GROUP 13

THE VOID MANUAL

Video presentation link: https://youtu.be/61bnHnRBMSk

https://the-void-manual.vercel.app/#

GROUP 13

TEAM MEMBERS

Aiden Ramezani
Leader

Divyanshi Agarwal
Presenter

Kanika Poonia
Presenter

Vedansh Bhatt

Bella Xu

Sameer Bandha

https://the-void-manual.vercel.app/#

INTRODUCTION & PURPOSE

Introduction

We extend the Void Editor with built-in real-time collaborative editing, so multiple developers can edit the same file simultaneously. This enhancement is grounded in our A1 conceptual model and A2 concrete architecture.

Purpose

The report explains how collaboration changes Void's subsystems and connectors, and how it affects key NFRs like performance, scalability, modifiability, and security. It also compares architectural options and motivates a recommended design.

https://the-void-manual.vercel.app/#

GROUP 13

ENHANCEMENT SUMMARY: REAL-TIME COLLABORATIVE EDITING

What the Feature Does

- Multiple users edit the same file
- Live cursors & synchronized updates
- New Collaboration Service subsystem
- Integrated with editor, files, security, config

https://the-void-manual.vercel.app/#

GROUP 13

REPORT ROADMAP

- Updated architectures
- Feature behavior & use cases
- Alternatives + SAAM analysis
- Concurrency, risks, testing, conclusion

https://the-void-manual.vercel.app/#

GROUP 13

NEW SUBSYSTEM - COLLABORATION SERVICE

Core Responsibilities:

- Manage collaborative editing sessions
- Maintain local replica of shared document state
- Apply remote edits and broadcast local changes
- Integrate seamlessly with existing Editor Component
- Track user presence and cursor positions
- Handle network communication (WebSocket/VS Code remote APIs)

