## FINAL PROJECT DOCUMENTATION

**SwiftChatPro**



### Submitted By

|  |  |
| --- | --- |
| Hasnain Yousaf | 21-Arid-3416 |
| Muhammad Hamza | 21-Arid-3423 |
| Amir Imtiaz | 21-Arid-3409 |

**Supervisor**

Ms. Aliza Falak

**2024**

**Gujrat Institute of Management Sciences**

**PMAS-Arid Agriculture University, Rawalpindi**

**‘‘SwiftChatPro**

Project submitted to Department of Software Engineering

Gujrat Institute of Management Sciences

### By

|  |  |
| --- | --- |
| Hasnain Yousaf | 21-Arid-3416 |
| Muhammad Hamza | 21-Arid-3423 |
| Amir Imtiaz | 21-Arid-3409 |

#### In partial fulfillment of the requirements for the degree of (BSSE)

**Gujrat Institute of Management Sciences**

**PMAS-Arid Agriculture University, Rawalpindi**

## Members’ Detail

|  |  |
| --- | --- |
| **Project ID** | **GIMS-BSSE-F20211** |

|  |  |  |  |
| --- | --- | --- | --- |
| **SwiftChatZone Pro** | | | |
| **Group Leader:**Muhammad Hamza  **Group Members:**3 | | | |
| Name | Registration # | Email Address | Class |
| Muhammad Hamza | 21-arid-3423 | muhhamza453@gmail.com | BSSE |
| Amir Imtiaz | 21-arid-3409 | amirimtiaz626@gmail.com | BSSE |
| Hasnain Yousaf | 21-arid-3416 | hasnainy246@gmail.com | BSSE |

|  |  |  |
| --- | --- | --- |
| **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Mr. Awais Illyas**  PMO, GIMS |  |  |

|  |
| --- |
| **Dated: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |

## Dedication

### To our Fathers,

"To the guiding light and unwavering source of strength—our beloved fathers."

### To our Mothers,

"To the ones whose endless support and kindness have shaped our journey through life’s symphony. Thank you for being our steadfast foundation."

### To our Loved Ones,

"For their understanding and gentle encouragement, keeping us inspired to pursue our dreams."

### To our Teacher,

"To Ms Aliza Falak, whose wisdom illuminated our path, guiding us with grace and profound insight. With heartfelt gratitude, we dedicate this to you."

~Muhammad Hamza, Hasnain Yousaf, Amir Imtiaz ~

## Project Summary

|  |  |
| --- | --- |
| **Project Title** | SwiftChatPro |
| **Project ID** | GIMS-BSSE-F20211 |
| **Organization** | Gujrat Institute of Management Sciences |
| **Objective** | Write the objectives of project here   * Combines messaging, calling, and task management to streamline workflows and maintain focus. * Enables instant messaging and seamless audio/video calls using Socket.IO and WebRTC. * Tracks progress, sets deadlines, and provides reminders within the app. * Includes ChatBot for auto-responses, and conversation summaries for smarter interactions. * Facilitates reliable video conferencing with external teams and clients through Google Meet integration. * Simplifies secure file sharing and collaboration directly within the platform. |
| **Undertaken By** | Hasnain Yousaf 21-arid-3416,Muhammad Hamza 21-arid-3423,Amir Imtiaz 21-arid-3409 |
| **Supervised By** | Ms.Aliza Falak |
| **Date started** | 10-November-2024 |
| **Date Completed** |  |
| **Technologies Used** | Nodejs, BOOTSTRAP, Express, Angular, Vs code MongoDb |
| **System Used** | Window 11 |

**Proofreading Certificate**

##### This is to acknowledge that the project entitled

**SwiftChatPro [**GIMS-BSSE-F20211**]**

##### has been proofread By

**Ms.Aliza Falak**

**Dated:**

## Declaration

We hereby declare that we developed this project and this report entirely on the basis of our personal efforts made under the sincere guidance of our project supervisor. We further declare that, the titled project and all associated documents, reports are submitted as partial requirements for the degree of “BS (Hons.) in Software Engineering

|  |  |  |
| --- | --- | --- |
| **Members’ Name** | **Registration #** | **Signature** |
| Muhammad Hamza | 21-arid-3423 |  |
| Hasnain Yousaf | 21-arid-3416 |  |
| Amir Imtiaz | 21-arid-3409 |  |

**Dated:**

In today’s rapidly evolving business landscape, seamless communication is the cornerstone of successful teamwork, efficient project management, and heightened productivity. While numerous communication tools are available, many fall short in addressing the diverse and interconnected needs of modern organizations. These tools often lack a unified approach, forcing teams to juggle between separate platforms for messaging, task management, and collaboration, leading to inefficiencies and fragmented workflows. This disjointed experience is further compounded by the absence of essential features like integrated video conferencing, robust file sharing, and advanced AI-driven functionalities that streamline communication and decision-making.

SwiftChatZone Pro aims to bridge these gaps by offering an all-in-one communication and collaboration solution designed to cater to the demands of modern businesses. This platform will empower teams with real-time messaging for instant communication, group collaboration tools for streamlined teamwork, and smart AI features for enhanced efficiency. With integrated audio and video call capabilities and seamless Google Meet integration, SwiftChatZone Pro will provide users with the flexibility to switch effortlessly between messaging, video conferencing, and project management. Its AI-powered chatbots will further revolutionize user interaction by automating routine tasks, providing intelligent responses, and delivering insightful analytics. By combining these features into a single cohesive platform, SwiftChatZone Pro will not only enhance communication and collaboration but also reduce the reliance on multiple disconnected tools, driving efficiency and productivity across organizations.

# Chapter 1 Introduction

## Chapter 1 Introduction

This proposal outlines SwiftChatZone Pro, a real-time chat application aimed at providing a comprehensive communication platform for businesses and organizations. The application will support essential chat functionalities, such as one-to-one chat, group chat, and file sharing. In addition, it will integrate advanced features like audio and video calls and Google Meet integration to facilitate seamless remote collaboration. The inclusion of AI-powered modules, such as ChatBot will enhance user experience and boost productivity. This project aims to create a robust communication tool that caters to the needs of modern organizations by combining messaging, calling, and task management into one platform.

SwiftChatZone Pro aims to bridge these gaps by offering an all-in-one communication and collaboration solution designed to cater to the demands of modern businesses. This platform will empower teams with real-time messaging for instant communication, group collaboration tools for streamlined teamwork, and smart AI features for enhanced efficiency. With integrated audio and video call capabilities and seamless Google Meet integration, SwiftChatZone Pro will provide users with the flexibility to switch effortlessly between messaging, video conferencing, and project management. Its AI-powered chatbots will further revolutionize user interaction by automating routine tasks, providing intelligent responses, and delivering insightful analytics. By combining these features into a single cohesive platform, SwiftChatZone Pro will not only enhance communication and collaboration but also reduce the reliance on multiple disconnected tools, driving efficiency and productivity across organizations.

During the development of a final year project, is that the report that PMO used to generate on Excel will now be generated by the system. In this application, a website has designed where users can register with username and password. They can search, upload, from the website. Supervisor and members easily communicate according to their timetable that are uploaded by the supervisor for final year project. This proposed system also saves time of members of final year project and supervisor. This proposed system keeps track of the flow of final year project. The members can easily discuss their project and save their time.

### Significance

SwiftChatZonePro will bridge the gap between instant messaging and project management, offering a powerful yet simple-to-use tool for effective communication. The application’s significant features include:

One-to-One and Group Chat: Enables users to engage in private conversations or collaborate in team-oriented group chats. This feature ensures secure and real-time communication for both personal and group discussions.

Audio and Video Calls: Provides users with an option for real-time voice and video communication, making remote discussions more dynamic. This feature supports both one-on-one and group calls, allowing for diverse meeting formats.

File Sharing: Enables users to easily share documents, images, and multimedia files within conversations, enhancing collaboration and reducing the need for external sharing services.

Google Meet Integration: Allows users to seamlessly schedule or join Google Meet calls directly within the chat application. This feature provides the flexibility of external video conferencing and makes it convenient to connect with people outside the organization who may prefer or require Google Meet.

Task Management: Integrated task management allows users to create, assign, and track tasks within the chat application. It includes deadline reminders, a Kanban board, and real-time updates to improve productivity and workflow management.

AI-Driven Chatbot:These tools can offer proactive responses, suggestions, and summarize lengthy conversations, saving time.

### Objectives

* Enhanced Productivity: SwiftChatZone Pro consolidates messaging, calling, and task management into one platform, reducing the need to switch between applications and helping teams stay focused on their tasks.
* Real-Time Communication: Using Socket.IO for instant messaging and WebRTC for audio/video calls ensures real-time data transfer, fostering efficient communication and reducing delays in response times.
* Task Management Integration: Built-in task management allows users to track progress, set deadlines, and receive reminders directly within the app, which keeps team members accountable and projects on schedule.
* AI-Enhanced Features: AI-powered capabilities like sentiment analysis, auto-response, and conversation summarization provide intelligent assistance, improving user engagement and response time.
* Google Meet Integration: Google Meet integration offers a well-known, reliable option for video conferencing, especially useful for collaborating with external teams and clients who may prefer or already use Google Meet.
* File Sharing and Collaboration: Direct file-sharing capabilities make it easy for teams to collaborate without relying on external file transfer platforms, enhancing productivity and security.

### Limitations and Restrictions:

* Dependency on Internet and Third-Party Services: Reliable internet is crucial for real-time features, and integrations like Google Meet depend on third-party services, subject to API rate limits and potential downtime.
* Scalability and Performance Challenges: Increased user volume may require significant infrastructure to maintain real-time responsiveness, potentially driving up hosting and operational costs.
* Data Security and Privacy Compliance: Handling real-time communications and file sharing demands strong security measures and compliance with data regulations, which can add complexity and restrict deployment in certain regions.
* Complexity and Learning Curve: A feature-rich application may be overwhelming for new users and incur higher initial development and maintenance costs.
* File and Device Limitations: Restrictions on file size/types for sharing, and potential compatibility issues across devices, may limit accessibility and functionality for some users.

##### Project Goal:

The primary goal of this project is to develop a comprehensive, real-time chat application designed to facilitate seamless communication, collaboration, and project management for individuals and teams. The application will support a range of interactive features, including one-to-one and group chats, audio and video calls, file sharing, and integrated tools like Google Meet for additional meeting capabilities.



**Type of project:**

R&D

Development

##### Project Success criteria:

* Real-Time Communication: Implement seamless real-time messaging, audio/video calls, and instant notifications across all devices.
* Feature Integration: Ensure full functionality of core features, including chat, file sharing, task management, video conferencing, and an AI chatbot.
* User Experience: Deliver a responsive and intuitive interface that enhances individual and group communication.
* Reliability and Performance: Maintain consistent uptime, low latency, and a scalable architecture to accommodate a growing user base.
* AI-Driven Enhancements: Integrate AI for automated responses, message summarization, and other features to enrich user interaction and improve efficiency.

##### Risks of the Project:

[Please describe the factors that may cause delays in, or prevent implementation of, the project as proposed above; estimate the degree of risk.]

