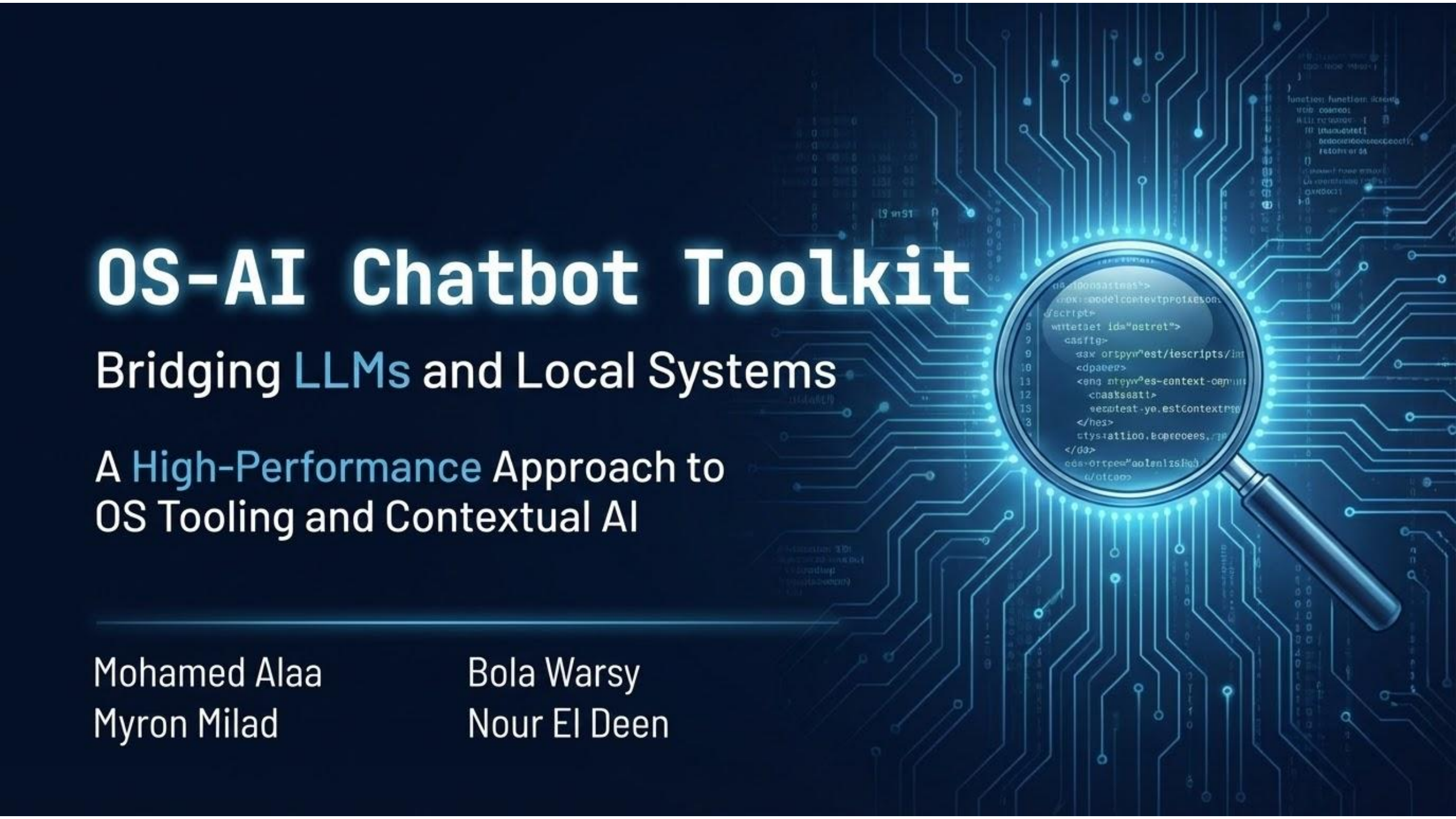
OS-AI Chatbot Toolkit

Bridging LLMs and Local Systems

A High-Performance Approach to OS Tooling and Contextual AI

Mohamed Alaa
Bola Warsy
Myron Milad
Nour El Deen



Core Functionality and User Interface

Performance Foundation

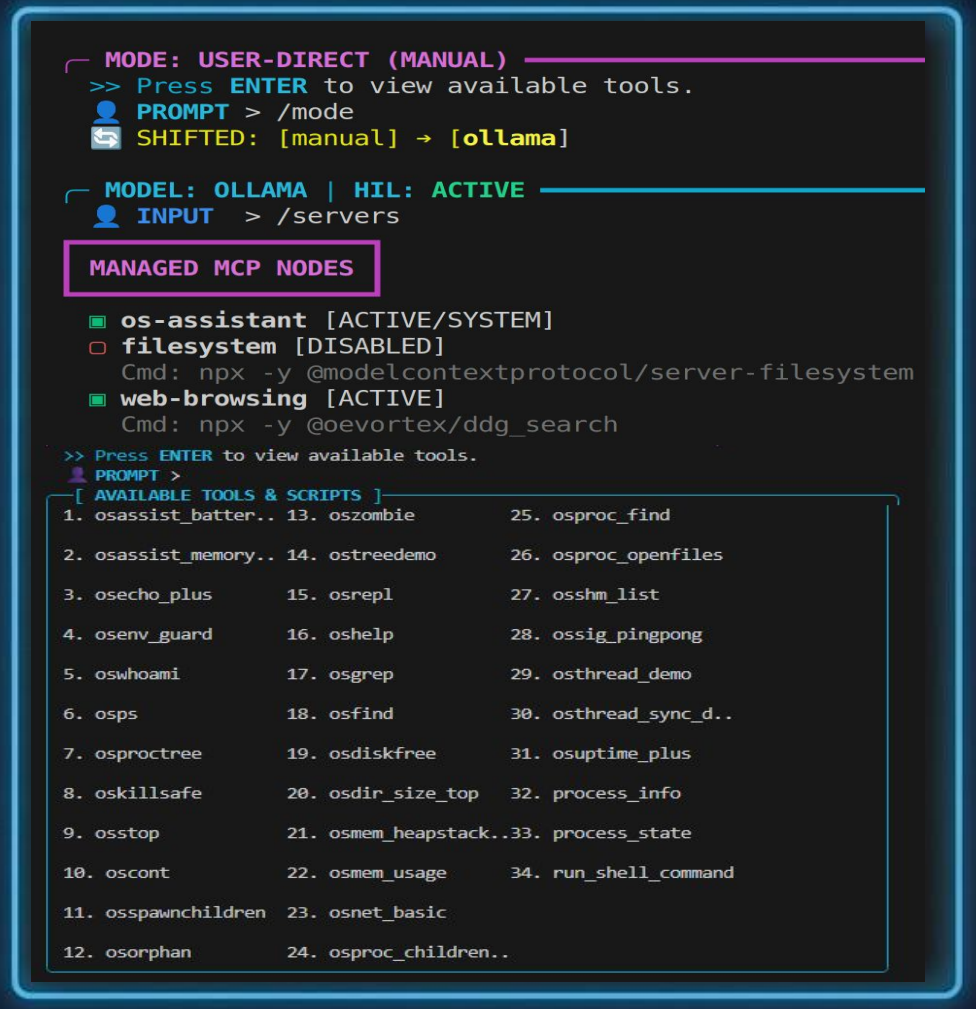
