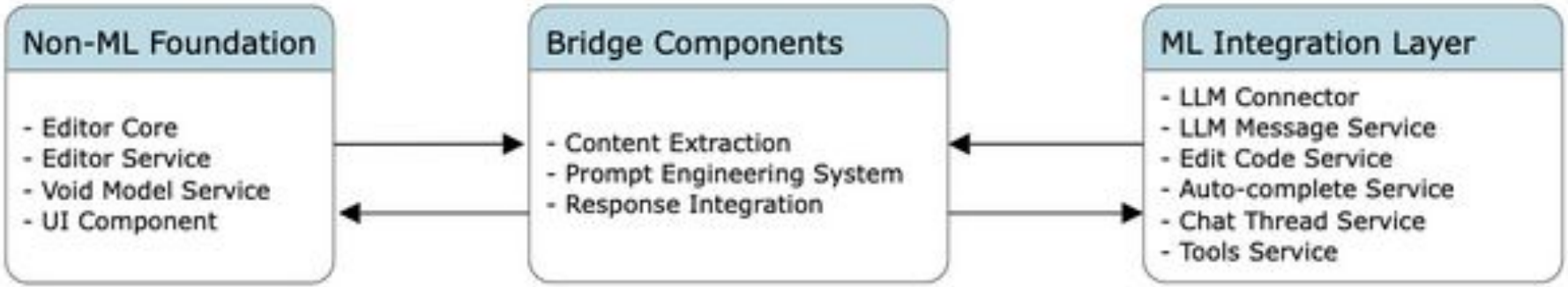
https://www.youtube.com/watch?v=-RGLYynPQGQ&feature=youtu.be

Group Members

Ibrahim Kettaneh (ibrahim.kettaneh@queensu.ca)
Sophie Liang (22whr@queensu.ca)
Noelle Morley (25gdb@queensu.ca) Presenter 2
Annika Tran (23LM5@queensu.ca)
Nicole Wu(22ll20@queensu.ca) Group Leader
Joshua Zheng (23SBN1@queensu.ca) Presenter 1

Architectural Style

- Primary Style: Layered Architecture
- Supporting Style: Implicit Invocation (Publish/Subscribe)
- Base Framework: Forked from Visual Studio Code
- Purpose: Enables modularity, scalability, and AI integration



Layered Subsystems

- Foundational Core:
  - Inherits from VS Code’s core modules
  - Handles editor functions, UI rendering, and file management
- Bridge Layer:
  - Mediates between the core and AI components
  - Manages API calls, model selection, and prompt dispatching
- ML Integration Layer:
  - Interfaces with external or local AI providers
  - Handles token management and response parsing

Interactions between subsystems

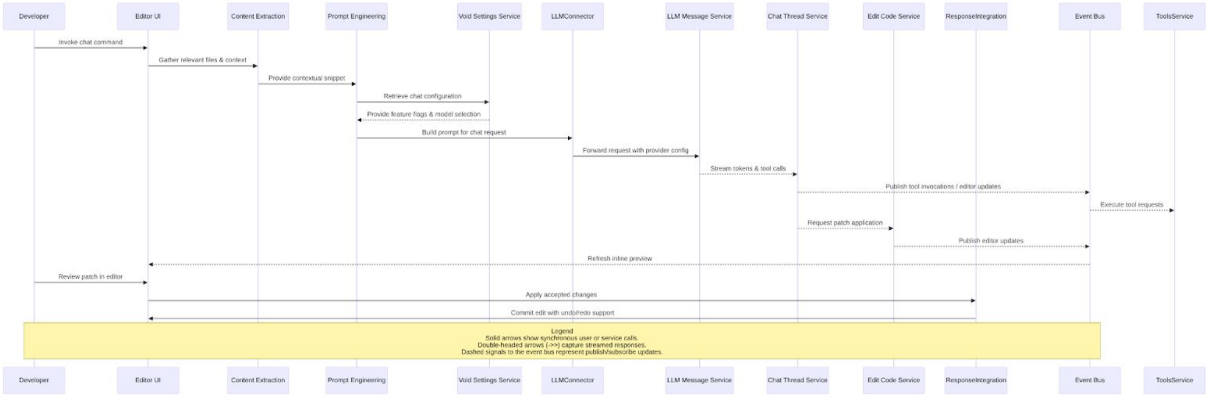
- Layered Communication: maintains clarity and modular design
- Bridge layer abstraction: reduced dependency between IDE and AI backends
- Pub/Sub messaging: enables asynchronous updates and event driven behaviour
- Security concerns: User control & System stability

