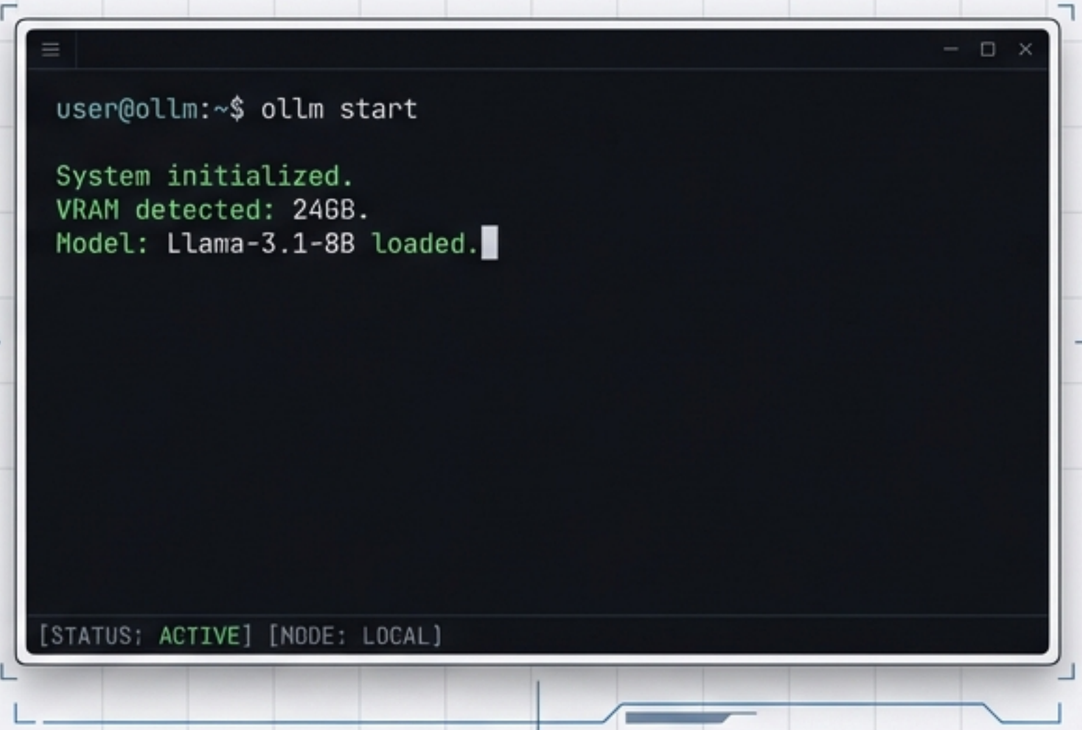
OLLM CLI

The Anatomy of a Local-First AI Terminal

A provider-agnostic, VRAM-aware environment for intelligent development.



Local-First

Deep integration with Ollama for privacy and offline capability.

