionary & Naming Conventions

- **Architecture Styles:** Software styles that describe the behaviour and relationships of components in an application.
- **MCP:** Model Context Protocol, a standardized way for a language model to receive, interpret, and use contextual information from a system or application.
- **Server:** A computer that provides services, data, or resources for clients over a network.
- **NodeJS:** Open source Javascript runtime environment that allows Javascript to run outside of the browser.
- **LLM:** Large Language Model, generally a way to refer to AI.
- **API:** Application programming interface, a set of functions and procedures allowing the creation of applications that access the features or data of an operating system, application, or other service.
- **GUI:** Graphic user interface, a system that allows users to interact with digital devices through visual elements like icons, windows, menus, and pointers, rather than by typing-text based commands.
- **CLI:** Command line interface, terminal through which the user interacts with the program at a lower level. Alternative to GUI.
- **Roadmap:** A project board linked on the Github that visualizes the weekly division of tasks as a byproduct of the developer meetings hosted on the Discord server. Tasks are organized into categories, such as new features, improvements, and backlog items.
- **Github:** Developer platform allowing developers to store, manage, and share code. Provides access control, bug tracking, feature requests, and other project resources.

- **GUI:** Graphic User Interface
- **LLM:** Large Language Model
- **AI:** Artificial intelligence
- **API:** Application Programming Interface
- **OS:** Operating System
- **CLI:** Command line interface

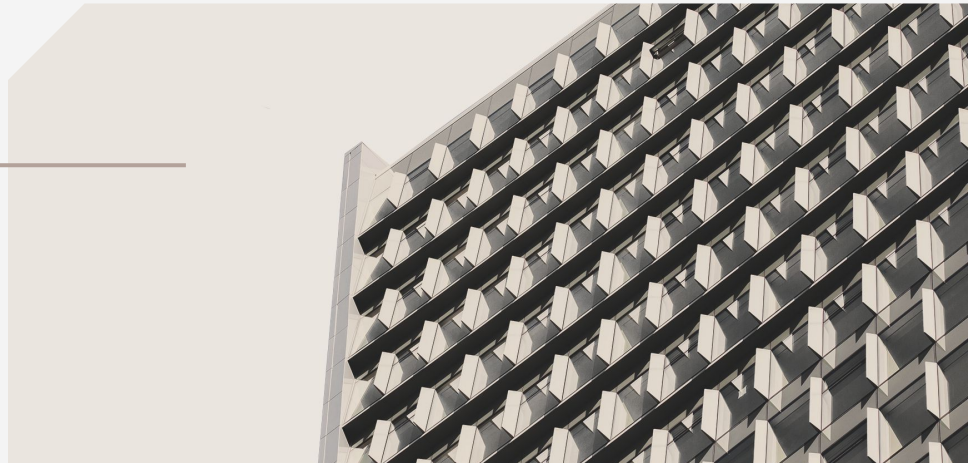
ARCHITECTURE

Main Architecture

Client-Server

Additional Styles

Pipe-and-Filter

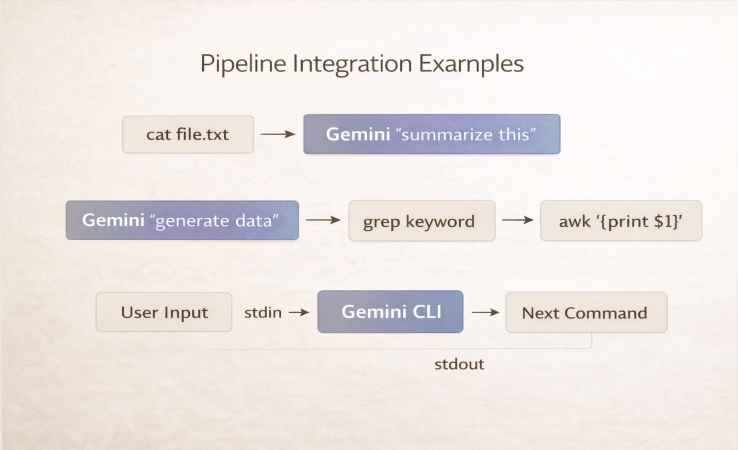


Client-Server Style

- Two Main Packages
  - client: **packages/cli**
  - server: **packages/core**

- Benefits
  - Scalability & Maintainability
- Challenges
  - Single point of failure
  - Added process time

