URL for video:

https://youtu.be/jVAg9ngf4SM

Group Breakdown

Outline, Research

Chat GPT 5.0

Use Cases, Ai Collab, Conclusions, Presentation

Karim Sponheimer
Presenter

Architecture, Diagrams, Use Cases

Zachary Stephens
Group leader

Abstract, Introduction, Presentation

Kasia Zaski
Presenter

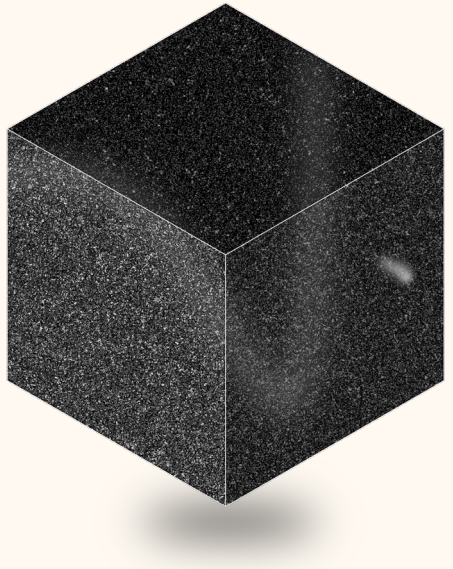
External Interfaces, Architecture, Data Dictionary

Zirui Wang

Group Name: Potential Guacamole

Conceptual Architecture

Void AI



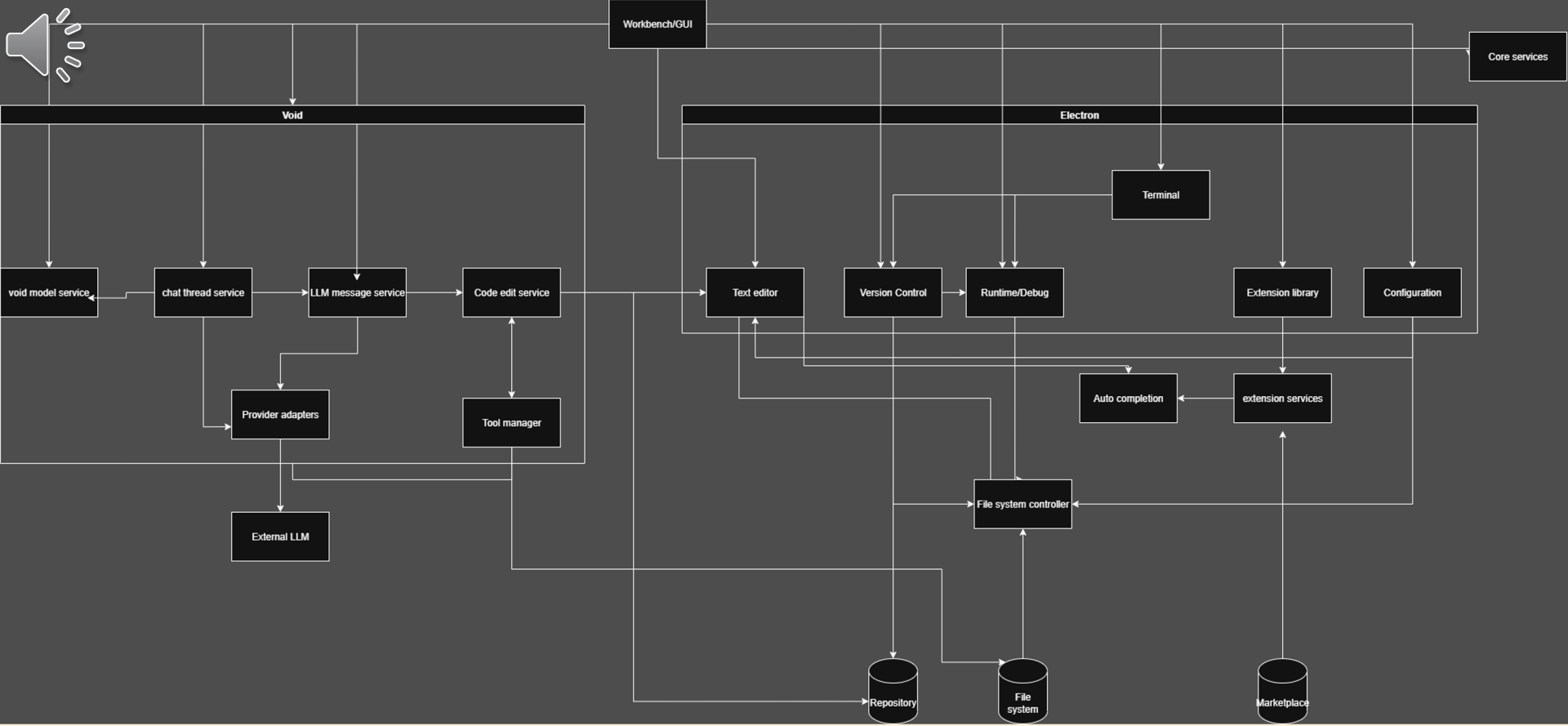
Group 14: Karim Sponheimer, Zachary Stephens, Zirui Wang, Kasia Zaski