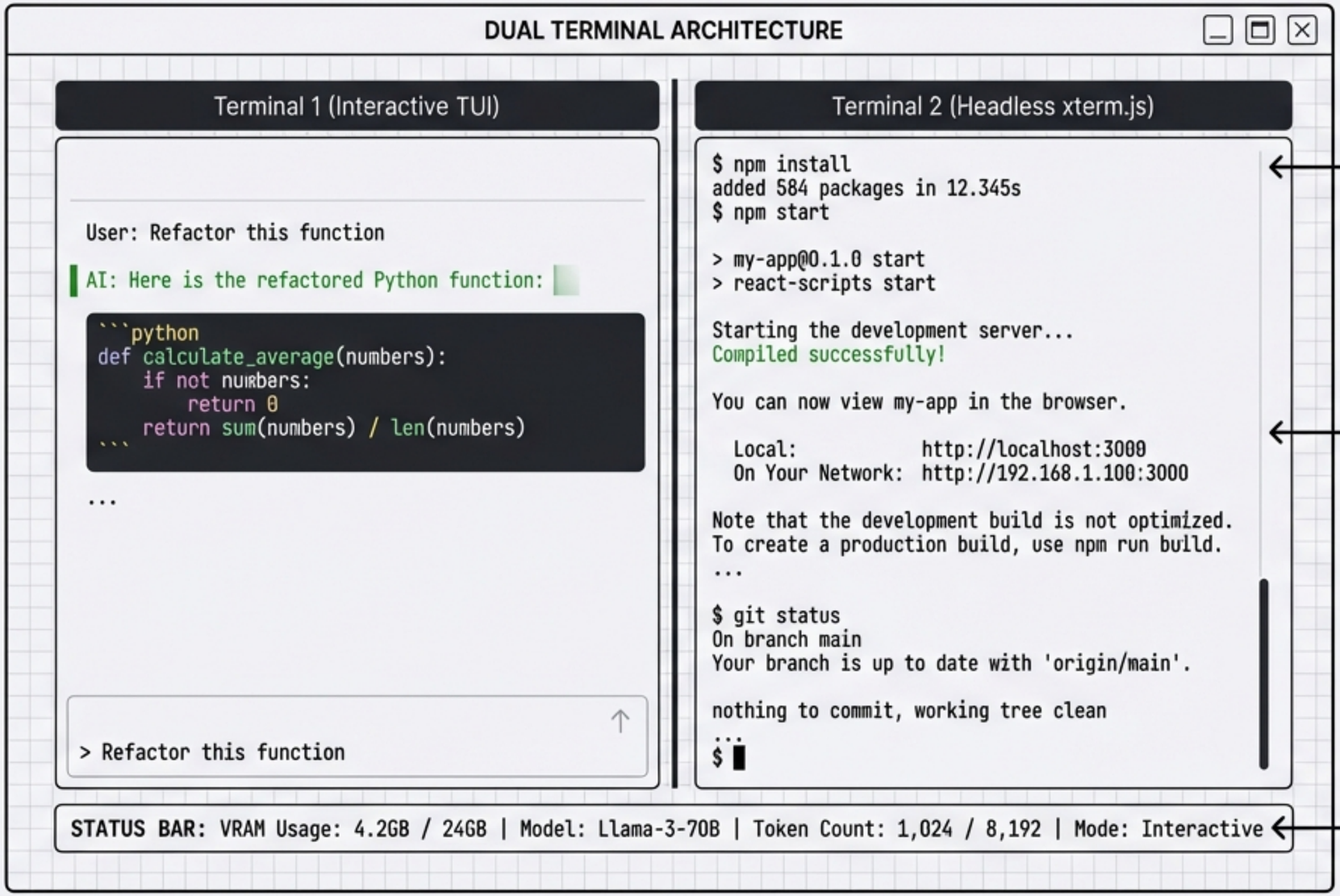
Resource-Aware

Mathematical context management to prevent Out-Of-Memory errors.

Tool-Enabled

Robust file system, shell, and web access via Model Context Protocol.

The Interface: React + Ink Rendering



React + Ink:
Dynamic component updates, not static text.

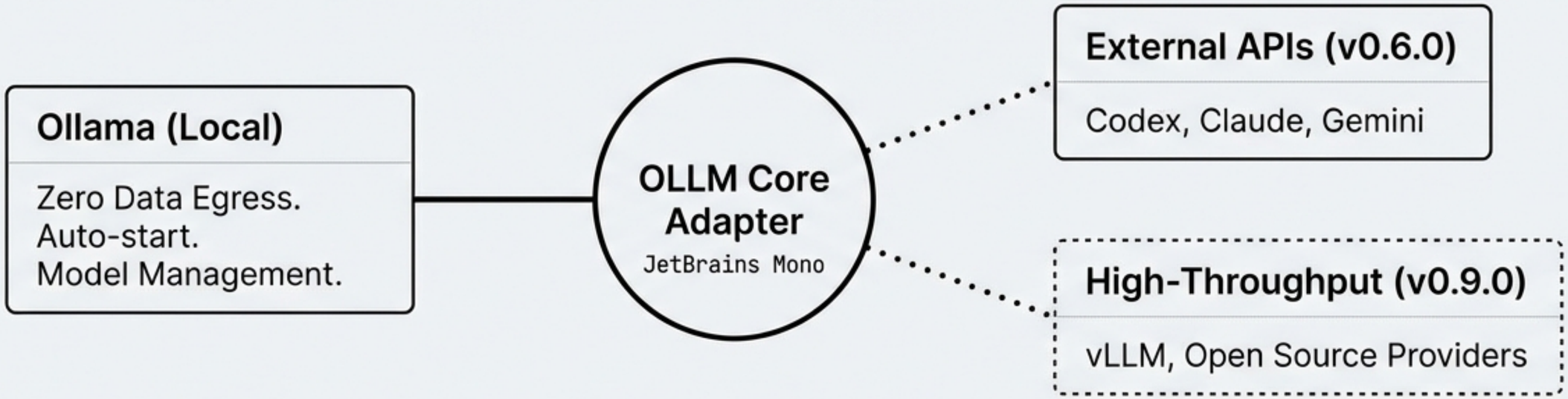
Dual PTY Sessions:
Concurrent shell execution and AI chat.

Status Bar:
VRAM usage, Model info, Token count.

