VOID: The Open-Source AI-Powered Code Editor

Group 23

CISC 322 Software Architecture
https://www.youtube.com/watch?v=xyLFo_VkChM

Team members

Jared

Charlie

Jaivik (Leader)

Kai

Robin

Zack

Background & Derivation Process

Void

How we derived the architecture

- Researched VOID docs and GitHub repository.
- Identified core subsystems from implementation and code structure.
- Mapped use cases → sequence diagrams → box-and-arrow model.
- Iteratively validated design with VOID's event-driven behavior and agent mode.

GitHub

Architectural Style & Design Rationale

Hybrid Architecture Styles

- Layered: clear separation of UI, logic, and data.
- Event-Driven: asynchronous communication via Event Bus.
- Plugin-Based: open extensibility for community contributions.
- Why this mix works: scalability + modularity + responsiveness.

Void

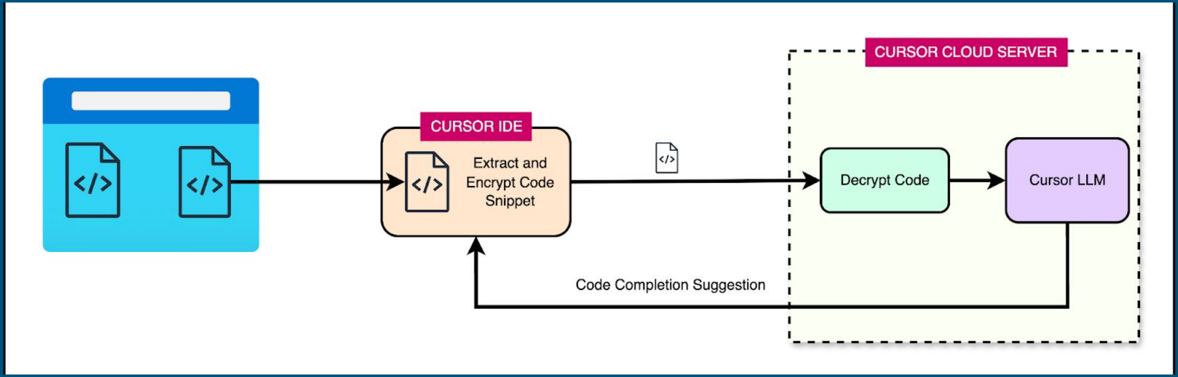
Top-Level Subsystems

Core components and responsibilities

1. Editor Core – text rendering, cursor movement, file I/O.
2. AI Bridge – prompt builder and context manager.
3. Model Connector – handles LLM requests and responses.
4. Plugin Manager – manages third-party tools and integrations.
5. Checkpoint Manager – project state and version control.
6. Security Sandbox – permission and safety enforcement.
7. Event Bus – async messaging backbone.

Interactions and Control Flow

1. User command triggers an event in the Editor Core.
2. Event Bus routes request to AI Bridge → Model Connector.
3. LLM returns response → AI Bridge formats → UI updates.
4. Checkpoints and Security Sandbox monitor the process.
5. Motivation: decoupled, parallel operations without UI blocking.