Written in modern C++17 to ensure minimal latency and a low resource footprint—critical for responsive local AI assistance.

Core Function

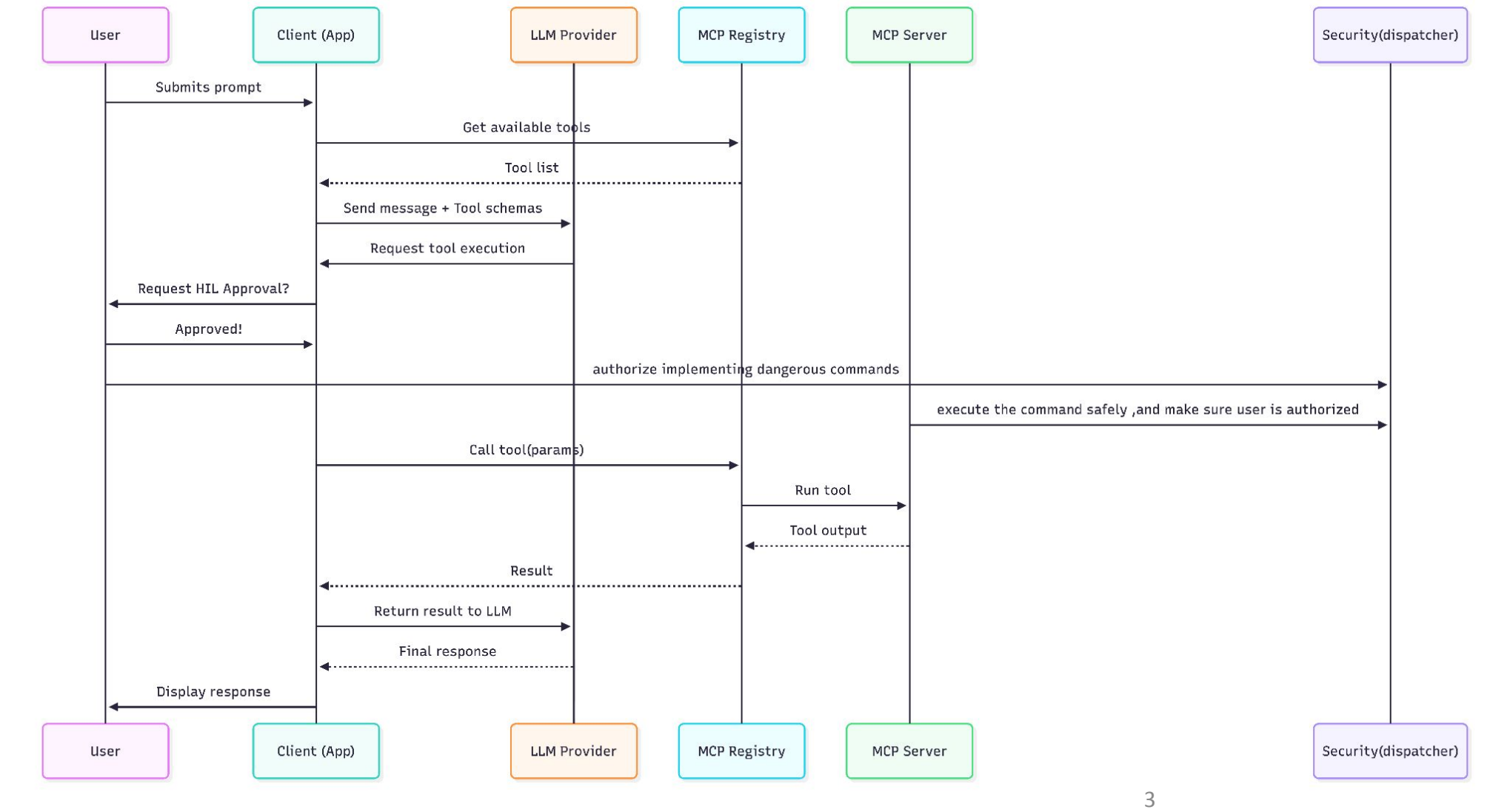
Acts as a secure, high-speed conduit between powerful LLMs and the local system environment.

