PHENIKAA UNIVERSITY

PHENIKAA SCHOOL OF COMPUTING

WEEKLY REPORT

TOPIC: DEVELOPING A MOBILE MESSAGING APPLICATION

Course Class:	Software Architecture-1-2-25 (N01)	
Instructor:	M.Sc. Vũ Quang Dũng	
Group:	04	
Group Members:	Đỗ Trịnh Lệ Quyên	23010485
	Vũ Viết Huy	23010699
	Nguyễn Mạnh Cường	23010064

Hanoi, December 16, 2025

CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 INTRODUCTION

1.1 Problem Statement

In the Industry 4.0 era, while instant messaging applications have become ubiquitous, users—especially students—face significant challenges in discovering quality study communities amidst information overload. The core problem addressed by this project is the development of a Mobile Messaging Application that not only supports high-performance Real-time Communication but also integrates an Intelligent Recommendation System. This system automatically suggests relevant "Public Channels" based on user interest tags (e.g., #Coding, #Piano).

1.2 Project Objectives

The primary objective of Lab 2 is to transition from the requirements elicitation phase (Lab 1) to the architectural design phase. Specifically, the team aims to design the Logical View of the system using the Layered Architecture Pattern. This structure is intended to manage system complexity by enforcing a strict separation of concerns, ensuring that the Recommendation Logic is decoupled from the User Interface and Data Access mechanisms.

2 LAYERED ARCHITECTURE SPECIFICATION

2.1 Layer Definition & Responsibility Strategy

Based on the principles of the Layered Architecture pattern, the system is structured into four distinct layers. A strict downward dependency rule is enforced: a layer can only interact with the layer directly below it.

Layer	Primary Responsibility	Key Artifacts
1. Presentation Layer	Handles all interactions with the mobile user. It is responsible for rendering the UI, capturing user inputs (touches, swipes), and displaying data (JSON) received from the Business Layer. It does not contain core business rules.	RecommendationController, ChatViewModel, Mobile Views/Activities.

2. Business Logic Layer	Serves as the core of the application. It encapsulates business rules, such as the recommendation algorithms (matching user tags to channel tags) and real-time messaging logic. It orchestrates data flow and enforces security checks.	RecommendationService, ChatService, AuthService.
3. Persistence Layer	Abstracts the underlying database technology. It translates high-level business requests into specific database queries (SQL/NoSQL) and maps the raw results back to domain entities/objects.	ChannelRepository, MessageRepository, Data Transfer Objects (DTOs).
4. Data Layer	The physical storage infrastructure ensures data durability and consistency.	PostgreSQL (User/Channel Data), MongoDB (Chat Logs), Redis (Caching).

Table 2.1: Layer Definition and Responsibility Allocation Strategy

2.2 Component Identification: Recommendation Subsystem

To demonstrate the architectural structure, the team focused on the specific feature: "View Recommended Channels" (FR-12).

1. Presentation Layer Component: RecommendationController
    - Responsibility: Receives HTTP GET requests from the mobile client, extracts the User ID from the authentication token, and invokes the IRecommendationService interface.
2. Business Logic Layer Component: RecommendationService
    - Responsibility: Implements the core recommendation logic. It retrieves the user's interest tags, matches them against active channels, filters out joined groups, and sorts the results by relevance.

- Provided Interface: IRecommendationService.

3. Persistence Layer Component: ChannelRepository

    - Responsibility: Constructs and executes optimized database queries (e.g., SELECT * FROM Channels WHERE tags IN ...) and maps the result set to ChannelEntity objects.

    - Provided Interface: IChannelRepository

2.3 Data Flow Analysis

The typical request flow for a user viewing recommendations is defined as follows [21]:

1. Client Request: The user opens the "Discover" tab. The Mobile App sends a request to the Presentation Layer (RecommendationController).
2. Layer 1 -> Layer 2: The Controller calls getRecommendedChannels() on the Business Logic Layer (RecommendationService).
3. Layer 2 -> Layer 3: The Service processes the User ID and calls findChannelsByTags() on the Persistence Layer (ChannelRepository).
4. Layer 3 -> Layer 4: The Repository executes a query against the Data Layer (Database).
5. Response Path: The raw data flows back up: Data -> Persistence (mapped to Entities) -> Business (filtered/sorted to DTOs) -> Presentation (JSON) -> Client Display.