https://the-void-manual.vercel.app/#

NEW SUBSYSTEM - COLLABORATION SERVICE

Key Interfaces

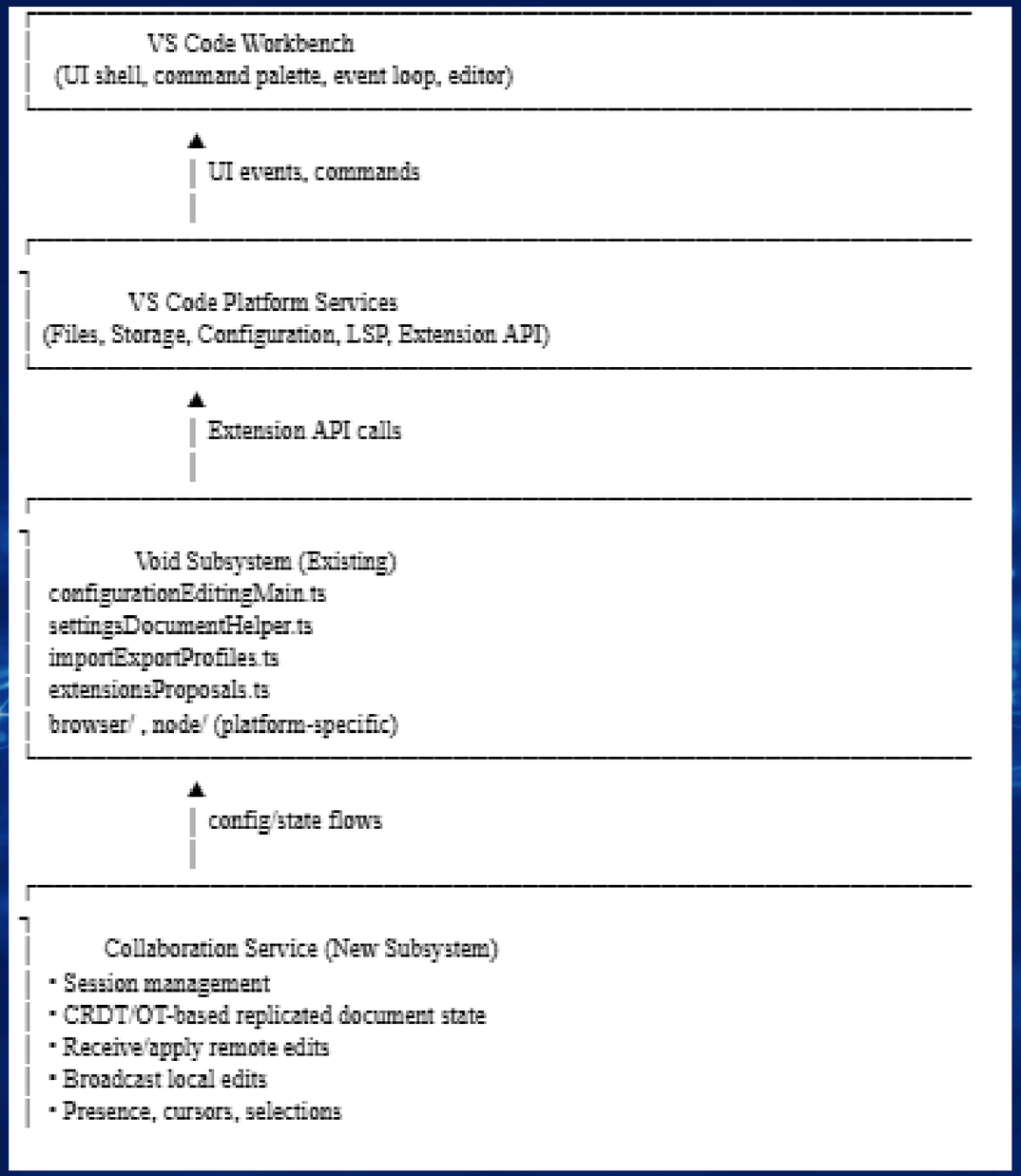
Inputs:

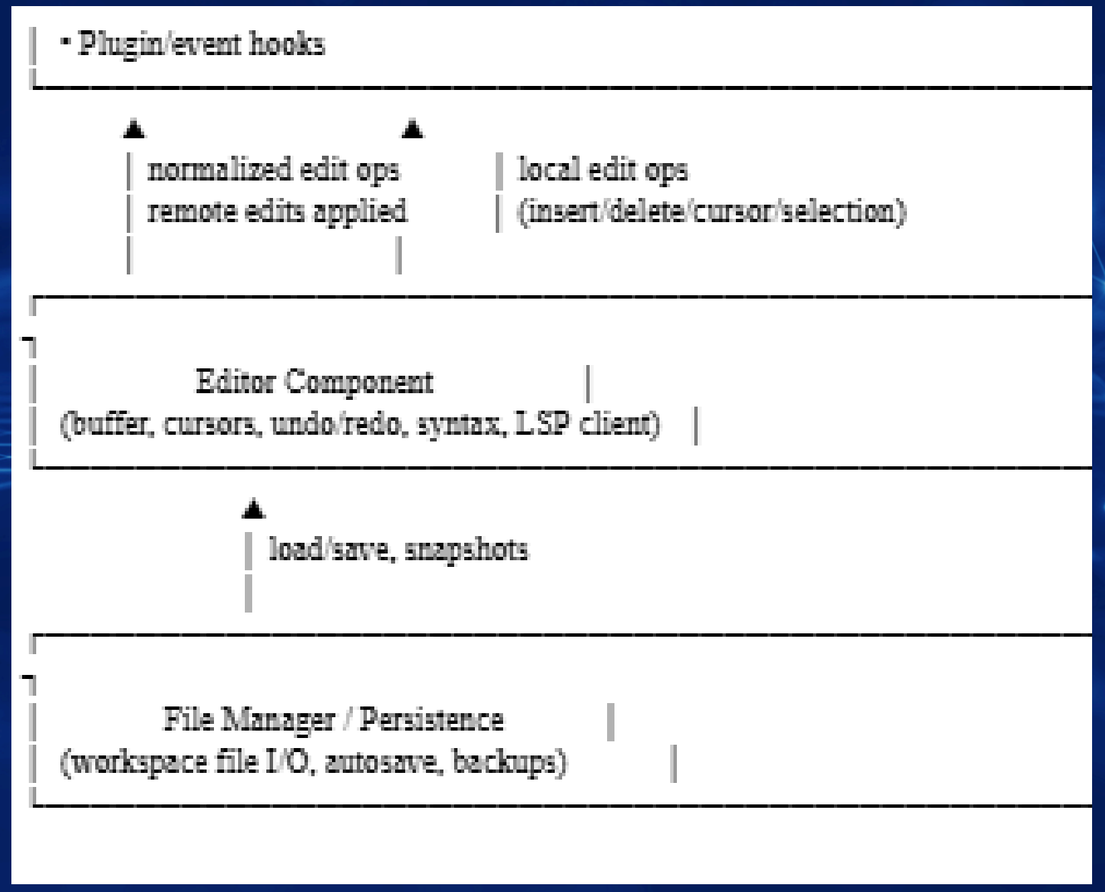
- Operations: insert, delete, cursorMove, selectionChange