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **(Please mark**  **where applicable)** | **Low** | **Medium** | **High** |  |
| Technical risk |  | Checkmark |  |
| Timing risk |  | Checkmark |  |
| Budget risk | Checkmark |  |  |

##### Organization Details (if any):

N/A

Checkmark

|  |
| --- |
| **Target End users:**  General public, professionals, students, and small businesses who need a versatile communication platform for both casual and professional use. |
| **Development Technology/ Languages: FOR EXAMPLE:**   * Node js * BOOTSTRAP * MongoDB * Html 5 * Vs code * JavaScript * Angular * Express js * CSS |
| **Platform:**  Web based Distributed Setup Configurations  Desktop based Android iOS  Other |
| **Project Supervisor:** Mr, Bilal Mazhar |

**Client-side**

* UI Components
* Real-Time Messaging
* HTTP Requests
* AI Integration:
* Google Meet Integration

**Third Party APIs**

* Google Meet(API)
* WebRTC
* Meta Ai Tools

**Application Layer(Node+express)**

* User Authentication (JWT)
* API Layer
* Socket.IO Server
* AI Chatbot processing
* Task Management

**MongoDb**

* User Data
* Message History
* File Storage
* Chat History
* Task Data

*Figure 1.1: Model-View-Controller*

### Software/Hardware Requirements

* + - Operating system
    - Web browser
    - Laptop/system

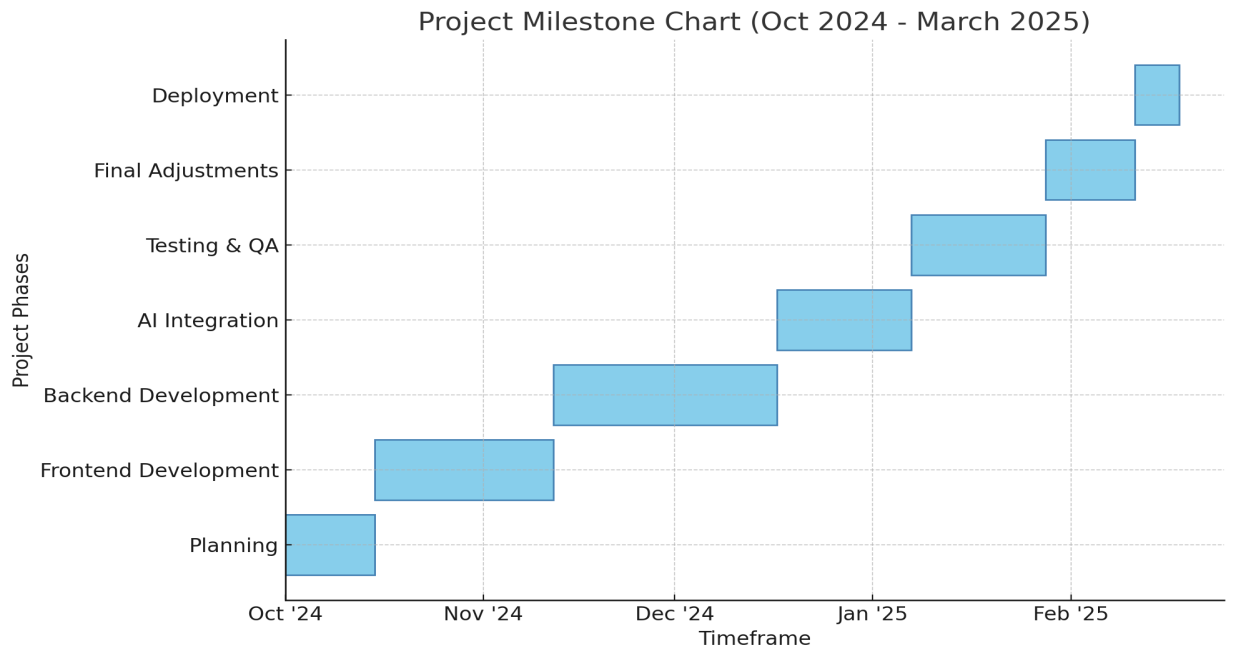
### Implementation Tools and Technology

* + - Node js
    - BOOTSTRAP
    - MongoDB
    - Express
    - Html 5
    - VS code
    - JavaScript
    - Typescript
    - Angular

### Deliverable Items

* + - * Project Proposal
      * Software Requirement Specification (SRS)
      * Project Presentation and Final Documentation
      * Executable Files
      * CD
      * User manual

### Milestone Chart



*Figure 1.2: Gantt chart for milestones*

# Chapter 2 Requirement Analysis

## Chapter 2 Requirement Analysis

### Functional Requirements

### Registarion and login

### 

##### Registration:

Users can create an account using their email and password.

Option for OAuth-based registration (e.g., Google, Facebook).

##### Login:

Users log in using email/password or third-party OAuth.

Remember Me functionality for seamless login.

##### Password Management:

Option to reset passwords via email link.

Secure storage of passwords using hashing algorithms (e.g., bcrypt).

##### Security Features:

Implement JWT for secure session management.

### Real Time Messaging

##### One-to-One Messaging:

Private chat sessions between two users.

Real-time message delivery and read receipts.

##### Group Messaging:

Chat rooms for multiple participants.

Notifications for new members or messages.

##### Typing Indicators:

Shows when a user is typing.

##### Message Features:

Edit, delete, or reply to messages.

Emojis and reactions for better engagement.

##### Technology:

Implemented using Socket.IO for low-latency, real-time messaging.

### Audio&Video Calls

##### One-to-One Calls:

Audio and video call options for private discussions.

Mute/unmute and video on/off controls.

##### Group Calls:

Enables multiple users to join the same call.

Features like speaker view and participant management.

##### Notifications:

Incoming call alerts with an option to accept/decline.

##### Technology:

Built using WebRTC for peer-to-peer communication.

### File Sharing

##### Upload and share files within chats.

##### Preview feature for supported file types (e.g., images, PDFs).

##### Download option with access control.

##### File size and type restrictions for better performance.

##### Secure storage using encrypted protocols.

### Google Meet

##### Generate Google Meet links directly within the app.

##### Schedule and share meeting links in chat.

##### Single-click join functionality.

##### Access Google Meet features (e.g., recording, captions).

##### Technology: Use Google Meet API for integration.

##### 

### Task Management

##### Create and Assign Tasks:

Users can assign tasks to individuals or groups.

Set deadlines, priorities, and task descriptions.

##### Progress Tracking:

Status updates (To-Do, In Progress, Done).

Real-time updates visible to all members.

##### Reminders and Notifications:

Automated reminders for approaching deadlines.

##### Visualization Tools:

Kanban boards or list views for task management.

### Notification

##### Real-time push notifications for new messages, calls, and task updates.

##### Customizable notification settings:

##### Control frequency and type of notifications.

##### Mute options for specific chats or tasks.

##### Technology:

##### Use Web Push API and Socket.IO for real-time updates.

### AI Powered ChatBot

##### Auto-Responses:

Suggest common replies based on context.

##### Task Assistance:

Help create tasks or set reminders through natural language commands.

##### Technology:

Integrate AI using APIs like OpenAI GPT-4 or Meta AI and Gemini.

### Search and Filter

##### Search for:

Messages by keyword.

Files shared in chats.

Tasks and their status.

Users by name or profile.

##### Filtering Options:

By time, sender, content type (messages, files, tasks).

### User Profile

##### Profile Customization:

Add profile picture, status, and bio.

##### Account Settings:

Manage email, password, and privacy preferences.

##### Activity Logs:

View chat history, call logs, and task activity.

##### Privacy Controls:

Block users or control visibility of profile information.

##### Technology:

Use secure API endpoints to manage user data.

### Non-Functional Requirement

All non-functional requirements of proposed system are as followed:

### Performance Requirements:

Fast response times (e.g., <500ms for messaging).

Efficient server throughput and optimized database operations.

### Scalability Requirements:

Support increased users and features via horizontal scaling, cloud integration, and microservices architecture.

### Security and Data Protection:

Encrypt data, secure APIs, and ensure compliance with privacy laws (e.g., GDPR).

Regular backups, penetration testing, and access controls.

### Usability and Responsiveness:

Intuitive UI, mobile responsiveness, accessibility (e.g., WCAG standards).

Fast loading times (<2 seconds) and feedback mechanisms.

### Availability and Reliability:

Ensure 99.99% uptime with redundancy, multi-region deployment, and disaster recovery systems.

### Maintainability:

Modular code for easy updates, CI/CD pipelines, automated testing, and detailed documentation.

Use monitoring tools to track and alert for system issues.

### Effective

Our system works effectively

### Use Cases

### User Registration Use Case

*Table 2.1* User Registration *use case*

|  |  |
| --- | --- |
| **Use Case Name** | User Registration |
| **Actors** | User, System |
| **Use Case Number** | UC-1 |
| **Description** | Users register to create an account on the platform. |
| **Preconditions** | User must have a valid email address and internet access. |
| **Triggers** | User clicks on the "Register" button. |
| **Main Flow** | 1. User enters registration details (e.g., email, password).   2. System validates input.  3. System sends a confirmation email.  4. User verifies email.  5. System creates the user account. |
| **Alternate Flow** | 1. User provides third-party OAuth credentials.   2. System fetches data and registers the user. |
| **Exceptions** | Invalid email, weak password, duplicate account, or network failure. |
| **Output** | User registered successfully. |

**2.3.2 User Login Use Case**

*Table 2.2 User Login Use Case*

|  |  |
| --- | --- |
| **Use Case Name** | User Login |
| **Actors** | User, System |
| **Use Case Number** | UC-2 |
| **Description** | Users log in to access their account and platform features. |
| **Preconditions** | User must have a registered account and internet access. |
| **Triggers** | User clicks on the "Login" button. |
| **Main Flow** | 1.User enters login credentials (email and password).  2. System validates credentials.  3. If valid, grants access to the platform. |
| **Alternate Flow** | 1.User uses third-party OAuth to log in.  2. System verifies and grants access. |
| **Exceptions** | Incorrect credentials, account locked, or network failure. |
| **Output** | User logged in successfully. |

**2.3.3 Sign in With Google Use Case**

*Table 2.3 Sign in with google Use Case*

|  |  |
| --- | --- |
| **Use Case Name** | Sign in with Google |
| **Actors** | User, System, Google API |
| **Use Case Number** | UC-3 |
| **Description** | Users log in using their Google account credentials. |
| **Preconditions** | User must have a Google account and internet access. |
| **Triggers** | User clicks on "Sign in with Google." |
| **Main Flow** | 1. User selects the "Sign in with Google" option.  2. System redirects to Google authentication page.  3. User authenticates via Google.  4. System retrieves user information from Google.  5. System grants access to the platform. |
| **Alternate Flow** | 1. User cancels the authentication process on the Google page.  2. System redirects back to the login page without granting access. |
| **Exceptions** | Invalid Google credentials, account not linked, or network failure. |
| **Output** | User signed in successfully using Google. |

**2.3.4 Forgot Password Use Case**

*Table 2.4 Forgot password Use Case*

|  |  |
| --- | --- |
| **Use Case Name** | Forgot Password |
| **Actors** | User, System, Google API |
| **Use Case Number** | UC-4 |
| **Description** | Users reset their password when they forget it.. |
| **Preconditions** | User must have a registered email address and internet access. |
| **Triggers** | User clicks on "Forgot Password." |
| **Main Flow** | 1. User clicks "Forgot Password."  2. System prompts for the registered email.  3. User provides the email.  4. System sends a password reset email.  5. User clicks on the link and resets the password. |
| **Alternative Flow** | 1. User requests a reset email but does not receive it due to spam filtering or incorrect email.  2. System provides a message to retry or contact support. |
| **Exceptions** | Invalid email address, expired reset link, or network failure. |
| **Output** | User resets password successfully. |