NAVIGATION CONTROLS

- `J` / `K` : Vim-style Scrolling
- `Ctrl+1` : Chat Tab
- `Ctrl+2` : Tools Tab
- `Ctrl+4` : Files Tab
- `Ctrl+8` : MCP Tab

The Brain: Provider-Agnostic Intelligence



Model Database (35+ Profiles)

Llama 3 Latest

[Standard] [Tools: Yes] [Context: 8K]

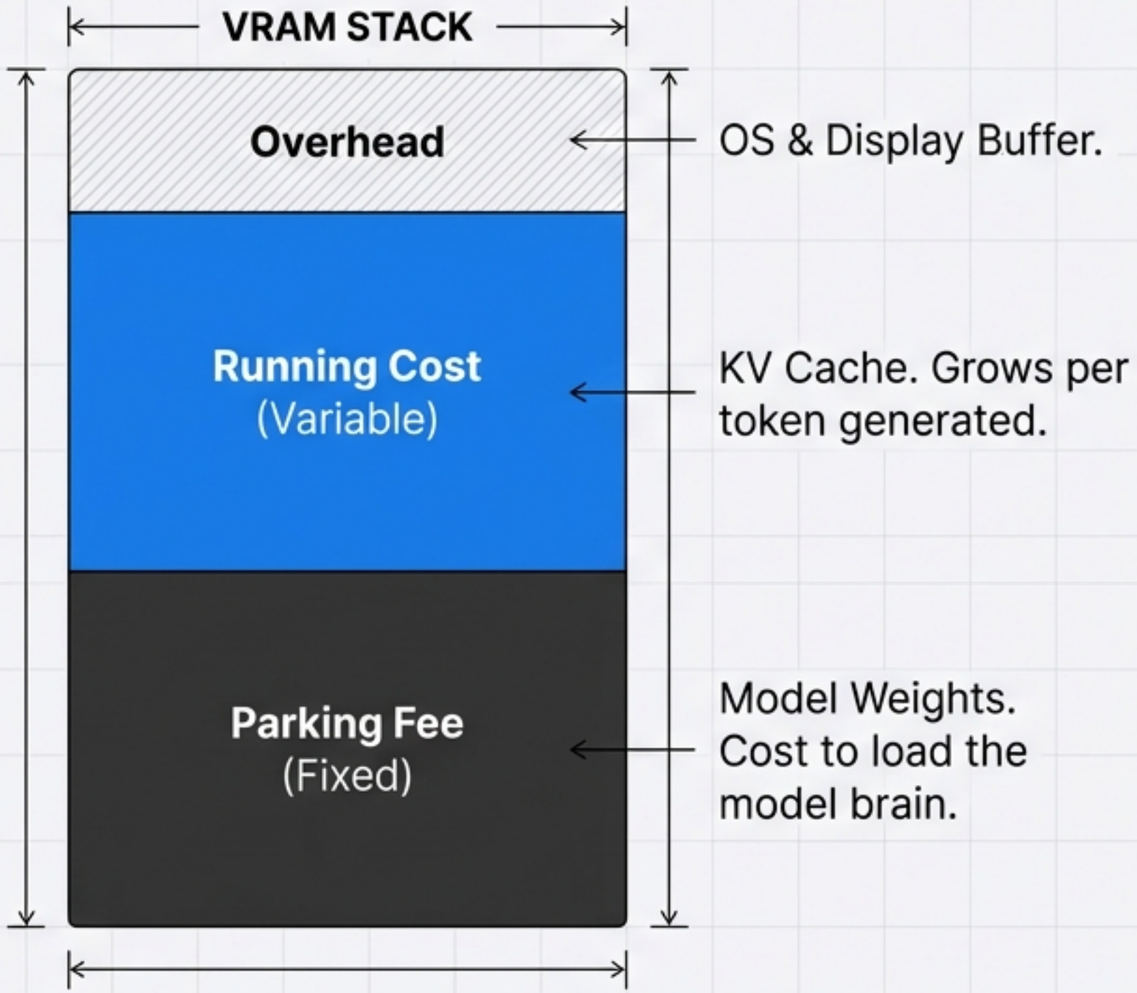
DeepSeek R1

[Reasoning: Yes] [Math Specialized]
[Context: 128K]

Qwen 2.5 Coder

[Coding Expert] [Tools: Yes]
[Context: 32K]

Resource Logic: Math, Not Guesswork



The Formula

```
Total VRAM = Model Weights +
      (Context Length × KV Cache per Token)
```

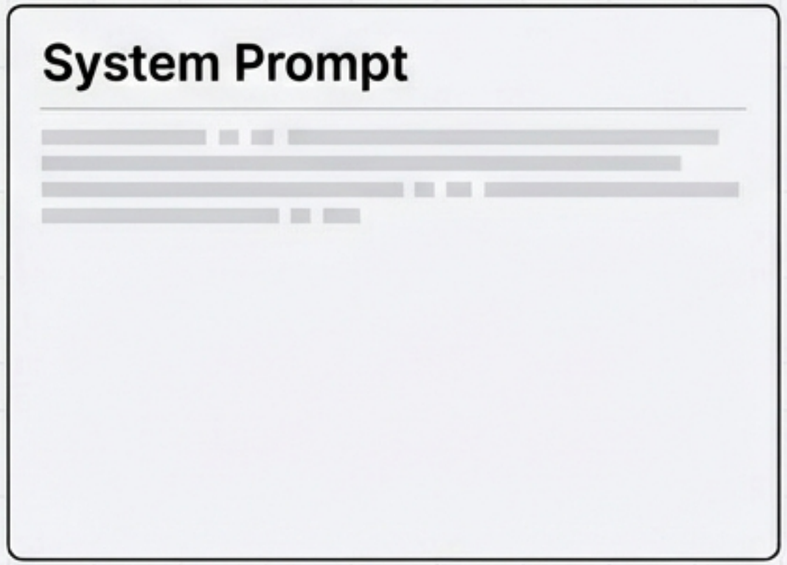
Auto-Sizing Mechanism

- **Target:** ~85% VRAM Utilization
- **Action:** Automatic context compression when limit reached.
- **Example:** 8GB Card → ~8k Context | 24GB Card → ~32k+ Context.