PIPE-AND-FILTER STYLE



- Non-interactive CLI mode
  - Integrate shell scripts by accepting standard input
  - Chain output to other tools
- Benefits
  - Reusability of components
- Challenges
  - Unidirectional flow over bidirectional conversation model

Performance Criticality & Evolution

Interactive Mode

Shell, Agent, Execution

- **AI Chat**
  - Increased Out of Memory crashes
- **Scalability Concerns**
  - Actively being worked on
- **Performance**
  - Roadmap prioritizes performance tuning + user experience issues
  - Working on large input token estimation, reducing out of memory crashes
  - Some user reports on buggy output for diverse workflows

SUBSYSTEMS OVERVIEW

Prompt Construction:

- Creates structured prompts to send to the API client for transmission to Gemini API
- Manages token limits and provides context to Agent

API Client:

- Communicates with Google Gemini API
- Communicates with prompt construction module, tool registry, tool execution module

Tool Registration and Execution Logic:

- Stores the resources that extend Gemini CLI capabilities (built in tools + functions)
- ToolRegistry and discovery mechanisms: command based and MCP-based

SUBSYSTEMS OVERVIEW - cont.

State management:

- Maintain conversation and session history, ensuring multi-turn interactions remain ready to resume after an interruption
- Feeds history to the prompt construction module to maintain continuous, multistream workflows. User can switch directories without losing progress.

Server Configurations:

- Responsible for backend behaviors, including endpoint definitions, security policies, and management of environment variables
- Connects to MCP servers to enable custom tools and allow administrators to enforce system level overrides.

Input Processing :

- Entry point within the CLI package, converting raw user prompts, commands, and flags into structured requests that the Core package can process
- Ensures model output and API responses are formatted for the user

SUBSYSTEMS OVERVIEW - cont.

Display Rendering:

- Uses React and Ink to render terminal Interface
- Integrates with user themes to display user commands and API responses smoothly

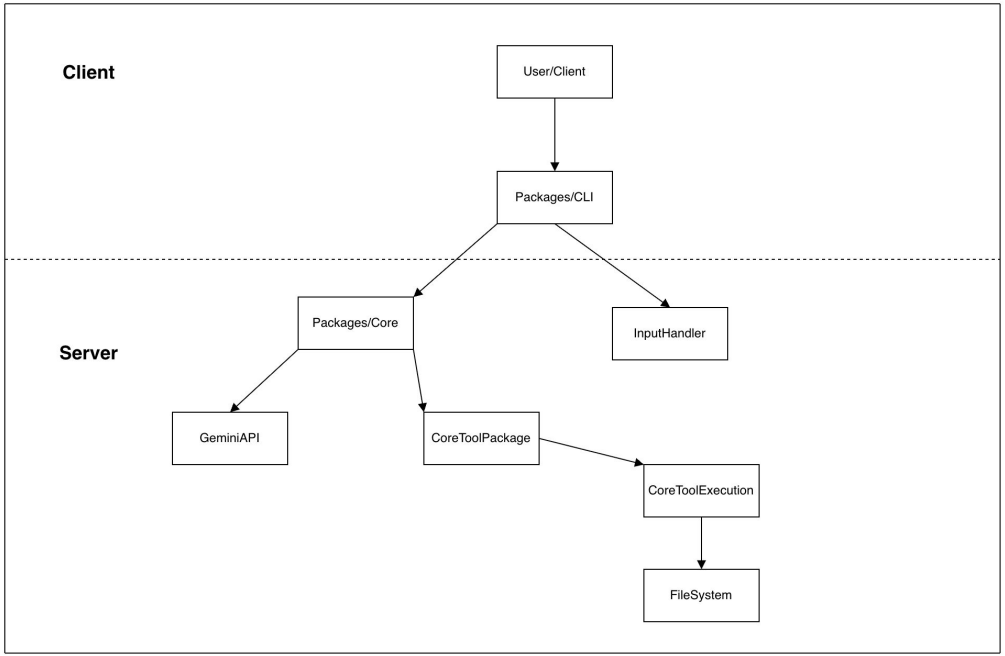
CLI configurations:

