# Software Design Specification

## for

## The Student Assistant: AI-Enabled Academic and Financial Management

**Version 1.0**


**Sukkur IBA University**

**Prepared by 20F-33**

**09 November, 2023**

# Supervisor Approval

**Team Members:**

Muhammad Aizaz Ullah Khan

Agha Kaleemullah Khan

Asghar Ali Shah

_____

**Dr. Asif Khan**

# Table of Contents

# 1. Introduction

## 1.1 Purpose

The purpose of this Software Design Specification (SDS) document for the The Student Assistant is to provide a comprehensive and detailed account of the software's design and architecture. It is a critical reference for the development team, as well as other stakeholders involved in the software development process. The primary objectives of this SDS are as follows:

- **Design Communication:** The SDS aims to communicate the design intent of the The Student Assistant. It provides a clear and detailed description of how the software will be structured, the relationships between its components, and the flow of data and control.

- **Development Guidance:** Developing team will use this document as a road map for implementing the software. It offers detailed insights into the software's architecture, including the design of individual modules, data storage, and how different parts of the system interact with each other.

- **Quality Assurance:** Development team will rely on the SDS to understand the expected behavior of the software. This understanding is crucial for the creation of effective test cases and ensuring that the software is thoroughly tested.

- **Maintenance Support:** Over the software's lifecycle, this SDS will serve as a reference for maintenance activities and future enhancements. It helps in identifying the underlying design choices and architectural decisions.

- **Collaboration:** The SDS document promotes collaboration among various stakeholders by providing a common reference point for the software's design. It ensures that all team members are aligned with the software's architecture and design principles.

- **Design Compliance:** The document may specify design standards, constraints, and best practices that need to be adhered to. This ensures that the software's design aligns with established architectural standards and guidelines.

- **Informing Decision-Making:** The SDS also communicates design constraints and potential impacts on the development process, which can influence project planning and decision-making.

The primary audience for this document includes software development team, project supervisor and any other stakeholders who need a deep understanding of the software's design. It is assumed that the readers of this document have a foundational understanding of software design principles.

As the development process progresses, the SDS will be periodically updated to reflect any design modifications, enhancements, or clarifications that may arise during the software's development lifecycle. This ensures that the document remains a valuable and relevant resource throughout the project.

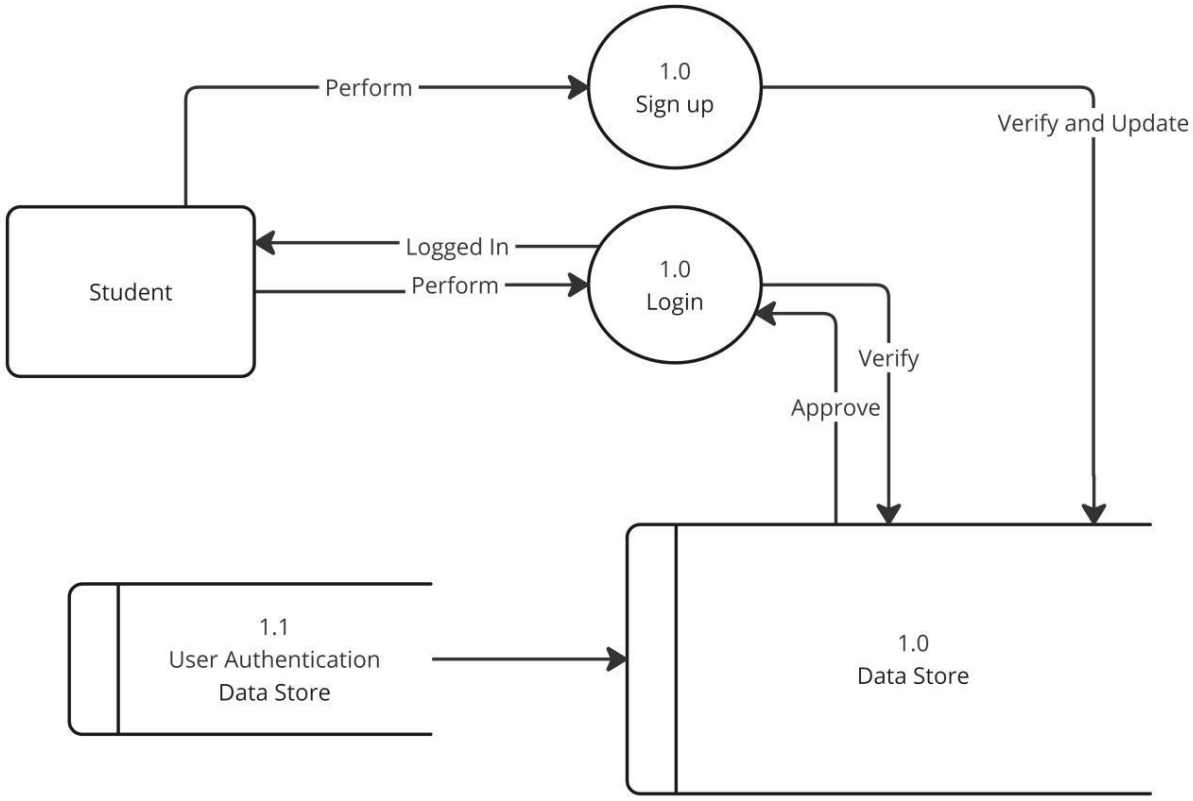
The SDS is organized into sections that comprehensively cover different aspects of the software's design, including architecture, interfaces, data flow, functional flow,

and more. Each section provides detailed insights into a specific aspect of the design, contributing to a holistic understanding of the software's architecture.