Outputs:

- Normalized operations → Editor Component
- Event hooks → Plugin System

Role: Acts as mediator between local editor and remote collaborators


VS Code Workbench
(UI shell, command palette, event loop, editor)
UI events, commands
VS Code Platform Services
(Files, Storage, Configuration, LSP, Extension API)
Extension API calls
Void Subsystem (Existing)
configurationEditingMain.ts
settingsDocumentHelper.ts
importExportProfiles.ts
extensionsProposals.ts
browser/ , node/ (platform-specific)
config/state flows
Collaboration Service (New Subsystem)
• Session management
• CRDT/OT-based replicated document state
• Receive/apply remote edits
• Broadcast local edits
• Presence, cursors, selections


• Plugin/event hooks
normalized edit ops
remote edits applied
local edit ops
(insert/delete/cursor/selection)
Editor Component
(buffer, cursors, undo/redo, syntax, LSP client)
load/save, snapshots
File Manager / Persistence
(workspace file I/O, autosave, backups)

ENHANCEMENT DESCRIPTION

Goals

- Native real-time collaboration in Void
- “Feels local” for all participants

Motivation

- Remote / distributed teams
- Built-in, not an external plugin

https://the-void-manual.vercel.app/#

KEY FEATURES

- Concurrent edits, live updates
- Conflict-free sync (CRDT / OT)
- Remote cursors & presence
- Sessions: create / join / leave / resume
- Integrated security checks

https://the-void-manual.vercel.app/#

HOW IT WORKS INTERNALLY

- New Collaboration Service subsystem
- Maintains shared document state
- Broadcasts & applies operations
- Handles metadata + networking

ARCHITECTURAL RATIONALE & FUTURE

- First-class subsystem, not a plugin
- Consistent buffer & undo/redo semantics
- Clear separation: editing vs. sync
- Foundation for shared terminals, debugging, etc.

https://the-void-manual.vercel.app/#

USE CASE - JOINING A COLLABORATION SESSION

Preconditions:

- VOID Editor running
- Session host has created collaboration session

Postconditions:

- User fully synced with session
- Background workers maintain real-time connectivity