Abstract

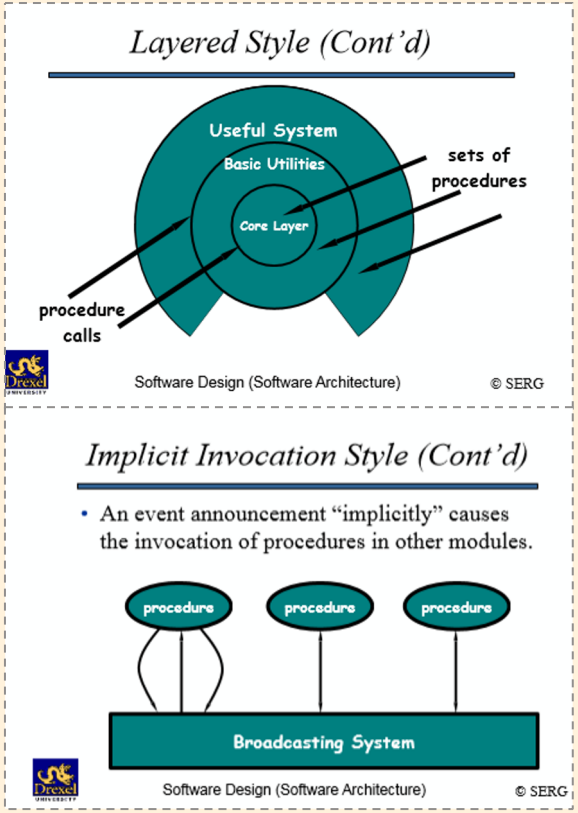
- Analyzed Void, an open source, AI powered IDE forked from VS Code
- Explored architecture principles
- Key Insights
  - Renderer, Main, Extension Host Separation
  - AI integration via Provider Adapters
- Deliverables
- Lessons Learned

Introduction/Overview

- VS Code: Lightweight, extensible editor
- Void IDE: adds AI assistant whole keeping privacy, customization, and open source model
- Project goal: Apply IDE architecture principles to a web based project
- Layered structure
  - Self contained component
  - Demonstrates separation of concerns and modularity
- Report structure



Conceptual Architecture



Architecture Style

1. Layered

- Renderer manages user interface and panels.
- Core Services handle editing logic, chat, and message flow.
- Main Process (Electron) handles file access and LLM connections.

2. Implicit Invocation (Pub-Sub)

- Event-driven design for extensions, AI providers, and chat interactions.
- Allows components to react dynamically to user actions or system events.