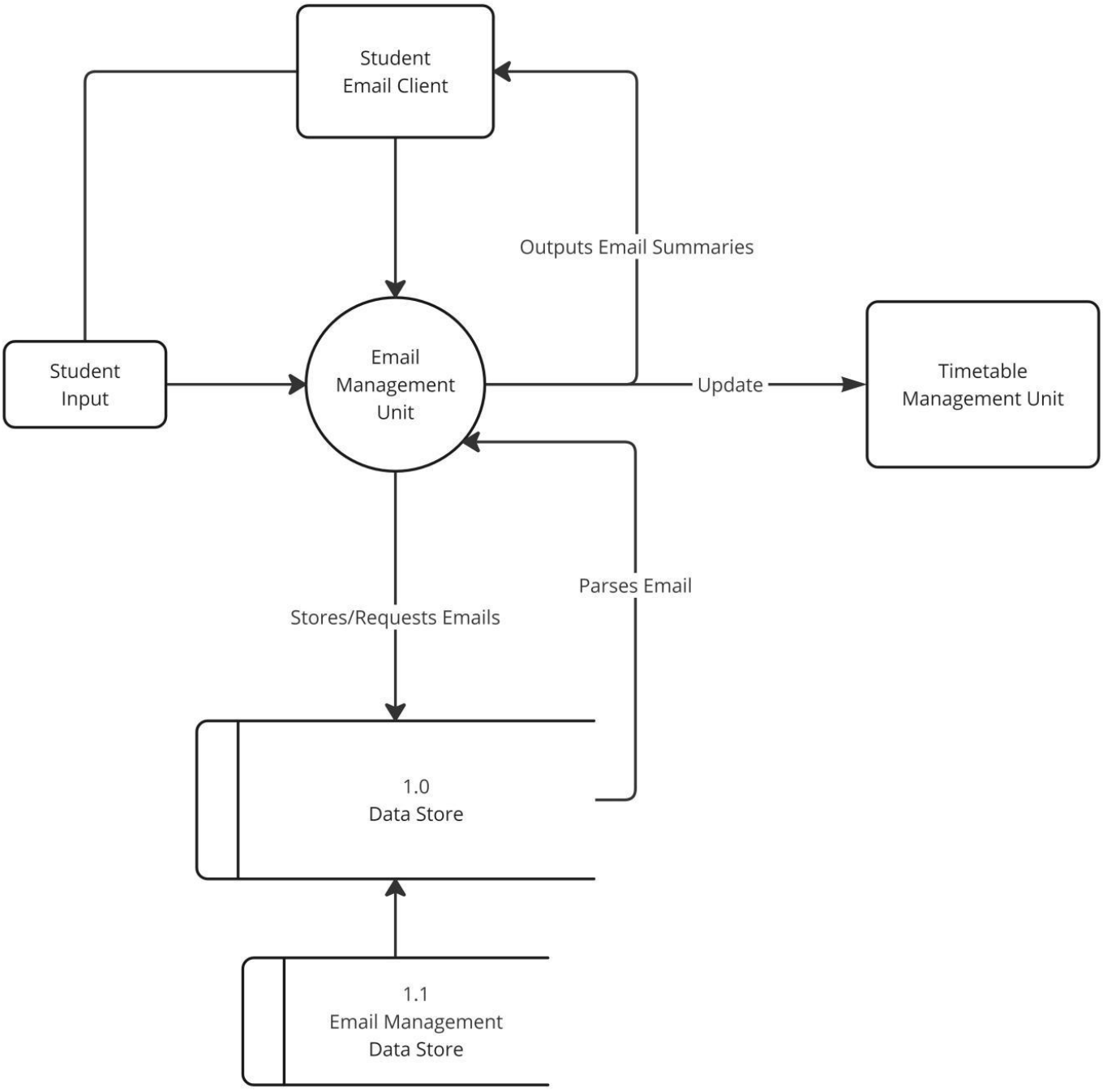
# 2. Software Design Diagrams

## 2.1 Data Flow Diagram

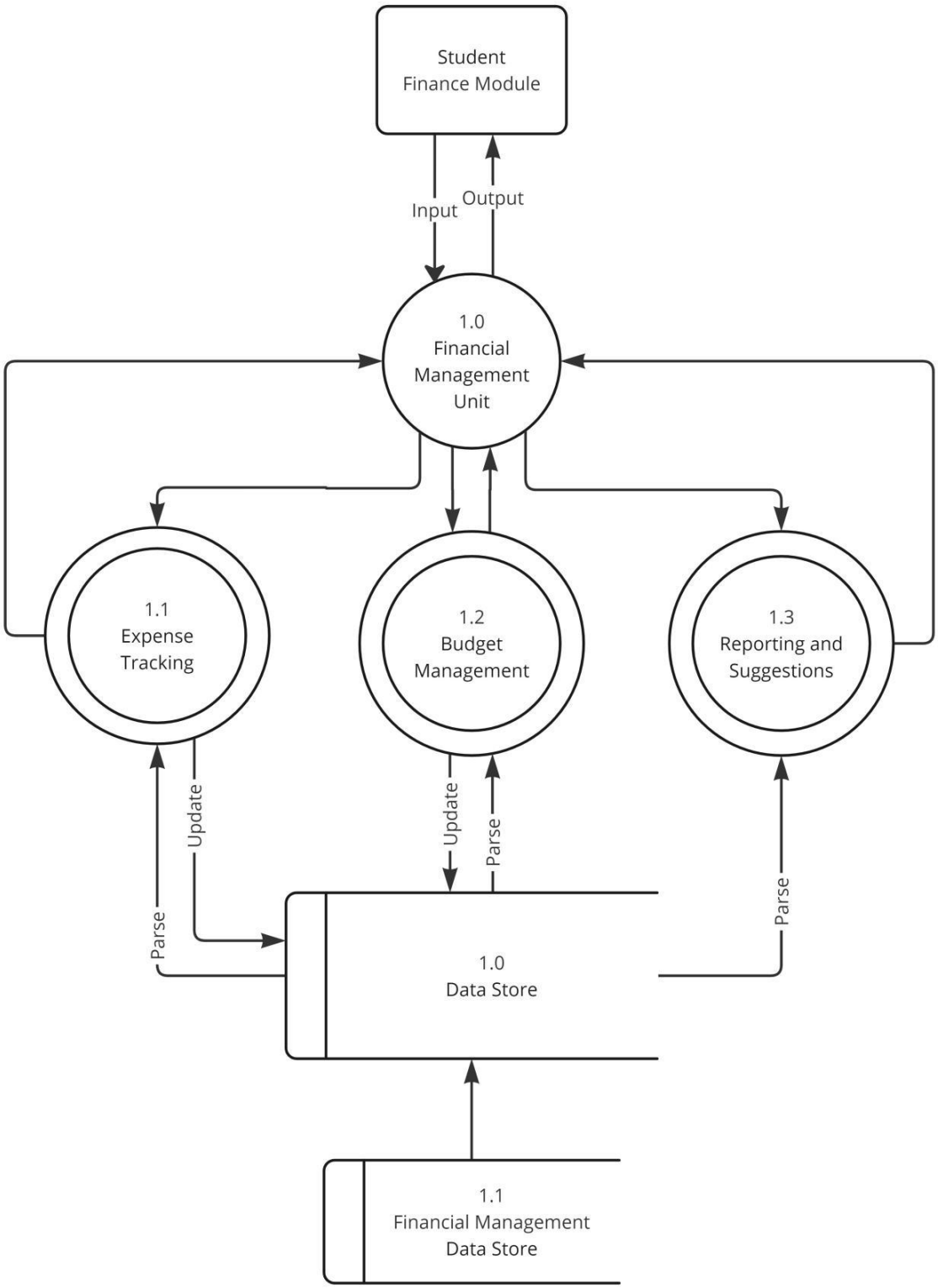
### 2.1.1 Login/Sign up



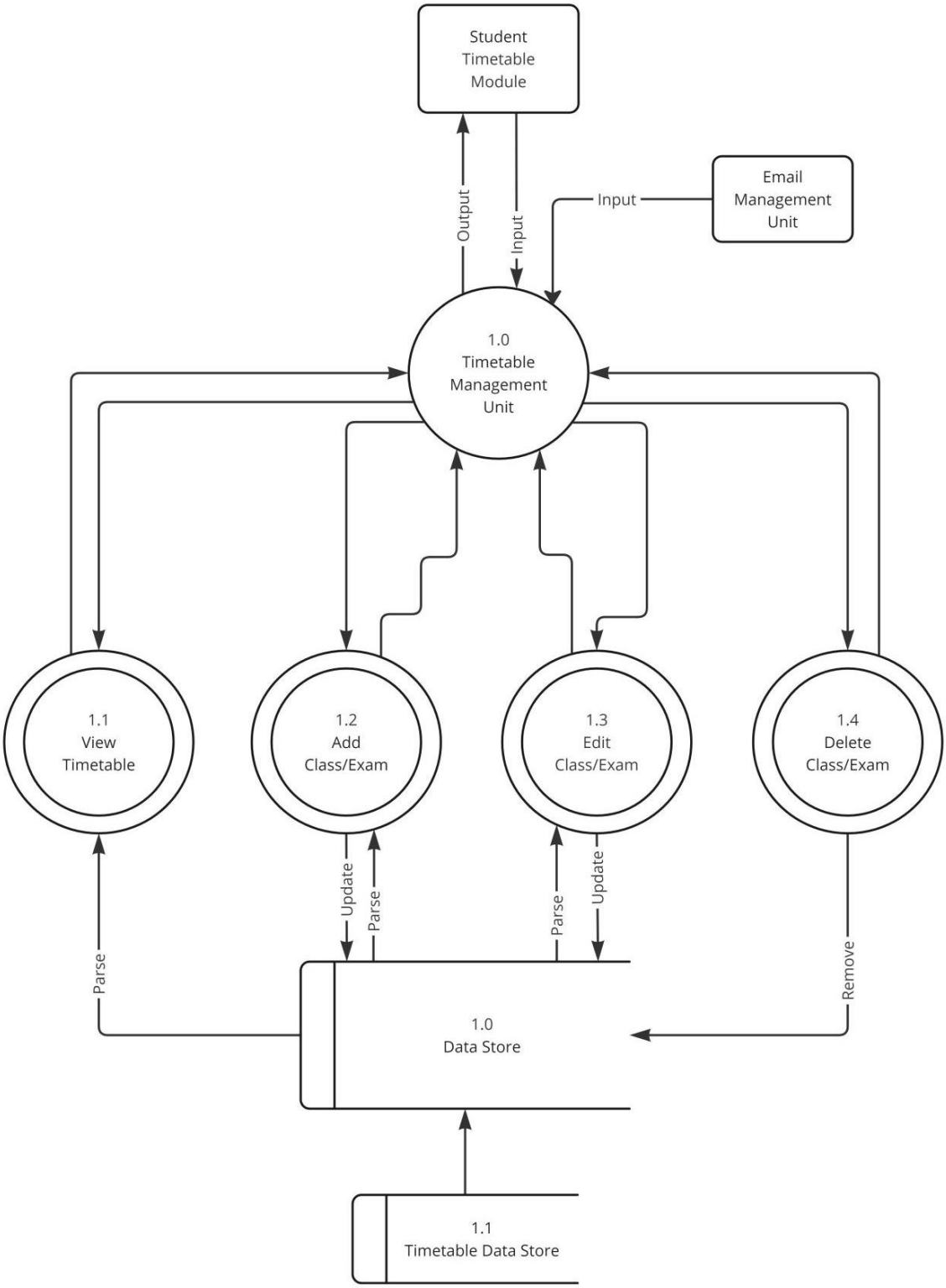