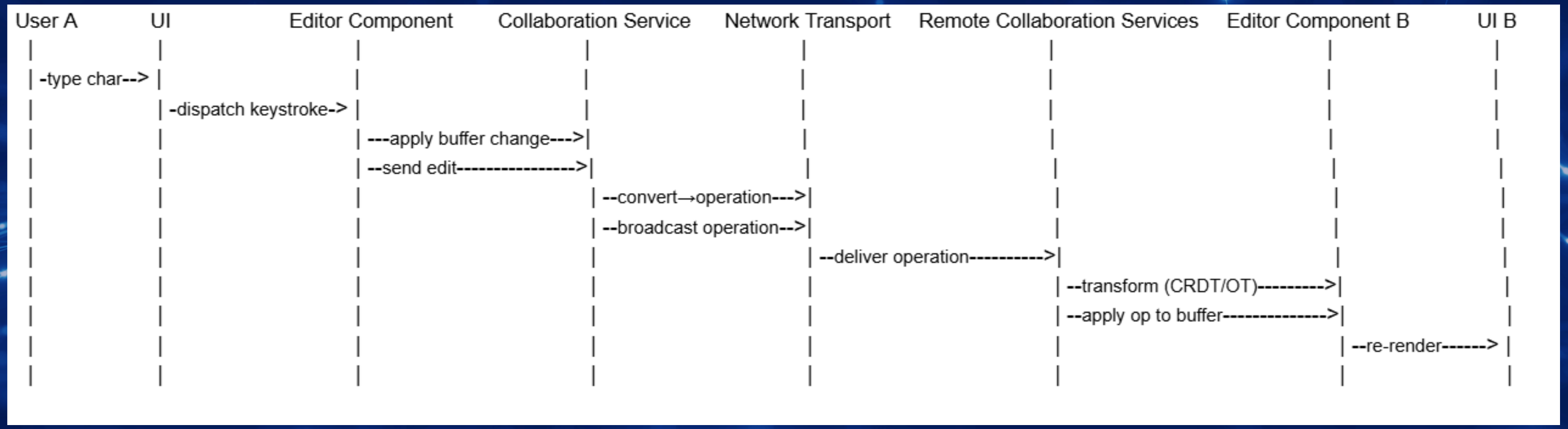
Flow:

- User enters invitation code/URL
- Collaboration Service requests session metadata from host
- Host validates permissions via Security Layer
- Collaboration Service loads shared document state
- Editor Component initializes shared buffer
- UI displays remote cursors

GROUP 13

USE CASE - CONCURRENT EDITING

Goal: Two users edit the same file simultaneously; both see updates instantly.



https://the-void-manual.vercel.app/#

ARCHITECTURAL MODIFICATIONS

Big Picture

- Cross-cutting change
- Collaboration touches core subsystems

KEY SUBSYSTEMS CHANGED

- Editor Component → accepts remote ops
- Collaboration Service → new subsystem
- File Manager & Persistence → collab-aware
- Config & Security → session policies

IMPACTED FILES & DIRECTORIES

Platform-Level Impact

- src/vs/editor/*
- src/vs/platform/files/*
- src/vs/platform/storage/*
- src/vs/platform/configuration/*
- src/vs/platform/security/*

NEW COLLABORATION PACKAGE & QA IMPACT

New Void Package

- void/collaboration/
- collaborationService.ts
- operations.ts
- sessionManager.ts

Quality Impact

- Modifiability / Maintainability
- Reliability / Testability

https://the-void-manual.vercel.app/#

GROUP 13

SAAM ANALYSIS: COMPARING APPROACHES

Alternative 1: CRDT-Based Collaboration

- Conflict-free Replicated Data Types
- Each client maintains local replica
- Asynchronous propagation with guaranteed convergence
- Key strength: Offline-first, decentralized

Alternative 2: Operational Transformation (OT)

- Central coordination server transforms operations
- Used in Google Docs, mature ecosystem
- Server maintains global operation history
- Key strength: Efficient under high concurrency

https://the-void-manual.vercel.app/#

GROUP 13

CRDT-BASED COLLABORATION

How it Works:

- Local edits applied immediately
- Changes propagate asynchronously to peers
- Mathematical guarantees ensure convergence

Advantages:

