PHENIKAA UNIVERSITY

PHENIKAA SCHOOL OF COMPUTING

COURSE: SOFTWARE ARCHITECTURE

TOPIC: DEVELOPING A MOBILE MESSAGING APPLICATION

GROUP 04

Đỗ Trịnh Lệ Quyên	23010485	K17-KTPMEL2	Leader
Vũ Viết Huy	23010699	K17-CNTT7	Member
Nguyễn Mạnh Cường	23010064	K17-KTPMEL1	Member

Hanoi, December 2025

CONTENTS

1. INTRODUCTION

1.1. Problem Statement

In the era of Industry 4.0, the rapid growth of smart mobile devices (smartphones) and Internet connectivity (4G/5G/Wi-Fi) has fundamentally transformed the way people communicate. Traditional communication methods such as SMS or regular phone calls are gradually being replaced by internet-based messaging applications like Facebook Messenger, Zalo, WhatsApp, Telegram, etc., thanks to their convenience, low cost, and support for multimedia transmission.

Modern communication needs are no longer limited to sending text messages but also require instant responses (real-time), the ability to share images, videos, files, and especially high demands for security and user experience (UI/UX).

Additionally, with the increasing prevalence of online learning and remote work, the need to exchange knowledge and find study partners with similar interests is growing rapidly. However, most popular messaging applications (Zalo, Messenger) mainly focus on connecting people who already know each other. Users—especially students—often struggle to find quality communities on specialized topics such as programming, foreign languages, arts, or science to join and learn from.

Moreover, in the vast ocean of information, users often spend a lot of time searching for the right discussion groups that match their interests and skill levels. They need a tool that not only enables messaging but also acts as a guide, recommending the most suitable communities.

1.2. Reasons for Choosing the Topic

From this reality, researching and developing a mobile-based messaging application is both necessary and highly practical for Information Technology students. This project not only strengthens knowledge in mobile programming but also provides opportunities to work with real-time communication technologies, database management, and the design of Client–Server system architecture.

In addition, the key differentiator of this application compared to traditional chat platforms is the integration of an intelligent Recommendation System. This system automatically suggests relevant channels based on the user's interest profile, making community discovery, learning, and collaboration easier, more proactive, and more efficient.

For these reasons, our team decided to carry out the project: “Developing a Mobile Messaging Application with an Integrated Intelligent Recommendation System.”

1.3.Project Objectives

To successfully build a fully functional mobile messaging application (Android/iOS) that allows users to register, log in, and chat in real time, ensuring system stability and providing a user-friendly experience.

2. SOFTWARE REQUIREMENTS SPECIFICATION (SRS)

2.1. Functional Requirements (FRs)

ID	Description	Priority
FR-01	The system must allow users to Register and Login using authentication via phone number or email address.	Critical
FR-02	The system must allow users to send and receive text messages with other users in Real-time.v	Critical
FR-03	The system must allow users to search for friends by name or ID and manage their **Contact List** (add friend, unfriend).	High
FR-04	The system must support sending **multimedia files** (images, videos) and file attachments within conversations.	High
FR-05	The system must allow users to create **Group Chats**, add members to the group, and modify group names.	High
FR-06	The system must send **Push Notifications** to the device when new messages arrive, even if the application is running in the background or the screen is locked.	High
FR-07	The system must allow users to update their **Personal Profile** (avatar, display name)	Medium

FR-08	The system must categorize chat groups into Public Channels based on specific topics (e.g., Information Technology, Foreign Languages, History…).	High
FR-09	The system must allow users to **Search** and **Discover** public channels based on keywords or topic categories.	High
FR-10	The system must allow users to **Join** public channels without requiring an invitation from an Admin.	High
FR-11	The system must allow users to select interests/topics (Tags/Interests) during profile creation or when updating their profile (e.g., #Piano, #Biology, #Coding)	High
FR-12	The system must automatically recommend a list of suitable channels/groups based on the matching between the user's interest tags and the channel's topic tags	Medium