## 2.1.2 Email Management



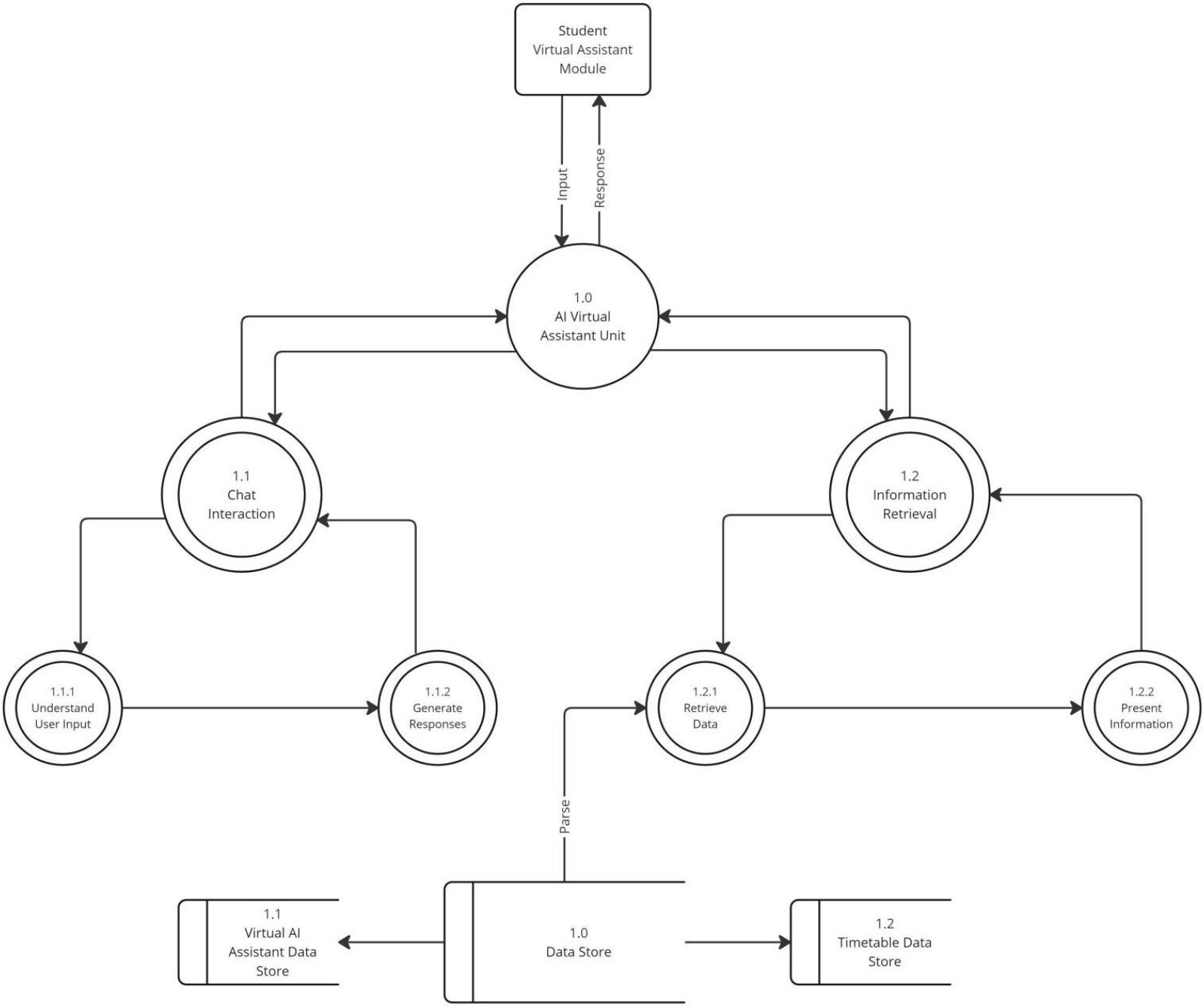
### 2.1.3 Finance Management



## 2.1.4 Timetable Management

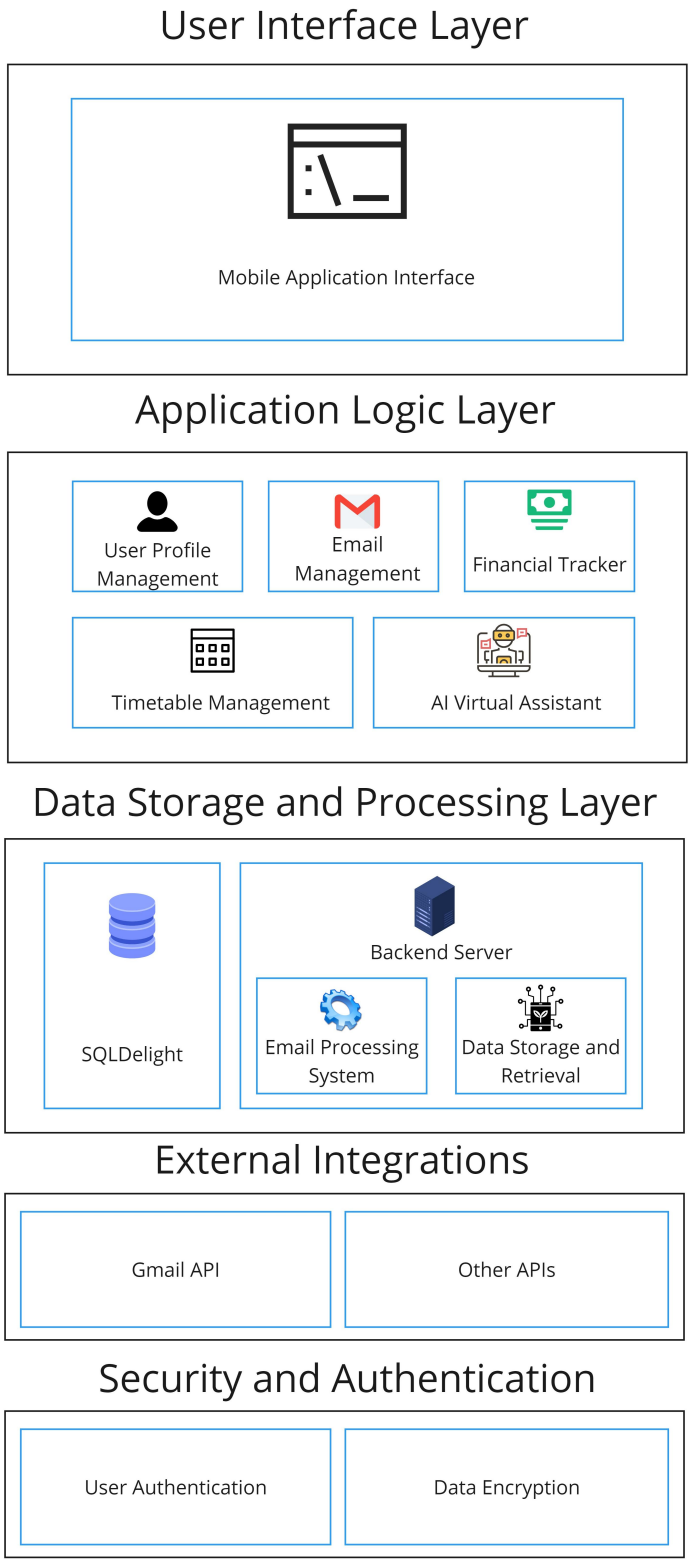


## 2.1.5 Virtual AI Assistant