Adaptive System Prompts

Tier 1 (Minimal)

`2K-4K Context | ~200 Token Budget`

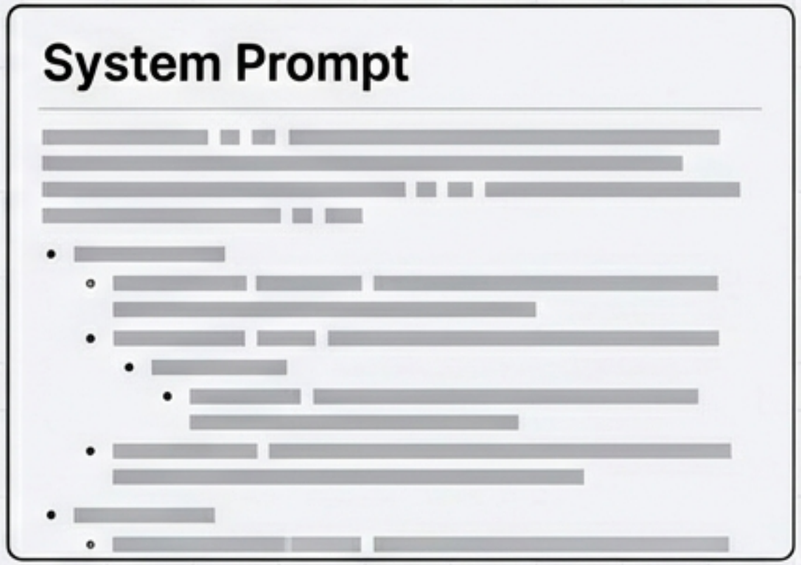


Focus: Essential guardrails only.
High efficiency for low-end hardware.

Scales with Hardware

Tier 5 (Ultra)

`128K Context | ~1500 Token Budget`

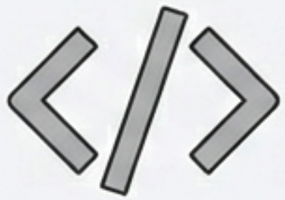


Focus: Expert-level sophistication, mentoring, advanced reasoning patterns.

Maximizes quality on high-end rigs. Maximizes workspace on low-end laptops.

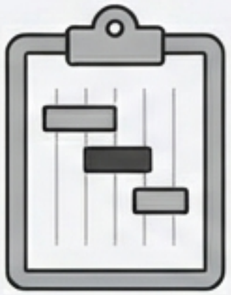
Operational Modes

Context-aware personas for specific tasks.



Dev Mode

- **Focus:** `SOLID` principles, Code Quality, `Testing`, `Testing`
- **Output:** `Production-ready` code.



Plan Mode

- **Focus:** Risk `Assessment`, `Dependencies`, `Estimation`
- **Output:** Actionable plans.