2.2. Non-Functional Requirements (NFRs)

ID	**Attribute**	**Description**	**Impact**
NFR-01	Performance (Latency)	The system must ensure text messages are delivered and displayed to the recipient (who is online) within **1.0 second** under stable network conditions.	Critical
NFR-02	Security (Integrity)	User passwords must be **hashed** before storage. Data transmission between the Client and Server must be encrypted using **SSL/TLS** protocols.	Critical
NFR-03	Scalability	The backend system must be capable of handling at least **1,000 concurrent connections** (users) simultaneously without service interruption.	High

NFR-04	Compatibility	The application must be compatible with and operate stably on mobile devices running **Android 8.0+** and **iOS 13.0+**.	High
NFR-05	Efficiency (Resource Usage)	The application must not consume more than **5% of the device's battery** per hour while running in background mode.	Medium

2.3.Yêu cầu quan trọng về kiến trúc (ASRs)

ASR ID	**Quality Attribute**	**Requirement Statement**	**Architectural Rationale**
NFR-01	Performance (Latency)	The system must maintain persistent WebSocket connections for **10,000 concurrent users** with message latency **under 200ms**.	This requirement negates the traditional HTTP Request-Response model and drives the adoption of an **Event-Driven Architecture** using Message Brokers (e.g., Redis or Kafka) to handle high-throughput real-time data.
NFR-02	Availability (Fault Tolerance)	If a server instance fails, client connections must automatically failover to a healthy instance without any message loss.	Directly necessitates a **Load Balancer** to distribute traffic and requires **Database Replication** strategies to ensure high availability and data persistence.
NFR-03	Modifiability (Extensibility)	Integrating new features such as "Sticker Packs" must not disrupt or require downtime for the core "Text Chat" functionality.	Promotes a **Microservices** or **Modular Monolith** design, where distinct modules (Chat, Call, Notification) operate independently, strictly

			following the Separation of Concerns principle.

2.4. Critical Architectural Requirements

2.4.1. Overall Use Case Diagram

This diagram provides the most comprehensive overview of the system, displaying only the core functional groups (Core Use Cases) without going into fine-grained details.