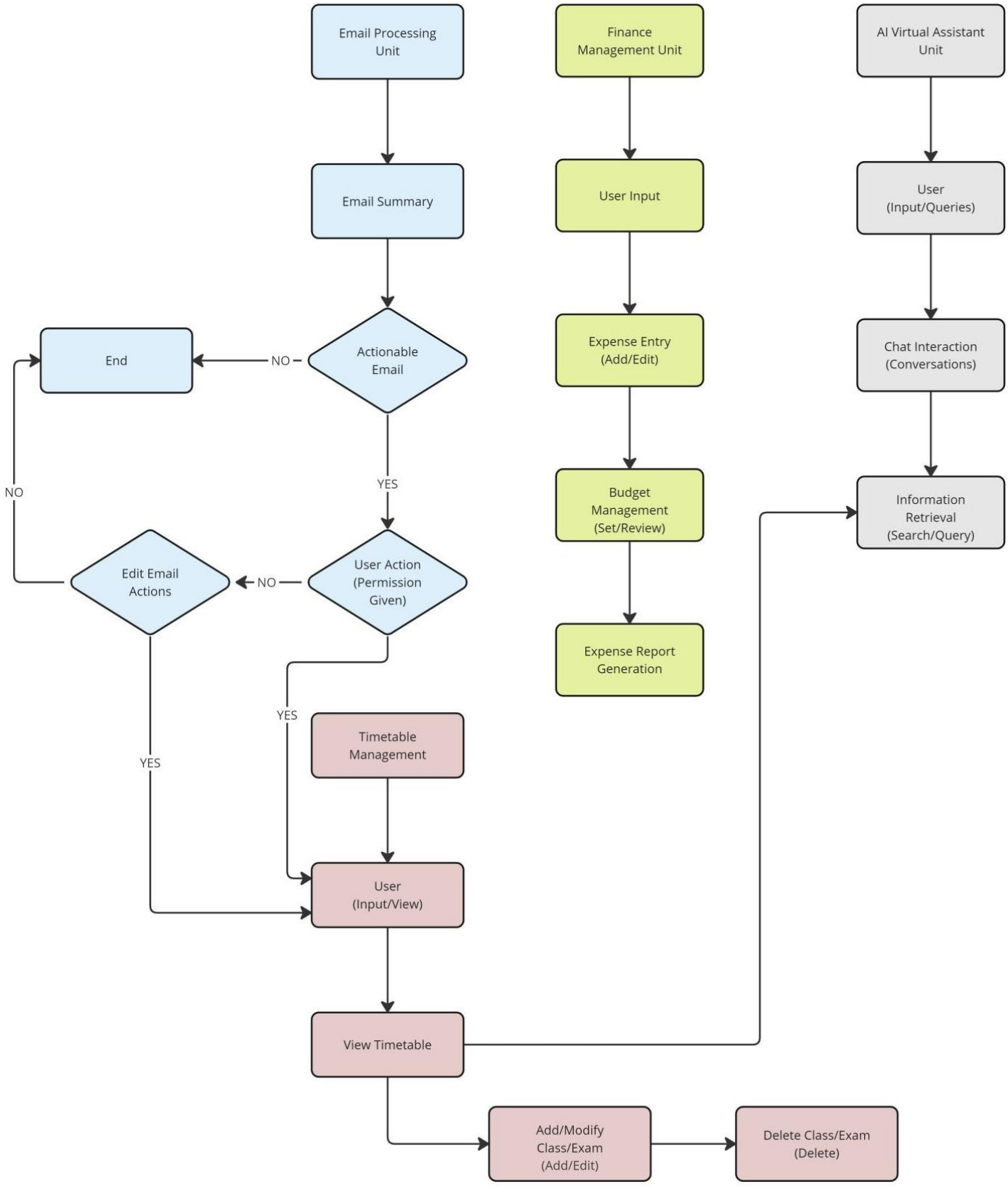


## 2.2 Architecture Design Diagram

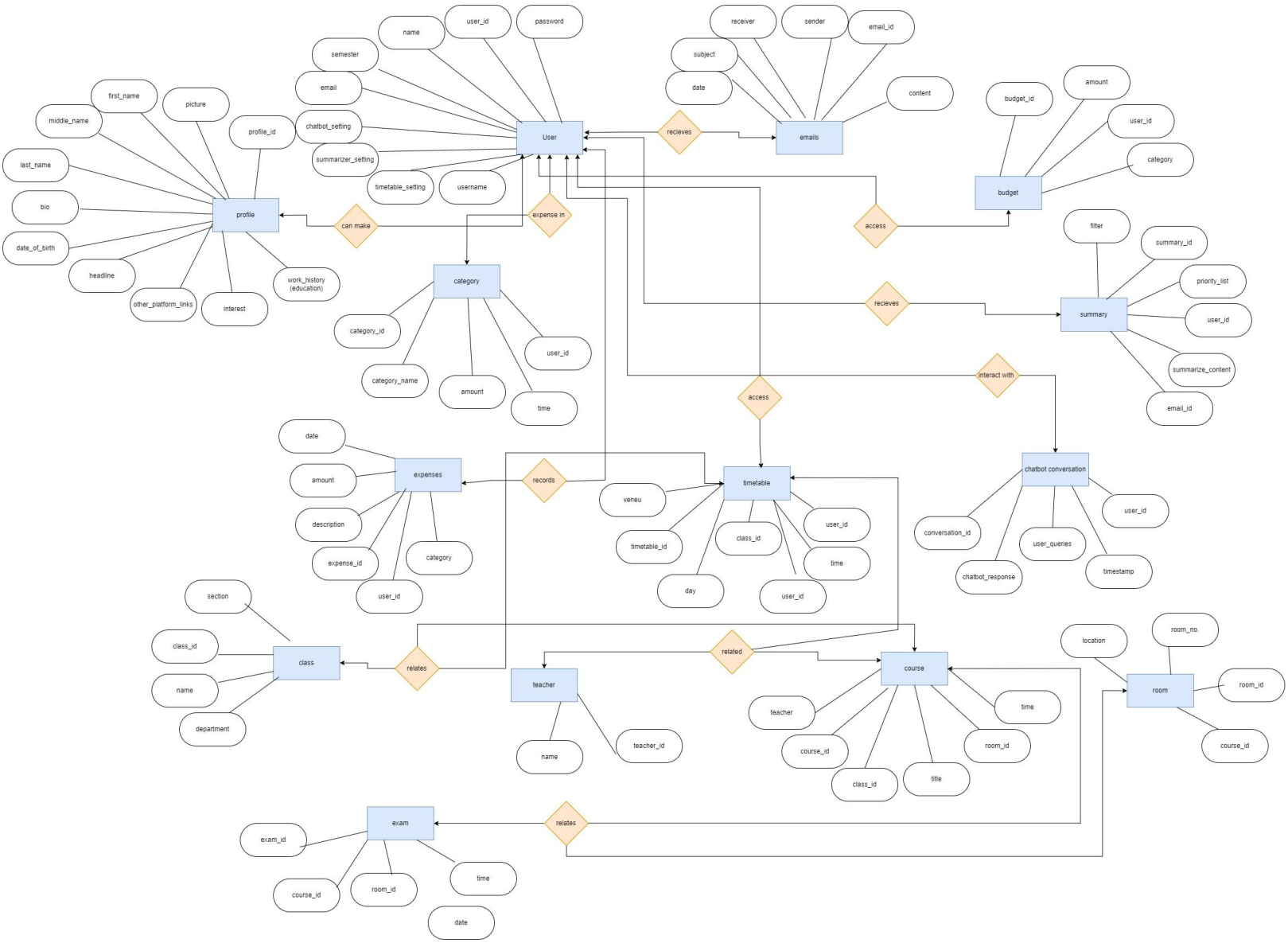
### 2.2.1 Layered Approach



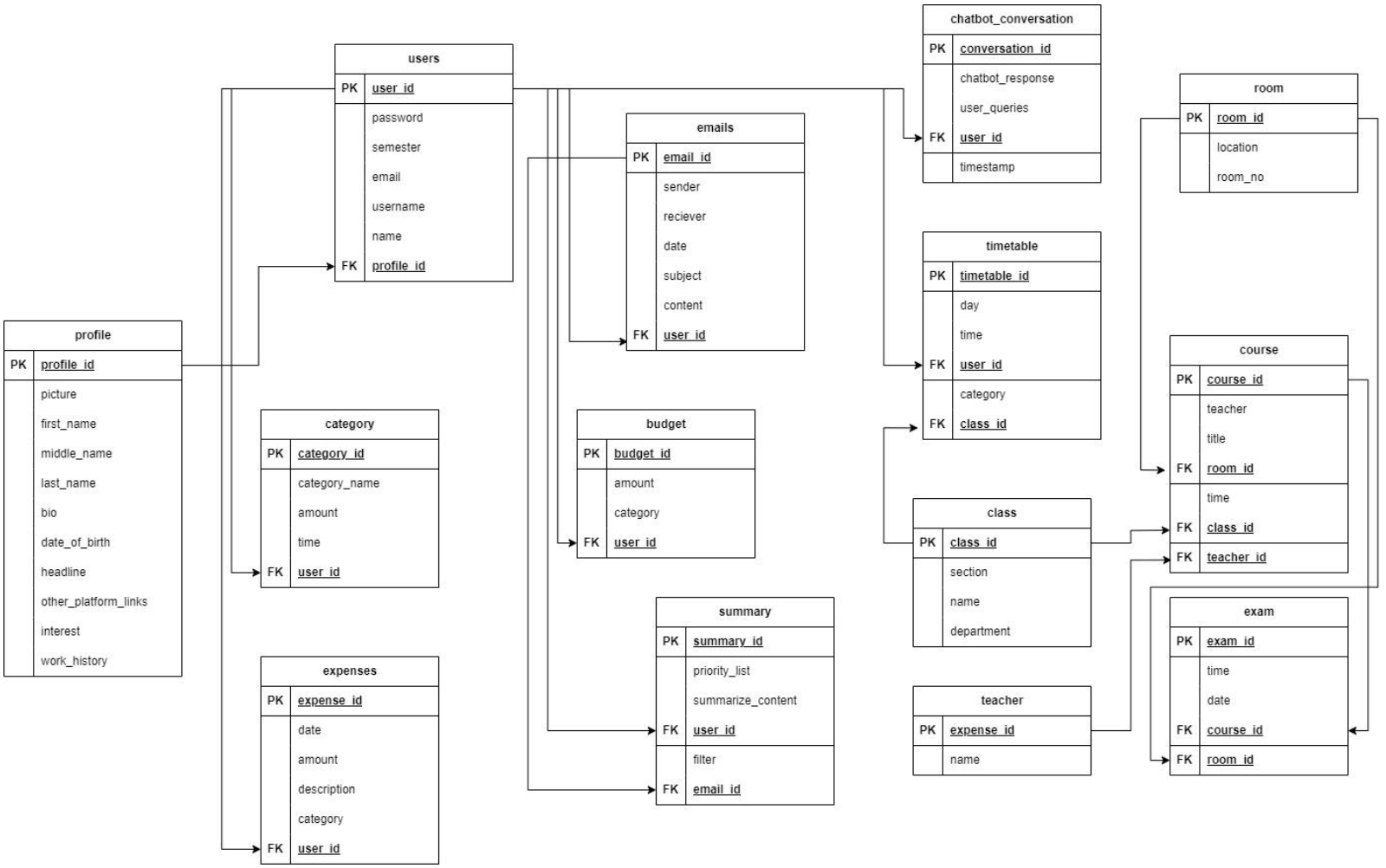