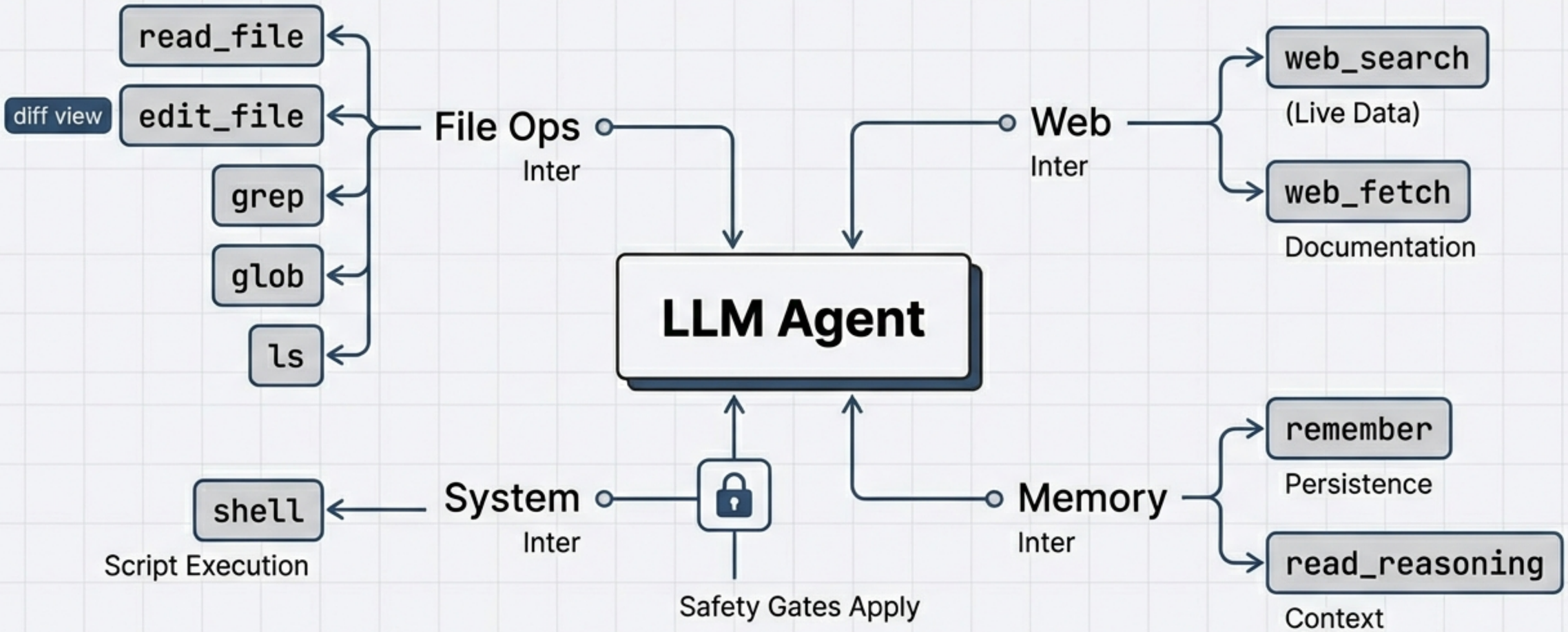
Debug Mode

- **Focus:** Root Cause `Analysis`, `Reproduction`, `Hypothesis Testing`
- **Output:** Systematic fixes.

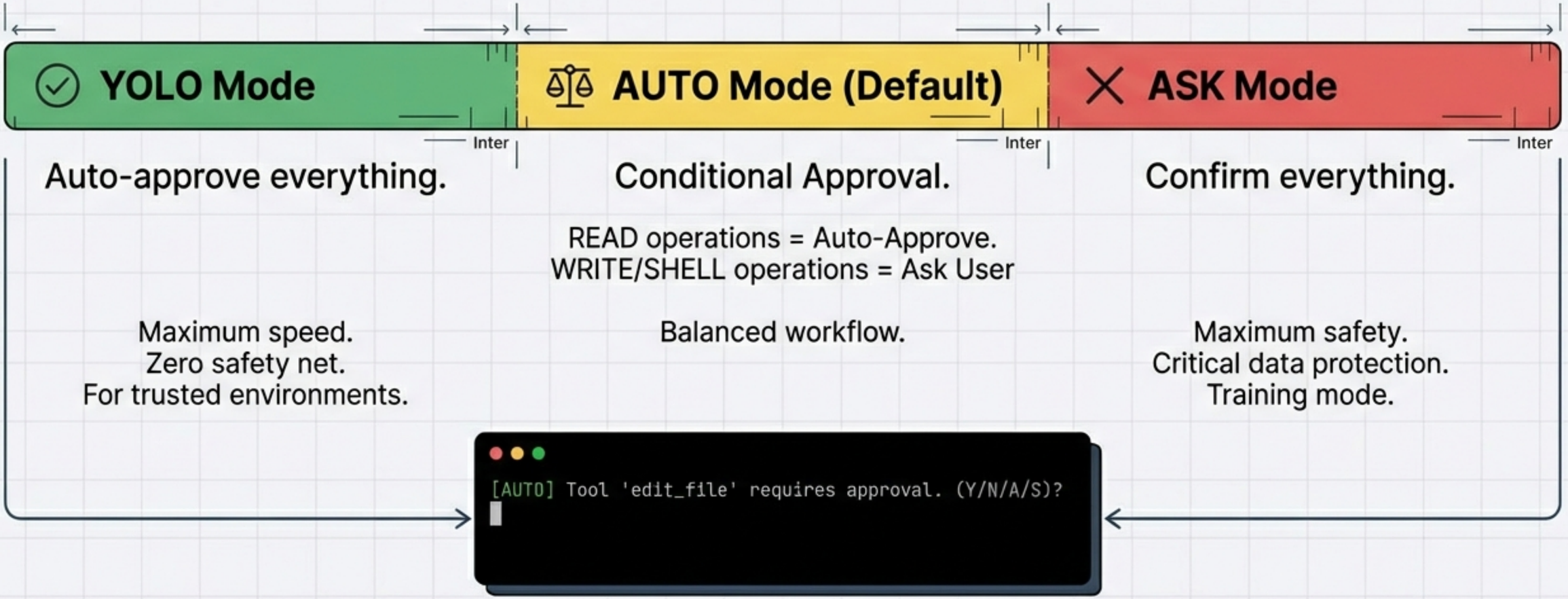
Assistant Mode

- **Focus:** General Help, Communication, User Preferences
- **Output:** Clear explanations.