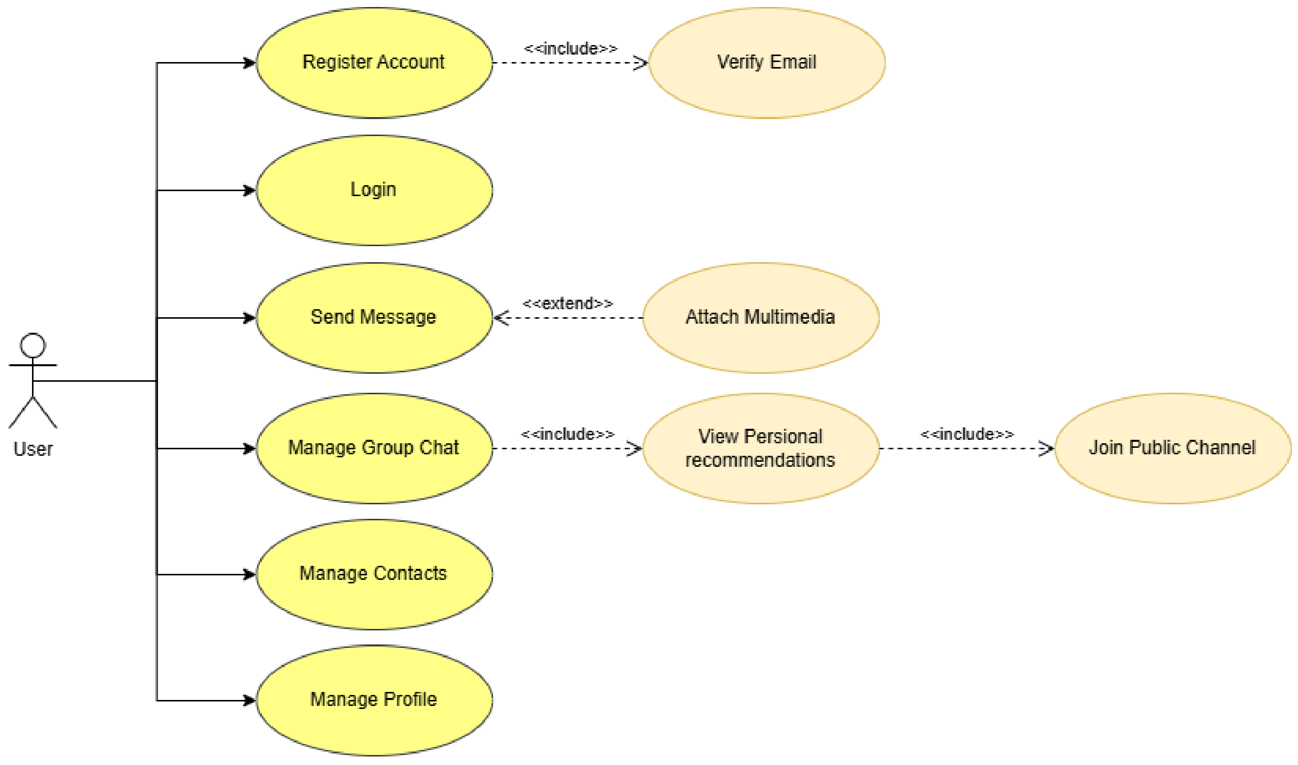

Figure 1.1: Overall Use Case Diagram of the Mobile Messaging Application

2.4.2. Account Management and Authentication Subsystem

This is the first subsystem that users interact with. The key highlight here is the mandatory **Include** relationship between **Register** and **Email Verification**.