**2.3.5 Logout Use Case**

*Table 2.5 LogoutUse Case*

|  |  |
| --- | --- |
| **Use Case Name** | Logout |
| **Actors** | User, System, Google API |
| **Use Case Number** | UC-5 |
| **Description** | Users log out of their account to end the session. |
| **Preconditions** | User must be logged in. |
| **Triggers** | User clicks on the "Logout" button. |
| **Main Flow** | 1. User clicks "Logout."  2. System ends the session and redirects the user to the home or login page. |
| **Alternative Flow** | 1. User attempts to logout but navigates away before confirmation.  2. System keeps the session active until logout is explicitly confirmed. |
| **Exceptions** | Network failure during logout. |
| **Output** | User logged out successfully. |

**2.3.6 One to One Use Case**

*Table 2.6 One to One Use Case*

|  |  |
| --- | --- |
| **Use Case Name** | One-to-One Chat |
| **Actors** | User, System |
| **Use Case Number** | UC-6 |
| **Description** | Users can engage in private conversations with other individual users. |
| **Preconditions** | Both users are registered and logged in. |
| **Triggers** | User initiates a private chat with another user. |
| **Main Flow** | 1. User selects another user to chat with.  2. System establishes a private chat session.  3. Users send and receive messages in real time. |
| **Exceptions** | Network connectivity issues or user is unavailable. |
| **Output** | Private messages are exchanged successfully. |

### 2.3.7 Group Chat Use Case

*Table 2.7 Group Chat Use Case*

|  |  |
| --- | --- |
| **Use Case Name** | Group Chat |
| **Actors** | User, System |
| **Use Case Number** | UC-7 |
| **Description** | Users can create or participate in chat groups for team discussions or social interactions. |
| **Preconditions** | User is registered, logged in, and a member of the group. |
| **Triggers** | User initiates a group chat or joins an existing one. |
| **Main Flow** | 1. User creates a group and adds members.  2. Group members exchange messages.  3. System synchronizes messages across all devices. |
| **Alternative Flow** | Admin removes or adds members to the group. |
| **Exceptions** | Group reaches maximum member limit or a message fails to deliver due to server issues. |
| **Output** | Group messages are exchanged and visible to all members. |

### 2.3.8 Audio/Video Calls Use Case

T*able 2.8.1: Initiating a Call Use Case*

|  |  |
| --- | --- |
| **Use Case Name** | Initiating a Call |
| **Actors** | User, System |
| **Use Case Number** | UC-8.1 |
| **Description** | Users initiate one-on-one or group calls. |
| **Preconditions** | User is logged in, both parties have internet. |
| **Triggers** | User clicks on "Call" to initiate a call. |
| **Main Flow** | 1. User selects a contact or group and clicks "Call."  2. System establishes a WebRTC connection.  3. System notifies the recipient(s) of the incoming call. |
| **Alternative Flow** | 1. Call initiation fails due to recipient being unavailable or busy.  2. System shows a message to retry or leave a message. |
| **Exceptions** | Network interruptions, hardware issues. |
| **Output** | Call is initiated successfully, or an error is shown. |

Table 2.8.2: *Receiving a Call Use Case*

|  |  |
| --- | --- |
| **Use Case Name** | Receiving a Call |
| **Actors** | User, System |
| **Use Case Number** | UC-8.2 |
| **Description** | Users receive incoming audio/video calls. |
| **Preconditions** | User is logged in and has a stable internet connection. |
| **Triggers** | System notifies the user of an incoming call. |
| **Main Flow** | 1. System notifies the user of an incoming call.  2. User accepts the call.  3. System establishes a WebRTC connection and starts the call. |
| **Alternative Flow** | 1. User rejects the call.  2. System notifies the caller that the call was declined. |
| **Exceptions** | Network interruptions or user unavailable due to device issues. |
| **Output** | Call is received successfully, or an error is shown. |

Table 2.8.3: *Ending a Call Use Case*

|  |  |
| --- | --- |
| **Use Case Name** | Ending a Call |
| **Actors** | User, System |
| **Use Case Number** | UC-8.3 |
| **Description** | Users end ongoing audio/video calls. |
| **Preconditions** | An active call is in progress. |
| **Triggers** | User clicks on the "End Call" button.. |
| **Main Flow** | 1. User clicks on the "End Call" button.  2. System terminates the WebRTC connection. |
| **Alternative Flow** | None |
| **Exceptions** | Network failure prevents call termination synchronization. |
| **Output** | Call ends successfully, and both parties are notified. |

### 2.3.9 File Sharing Use Case

*Table 2.9:* File Sharing *Use Case*

|  |  |
| --- | --- |
| **Use Case Name** | File Sharing |
| **Actors** | User, System |
| **Use Case Number** | UC-9 |
| **Description** | Users share files during chats. |
| **Preconditions** | User is logged in. |
| **Triggers** | User selects a file to share. |
| **Main Flow** | 1. User uploads a file.  2. System checks size and format.  3. File is shared in the chat. |
| **Alternative Flow** | File exceeds size limit or is an unsupported format. |
| **Exceptions** | File upload failure, slow internet. |
| **Output** | File appears in the chat for recipient(s) |

### 2.3.10 Add User Use Case

*Table 2.10: Add user Use Case*

|  |  |
| --- | --- |
| **Use Case Name** | Add User |
| **Actors** | User, System |
| **Use Case Number** | UC-10 |
| **Description** | Users add other users to their contact list within the system. |
| **Preconditions** | User is logged in and has internet access. |
| **Triggers** | User clicks on "Add Contact." |
| **Main Flow** | 1. User clicks on the "Add Contact" button.  2. System prompts the user to enter the contact’s details (e.g., username, email, or unique identifier).  3. User provides the details and submits the request.  4. System validates the input and checks if the contact exists in the system.  5. If valid, the contact is added to the user’s list, and the user is notified. |
| **Alternative Flow** | 1. User searches for the contact using the search feature.  2. System displays matching results.  3. User selects a contact from the results and clicks "Add." |
| **Exceptions** | Invalid user details, contact already exists, or system failure. |
| **Output** | Contact added successfully, or an appropriate error message is displayed. |

### 2.3.11 Emoji Use Case

*Table 2.11: Emoji Use Case*

|  |  |
| --- | --- |
| **Use Case Name** | Emoji Use |
| **Actors** | User, System |
| **Use Case Number** | UC-11 |
| **Description** | Users select and use emojis in communication and tasks. |
| **Preconditions** | User is logged in and interacting with a feature that supports emojis. |
| **Triggers** | User clicks on the emoji icon or enters an emoji shortcut. |
| **Main Flow** | 1. User opens the emoji picker or types an emoji shortcut.  2. System displays a list of emojis or auto-suggests matching emojis.  3. User selects an emoji or confirms the shortcut.  4. System inserts the emoji into the message, task, or comment. |
| **Alternative Flow** | 1. User searches for a specific emoji in the emoji picker.  2. System filters and displays the relevant emojis.  3. User selects the desired emoji. |
| **Exceptions** | Emoji not supported by the platform, outdated emoji set, or network failure preventing emoji rendering. |
| **Output** | Emoji added to the intended message, task, or comment. |

### 2.3.12 Voice Messaging Case

*Table 2.12: VoiceUse Case*

|  |  |
| --- | --- |
| **Use Case Name** | Voice Messaging |
| **Actors** | User, System |
| **Use Case Number** | UC-12 |
| **Description** | Users record and send voice messages |
| **Preconditions** | User is logged in, has access to a microphone, and has granted necessary permissions. |
| **Triggers** | User clicks on the "Voice" button. |
| **Main Flow** | 1. User clicks the "Record" button.  2. System activates the microphone and starts recording.  3. User speaks into the microphone.  4. User clicks the "Stop" button to end the recording.  5. System saves the recording and presents options to play, delete, or send. |
| **Alternative Flow** | 1. User pauses the recording and resumes later.  2. System resumes recording from where it left off. |
| **Exceptions** | No microphone detected, permissions not granted, or storage limit exceeded. |
| **Output** | Voice recording is saved, sent, or an appropriate error message is displayed. |

**2.3.13 Task Management Use Case**

*Table 2.13.1: Task Creation Use Case*

|  |  |
| --- | --- |
| **Use Case Name** | Task Creation |
| **Actors** | User, System |
| **Use Case Number** | UC-13.1 |
| **Description** | Users create tasks with specific details. |
| **Preconditions** | User is logged in and part of a group. |
| **Triggers** | User clicks on "Create Task." |
| **Main Flow** | 1. User enters task details (e.g., title, deadline, assignee).  2. System validates the input.  3. System saves the task and notifies the assignee. |
| **Alternative Flow** | 1. User does not assign a task to anyone.  2. System saves the task as "Unassigned" and notifies the group or creator. |
| **Exceptions** | Deadline conflicts, invalid input, or network failure. |
| **Output** | Task created successfully. |

*Table 2.13.2: Task Editing Use Case*

|  |  |
| --- | --- |
| **Use Case Name** | Task Editing |
| **Actors** | User, System |
| **Use Case Number** | UC-13.2 |
| **Description** | Users modify existing task details. |
| **Preconditions** | User is logged in and part of a group. |
| **Triggers** | User clicks on "Edit Task." |
| **Main Flow** | 1. User selects a task to edit.  2. User updates task details (e.g., title, deadline, assignee).  3. System validates and saves the changes. |
| **Alternative Flow** | 1. User removes the assignee.  2. System saves the task as "Unassigned" and updates group notifications. |
| **Exceptions** | Deadline conflicts, invalid input, or network failure. |
| **Output** | Task updated successfully. |

*Table 2.13.3: Task Deletion Use Case*

|  |  |
| --- | --- |
| **Use Case Name** | Task Deletion |
| **Actors** | User, System |
| **Use Case Number** | UC-13.3 |
| **Description** | Users delete tasks from the system. |
| **Preconditions** | User is logged in and part of a group. |
| **Triggers** | User clicks on "Delete Task." |
| **Main Flow** | 1. User selects a task to delete.  2. System confirms the deletion action.  3. System deletes the task and updates task-related notifications. |
| **Alternative Flow** | 1. User cancels the deletion after confirmation prompt.  2. System retains the task and returns the user to the task view. |
| **Exceptions** | Task not found or network failure. |
| **Output** | Task deleted successfully. |

*Table 2.13.4: Task Viewing Use Case*

|  |  |
| --- | --- |
| **Use Case Name** | Task Viewing |
| **Actors** | User, System |
| **Use Case Number** | UC-13.4 |
| **Description** | Users view a list of tasks or specific task details. |
| **Preconditions** | User is logged in and part of a group. |
| **Triggers** | User navigates to the "Tasks" section. |
| **Main Flow** | 1. User requests a list of tasks or task details.  2. System retrieves and displays the requested tasks. |
| **Alternative Flow** | 1. User applies filters (e.g., by deadline, status, assignee).  2. System retrieves and displays tasks based on the applied filters. |
| **Exceptions** | No tasks available or network failure. |
| **Output** | Tasks displayed successfully. |

**2.3.14 Notifications Use Case**

*Table 2.14:* Notification *Use Case*

|  |  |
| --- | --- |
| **Use Case Name** | Notifications |
| **Actors** | User, System |
| **Use Case Number** | UC-14 |
| **Description** | Notify users of new messages, calls, task updates, and deadlines. |
| **Preconditions** | User is logged in and notifications are enabled. |
| **Triggers** | A new event occurs (e.g., message received, task updated). |
| **Main Flow** | 1. System detects the event.  2. Notification is sent to the user. |
| **Alternative Flow** | User misses a notification; the system logs it in the notification panel. |
| **Exceptions** | Notification delivery failure due to network issues. |
| **Output** | Notification displayed to the user. |