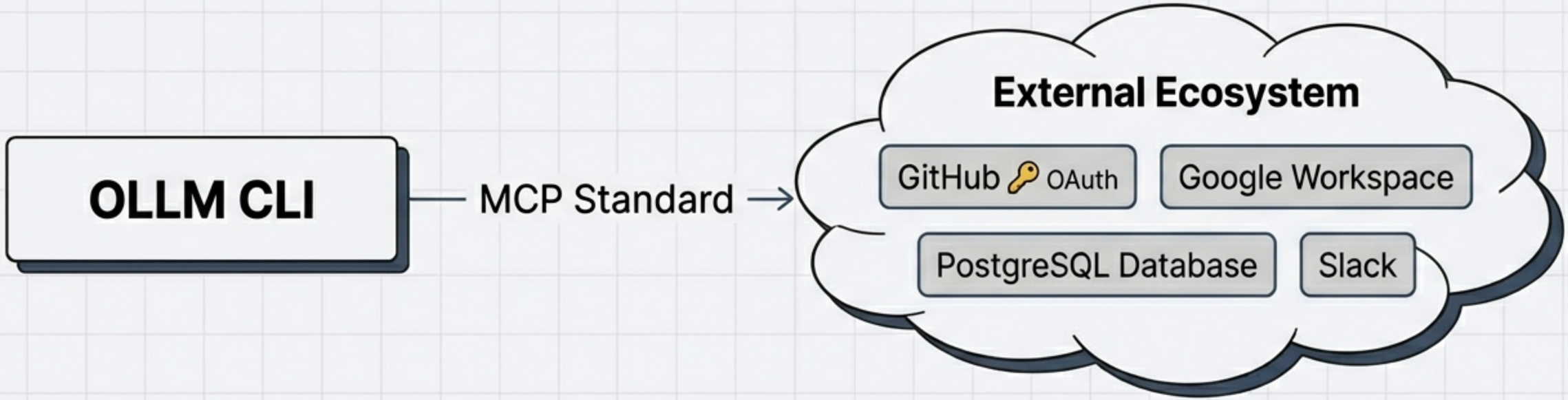
The Tool System: Agency & Action



Safety Spectrum: The Approval Protocol



Extensibility: Model Context Protocol (MCP)



Standardized Protocol

Connects AI to external tools, resources, and prompt templates.

Transports

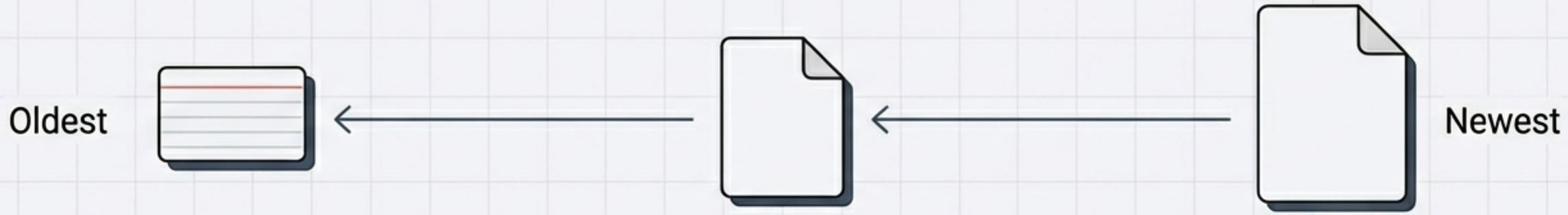
- Supports `Stdio` (Local Processes)
- Supports `SSE` (Remote Streams)

Capabilities

- Marketplace browsing (`/mcp`)
- Built-in OAuth token management
- Automatic Health Monitoring

Memory Architecture & Persistence

Checkpoint Aging Strategy



Compact Checkpoint
One-line summary.

Moderate Checkpoint
Top 3 decisions +
Reduced metadata.