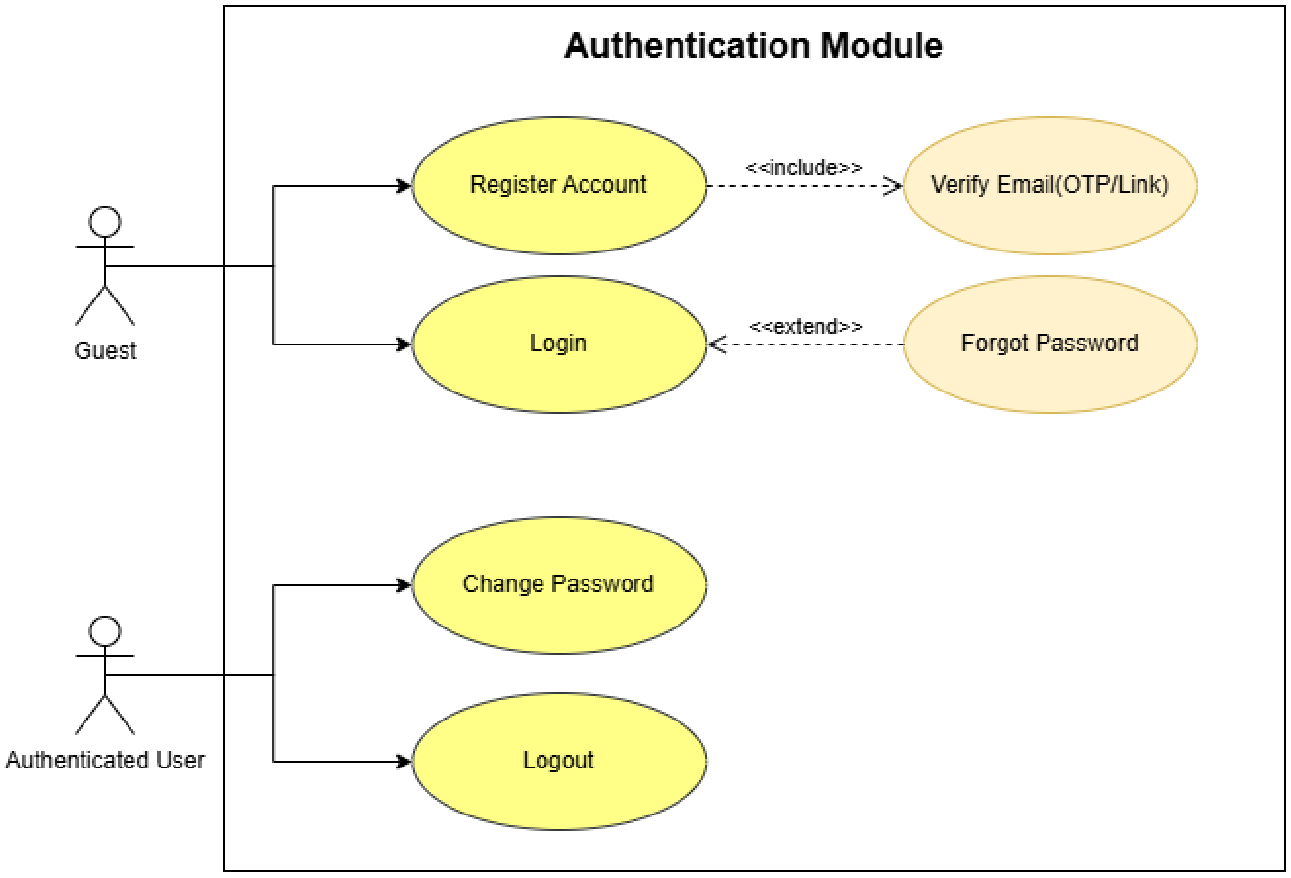

Figure 1.1.1: Account Management and Authentication Subsystem

2.4.3. Messaging Management Subsystem

This diagram focuses on the features related to message operations.