>_Modern TUI

A beautiful terminal interface designed for developer experience, offering colorized, interactive, and efficient command-line interaction.



System Architecture



The Model Context Protocol (MCP)

MCP Registry

Maintains a list of available tools and their schemas, enabling automated tool discovery by the AI.

MCP Server

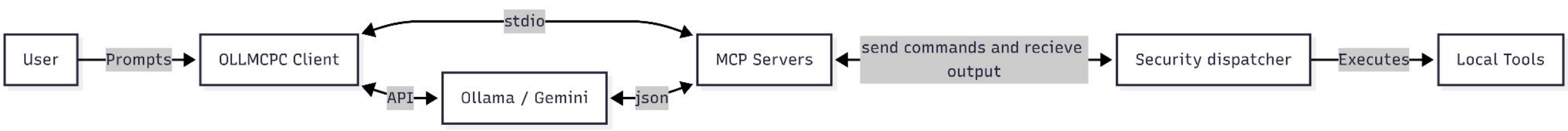
Hosts the actual tool logic (e.g., the Mini OS scripts), ready for execution upon request.

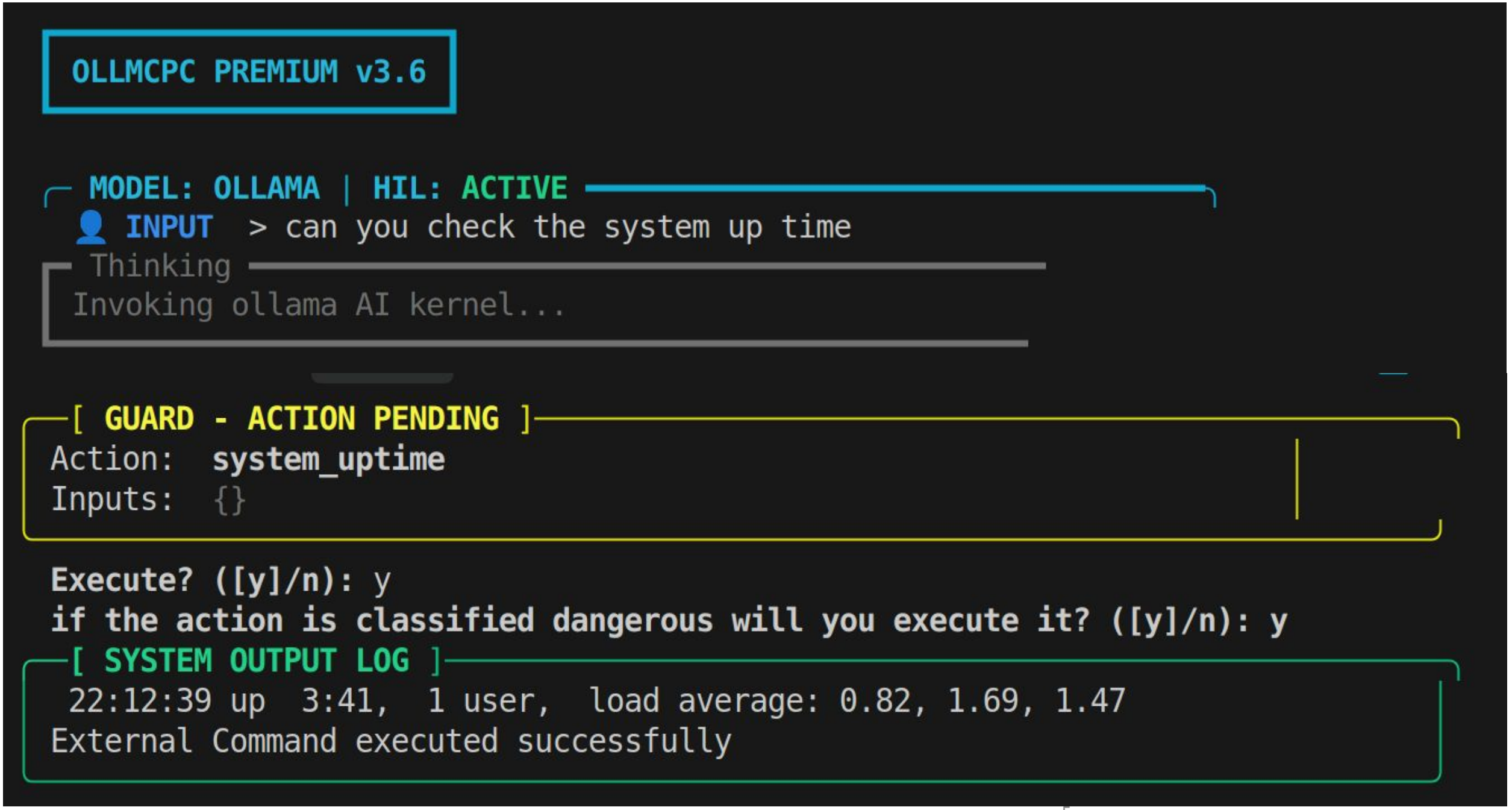
Security Dispatcher

Enforces authorization and safety checks, acting as a firewall for dangerous commands.