- No central server needed
- Strong eventual consistency
- Works offline, syncs later
- Resilient to network partitions

Disadvantages:

- Higher memory usage (metadata per operation)
- Larger network payloads

Best for: Decentralized, offline-tolerant collaboration

https://the-void-manual.vercel.app/#

GROUP 13

OPERATIONAL TRANSFORMATION (OT)

How it Works:

- Operations sent to central coordination server
- Server transforms operations based on global history
- Transformed ops broadcast to all clients

Advantages:

- Highly efficient with many concurrent users
- Low memory footprint
- Mature, proven approach (Google Docs)

Disadvantages:

- Requires central server (single point of failure)
- Complex transformation logic
- Poor offline support

Best for: Server-based, always-online scenarios

https://the-void-manual.vercel.app/#

Stakeholder	Primary Needs
Developers	Low latency, correctness, fluid workflow
Team Leads	Reliability, consistency, stable coordination
Instructors	Predictable real-time collaboration
Plugin Developers	Stable APIs, extensibility
System Integrators	Clear interfaces, security
Performance Testers	Responsiveness, concurrency handling
DevOps/Maintainers	Deployment stability, monitoring

GROUP 13

NFRS DRIVING ARCHITECTURE DECISIONS

7 Critical NFRs for Evaluation:

1. Latency
2. Scalability
3. Offline Tolerance
4. Maintainability
5. Modifiability
6. Reliability
7. Security

These NFRs form the basis for comparing CRDT vs OT

https://the-void-manual.vercel.app/#

HEAD-TO-HEAD COMPARISON (1-10 SCALE)

NFR	CRDT SCORE	OT SCORE	NOTES
Latency	8	9	OT operations are smaller and propagate faster.
Scalability	9	7	CRDT avoids central server bottlenecks.
Reliability	8	6	CRDT ensures reliable convergence under failures.
Maintainability	9	6	OT transformation is more complex.

HEAD-TO-HEAD COMPARISON (1-10 SCALE)

NFR	CRDT SCORE	OT SCORE	NOTES
Modifiability	10	4	CRDT's model is easier to extend.
Offline Support	8	8	CRDT is inherently offline-first.
Security	9	7	Security depends mostly on session authentication.

RECOMMENDATION: CRDT APPROACH

Aligns with user needs:

- Works seamlessly with unstable networks
- Enables true offline editing
- No single point of failure

Architectural benefits:

- Naturally scales without central bottleneck
- Easier to maintain and extend
- More reliable convergence under failures

GROUP 13

ARCHITECTURE & TEAM PROCESS

- Ownership: void/collaboration/ owned by core Void team
- Concurrency focus: shared buffer & session logic reviewed centrally
- New workflow: replayable collab test cases + stricter code reviews
- Cross-team coordination: editor, platform, and security teams align on APIs & policies

https://the-void-manual.vercel.app/#

KEY RISKS

- Network instability
- Unauthorized session access
- Performance under load
- Buffer consistency bugs
- Concurrent save conflicts

INTERACTIONS & ARCHITECTURAL IMPACT

Feature Interactions

- Syntax highlighting & LSP
- File Manager & plugins
- Undo/redo per user

Impact

- Maintainability, reliability, UX

https://the-void-manual.vercel.app/#

LIMITATIONS OF OUR FINDINGS

- Analysis is model-based, not from a live prototype
- Assumes a well-behaved network and correct CRDT implementation
- Performance & memory costs of CRDT are only estimated, not measured
- SAAM focuses on typical collab scenarios, not extreme edge cases

GROUP 13

WRAPPING UP: COLLABORATION IN VOID

What We've Achieved:

- Designed a real-time collaborative editing feature for VOID
- Integrated new Collaboration Service with existing architecture
- Systematically evaluated CRDT vs OT approaches
- Selected CRDT as optimal solution based on NFRs

