PHENIKAA UNIVERSITY - SCHOOL OF COMPUTING

Group Submission Software Architecture

Project Title: Kanban Board Application

Group Submission: N02 – Group 2

Name	Student ID	Position Title
Nguyễn Xuân Mạnh	23010045	Document + Diagram & Quality Assurance

Class: CSE703110-1-2-25(N02)

Instructor in charge : Vũ Quang Dũng

Hà Nội - Feb 7, 2026

Architecture Design & Documentation

Project: Kyte - Kanban Board Application
Course: Software Architecture
Student: Nguyễn Xuân Mạnh (23010045)
Role: Architectural Designer & QA
Class: N02 - Group 2
Date: Feb 07, 2026

1. Executive Summary

As the Architectural Designer and QA lead for the Kyte project, my primary mission was to establish a robust, scalable, and secure structural framework for the application. My work bridged the gap between high-level functional requirements and concrete technical implementation. I was responsible for the end-to-end design of the system's architecture, translating complex business needs into detailed visual models using the C4 framework, and authoring a professional-grade technical report. Furthermore, I designed the comprehensive verification strategy that validated the system's reliability, concurrency, and security attributes.

2. Contribution Scope & Responsibilities

My contributions spanned the entire architectural lifecycle, focusing on the following key areas:

- **Architectural Modeling:** Implementation of the full C4 model (Context, Container, Component, and Code diagrams) to ensure multi-level system clarity.
- **Use Case Engineering:** Designing structured use case models that mapped user interactions to back-end modules, ensuring complete functional coverage.
- **Documentation Leadership:** Authoring the 10+ section Final Architecture Report, ensuring technical accuracy, professional tone, and clear communication of architectural decisions.
- **QA Strategy & Verification:** Defining the verification matrix and mapping Non-Functional Requirements (NFRs) to empirical test evidence to prove system quality.
- **Visual Asset Management:** Creating and maintaining 20+ professional diagrams in Draw.io, including ERDs and detailed layered architecture views.

3. Detailed Architectural Design & Visualization

3.1 C4 Model Implementation (Levels 1-4)

I adopted the C4 model to provide a rigorous hierarchy of abstractions:

- **Level 1 (System Context):** I mapped Kyte's external dependencies, including GitHub OAuth for identity, Resend for transactional messaging, and SeaweedFS for distributed storage.
- **Level 2 (Container Diagram):** I designed the decoupling between the React SPA and the Bun/Elysia API, focusing on the "Eden Treaty" communication protocol as a core architectural constraint.
- **Level 3 (Component Diagram):** I detailed the internal structure of the API modules, enforcing the Controller-Service-Repository pattern to ensure separation of concerns.
- **Level 4 (Code Diagram):** I authored UML class diagrams for critical services (Task, Board, WebSocket), documenting interfaces and data flow logic.

3.2 Dynamic Modeling (Behavioral Views)

Beyond static structure, I modeled the system's dynamic behavior:

- **Real-Time Orchestration:** Designed the flow between the Internal Event Bus and the WebSocket Bridge to ensure low-latency synchronization.
- **Concurrency Flow:** Modeled the Optimistic Locking sequence to visualize how the system handles simultaneous updates via versioning.

4. Documentation & Technical Reporting

4.1 Requirement Analysis & Specification

I performed a deep dive into the system's requirements:

- **ASR Definition:** Identified and documented Architecturally Significant Requirements (ASRs) like multi-tenancy isolation and UI independence.
- **NFR Specification:** Defined measurable goals for performance (<100ms latency) and scalability (stateless API).

4.2 Professional Reporting

I focusing on:

- Technical rationale for the Modular Monolith pattern.
- Detailed API specifications with request/response payload examples.
- Comprehensive Testing & Verification methodology based on real-world integration.

5. Quality Assurance (QA) & Verification Strategy

5.1 Verification Traceability Matrix

I established a formal matrix to link architectural goals to testing evidence:

- **Security Verification:** Designed scenarios to prove RBAC enforcement (e.g., verifying 403 responses for unauthorized workspace access).
- **Reliability Verification:** Defined the criteria for testing request idempotency and optimistic concurrency control.

5.2 Quality Attribute Proof (NFR Mapping)

I mapped the system's NFRs to specific testing tiers, providing a "Pillar" based explanation of how the "Real-World Integration Testing" approach ensures data integrity and high fidelity.

6. Lessons Learned & Reflections

- **Abstract to Concrete:** Learning to translate abstract architectural patterns (like Modular Monolith) into concrete documentation that guides development was a key growth area.
- **The Value of Visualization:** I discovered that clear diagrams are not just documentation they are tools for finding design flaws before a single line of code is written.
- **Documentation-as-Code:** Maintaining all documentation and diagrams within the repository ensured that the architecture stayed synchronized with the evolving implementation.

7. Future Architectural Improvements

- **Automated Diagramming:** Moving towards PlantUML or Mermaid integration for auto-generating diagrams from code structure.
- **Advanced Sequence Diagrams:** Implementing more granular behavioral models for edge-case scenarios like offline synchronization and conflict resolution.
- **Infrastructure Modeling:** Expanding Level 2 diagrams to include detailed cloud deployment and network topology.