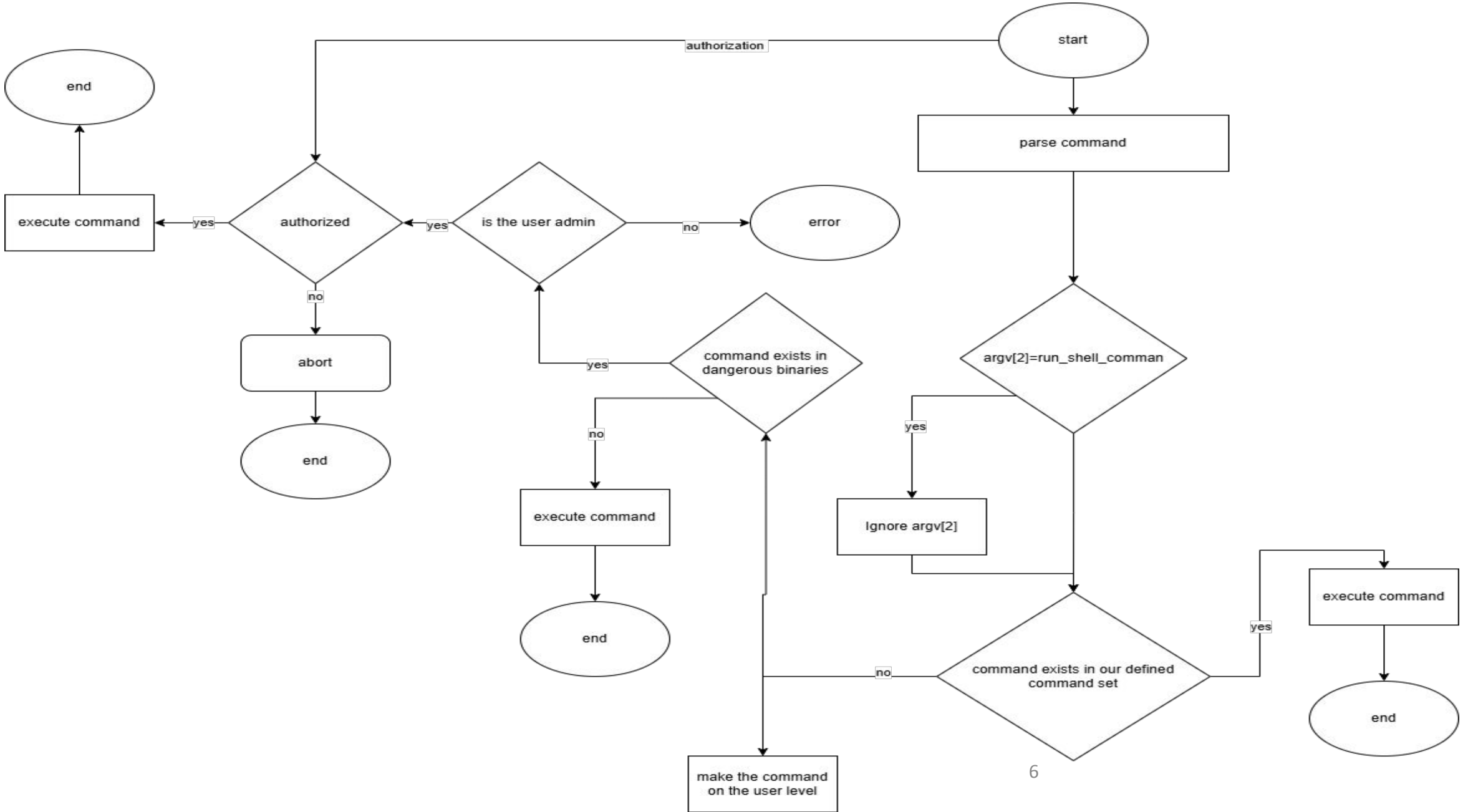
HIL (Human-in-the-Loop)

Pauses execution of sensitive operations to request explicit user approval before proceeding.




OLLMCPC PREMIUM v3.6
MODEL: OLLAMA | HIL: ACTIVE
INPUT > can you check the system up time
Thinking
Invoking ollama AI kernel...
[GUARD - ACTION PENDING]
Action: system_uptime
Inputs: {}
Execute? ([y]/n): y
if the action is classified dangerous will you execute it? ([y]/n): y
[SYSTEM OUTPUT LOG]
22:12:39 up 3:41, 1 user, load average: 0.82, 1.69, 1.47
External Command executed successfully

Dispatcher Security Flow



Command Groups

- User Groups (No Need for user to be admin and no need for authorization
- Admin Groups (Need user to be admin)

- Dangerous Groups (Need user to be admin and no need authorization)

OLLMCPC Key Features

Multi-Provider

Supports both local (Ollama) and cloud (Gemini) LLM providers, ensuring flexibility and vendor independence.

OS Assistant

Dedicated tools for filesystem access, process management, and system diagnostics.

30+ tools

Extensible

Connect external MCP servers written in Node.js, Python, or Go for unlimited customization.

Operating System Utility Scripts

Process Management, Memory Inspection, IPC, Threads, and System Monitoring

Process Management & Control

- Inspect process state, hierarchy, and ownership
- List child processes and threads
- Start, stop, and terminate processes safely

Threads & Concurrency

- Demonstrate thread creation and execution
- Show the importance of synchronization between threads

Memory Inspection

- Monitor system memory and swap usage
- Inspect heap and stack address ranges of running processes
- Understand memory pressure and process memory behavior

System & Network Monitoring

- Check uptime and load average
- Inspect network configuration and open ports
- Observe I/O and system activity

Inter-Process Communication (IPC)