- Applies local options that impact how the user interacts with the CLI and how interface is visually presented
- Handles CLI specific settings locally, passing only the necessary parameters to the Core for backend processing

System Tools:

- File system management: enables secure reading, writing, and directory listing
- Web integrations: provides outbound HTTP capabilities, allowing system to fetch URLS, interact with external APIs, or download web content
- Shell execution: ability to run OS-level commands and capture resulting streams

DATA FLOW



User → GUI

1) **GUI** → Packages/CLI,
2) (1) → **Packages/core** (←→ GeminiAPI or CoreToolPackage)

2.1) ToolExecution → Filesytem

3) (2) → **Input handler**

CONCURRENCY

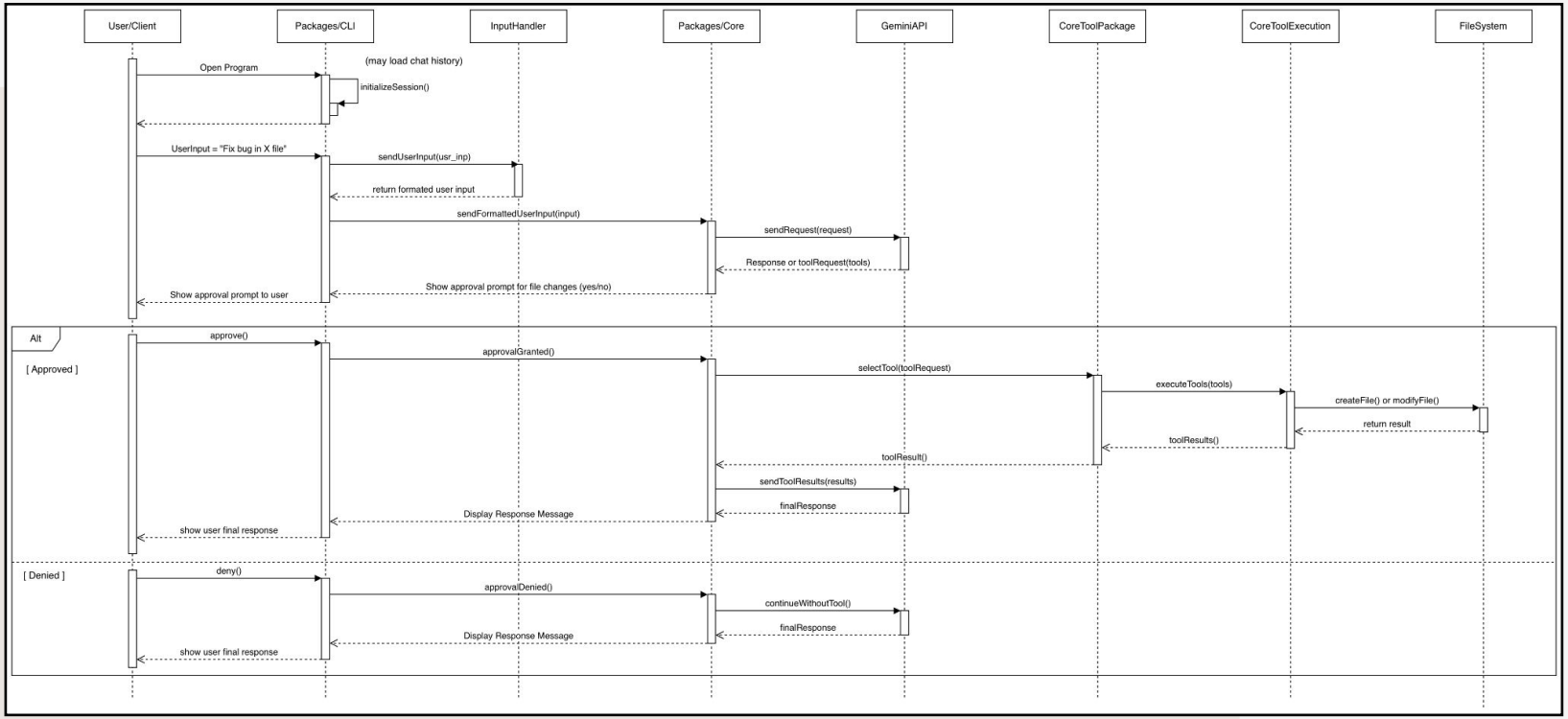
- **Asynchronous Node.js architecture:**
  - Relies on the Node.js event loop to avoid freezing UI
- **Response and Streaming**
  - AI responses split to prevent terminal lag
- **Coordinated workflow**
  - State management module allows for multiple continuous streams of execution