# 2.3 Functional Flow Diagram



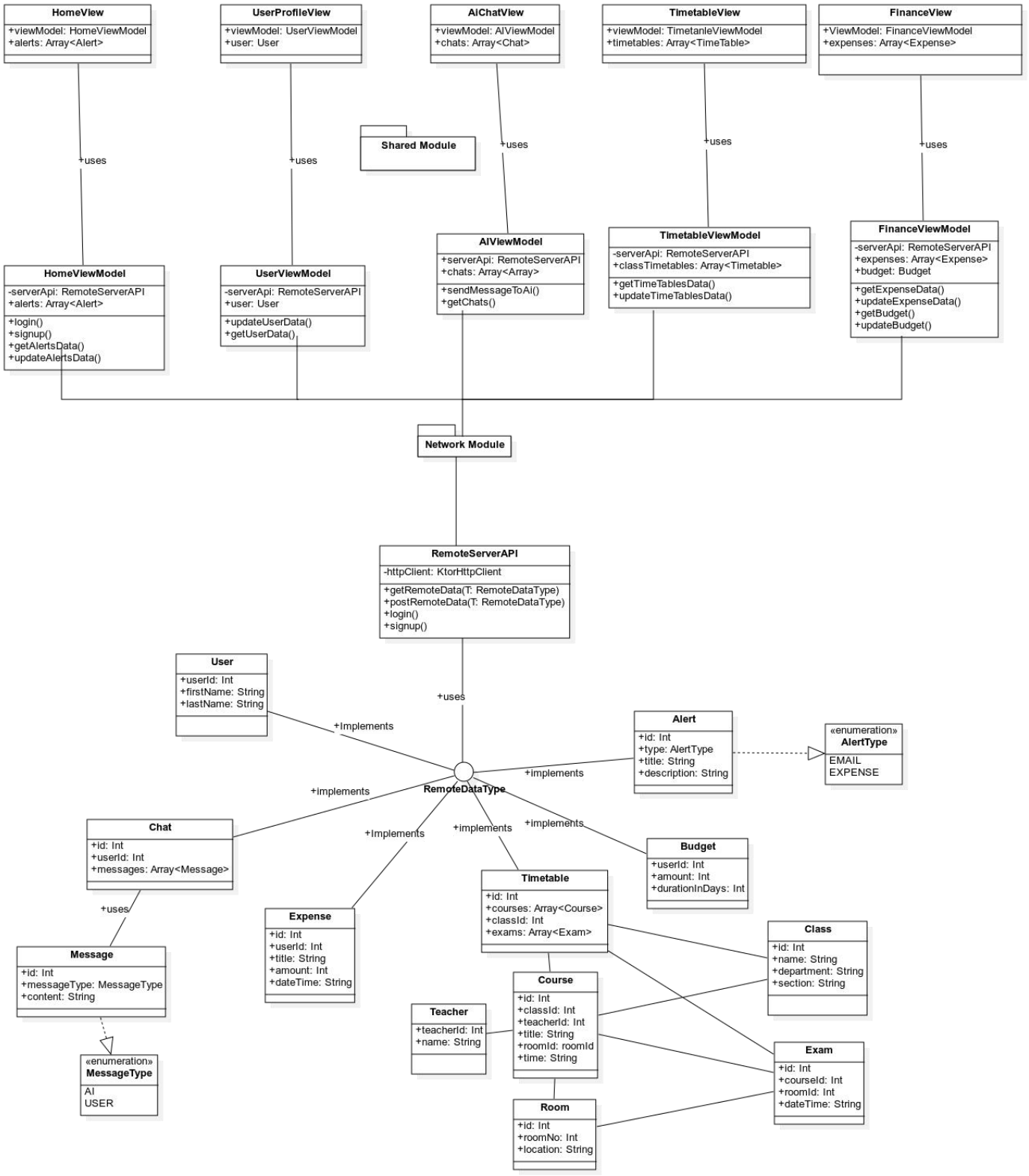
# 2.4 Entity Relationship Diagram



# 2.5 Database Diagram

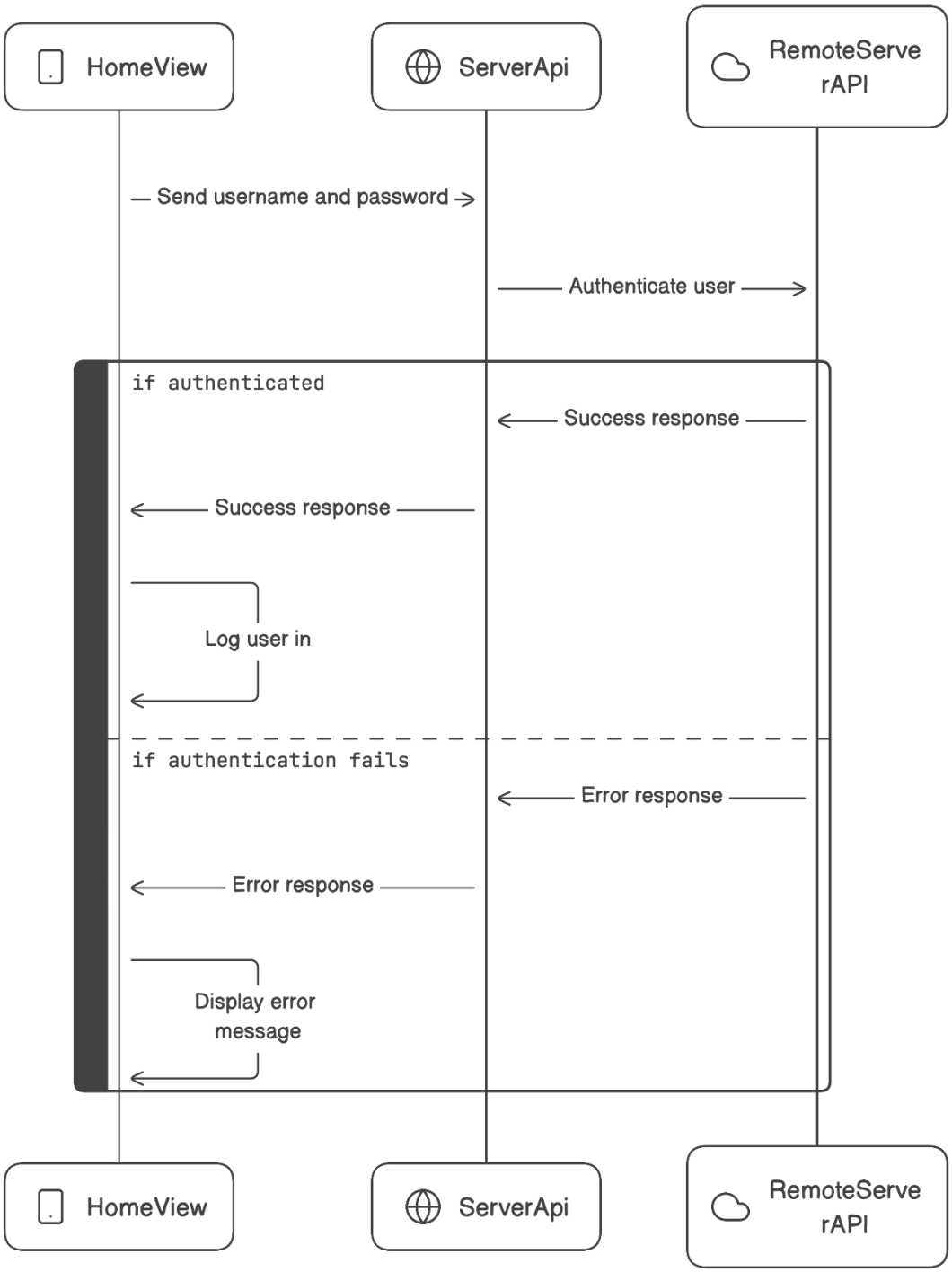


# 2.6 Class Diagram

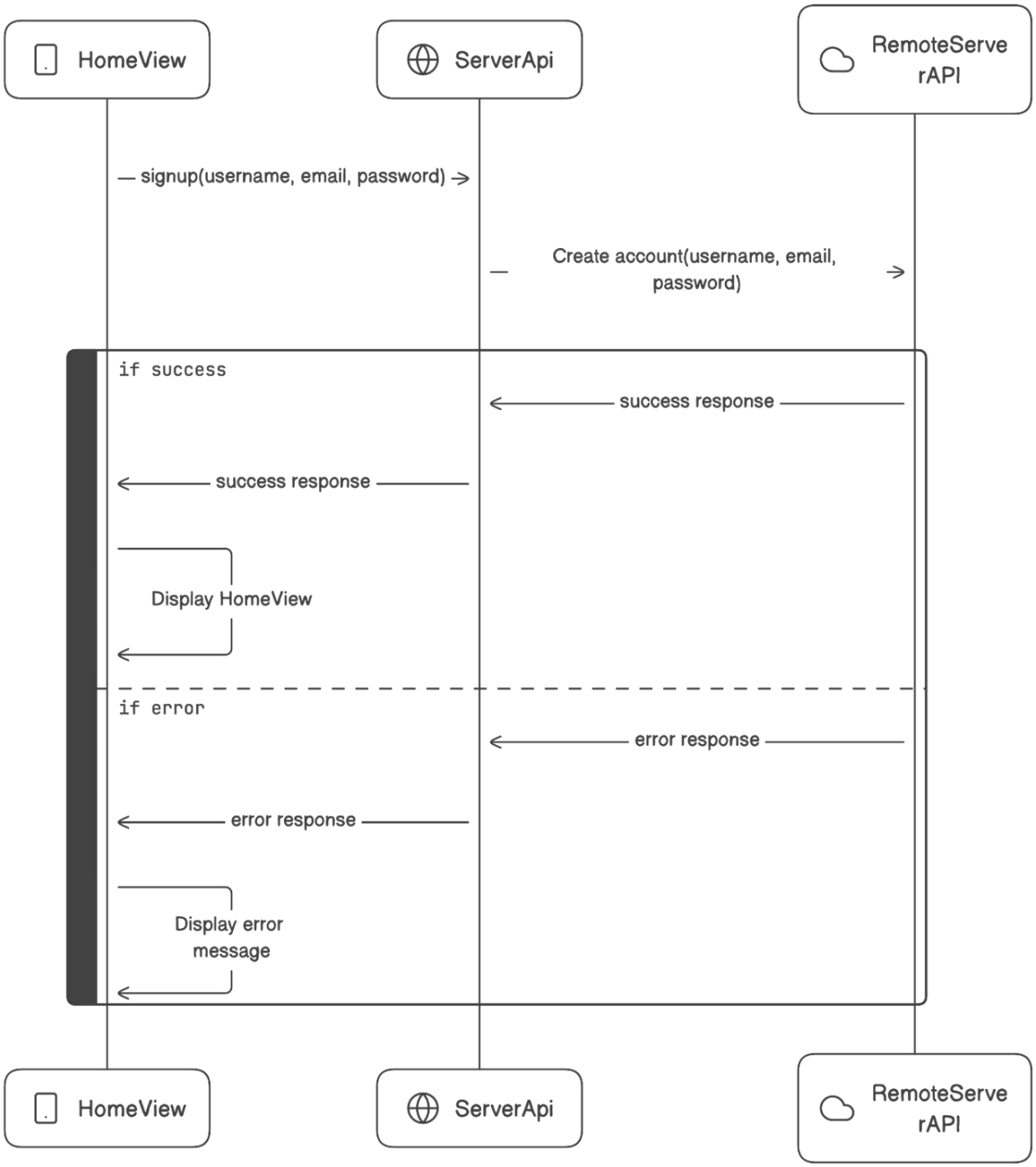