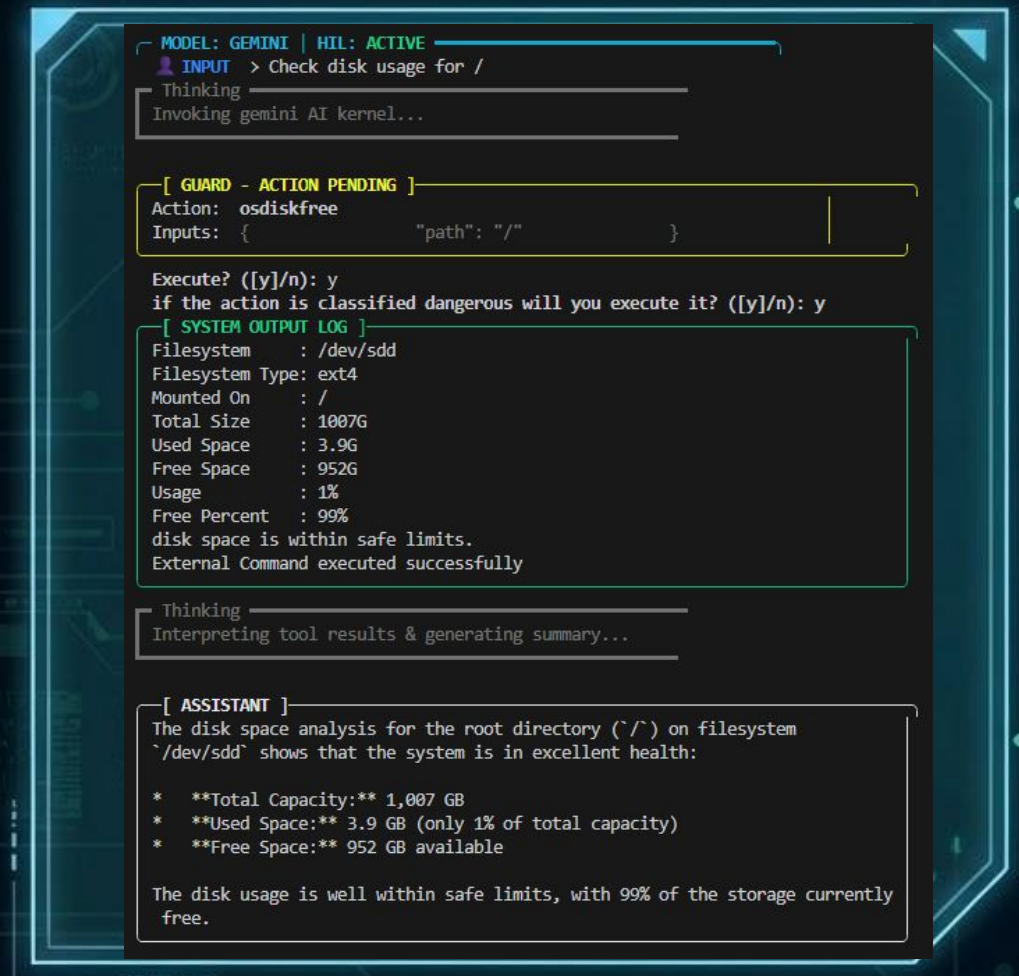
- Demonstrate signal-based communication between processes
- Observe parent–child coordination using OS signals

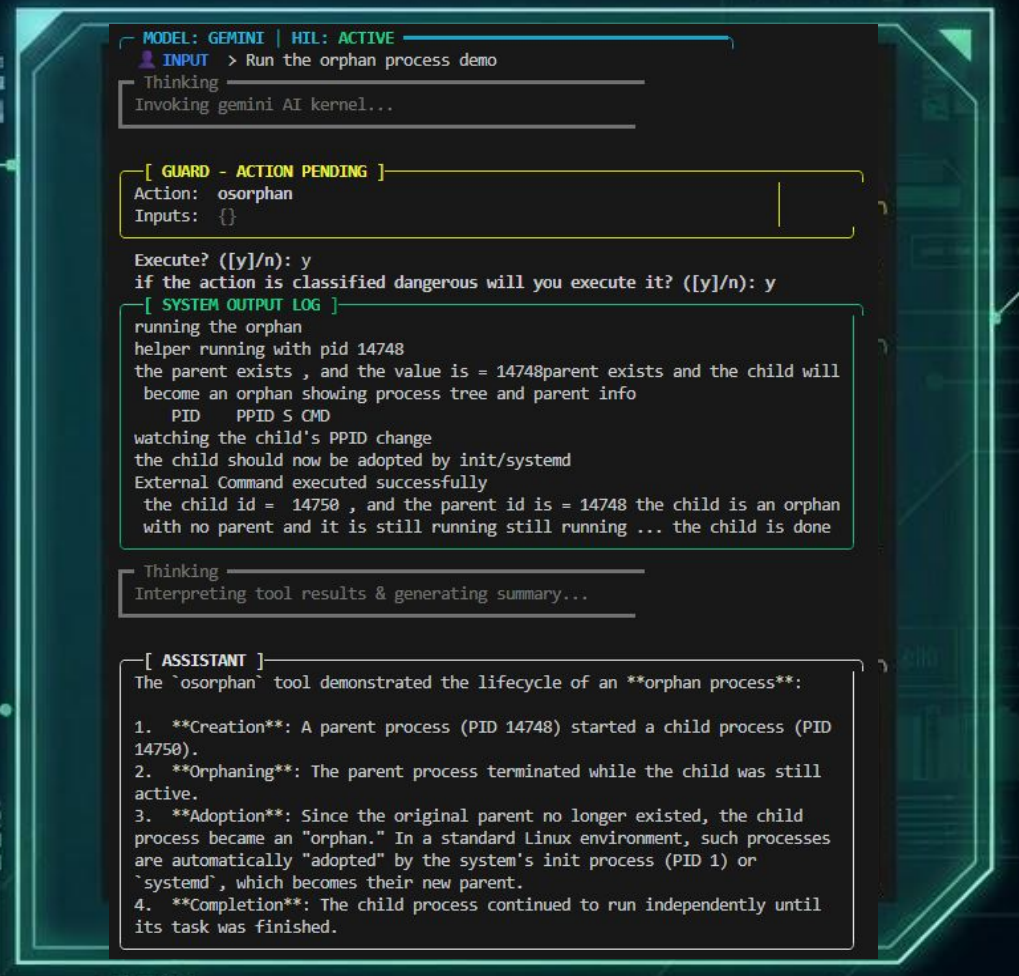
OS Assistant Utilities

- Answer human questions like:
  “Why is my system slow?”
  “Is my system under memory pressure?”
  “How healthy is my system right now?”