**2.3.15 AI Chatbot** **Use Case**

*Table* *2.15* AI Chatbot *Use Case*

|  |  |
| --- | --- |
| **Use Case Name** | AI Chatbot |
| **Actors** | User, System |
| **Use Case Number** | UC-15 |
| **Description** | AI-powered chatbot assists with queries, auto-responses, and conversation summaries. |
| **Preconditions** | User is logged in and chatbot is enabled. |
| **Triggers** | User sends a message to the chatbot. |
| **Main Flow** | 1. User sends a query or command.  2. Chatbot processes input using AI models.  3. Chatbot responds with relevant information. |
| **Alternative Flow** | Chatbot escalates the query to human support for complex issues. |
| **Exceptions** | Chatbot fails to respond due to API downtime or unclear input. |
| **Output** | User receives a response or solution. |

**2.3.16 Search Functionalities Use Case**

*Table 2.16* AI Chatbot *Use Case*

|  |  |
| --- | --- |
| **Use Case Name** | Search Functionalities |
| **Actors** | User, System |
| **Use Case Number** | UC-16 |
| **Description** | Users can search for messages, files, users, or tasks within the app. |
| **Preconditions** | User is logged in. |
| **Triggers** | User enters a keyword in the search bar. |
| **Main Flow** | 1. User inputs a search query.  2. System filters results based on keywords, time, or type.  3. Results are displayed to the user. |
| **Alternative Flow** | No matches found; system suggests refining the query. |
| **Exceptions** | System fails to return results due to a database query error. |
| **Output** | Relevant results are shown to the user. |

**2.3.17 Google Meet Integration Use Case**

*Table 2.17* Google Meet Integration *Use Case*

|  |  |
| --- | --- |
| **Use Case Name** | Google Meet Integration |
| **Actors** | User, System |
| **Use Case Number** | UC-17 |
| **Description** | Users can create and join Google Meet sessions directly from the app. |
| **Preconditions** | User is logged in and has a Google account linked. |
| **Triggers** | User clicks on the "Create Meet" button. |
| **Main Flow** | 1. User initiates Google Meet.  2. System interacts with the Google Meet API.  3. Meet link is generated and shared. |
| **Alternative Flow** | User receives an invite link to join an ongoing Meet. |
| **Exceptions** | API errors, network failure, or invalid Google credentials. |
| **Output** | Meet link created or user joins the meeting successfully. |

**2.3.18 Group Creation and Management**  **Use Case**

*Table 2.18* Group Creation and Management Use Case

|  |  |
| --- | --- |
| **Use Case Name** | Group Creation and Management |
| **Actors** | User, System |
| **Use Case Number** | UC-18 |
| **Description** | Users can create, manage, and leave groups |
| **Preconditions** | User is logged in. |
| **Triggers** | User initiates group creation. |
| **Main Flow** | User provides a group name and adds members.  2. System creates the group and sends invitations to added members. |
| **Alternative Flow** | User modifies the group settings (e.g., adds/removes members) |
| **Exceptions** | system fails to notify members of updates |
| **Output** | Group is created and members are notified. |

**2.3.19 Profile Management**  **Use Case**

*Table 2.19.1* Add/Update Profile Use Case

|  |  |
| --- | --- |
| **Use Case Name** | Add/Update Profile |
| **Actors** | User, System |
| **Use Case Number** | UC-19.1 |
| **Description** | Users add or update their profile information. |
| **Preconditions** | User is logged in. |
| **Triggers** | User navigates to the "Profile" section and clicks on "Edit Profile." |
| **Main Flow** | 1. User accesses the profile management section.  2. User enters or updates profile information (e.g., name, email, photo, contact information).  3. System validates the input.  4. System saves the updated profile information. |
| **Alternative Flow** | 1. User uploads a profile picture.  2. System validates the file type and size before saving. |
| **Exceptions** | Invalid input (e.g., incorrect email format), unsupported file type, or network failure. |
| **Output** | Profile added or updated successfully. |

*Table 2.19.2* View Profile Use Case

|  |  |
| --- | --- |
| **Use Case Name** | View Profile |
| **Actors** | User, System |
| **Use Case Number** | UC-19.2 |
| **Description** | Users view their profile information. |
| **Preconditions** | User is logged in. |
| **Triggers** | User navigates to the "Profile" section. |
| **Main Flow** | 1. User clicks on "View Profile."  2. System retrieves the user's profile information.  3. System displays the profile details (e.g., name, email, contact information, profile picture). |
| **Alternative Flow** | 1. User accesses a simplified profile view (e.g., compact layout for mobile devices). |
| **Exceptions** | Profile data not found due to database error or network failure. |
| **Output** | Profile information displayed successfully. |