# 2.7 Sequence Diagrams

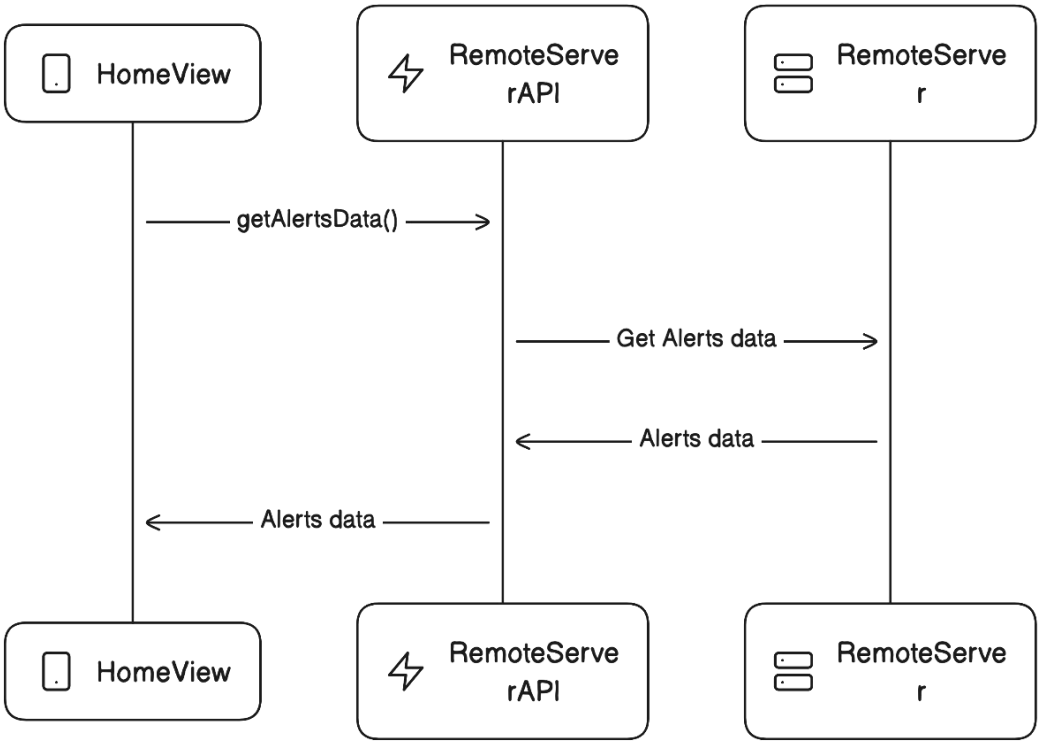
## 2.7.1 Login Sequence



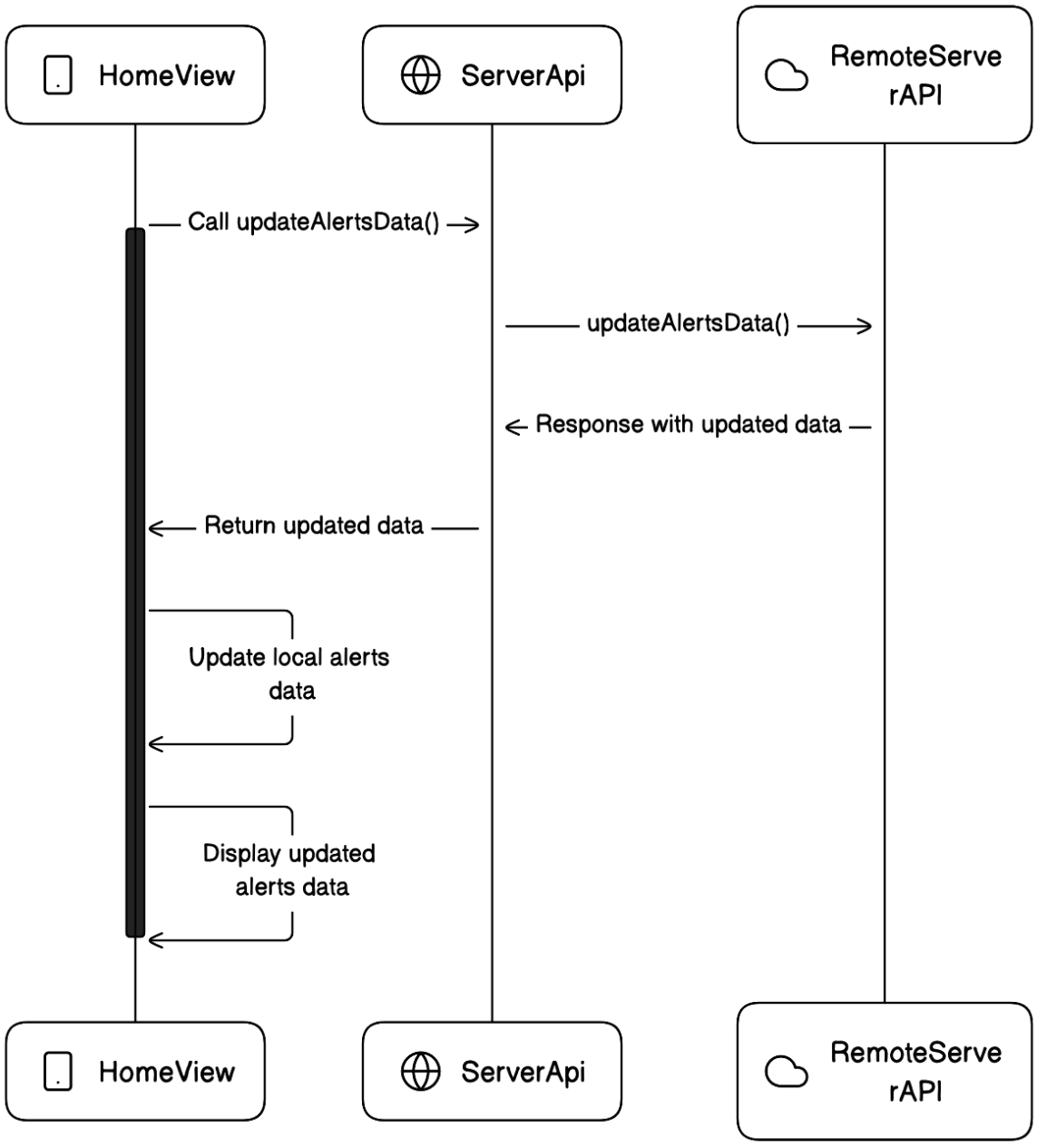
## 2.7.2 Sign up Sequence



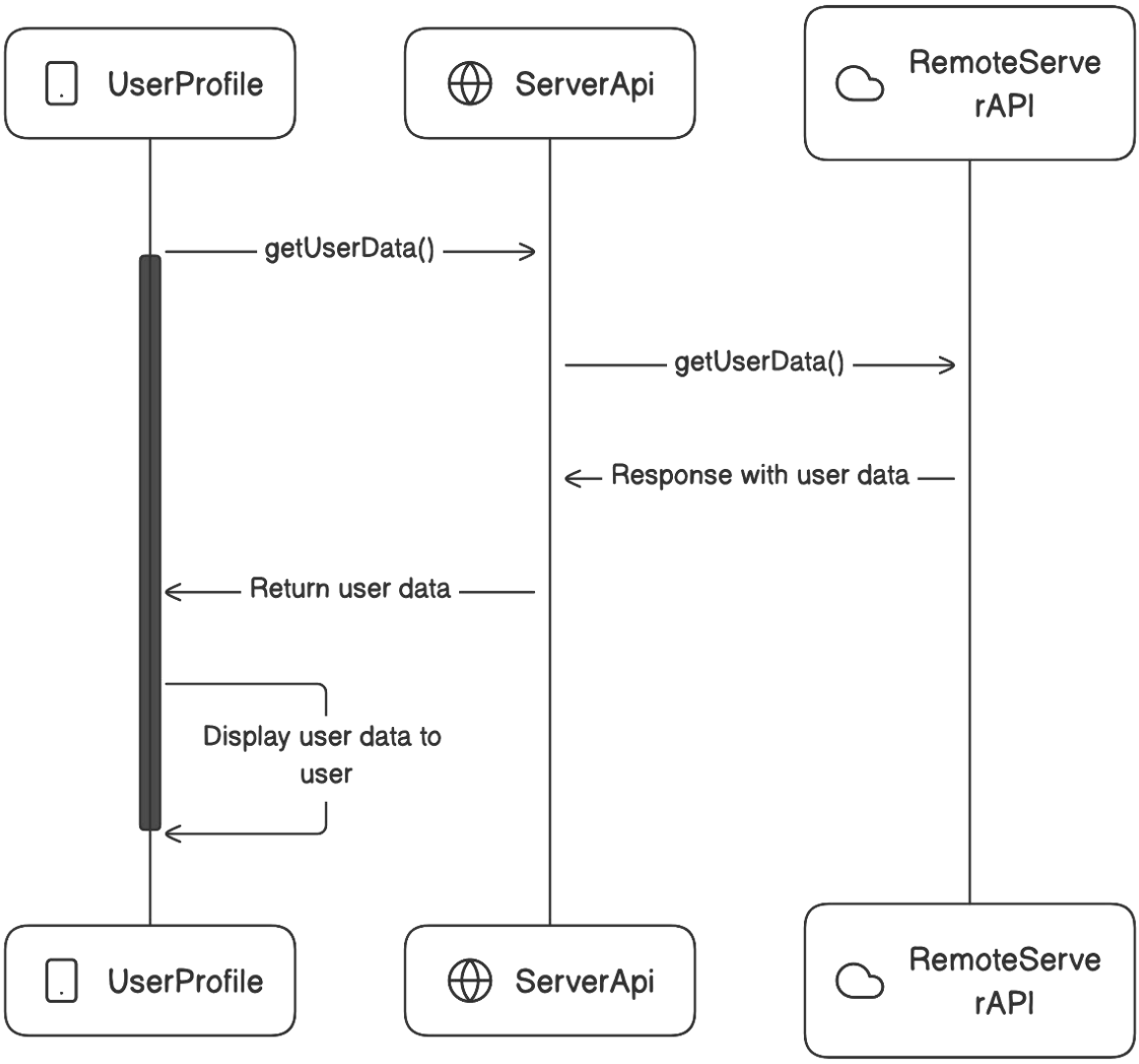