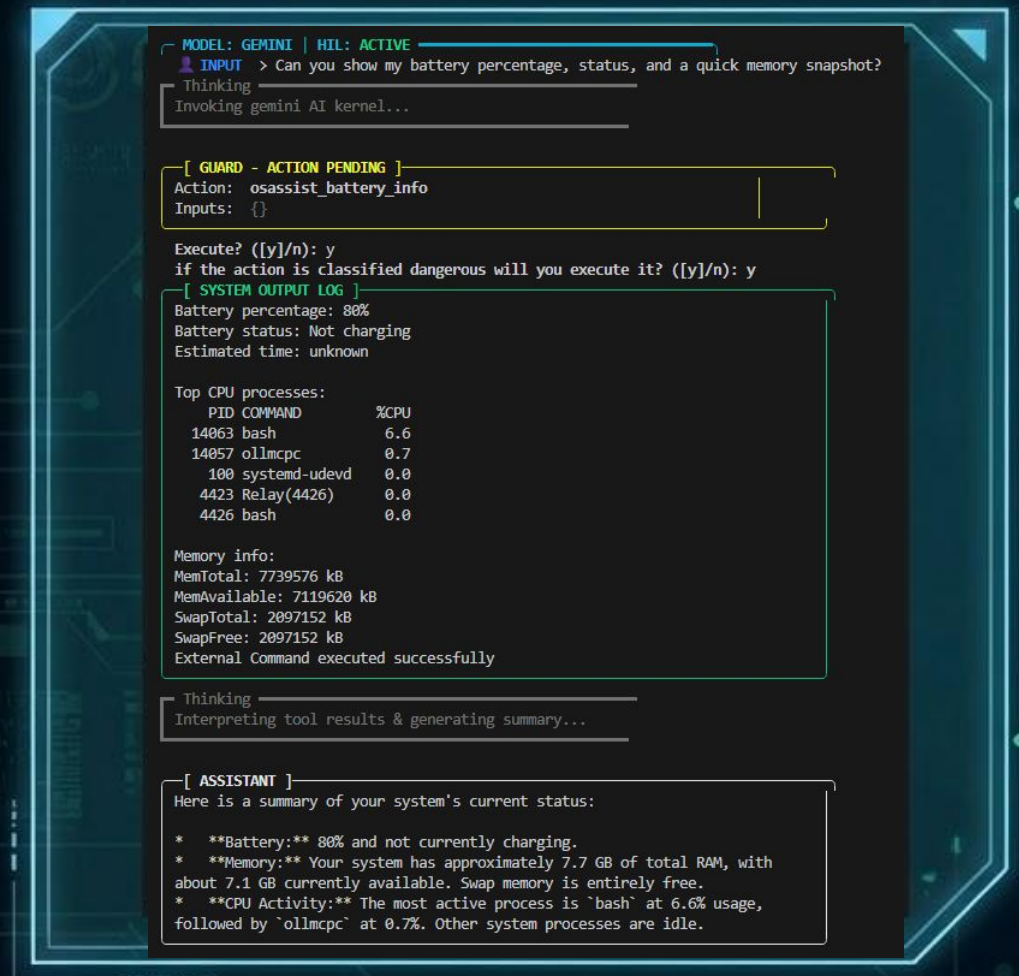
Built using shell scripts and C helpers on top of Linux OS primitives