2.4 Logical View Modeling (Component Diagram)

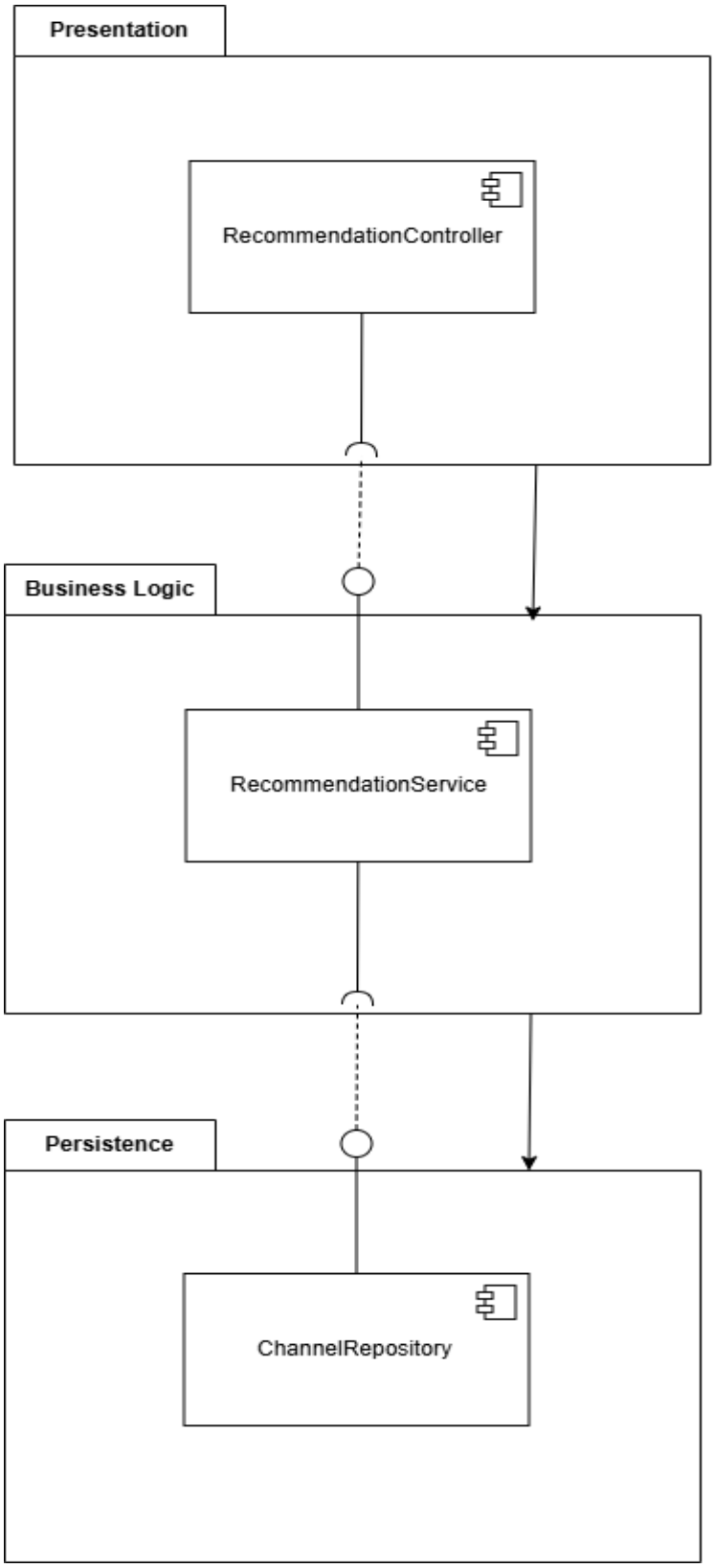

Figure 2.1: UML Component Diagram - Recommendation Subsystem (Logical View)

3 ARCHITECTURAL ANALYSIS

3.1 Design Challenge

The primary design challenge identified in Lab 1 is integrating two conflicting operational modes: Real-time Messaging (requiring low latency and persistent connections) and Intelligent Recommendations (requiring complex data processing). A poorly structured architecture would lead to tight coupling, where changes to the recommendation algorithm could inadvertently break the messaging functionality.

3.2 Impact of ASRs on Layered Design

The Architecturally Significant Requirements (ASRs) derived from Lab 1 have directly influenced the logical design:

- ASR-3: Modifiability (Extensibility):
    - *Requirement:* New features (e.g., advanced AI matching) must be added without disrupting existing modules.
    - *Design Impact:* The use of interfaces (e.g., IRecommendationService) creates a loose coupling between layers. This ensures that the internal logic of the RecommendationService can be refactored or replaced entirely without affecting the RecommendationController or the mobile client code.
- ASR-2: Security:
    - *Requirement:* Access to administrative functions and private data must be strictly controlled.
    - *Design Impact:* The Business Logic Layer acts as a centralized security gatekeeper. By forcing all requests to pass through this layer, the architecture ensures that authorization rules are consistently applied, regardless of the access point.
- ASR-1: Performance (Real-time):
    - *Requirement:* Support for high-concurrency WebSocket connections.
    - *Design Impact:* While the Layered Architecture organizes the code, the Business Layer is designed to be Stateless to support horizontal scaling

(Load Balancing), satisfying the high availability requirement defined in Lab 1.

4 CONCLUSION

Through the activities in Lab 2, the team has successfully translated the functional requirements of the ShopSphere (Messaging App) project into a concrete Logical Architecture. By decomposing the "Recommendation System" into specific components (Controller, Service, Repository) and defining their interactions via interfaces, the design achieves the critical goal of Separation of Concerns. This logical view provides a stable and scalable foundation for the subsequent Implementation Phase (Lab 3).