Subsystem Overview

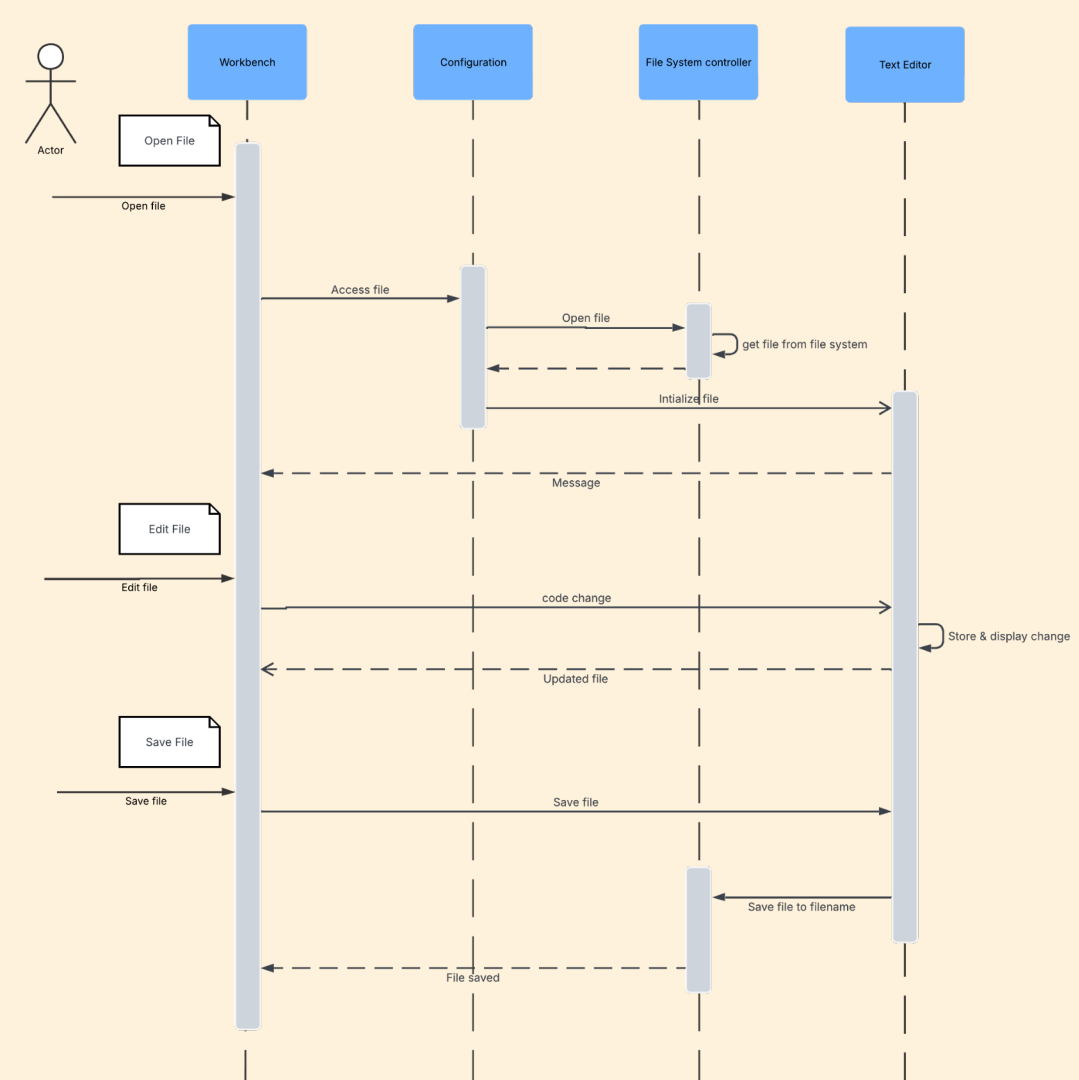
Renderer / Workbench UI: Manages the visible interface including chat, diff previews, and panes.

Core Services: Handles document state, editing buffers, LLM message handling, and version control.

Main Process (Electron): Executes system-level tasks like file I/O and API communication.

Extension Host: Runs extensions safely in a sandbox to protect performance and security.