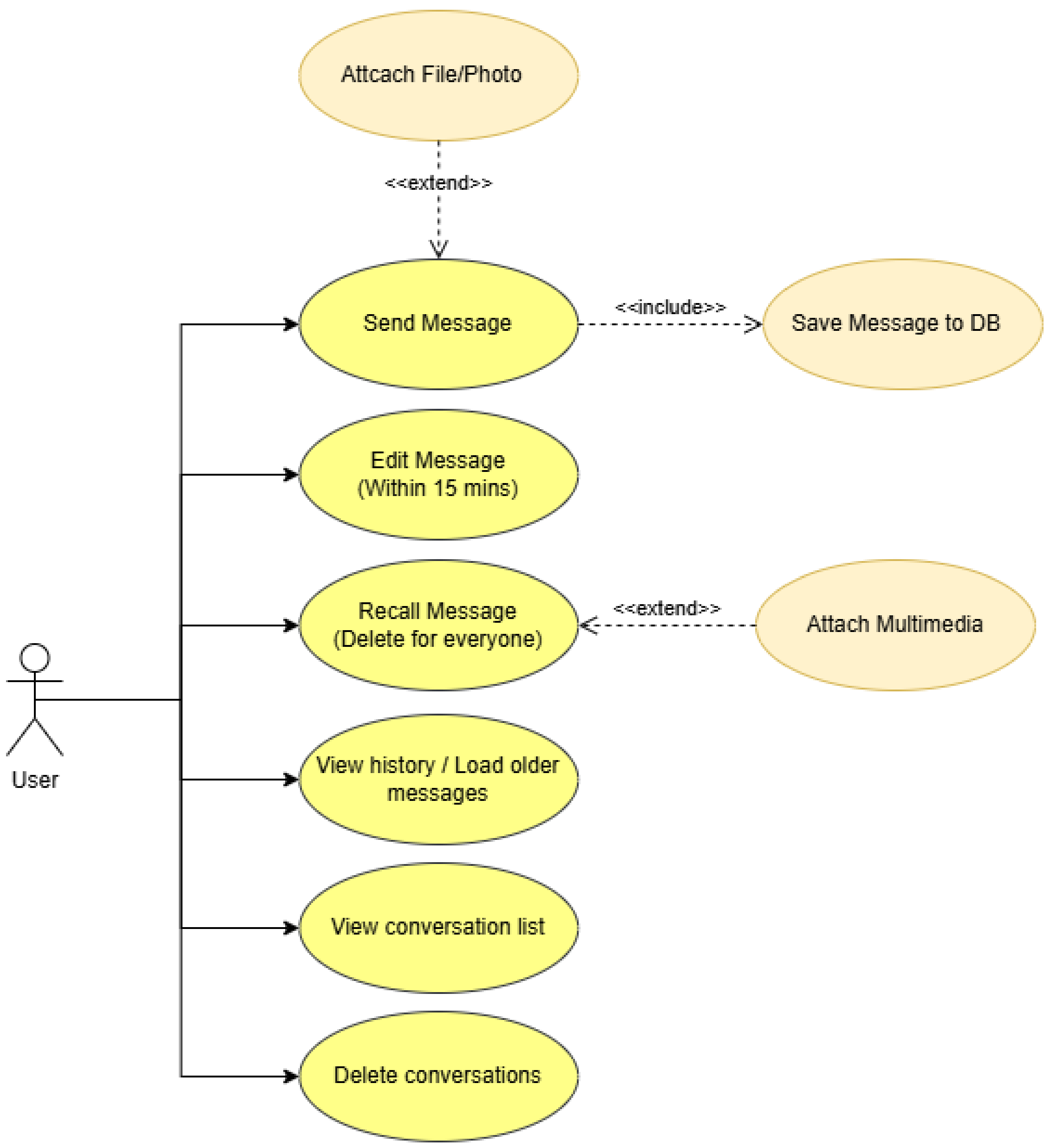

Figure 1.1.2: Personal Message Management Subsystem

2.4.4. Group Messaging Management Subsystem

This diagram clearly illustrates the inheritance of permissions: the **Admin** can perform all actions that a **Member** can, plus additional administrative privileges. It also includes other features available to users.