Key Outcome: Strengthens VOID Editor for distributed, synchronous development while maintaining architectural integrity

https://the-void-manual.vercel.app/#

KEY TAKEAWAYS FROM THIS PROJECT

5 Critical Lessons:

- Collaborative editing requires strict buffer ownership rules
- Conflict-resolution models differ drastically
- VS Code platform integration is essential
- Manage architectural drift early
- CRDT simplifies offline-first architecture

https://the-void-manual.vercel.app/#

GROUP 13

AI COLLABORATION REPORT

AI Teammate: OpenAI GPT-4

Used for: Drafting sections, rewriting for clarity, generating tables, assisting with SAAM construction

Quality Control:

- All technical claims manually verified
- Architectural decisions human-driven
- AI outputs reviewed by entire team

Contribution Split:

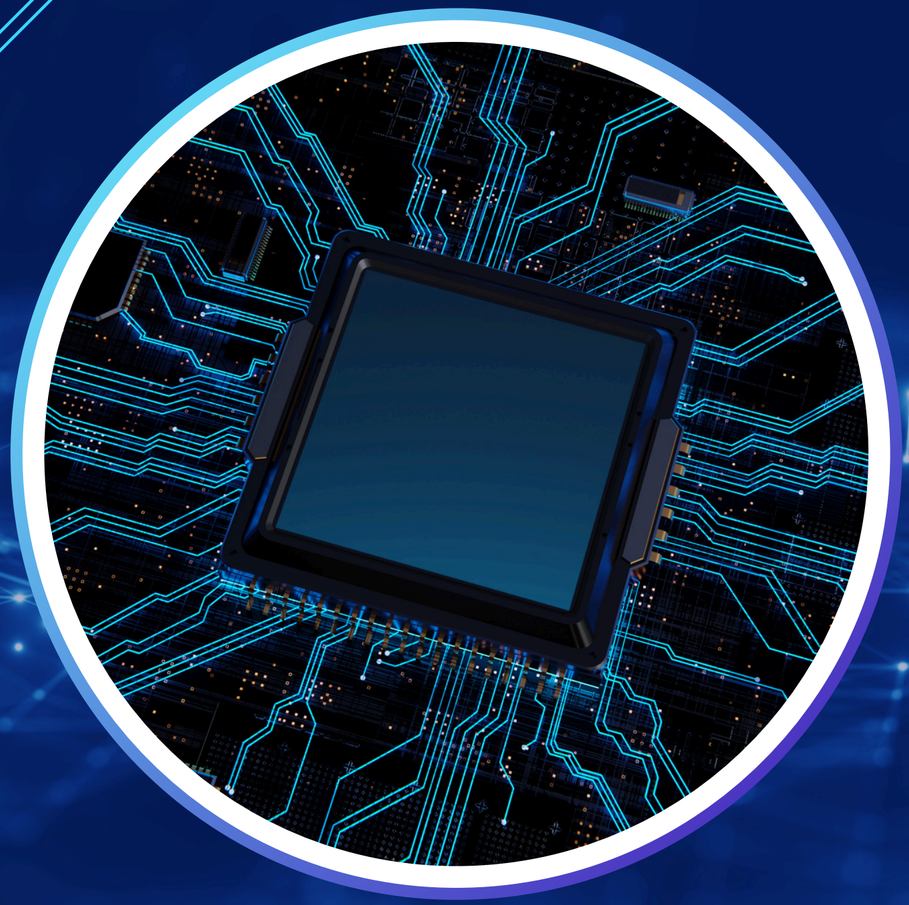
- ~85% human (analysis, design, decisions)
- ~15% AI (editing, structure, drafting)

Key Reflections:

- Strengths: Unified writing style, accelerated drafting
- Limitations: Cannot make architectural judgments
- Insight: Effective for structure/refinement, not technical analysis

https://the-void-manual.vercel.app/#

GROUP 13



THANK YOU

https://the-void-manual.vercel.app/#