OLLMCPC in Action: Mini OS Assistant

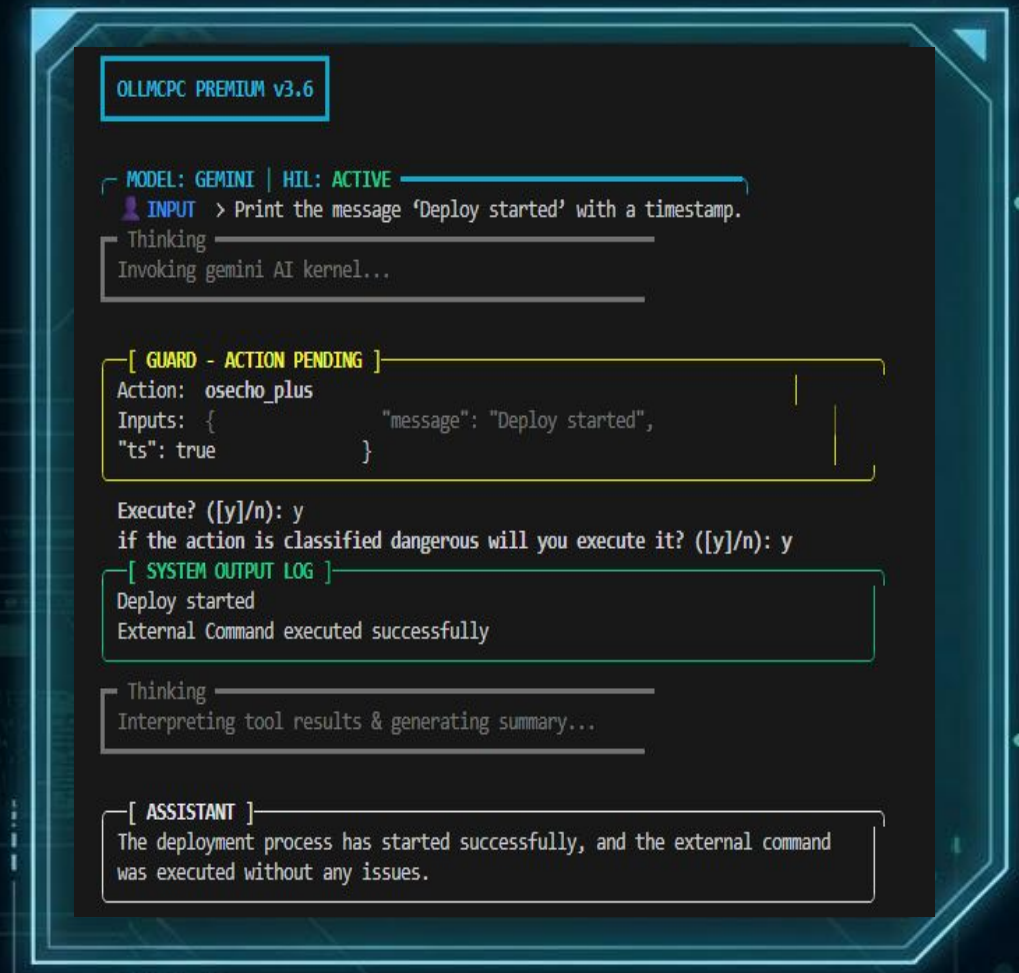


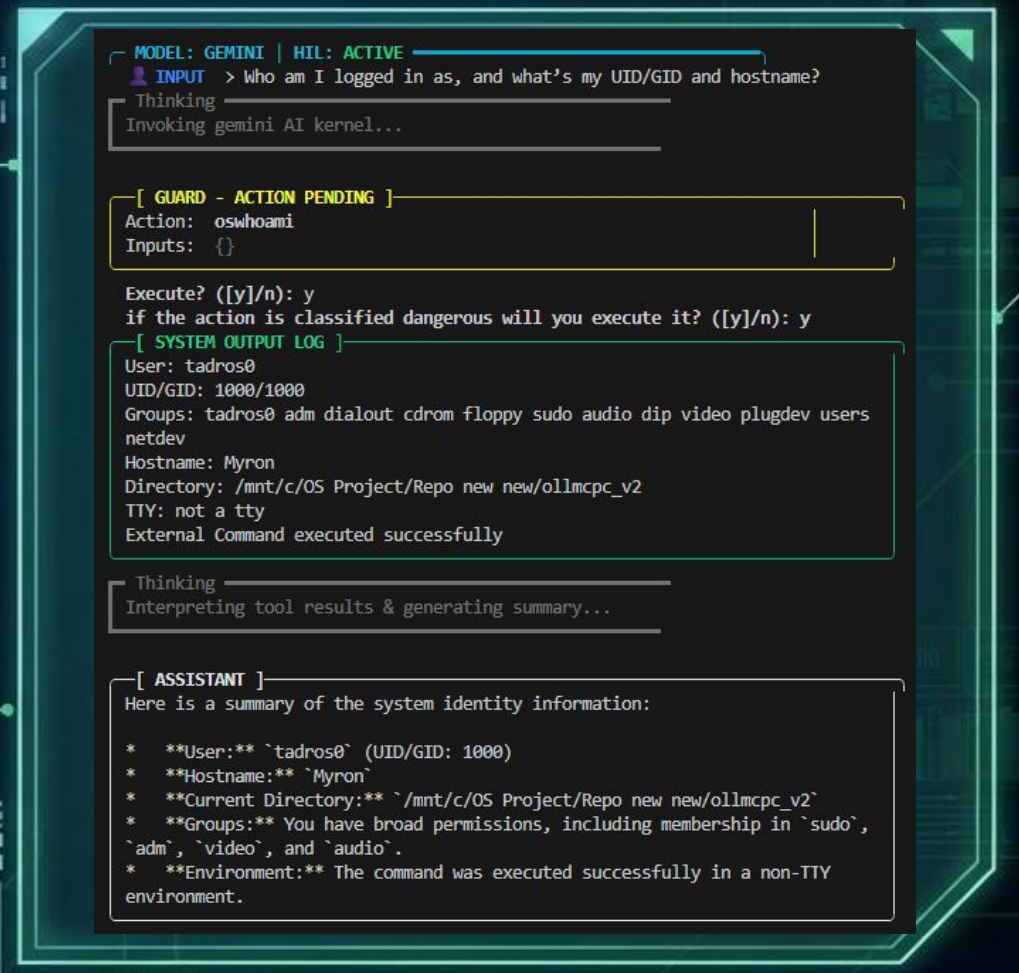


OLLMCPC in Action: Mini OS Assistant

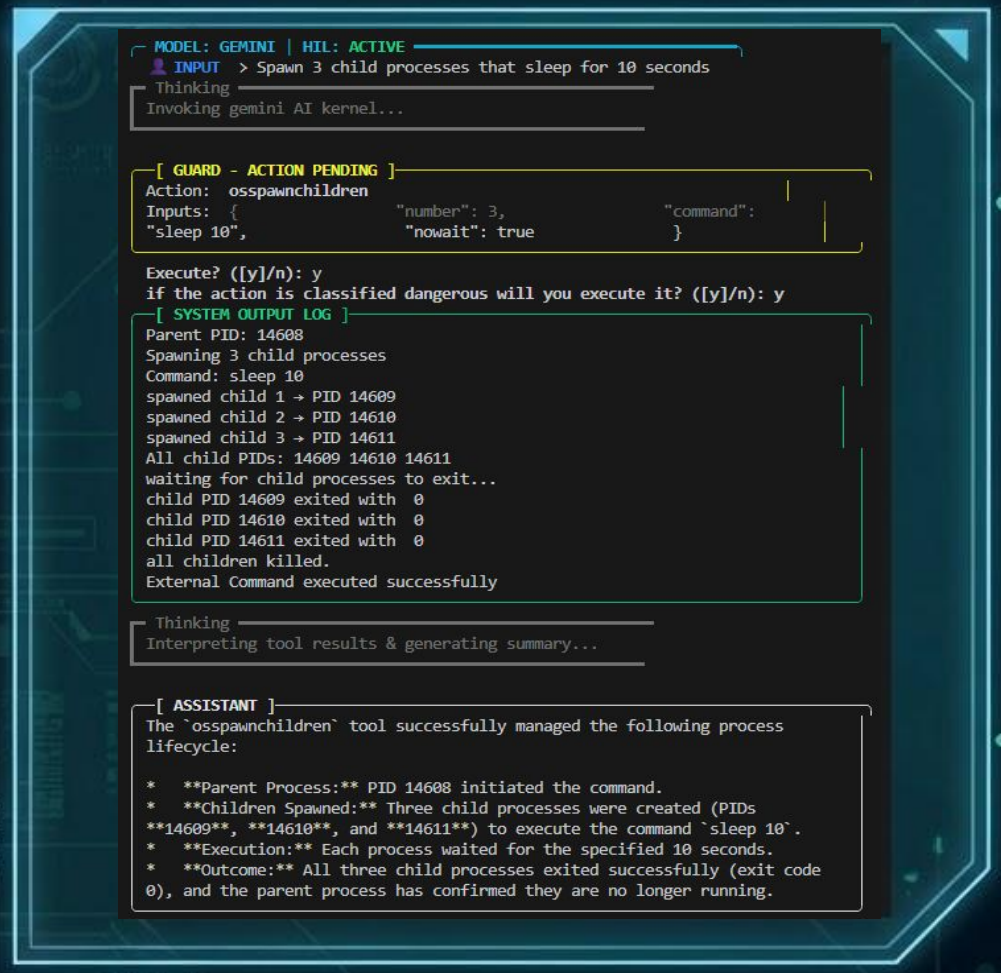


OLLMCPC in Action: Mini OS Assistant