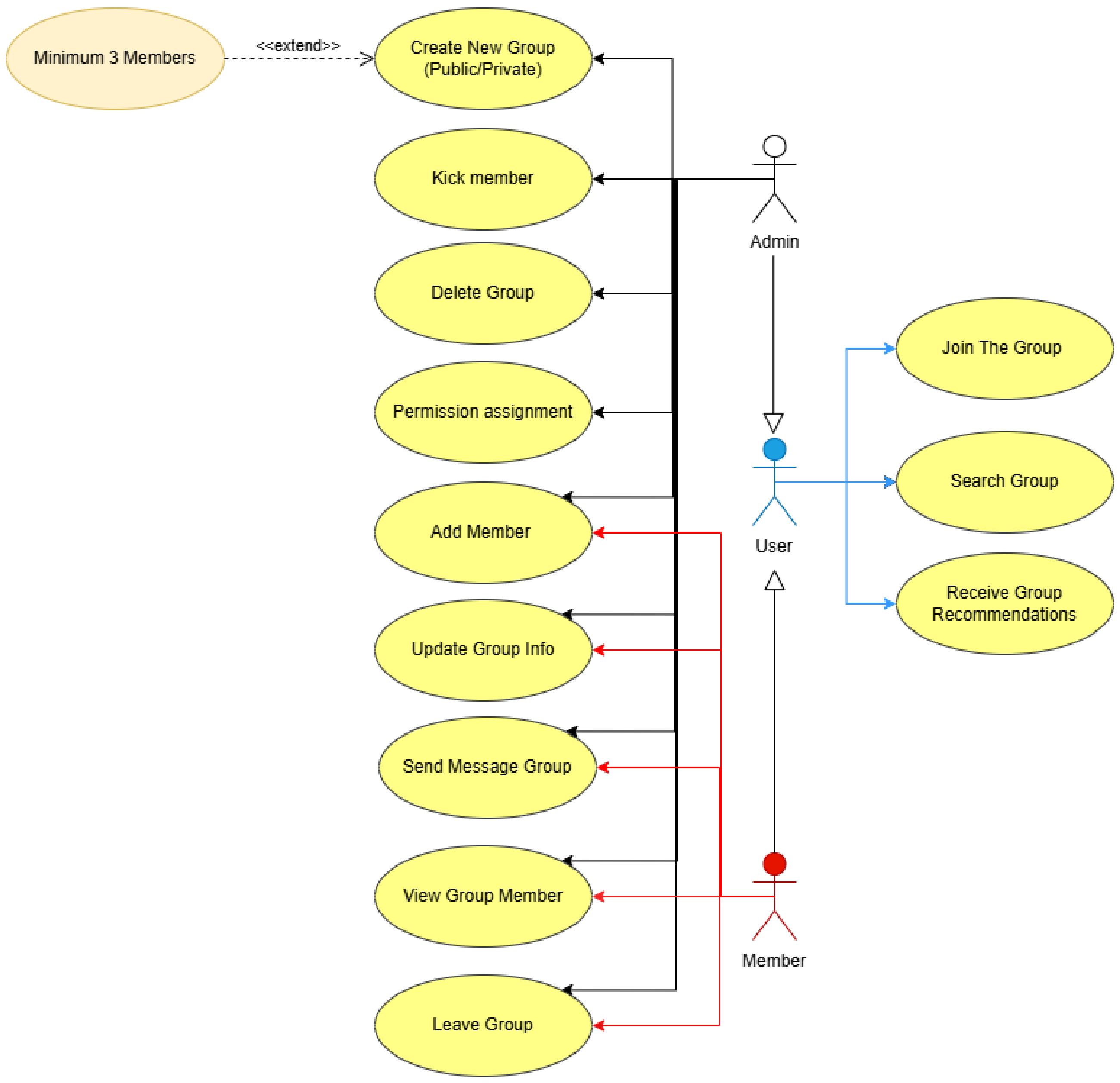

Figure 1.1.3: Group Messaging Management Subsystem

2.4.5. Friend Management Subsystem

The diagram describes the process of searching for users, adding friends, and blocking/unblocking users..

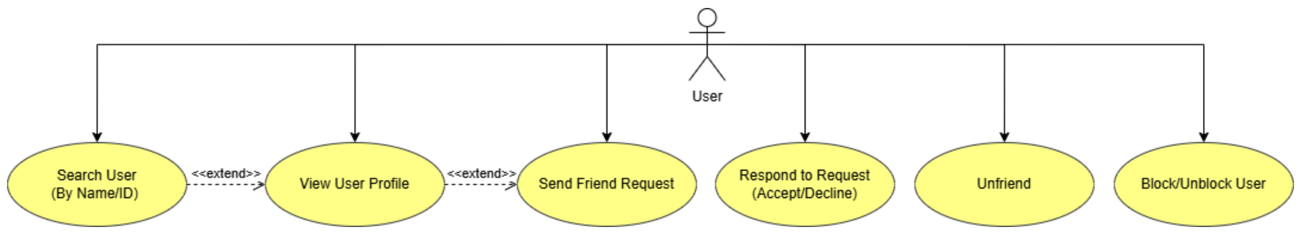

Figure 1.1.4: Friend Management Subsystem

2.4.6. Personal Profile Management Subsystem

The diagram describes the features for managing personal profiles along with several other application functionalities.