Provider Adapters: Bridge between Void and external or local LLMs (OpenAI, Anthropic, etc.).



Use Case 1

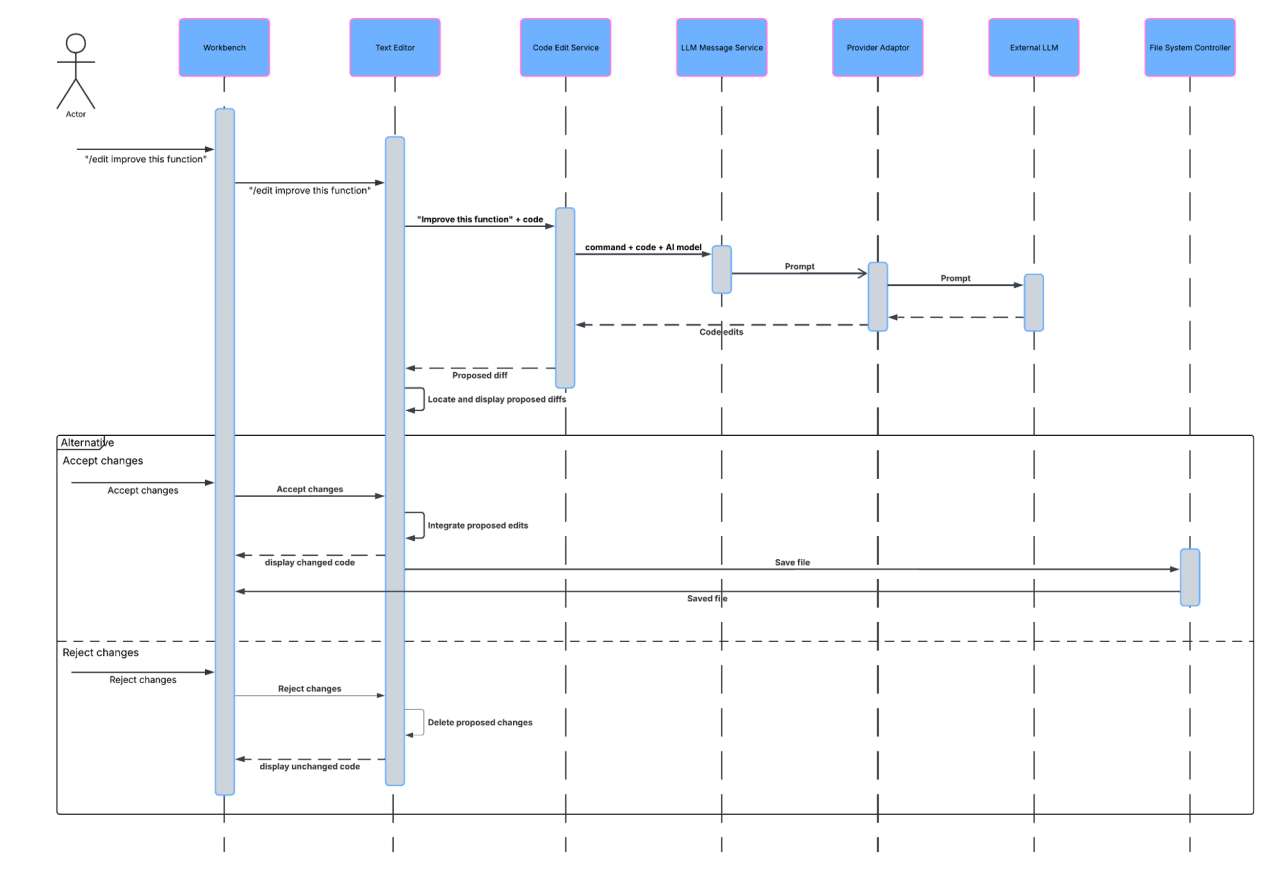
Open, Edit, and Save a File

- User opens a file from the workspace through the Renderer (UI).
- Renderer sends request to Core Services, which checks cache or disk.
- Core communicates with the Electron Main Process to read or write files.
- Main Process performs system-level file I/O and returns confirmation.
- Renderer updates the editor view once the file is opened or saved.
- Demonstrates the layered flow between UI, logic, and system access.