### 2.7.3 Get Alerts Sequence



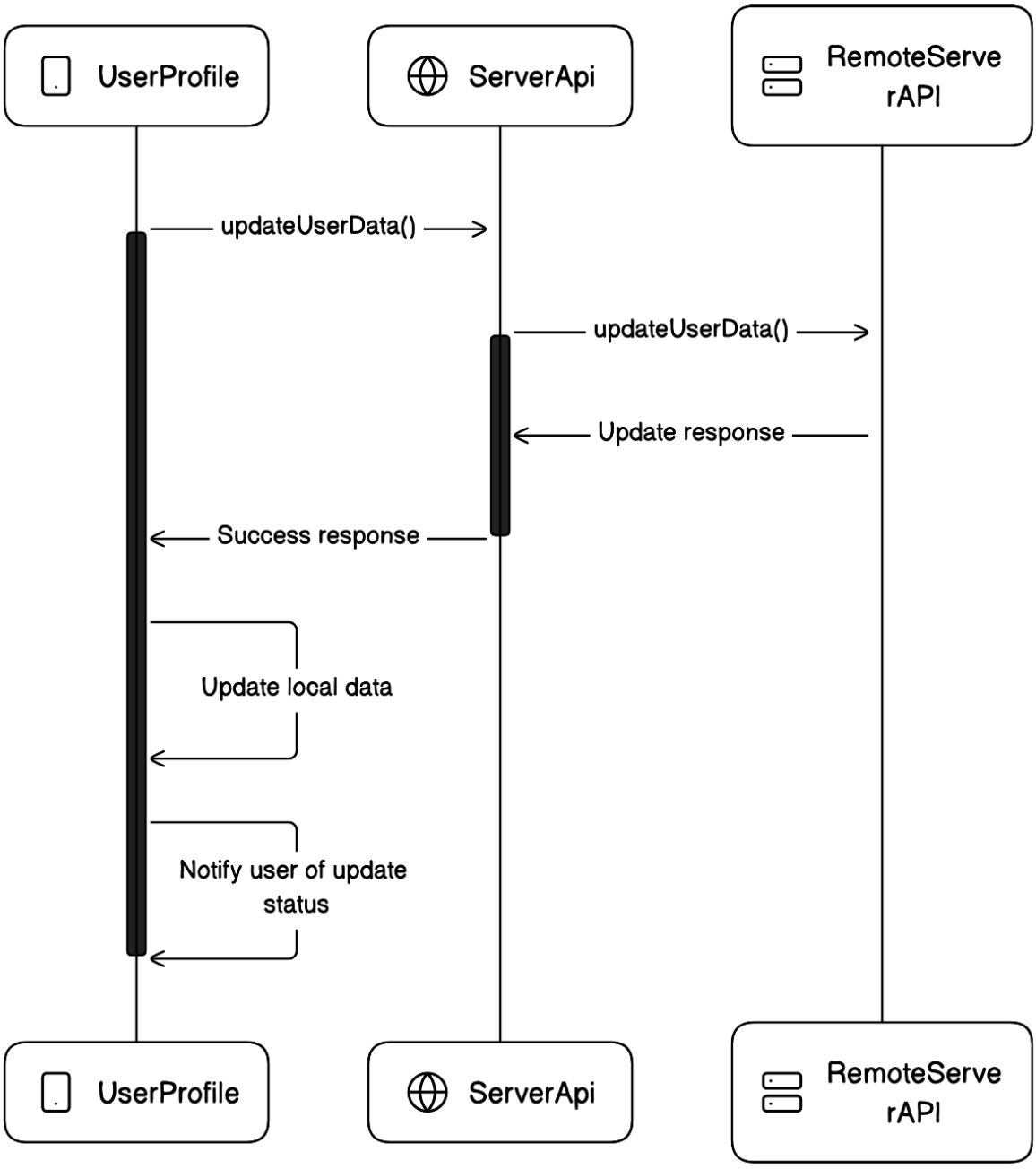
### 2.7.4 Update Alerts Sequence



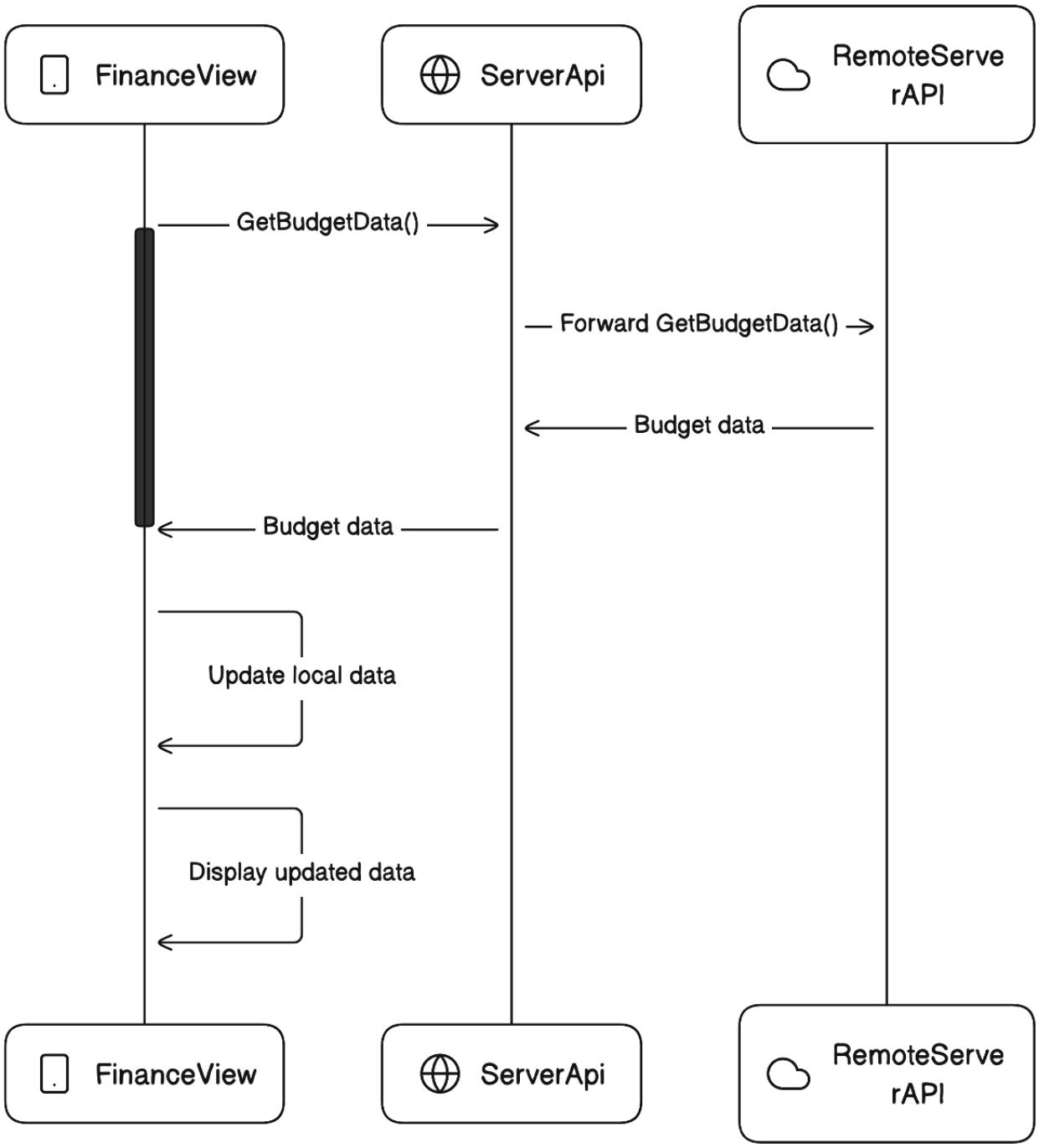
### 2.7.5 Get User Data Sequence



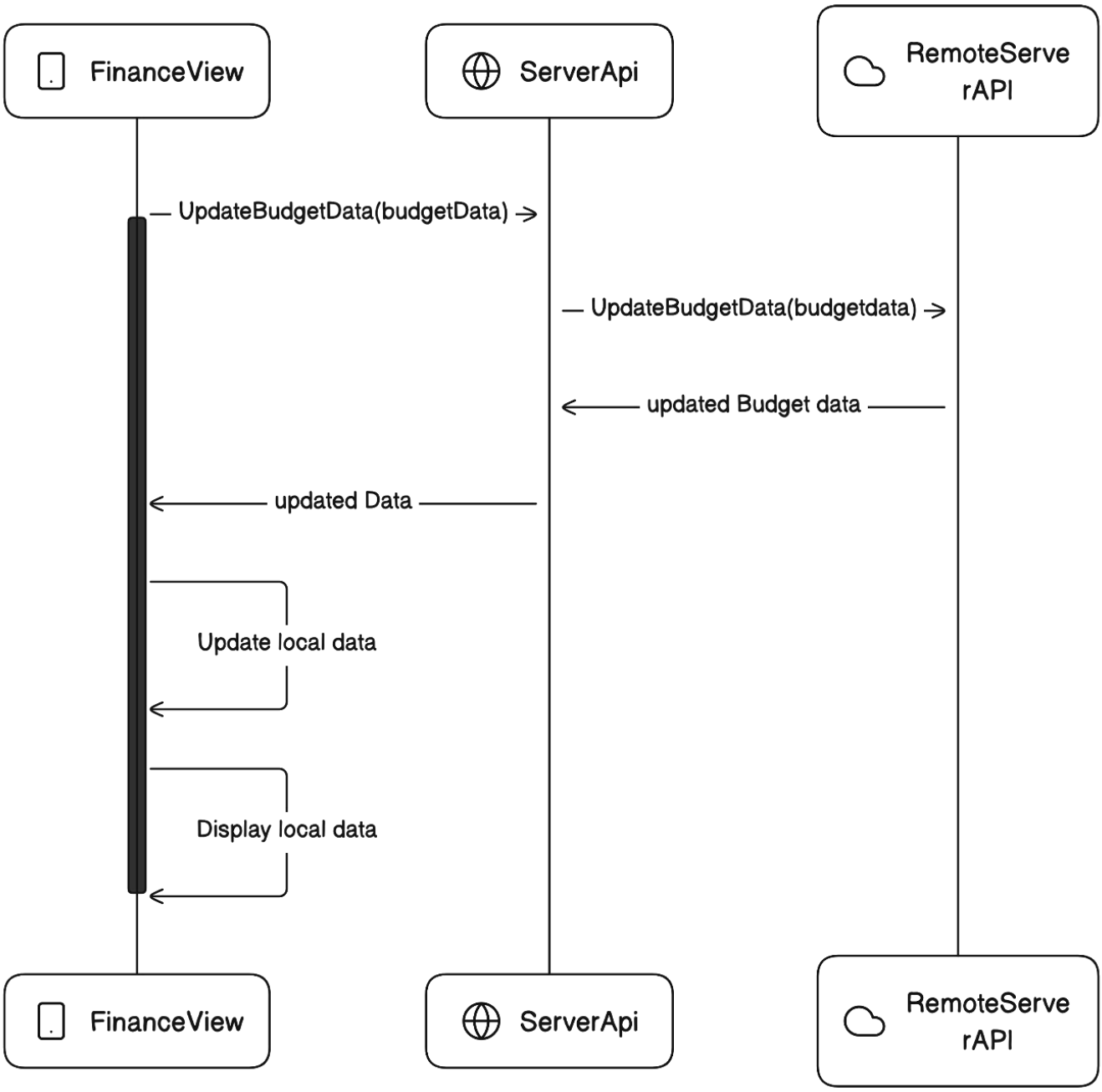