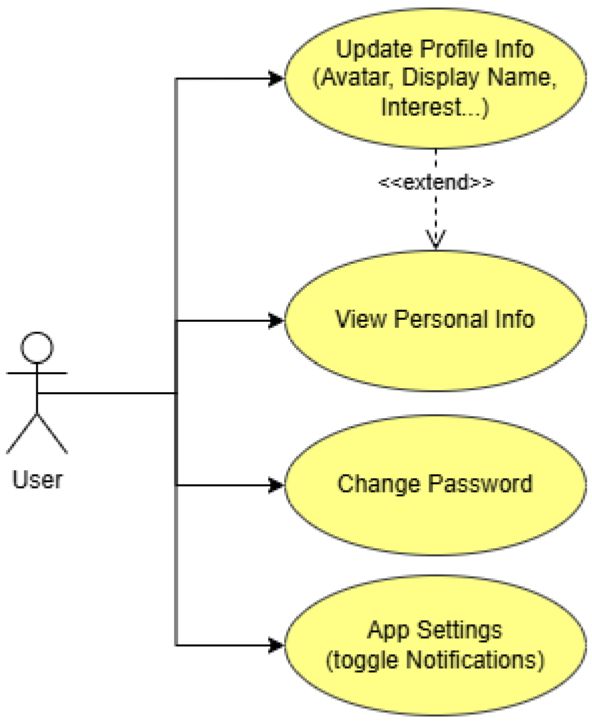
Update Profile Info (Avatar, Display Name, Interest...)
<<extend>>
View Personal Info
User
Change Password
App Settings (toggle Notifications)

Figure 1.1.5: Personal Profile Management Subsystem

3. ARCHITECTURAL DESIGN

3.1. Problem Statement

The core challenge is to design an architecture capable of harmoniously integrating two fundamentally different business flows: real-time communication (messaging) and data analysis (study-channel recommendations), while ensuring proper separation of concerns.

The Layered Architecture pattern is selected as the foundational structural approach to address the critical architectural requirements (ASRs), particularly the need for maintainability when the recommendation algorithm evolves and the strict security requirements for protecting user privacy.

3.2. Impact of ASRs on the Layered Architecture

The Architecturally Significant Requirements (ASRs) impose specific constraints on how the system layers must be designed:

• ASR-1 (Performance / Real-time) → *Business Layer & Presentation Layer:*

ASR requires handling thousands of concurrent WebSocket connections. Within the Layered Architecture, this strongly affects the Business Layer.

Instead of processing short-lived HTTP requests (Request–Response), the Business Layer must integrate Event Handlers to manage long-lived, real-time connections.

- Specific Implementation:
  The Business Layer should not communicate with the Database for every single message.
  Instead, it must utilize a Message Broker (such as Redis or Kafka) to distribute messages instantly, reducing load on the Persistence Layer.

• ASR-2 (Availability / Fault Tolerance) → *Stateless Business Layer:*

To meet ASR-2—automatic failover and load balancing—the components in the Business Layer must be designed as Stateless.

- Specific Implementation:
  The Business Layer must not store session data or socket connection states locally in the RAM of a single server.
  These states must be stored in a shared distributed store (e.g., Redis).

This ensures that the Presentation Layer (clients) can connect to *any* available server without losing session data.

• ASR-3 (Modifiability / Extensibility) → *Modularization within the Business Layer:*

ASR-3 requires that new features (e.g., Stickers) can be added without affecting existing ones (e.g., Chat).
This leads to adopting a Modular Monolith approach inside the Layered Architecture.

- Specific Implementation:
  The Business Layer is strictly divided into independent modules
  (e.g., ChatModule, NotificationModule)
  instead of combining all logic into a single codebase.

These modules interact through well-defined interfaces, allowing them to be split into microservices in the future if needed.

4. CONCLUSION

The requirements analysis phase for the Mobile Messaging Application has clearly defined the project scope through a comprehensive system of Functional and Non-functional Requirements.

The addition of the Channel Recommendation feature introduces a distinctive advantage over traditional messaging applications. The UML Use Case Diagrams offer a clear overview of the system’s two primary experience flows: real-time communication and community discovery.

The identified Architecturally Significant Requirements (ASRs)—especially Real-time Performance (ASR-1) and High Availability (ASR-2)—serve as crucial foundations guiding the selection of technologies in the subsequent design phase. These requirements call for a combination of Layered Architecture to manage business logic and Event-Driven Design patterns to handle WebSocket connections, ensuring system stability even as the user base scales.