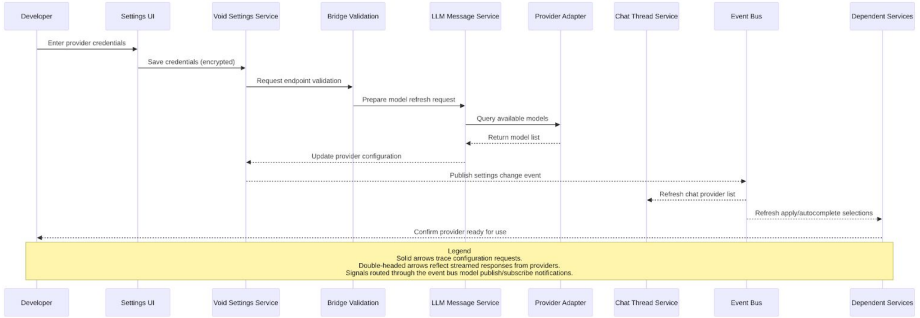
Concurrency & Open source model

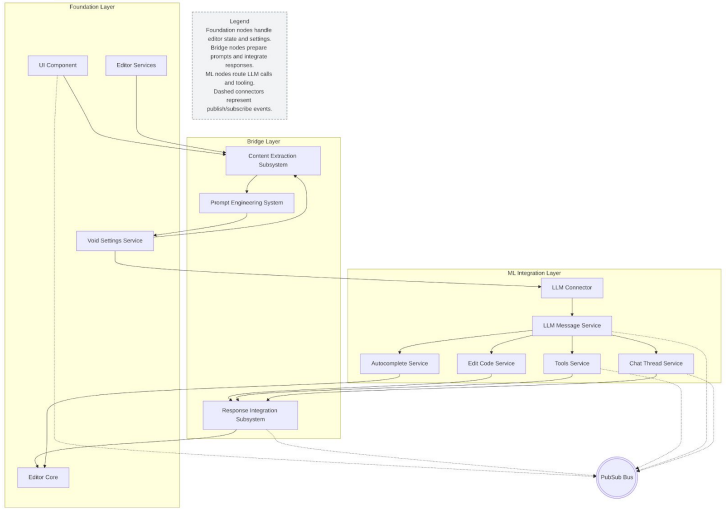
- Concurrency
  - Pub/Sub enables simultaneous processes (editor + AI feedback)
  - Avoids blocking operations through event driven communication
- Team collaboration
  - Layered design allows a team to focus on a layer
  - Open source model encourages distributed contributions

Quickened development cycle + potential for modular testing

One sequence diagram is presented and matches the conceptual architecture box-and-arrow diagram







Derivation Process

Methods uses:

- Research void’s GitHub source code page
- Analyze codebase guide document in the repository and consulted with external sources such as Kumar’s and Ahwan’s respective articles on Void
- Deeper analyzation analysis of the source code

Combine the key findings concluded identified pub/sub as another style that inspired the architecture of Void.

Discussion of considered alternatives (and why they went with the current one)

- Client/Server Architecture: Good as it reduces resources on local device, but sending commands to remote servers increases latency, and sending code to remote server poses security concerns
- Object-Oriented Architecture: good for reusing and simplifies unit testing, but has issues with tight coupling between editor, ML, and bridge logic.

Limitations Of Reported Findings

- Conceptual architecture will most likely be different than the concrete architecture
- Hard to get accurate view without being the actual developers of Void

Learned Lessons

- Void exemplifies a new generation of IDEs designed to work with AI
- Built on VS Code’s foundation, Void’s layered and modular architecture enables seamless AI integration
- Clear layer responsibilities
- The implicit invocation (pub/sub) model allows components to remain independent
- The design reflects a modern shift toward concurrent, scalable architectures that support rapid evolution and responsiveness.
- Global, community-driven development promotes innovation, transparency, and continuous improvement of the platform.