MAIN PROCESS

Display rendering Module
Tool Execution logic

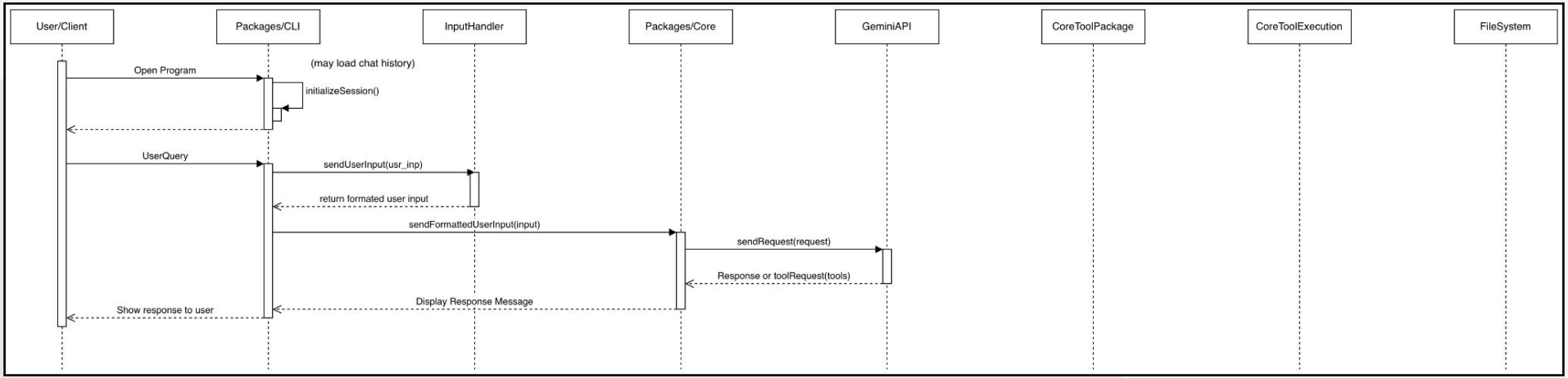
Sequence Diagram 1: Interactive

Use Case 1: A user types a prompt requesting file creation or modification



Sequence Diagram 2: Chat

Use Case 2: A user asks Gemini for an explanation about their code



LESSONS LEARNED

Clear documentation makes analysis possible

Clear and unambiguous official docs & third-party analyses are vital for understanding a system

Context-Dependent Architectural Styles must be Considered

A system can exhibit multiple styles depending on its usage mode, analysis must consider different constraints & perspectives

Clear component roles make architecture defensible

Defining what each part is responsible for makes the final architecture choice much easier to justify.