Use Case 2

Chat (/edit) AI Suggestion and Apply

- User types /edit command in the chat panel requesting a code change.
- Renderer passes request and code context to Core editing service.
- Core sends request through Main Process to the selected AI provider.
- Provider streams code suggestions back to the Core and Renderer.
- User previews suggested edits in the editor's diff view.
- On approval, Core applies the new code to the text buffer and saves.
- Shows implicit invocation and real-time, event-driven interaction between components.



External Interfaces

Graphical Interface: Same layout as VS Code but adds AI chat panels, inline diffs, and checkpoints.

APIs: Supports LLM providers and extensions via REST or streaming connections.

File System: Unified handling for local, remote, and cloud-based files.

OS Integration: Clipboard, terminal, and process execution managed through Electron.

Privacy Focus: Direct communication with model providers, no centralized proxy.

Lessons Learned

- Creating diagrams clarified how Void extends VS Code’s architecture.
- Separating inherited vs new components was complex but rewarding.
- AI teammates help accelerate planning but must be carefully guided.
- Strong collaboration even with a smaller group can create success.

Conclusions

- Void uses a layered architecture that provides separation of concerns and easy maintainability.
- The implicit invocation style enables responsive, event-based communication, ideal for AI streaming and plugin integration.
- Each subsystem handles distinct roles: Renderer for UI, Core for logic, Main for system access, and Provider Adapters for LLM connections.
- Architecture supports key goals: extensibility, privacy, and flexibility for AI workflows.
- Electron introduces some performance overhead but gives cross-platform consistency.
- Future improvements could include optimizing caching, improving IPC efficiency, and reducing runtime weight.
- Overall, Void’s architecture successfully merges classic IDE design with modern AI functionality.

AI Collab

- Used ChatGPT (GPT-5) as an AI teammate to help plan and organize the report.
- Included the official project guidelines and our team outline in prompts.
- AI produced a detailed report structure with section summaries and estimated lengths.
- Helped us connect Void’s architecture to Visual Studio Code and Electron.
- Only one member interacted with the AI; group reviewed and edited all outputs.
- AI contributed mainly to organization and outline development.
- Improved efficiency and clarity but needed fact-checking and context for accuracy.
- Best used as a research and planning assistant, not a substitute for technical work.

References

[1] “Understanding Visual Studio Code Architecture.” (n.d.). *Medium.* Retrieved from https://franz-ajit.medium.com/understanding-visual-studio-code-architecture-5fc411fca07 Medium

[2] “VSCode · Delft Students on Software Architecture – delftswa.” (n.d.). *Delft SWA / DESOSA.* Retrieved from https://delftswa.gitbooks.io/desosa-2017/content/vscode/chapter.html DelftSwa

[3] “VSCode – From Vision to Architecture.” (2021). *DESOSA Project.* Retrieved from https://2021.desosa.nl/projects/vscode/posts/essay2/ DESOSA

[4] “Void Editor – Architecture Overview.” (n.d.). *DeepWiki.* Retrieved from https://deepwiki.com/voideditor/void/1.1-getting-started#architecture-ov8631-AE22erview

[5] “Visual Studio Code – Wiki.” (n.d.). *GitHub.* Retrieved from https://github.com/microsoft/vscode/wiki/

[6] “Void – GitHub Repository.” (n.d.). *GitHub.* Retrieved from https://github.com/voideditor/void/

[7] “Electron: Things to Watch Out for Before You Dive In.” (n.d.). *Medium.* Retrieved from https://medium.com/@vishaldwivedi13/electron-things-to-watch-out-for-before-you-dive-in-e1c23f77f38f medium.com

[8] “Void IDE: The Comprehensive Guide to the Open-Source Cursor Alternative.” (n.d.). *Medium.* Retrieved from