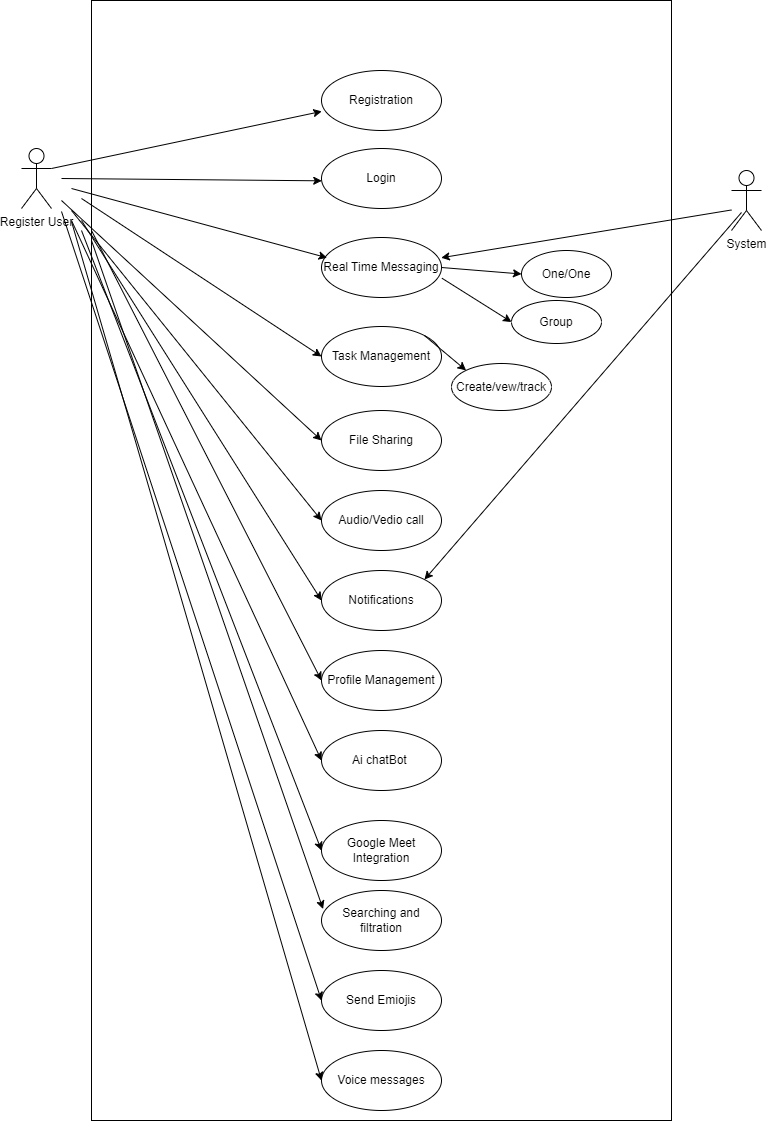
# Chapter 3 Design

## Chapter 3 DESIGN

### UML Diagrams

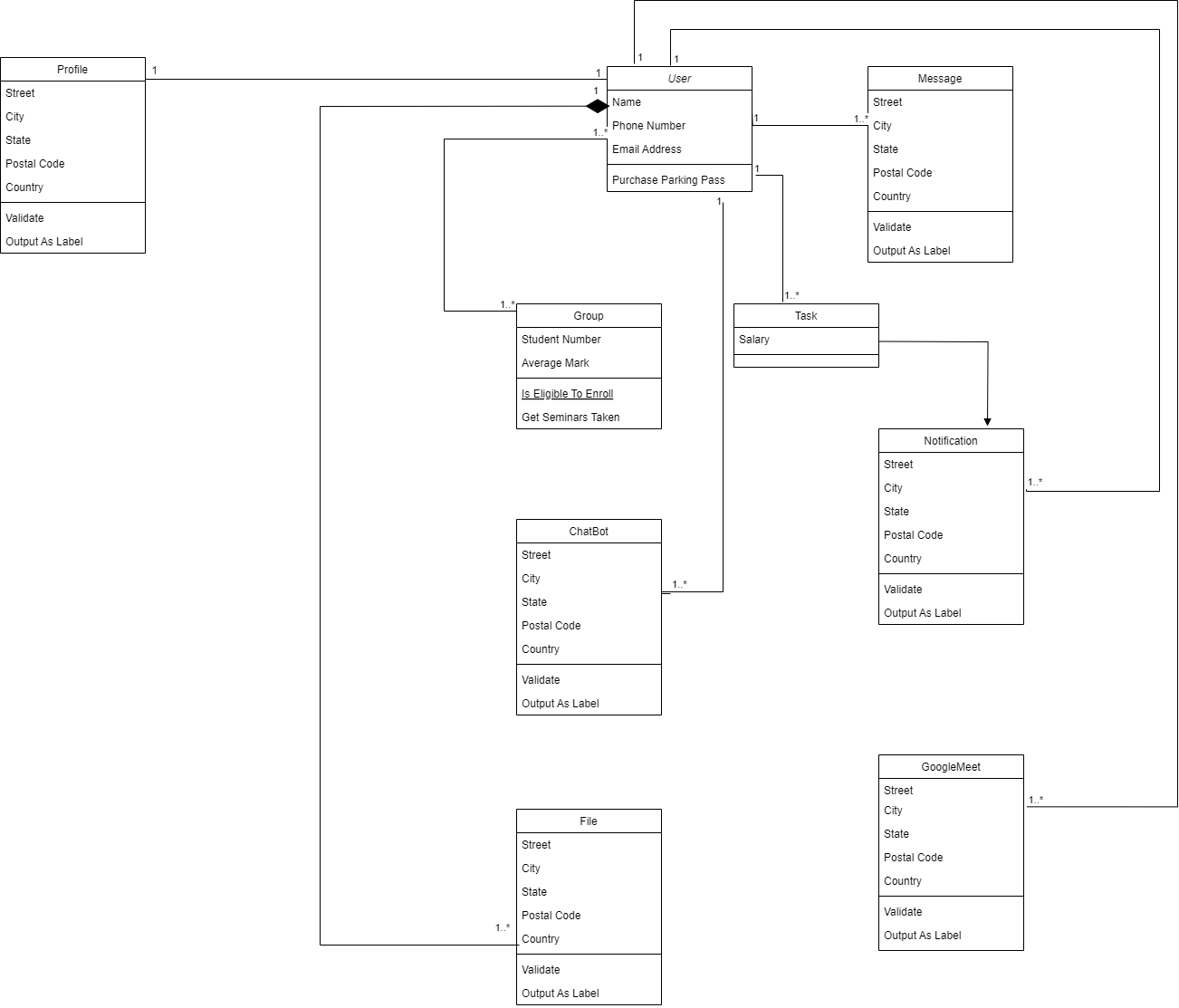
### Use-Case Diagram

### 3.2.1 Complete system



*Figure 3.1 Use-Case Diagram for complete system*

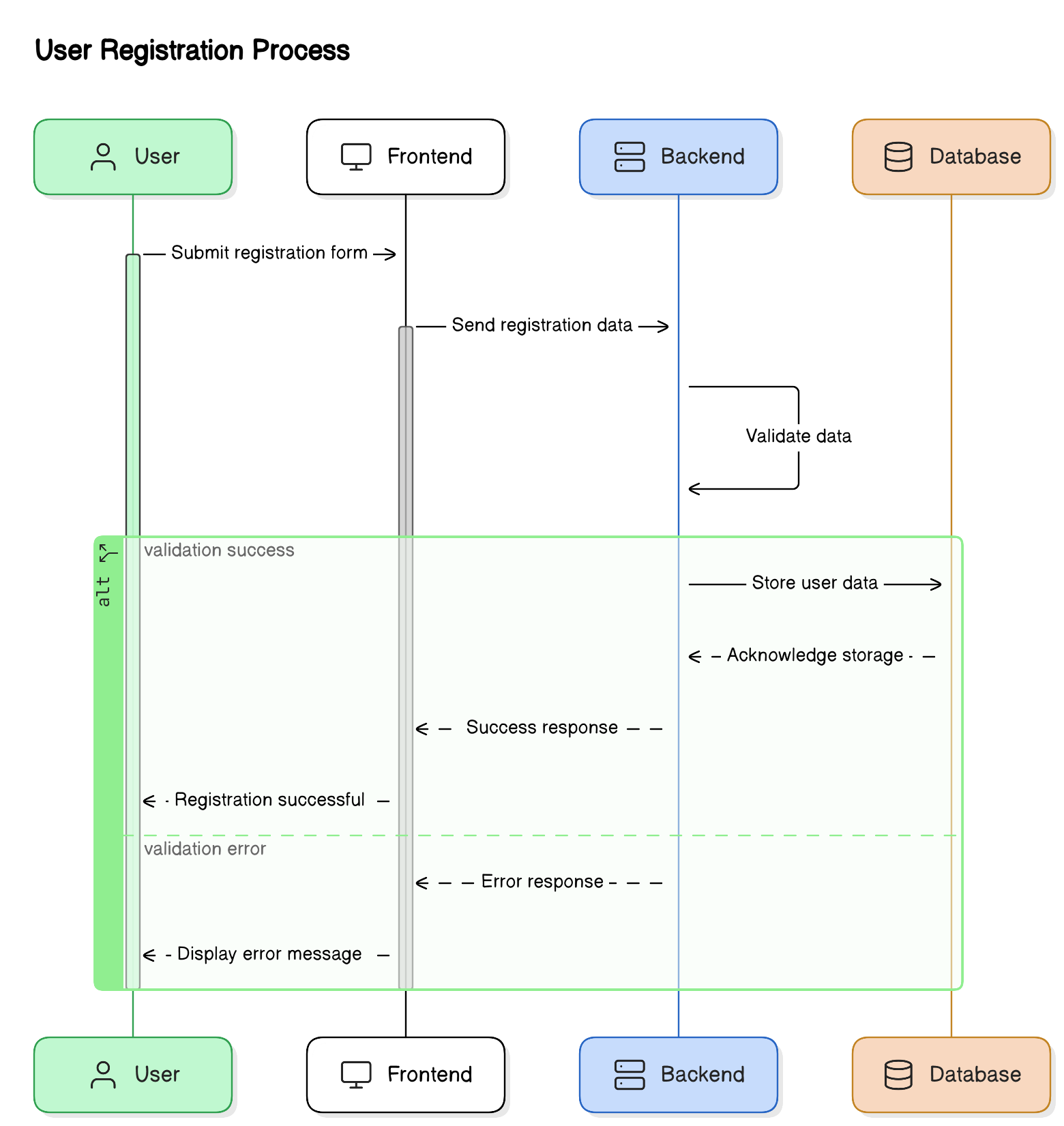
**3.3 Class Diagram**



*Figure 3.2 Class Diagram*

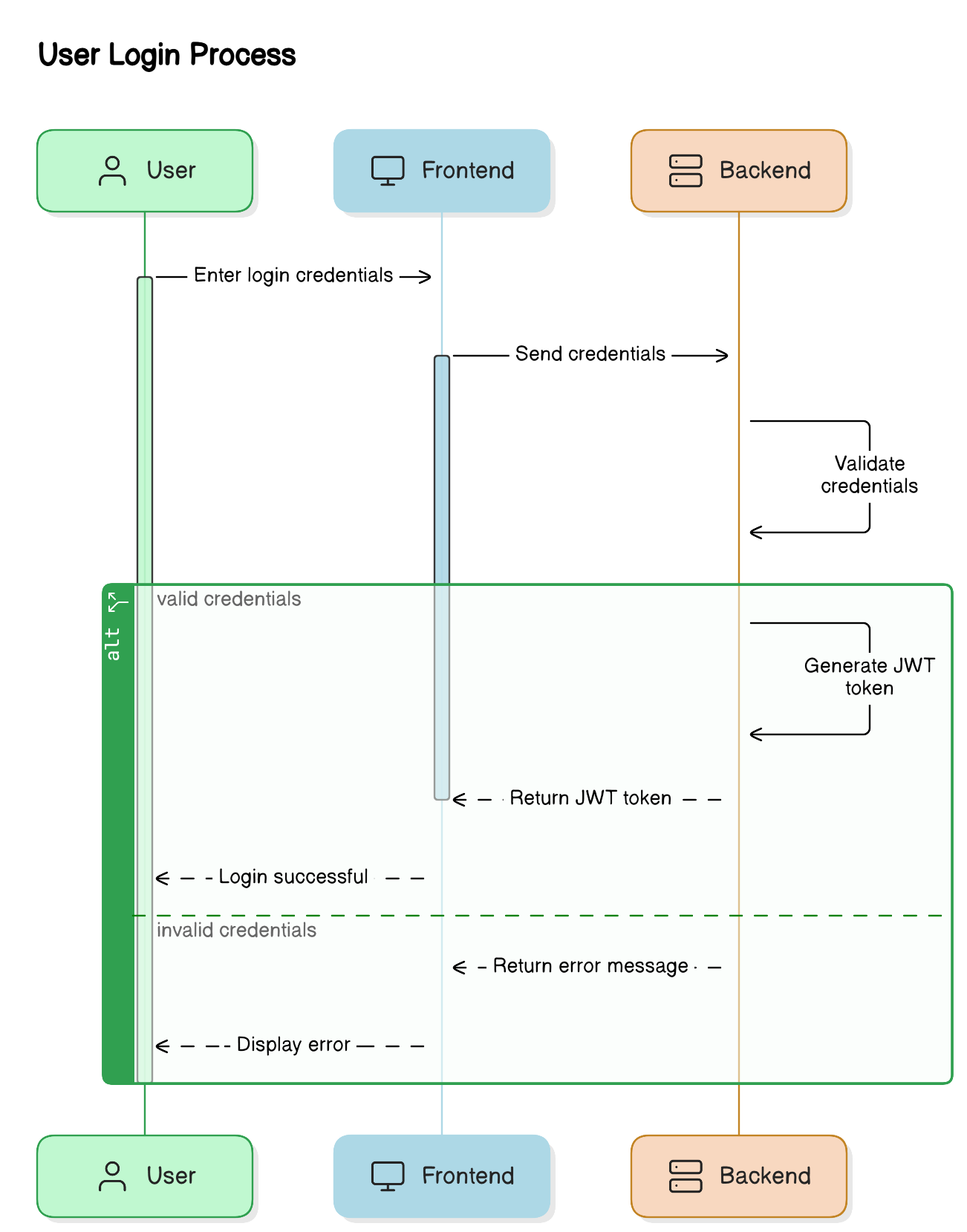