Our AI Teammate: **perplexity**

Tasks Assigned

- Flesh out ideas on architectural style and subsystems
- Summarize and find sources

Interaction Protocol & Workflow

- Collaborate within partners to craft queries
- Use sero shot and 1-shot prompts
- Refine AI output by asking for specifics

AI agent
January 2026 version

Quantitative impact: 13% of final deliverable

Validation & Quality Control

- Ask perplexity for sources
- Fact-check & cross-reference source material

Human-AI Team Dynamic

- Easy to collaborate and get efficient work done
- Importance of verifying AI-generated ideas

Conclusion

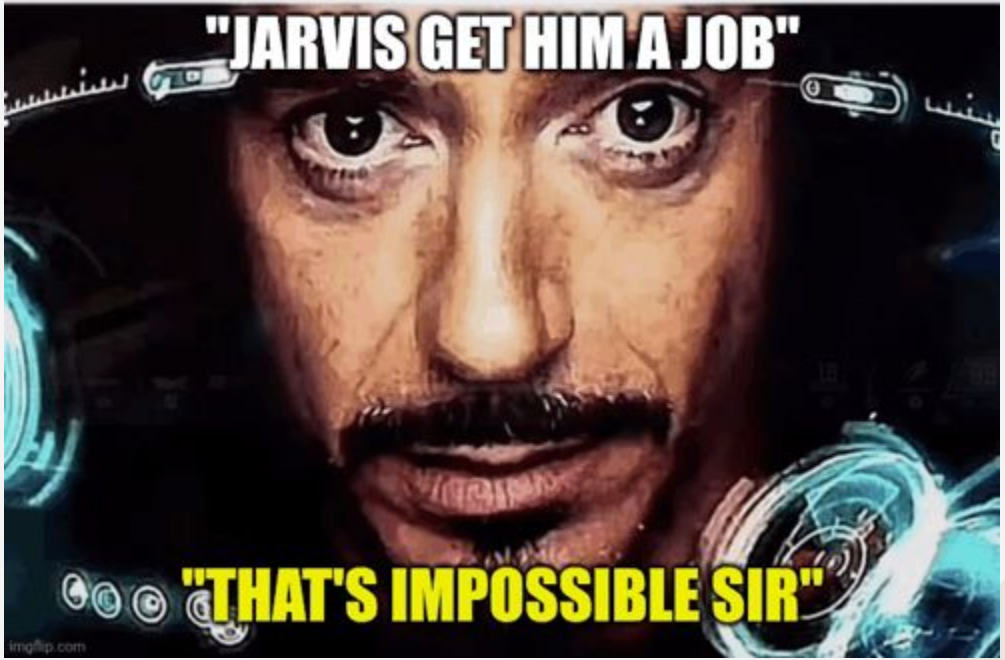
REFERENCES

- AI TL;DR. (2025, December 4). *Gemini CLI - The command line agent from Google.* YouTube. https://www.youtube.com/watch?v=QzJufbGhTeI
- Alateras, J. (2025, July 8). *Unpacking the Gemini CLI: A High-Level Architectural Overview*. Medium. https://medium.com/@jalateras/unpacking-the-gemini-cli-a-high-level-architectural-overview-99212f6780e7
- cz. (2025, June 28). *Gemini CLI Project Architecture Analysis*. DEV. https://dev.to/czmilo/gemini-cli-project-architecture-analysis-3onn
- Dutt, A. (2025, June 27). *Gemini CLI: A Guide With Practical Examples*. DataCamp. https://www.datacamp.com/tutorial/gemini-cli

REFERENCES - cont.

- *Gemini CLI documentation*. (2026). Gemini CLI. https://geminicli.com/docs/ *Gemini CLI Project Architecture Analysis*. (2026, February 6). AI Coding Tools Docs. https://aicodingtools.blog/en/gemini-cli/architecture-analysis
- google-gemini. (2025). *Gemini CLI Architecture Overview*. GitHub. https://github.com/google-gemini/gemini-cli/blob/main/docs/architecture.md
- myyorku2002. (2025, July 15). *Diving Deep into the Gemini CLI Architecture: A Source Code Analysis - BI Practice*. BI Practice. https://www.bipractice.ca/diving-deep-into-the-gemini-cli-architecture-a-source-code-analysis/*MCP servers with the Gemini CLI*. Gemini CLI. (n.d.). https://geminicli.com/docs/tools/mcp-server/

Thanks for watching!



- The Unemployed*