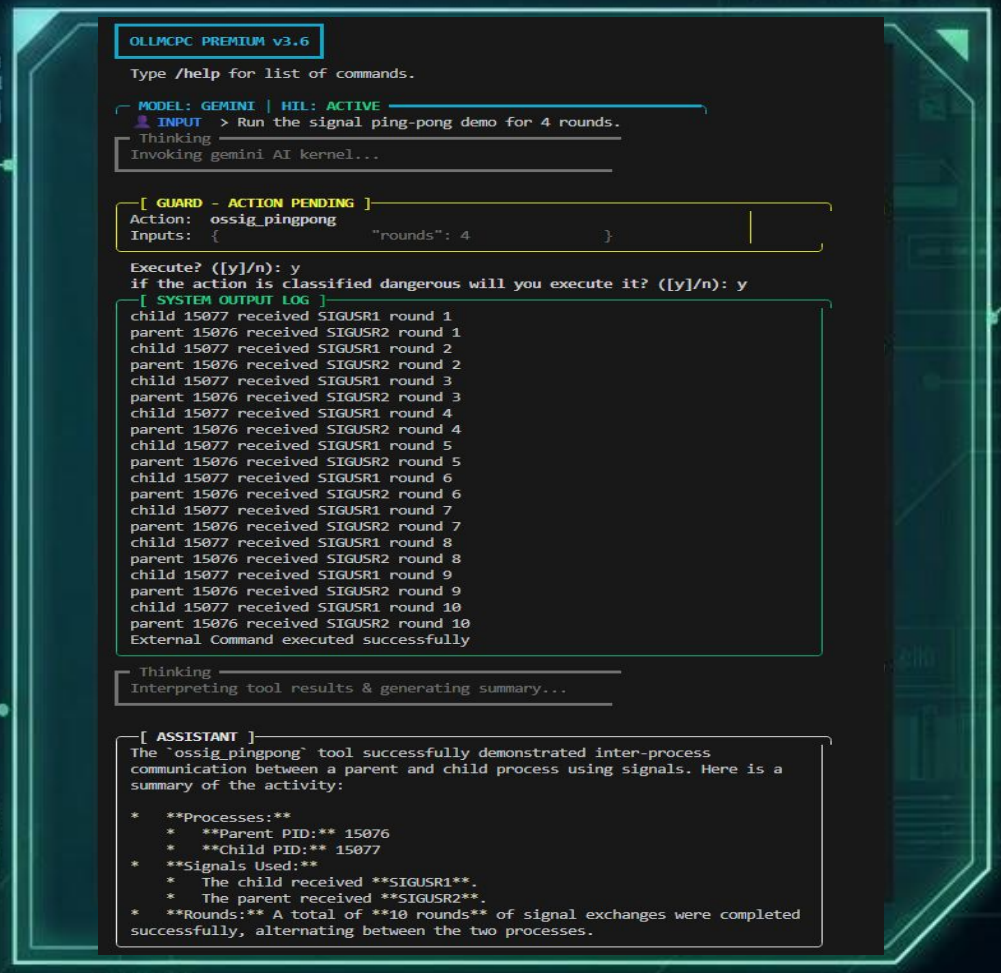




OLLMCPC in Action: Mini OS Assistant



OLLMCPC in Action: Mini OS Assistant



Thank You

Questions & Discussion

Documentation:
https://mohammed-alaa40123.github.io/ollmcpc_v2/