**3.4 Sequence Diagram**

### 3.4.1 User Registration



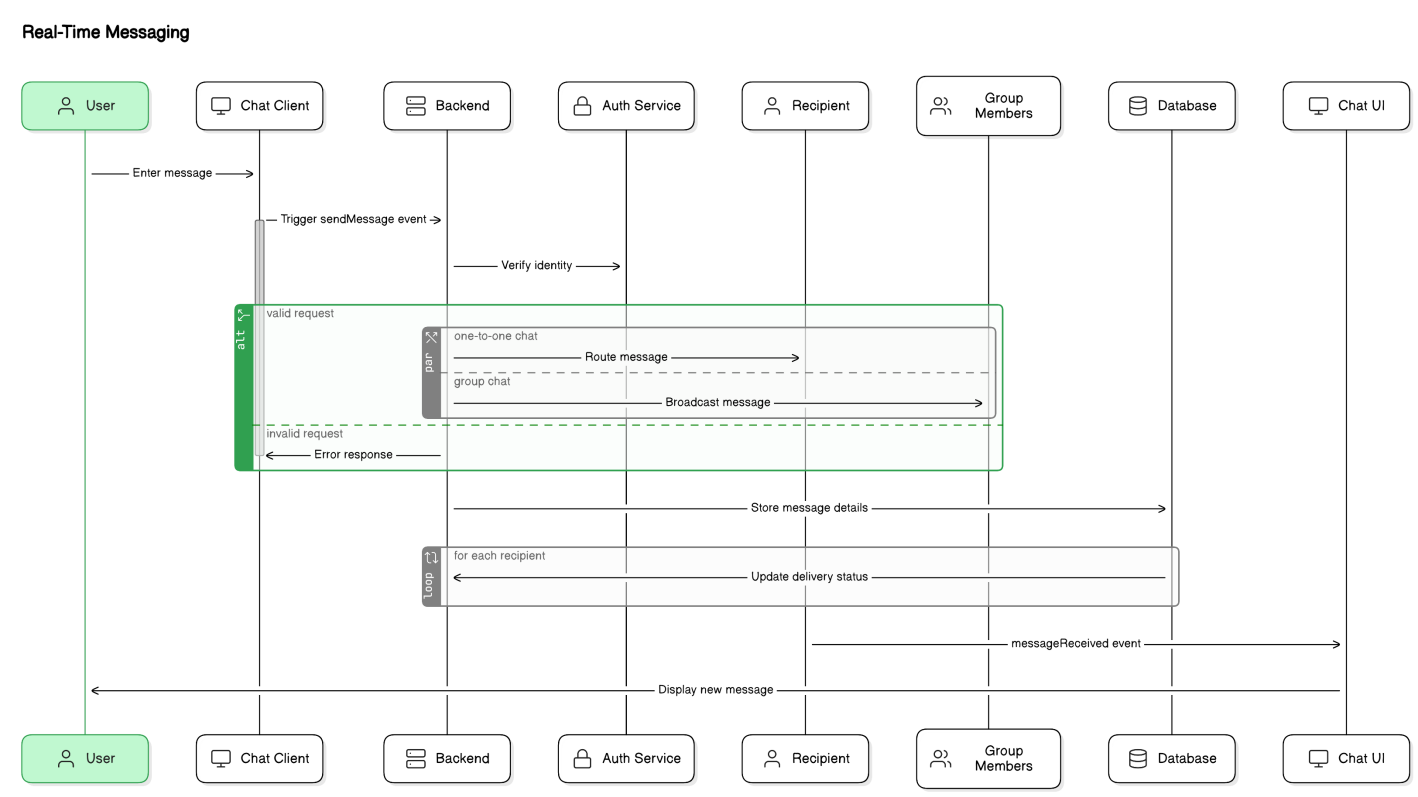
*Figure 3.3 Registration*

### 3.4.2 User Login



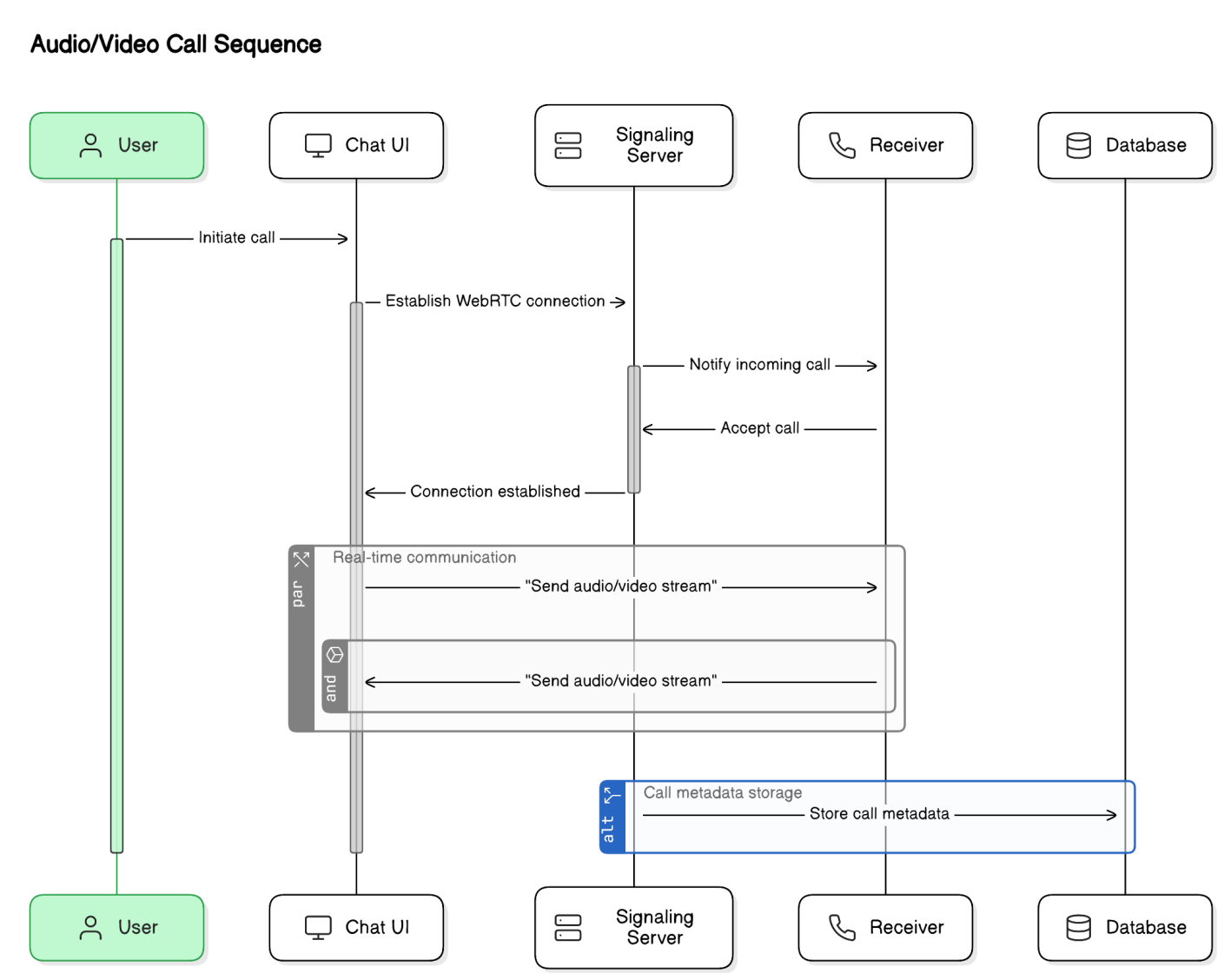
*Figure 3.4 Logins*

### 3.4.3 Real Time Messaging



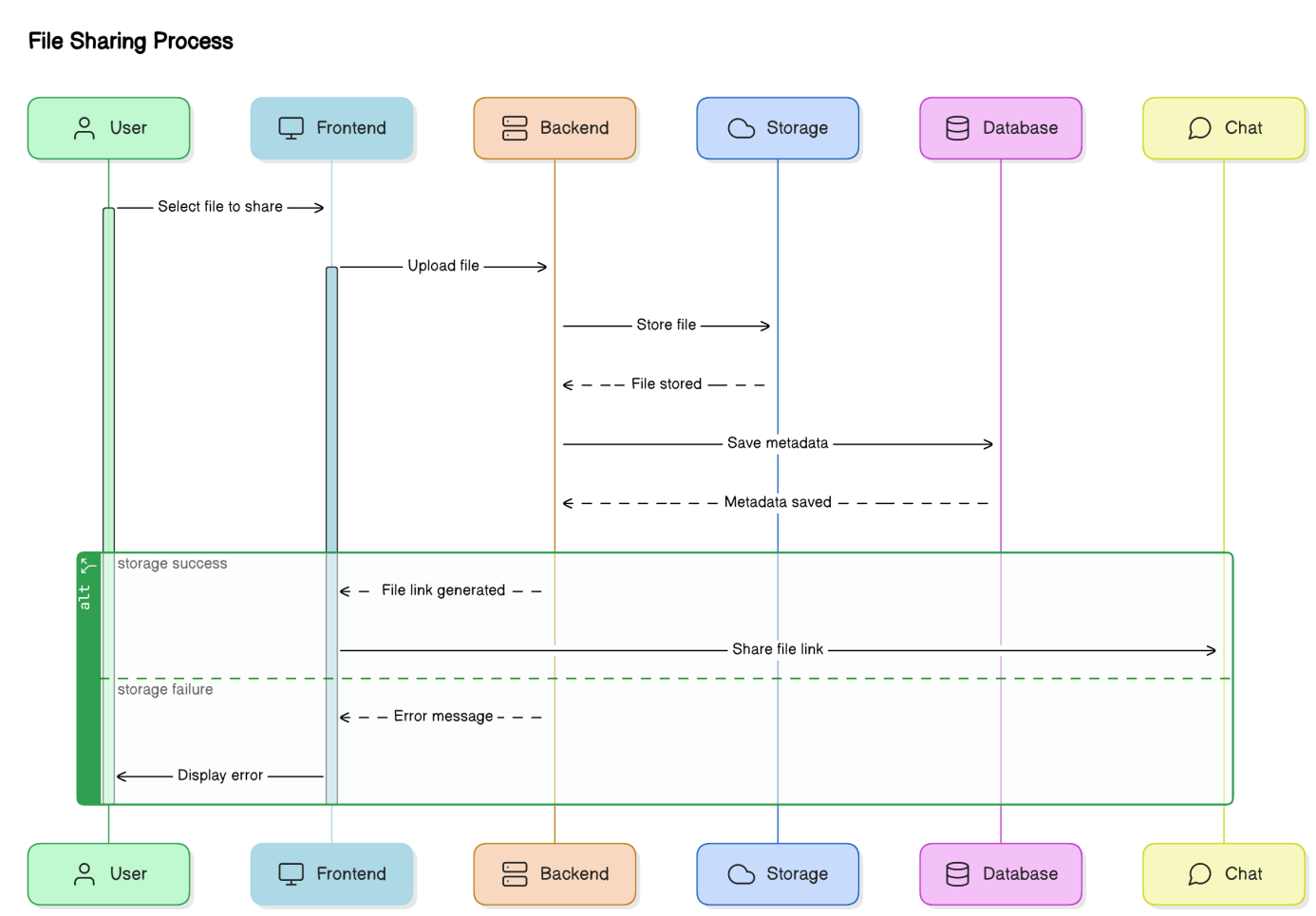
*Figure 3.5 Real Time Communication*

### 3.4.4 Audio/Video Call



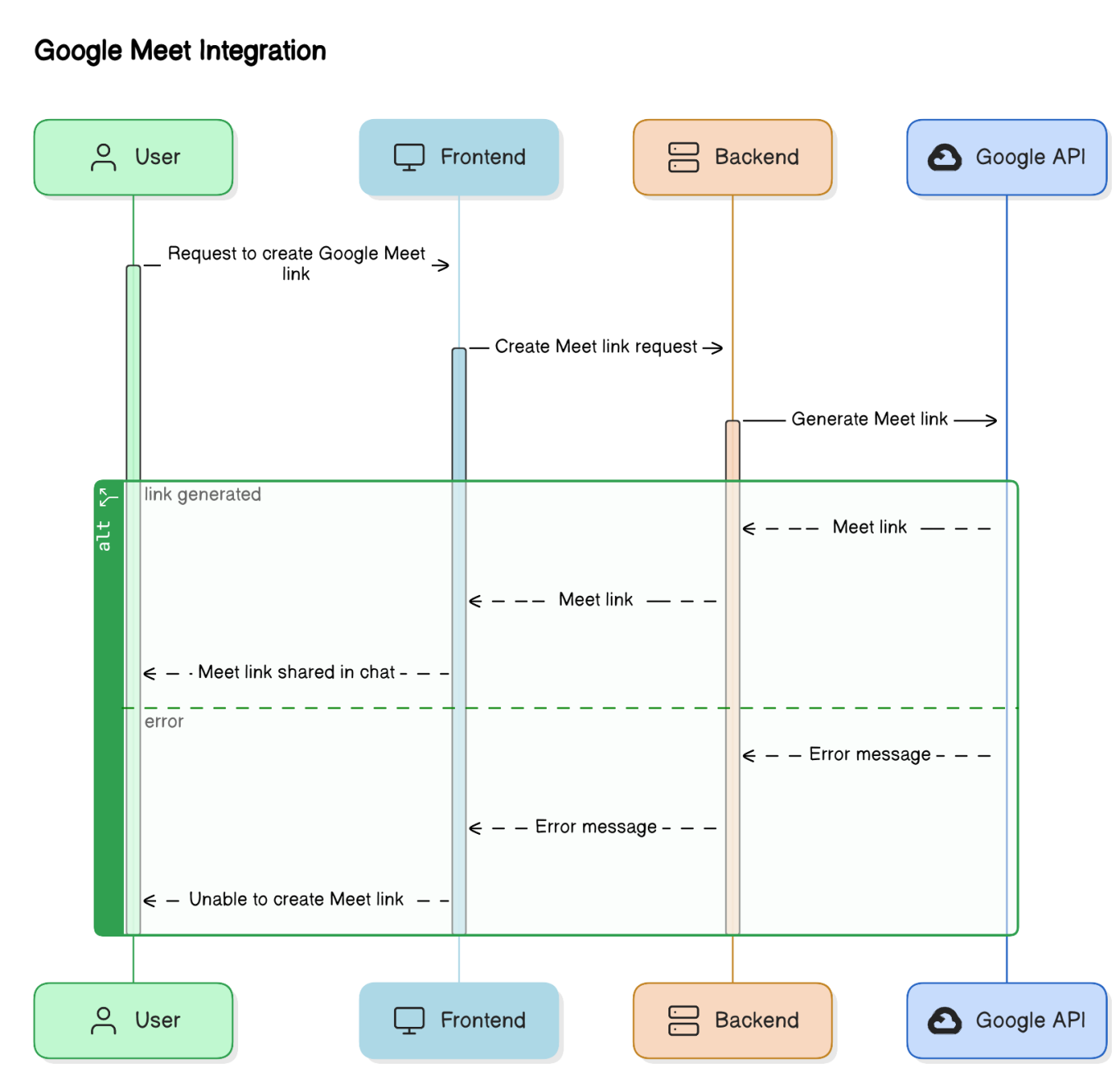