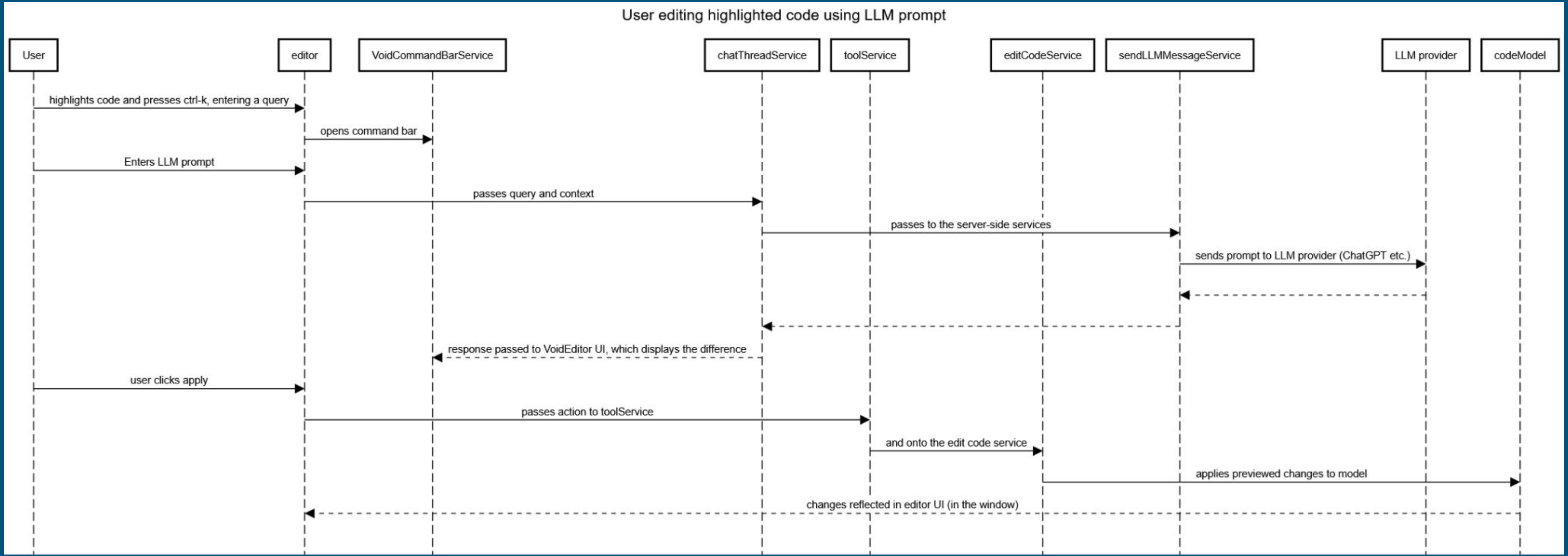


Concurrency in VOID

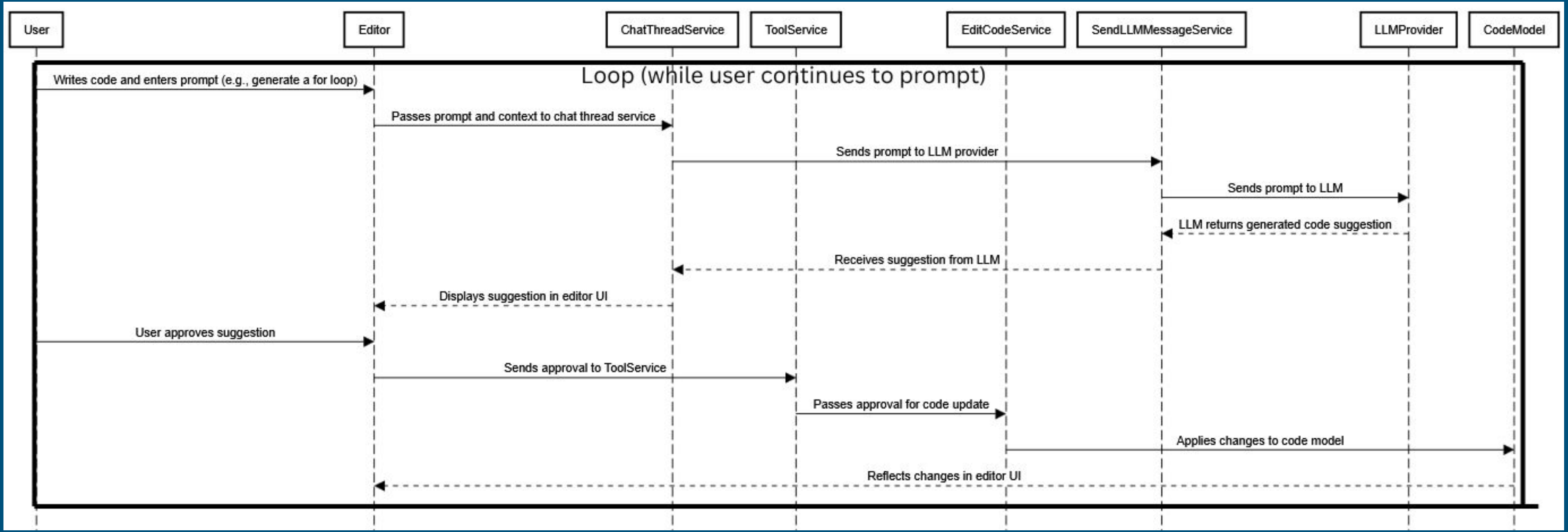
- Built on VS Code’s multi-process model (renderer, extension host, main).
- Language servers and AI connectors operate concurrently.
- Event loops and streaming prevent lock-ups.
- Enables real-time token updates and parallel diagnostics.
- Outcome: non-blocking asynchronous collaboration between human and AI.

Void

Use Case 1: Refactor Highlighted Code


User editing highlighted code using LLM prompt
User
editor
VoidCommandBarService
chatThreadService
toolService
editCodeService
sendLLMMessageService
LLM provider
codeModel
highlights code and presses ctrl-k, entering a query
opens command bar
Enters LLM prompt
passes query and context
passes to the server-side services
sends prompt to LLM provider (ChatGPT etc.)
response passed to VoidEditor UI, which displays the difference
user clicks apply
passes action to toolService
and onto the edit code service
applies previewed changes to model
changes reflected in editor UI (in the window)

Use Case 2: LLM lead code generation


User
Editor
ChatThreadService
ToolService
EditCodeService
SendLLMMessageService
LLMProvider
CodeModel
Loop (while user continues to prompt)
Writes code and enters prompt (e.g., generate a for loop)
Passes prompt and context to chat thread service
Sends prompt to LLM provider
Sends prompt to LLM
LLM returns generated code suggestion
Receives suggestion from LLM
Displays suggestion in editor UI
User approves suggestion
Sends approval to ToolService
Passes approval for code update
Applies changes to code model
Reflects changes in editor UI

System Evolution

Void

- v1.0 – Layered base (derived from VS Code).
- v1.5 – Event Bus for asynchronous tasks.
- v2.0 – Agent Mode + Security Sandbox.
- v2.5 – Checkpoint sync + API Gateway.
- v3.0 – Distributed multi-agent architecture.

Takeaway: evolution prioritizes modularity, concurrency, and privacy.

Alternative Architectures

- Microservices: scalable but over-engineered for local IDE.
- Monolithic: simpler but limits modular updates.
- Client–Server: great for remote collab, adds latency offline.
- Final Choice: Hybrid local + event-driven for speed and extensibility.

Lessons Learned & Limitations

1. Continuous iteration keeps VOID relevant and scalable.
2. Transparency and sandboxing build user trust.
3. Open-source ecosystem drives innovation and collaboration.
4. Limitations: balancing AI autonomy vs user control; distributed testing complexity.

AI Collaboration & Wrap-Up

- Used AI for drafting, summarization, and diagram mockups.
- We validated accuracy and architecture logic.
- Clear prompting protocol ensured focused results.
- Outcome: faster iteration and improved clarity in conceptual design.

References & Links

VOID Editor Official Site – https://voideditor.com

VOID GitHub Repo – https://github.com/voideditor/void

Microsoft Docs: VS Code Architecture Overview (2023)

Aditya Kumar, “Void IDE – A Comprehensive Guide,” Medium (2024)