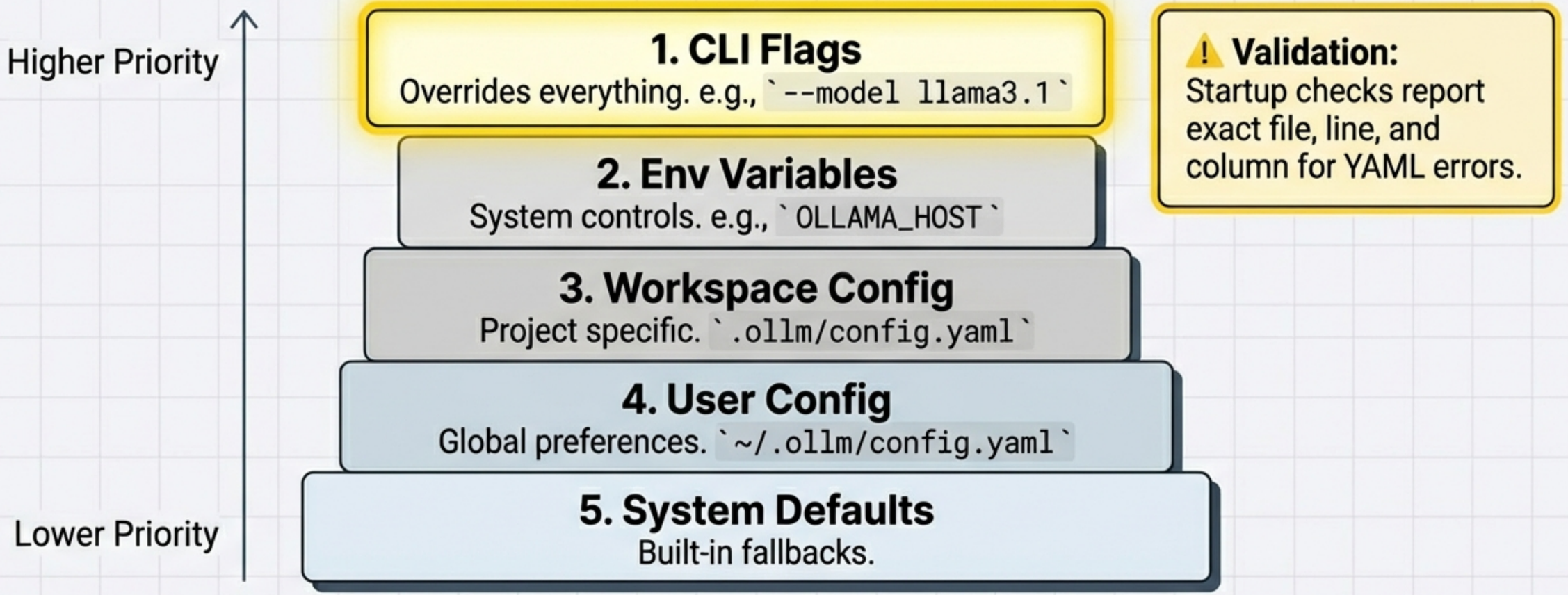
Detailed Checkpoint
Full summary + All
key decisions.

Persistence Layer

- **Session Management**: Save/Resume (`/session save`). Compression history.
- **Cross-Session Memory**: stored in `~/.ollm/memory.json`. Remembers user preferences and facts.

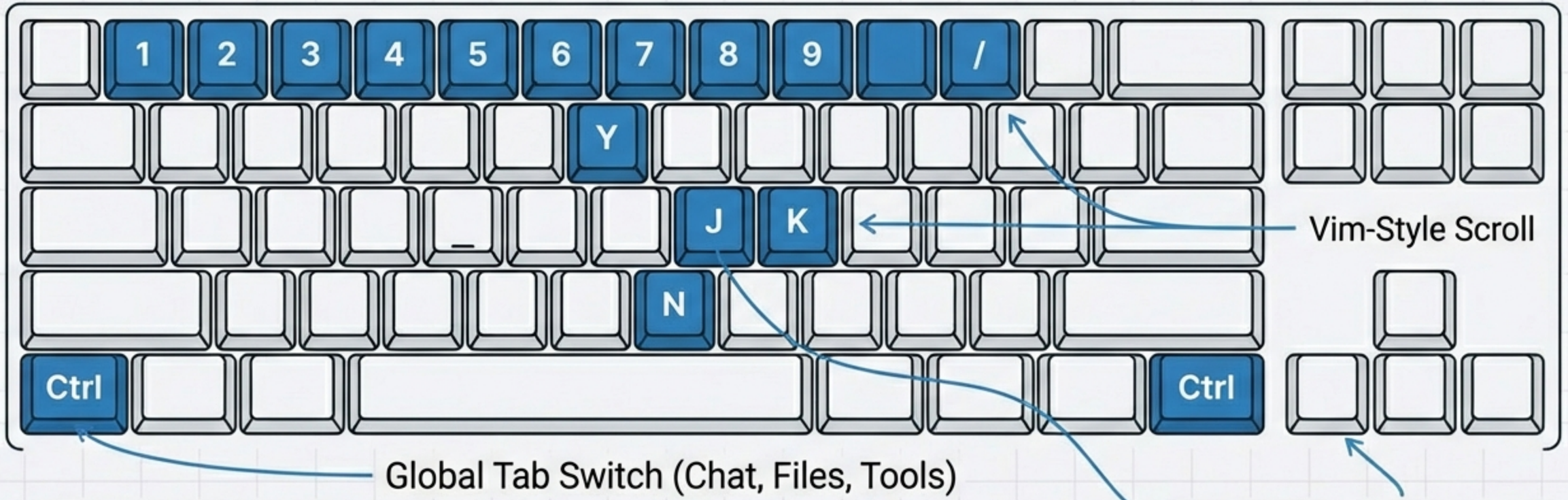
The Precedence Stack

Fully tunable behavior via layered configuration.