*Figure 3.6 Audio/Video Calls*

### 3.4.5 File Sharing



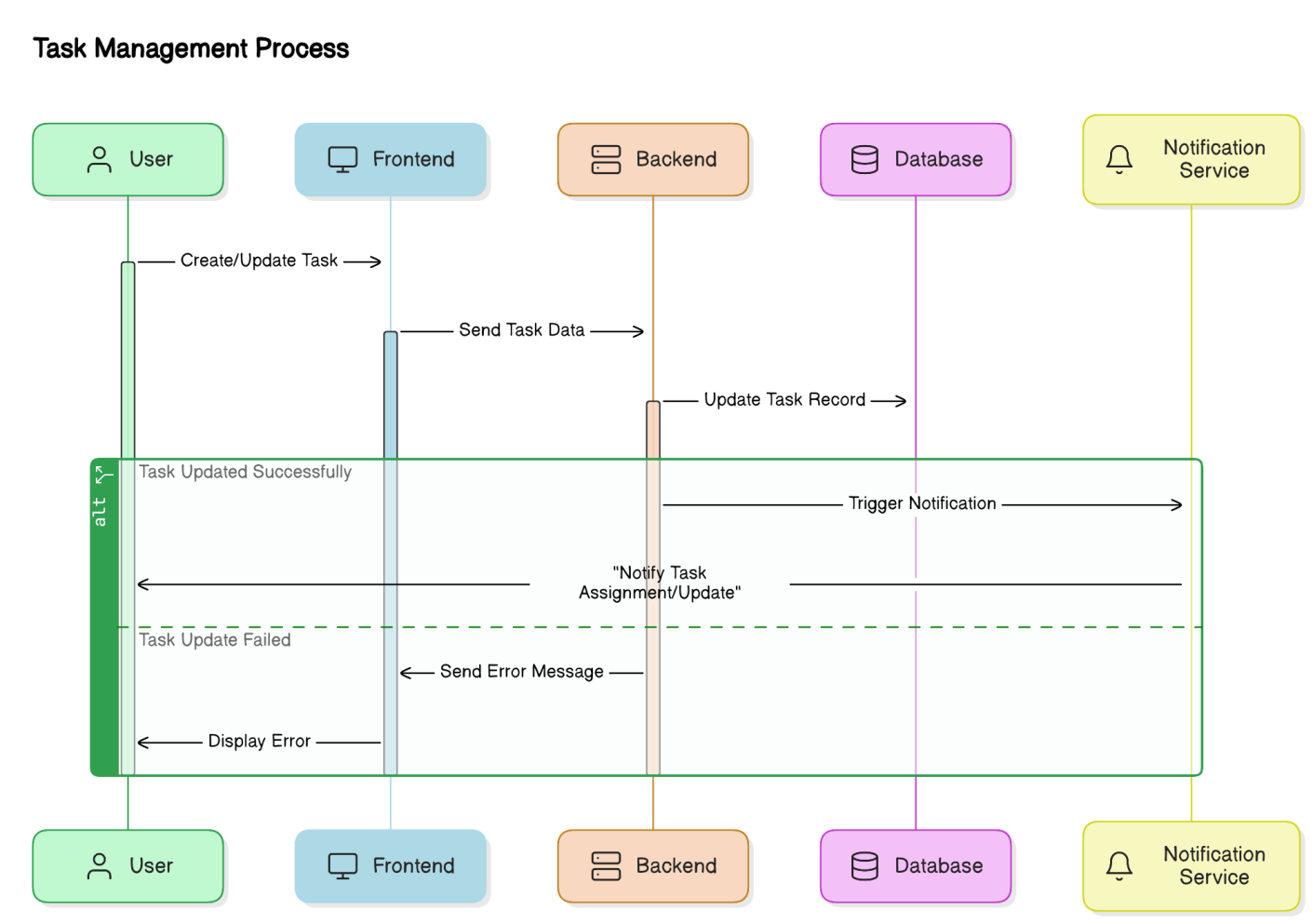
*Figure 3.7 file Sharing*

### 3.4.6 Google Meet Intergration



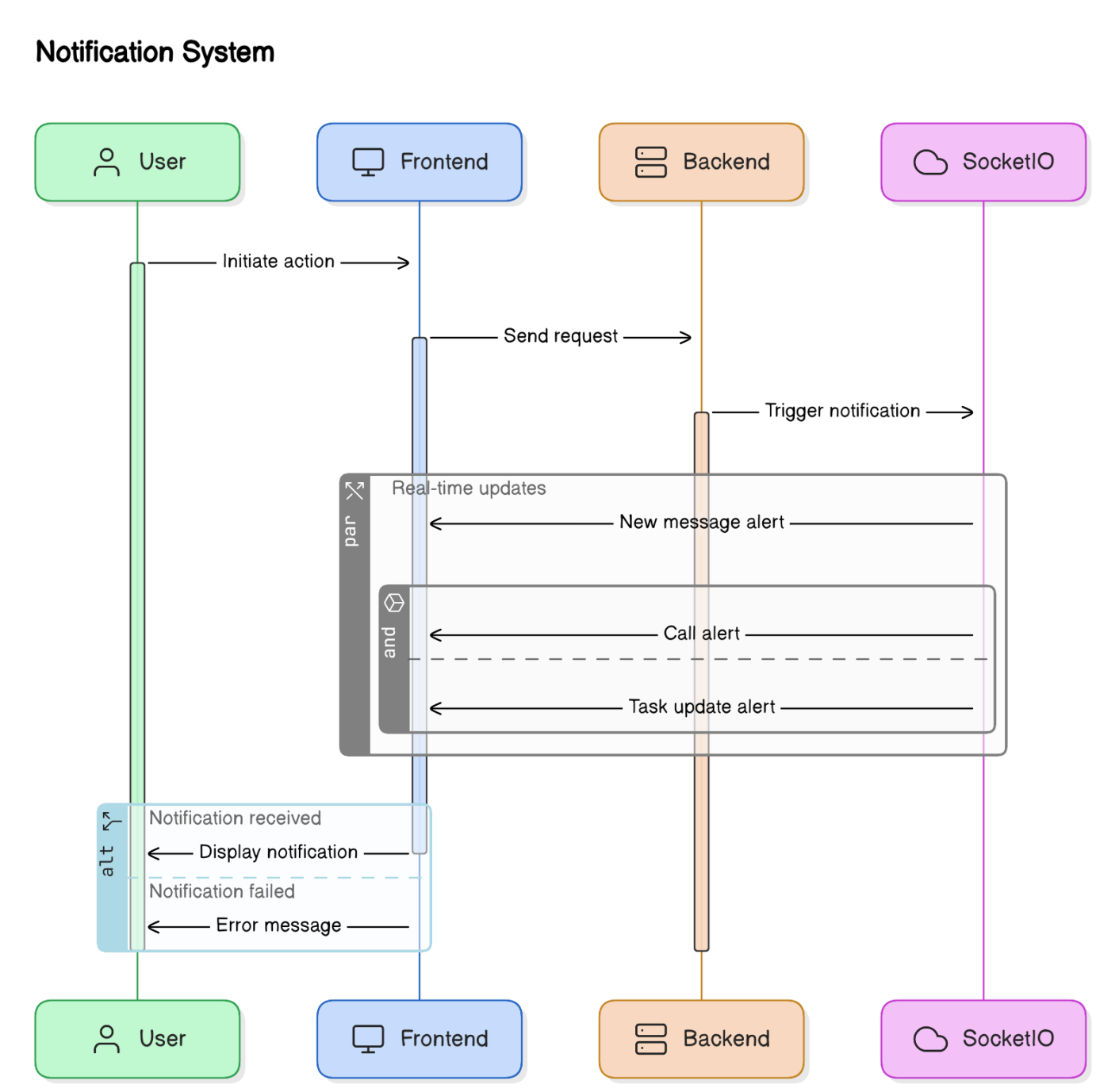
*Figure 3.8 Google Meet*

### 3.4.7 Task Managment



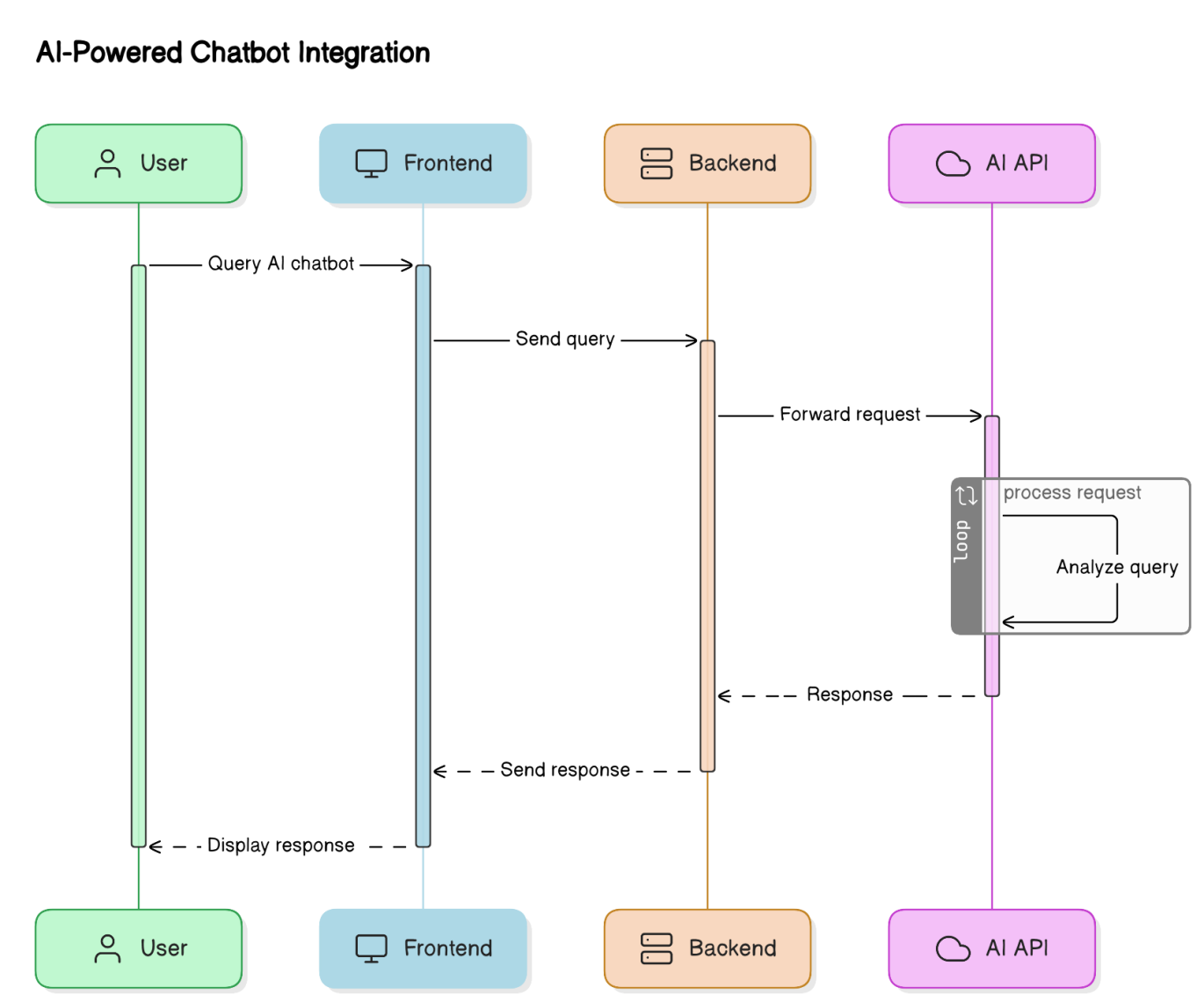
*Figure 3.9 Task Management*