⚠ **Validation:**
Startup checks report exact file, line, and column for YAML errors.

Developer-Centric Workflow

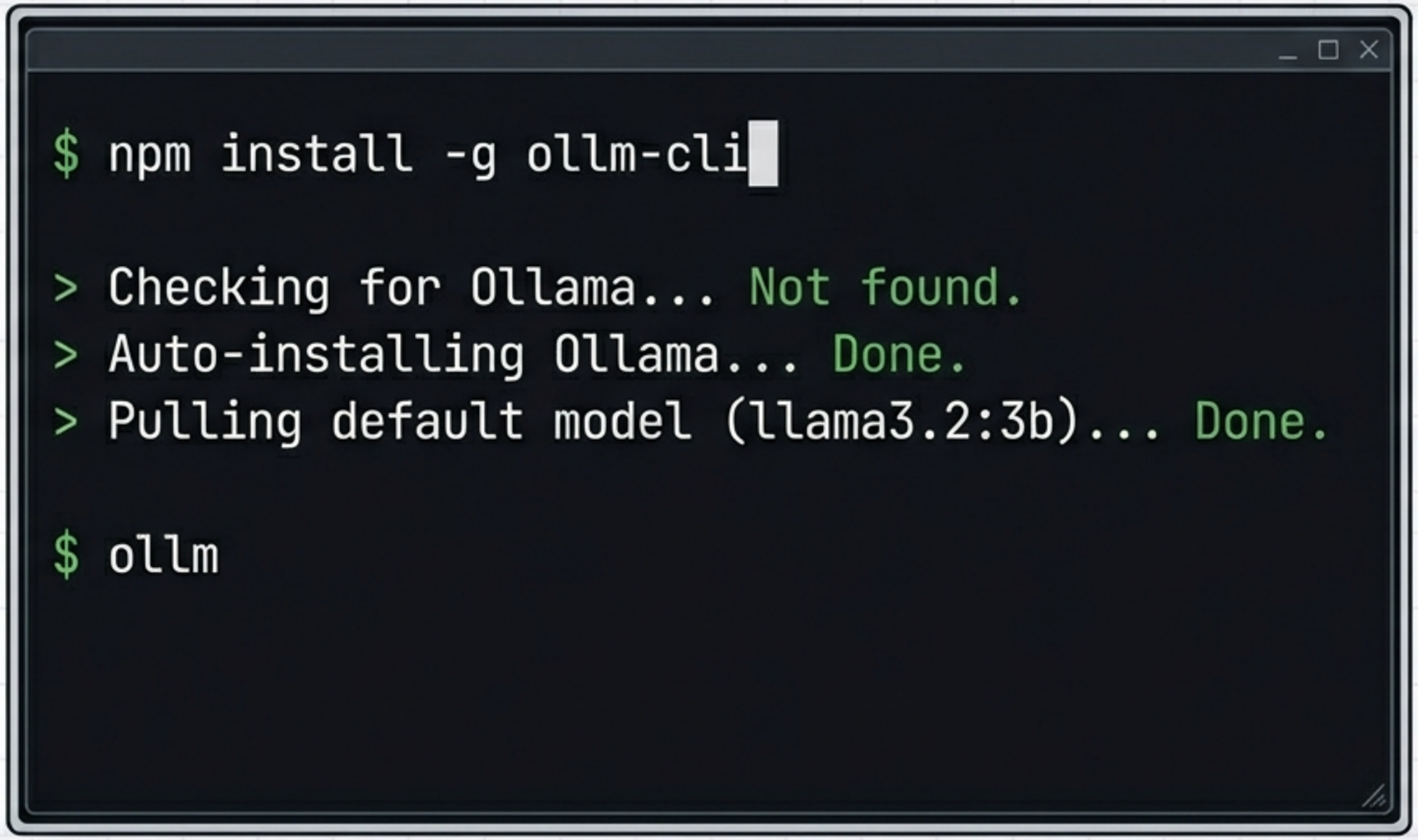


Power Features

- **Quick Open Files** (Ctrl+P) - in JetBrains Mono.
- **Command Palette** (`/model`, `/context`).
- **Review Mode** for rapid code diff approvals.

Development Roadmap (Alpha Phase)

Quick Start: Zero to AI in Minutes



Key Commands

- `/help`: Show all commands.
- `/model list`: Switch models.
- `/context stats`: View VRAM/Context usage.

Capabilities at a Glance

CAPABILITY	STATUS	DESCRIPTION
Local-First	☑	Offline capable, Privacy-focused, Zero data egress.
VRAM-Aware	☑	Mathematical context management prevents OOM crashes.
Tool-Enabled	☑	Native Files, Shell, and Web access with safety gates.
Provider-Agnostic	☑	Ready for `vLLM`, `OpenAI-compatible` backends.
Developer-Centric	☑	TUI, Vim keys, Configurable hierarchy.

Build the future of local AI tooling.

`github.com/ollm/ollm-cli`