### 3.4.8 Notification



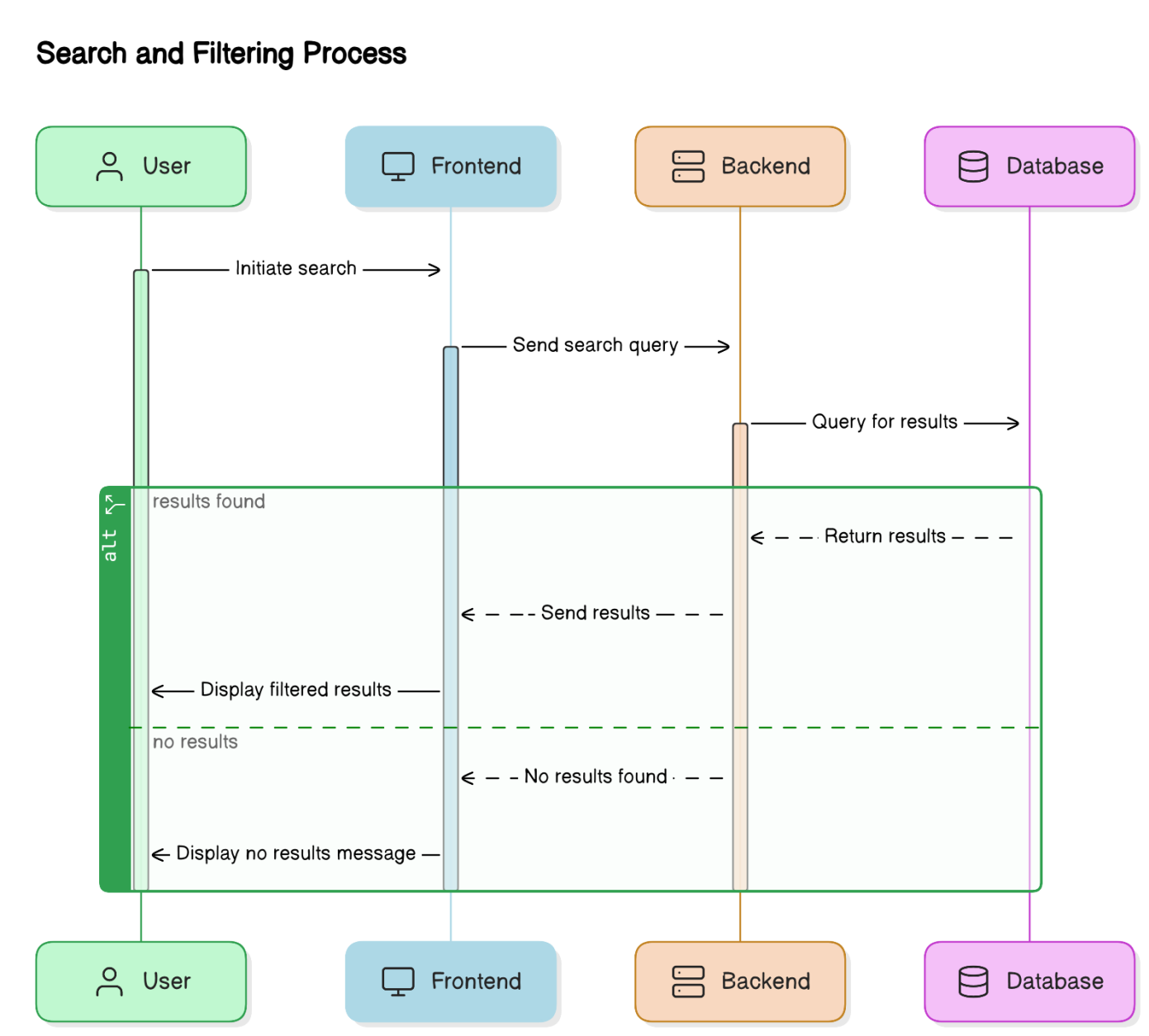
*Figure 3.10 Notification*

### 3.4.9 Ai Power Chatbot



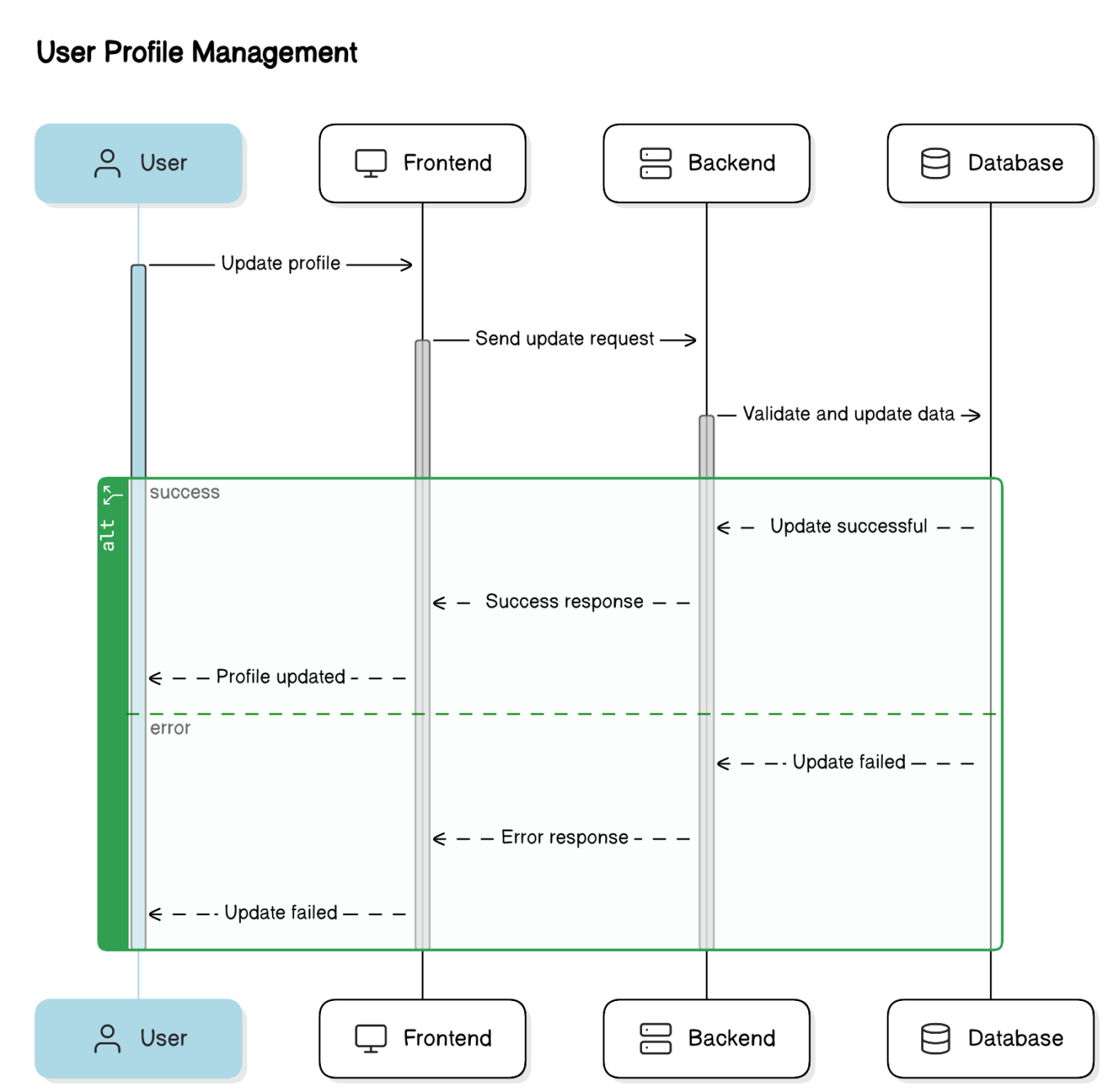
*Figure 3.11 Ai Power ChatBot*

### 3.4.10 Searching And filtering

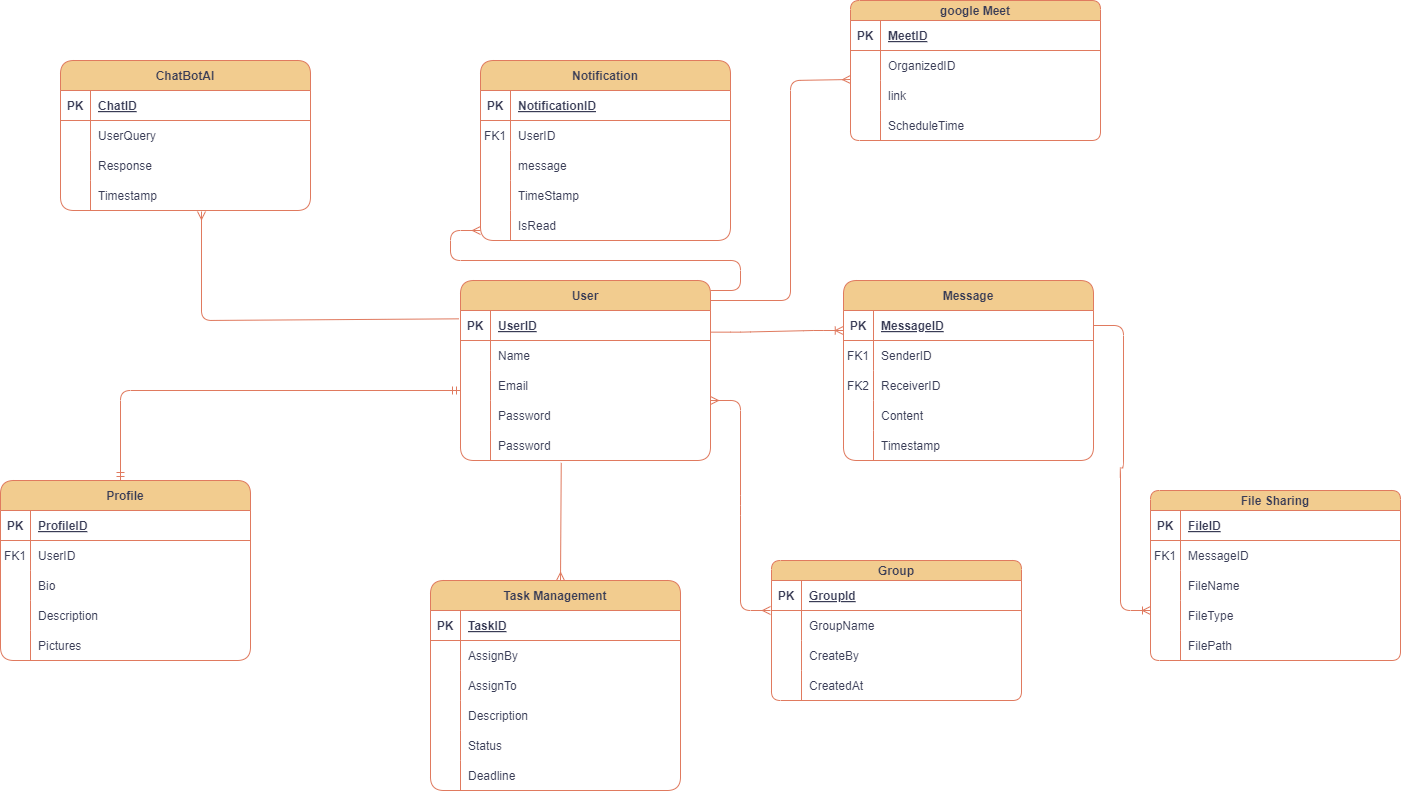


*Figure 3.12 Searching and Filtering*

### 3.4.10 Profile Management



*Figure 3.13 Profile Management*

**

*Figure 3.14 ERD*