### 2.7.6 Update User Data Sequence



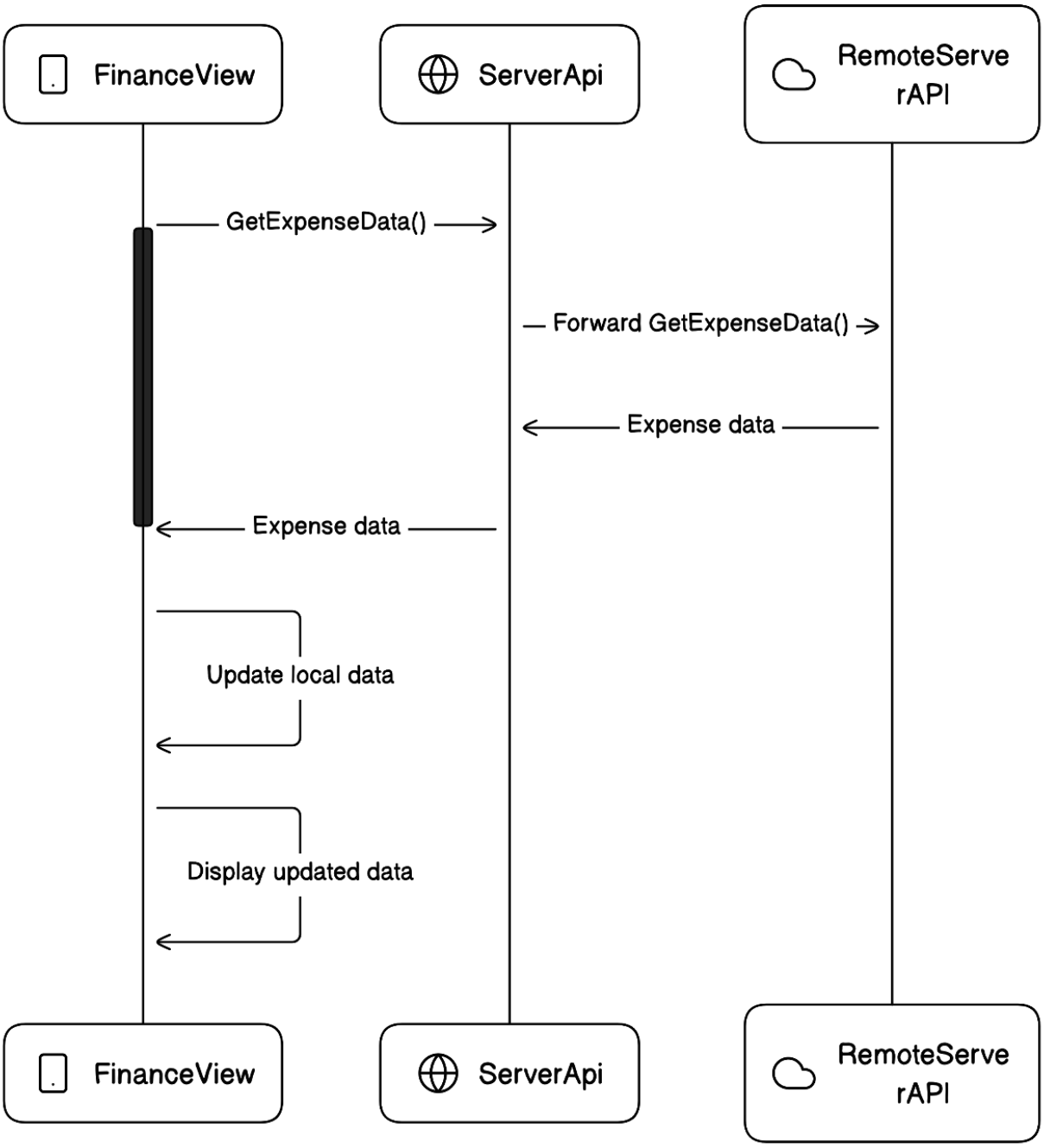
### 2.7.7 Get Budget Sequence



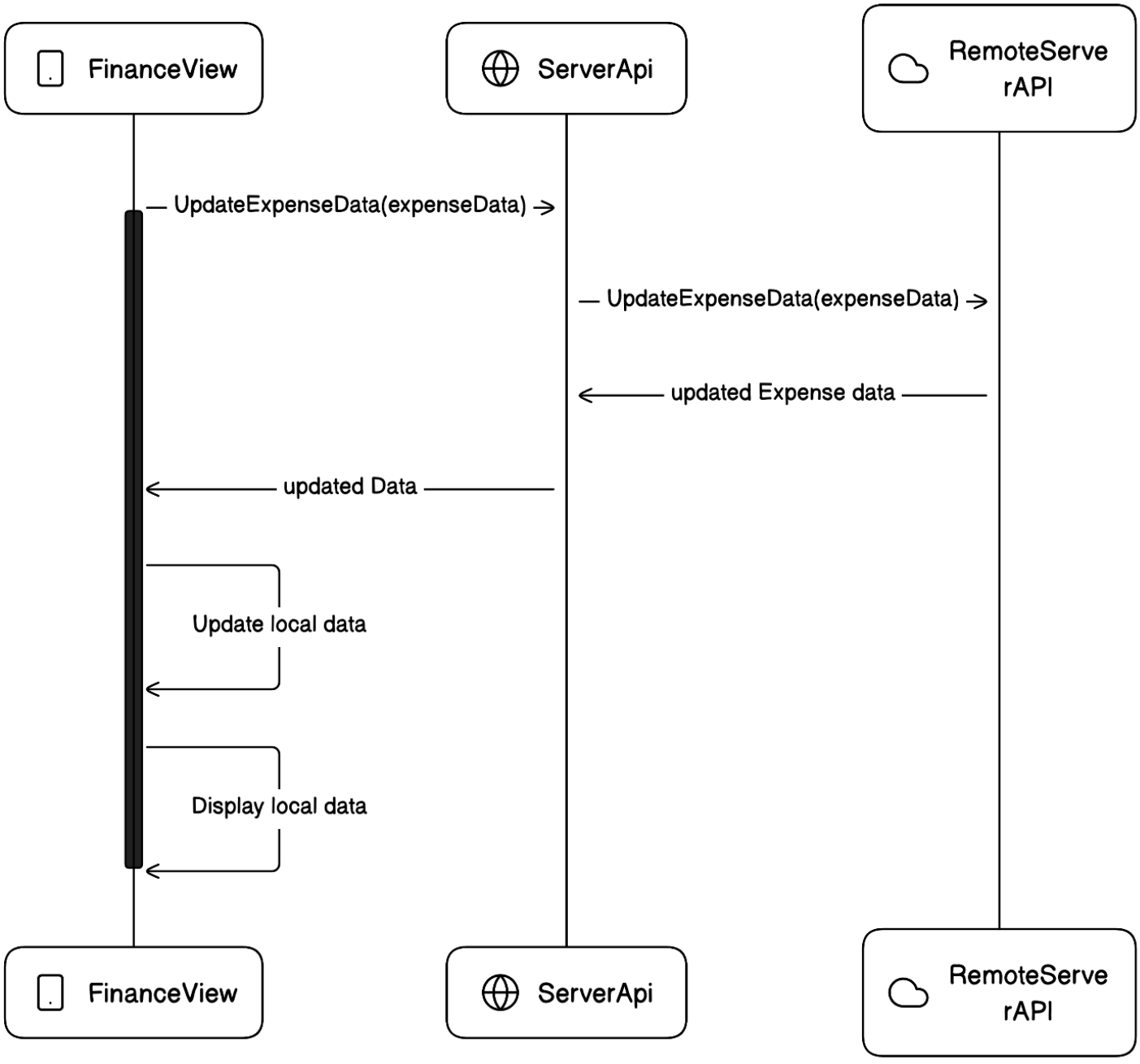
### 2.7.8 Update Budget Sequence



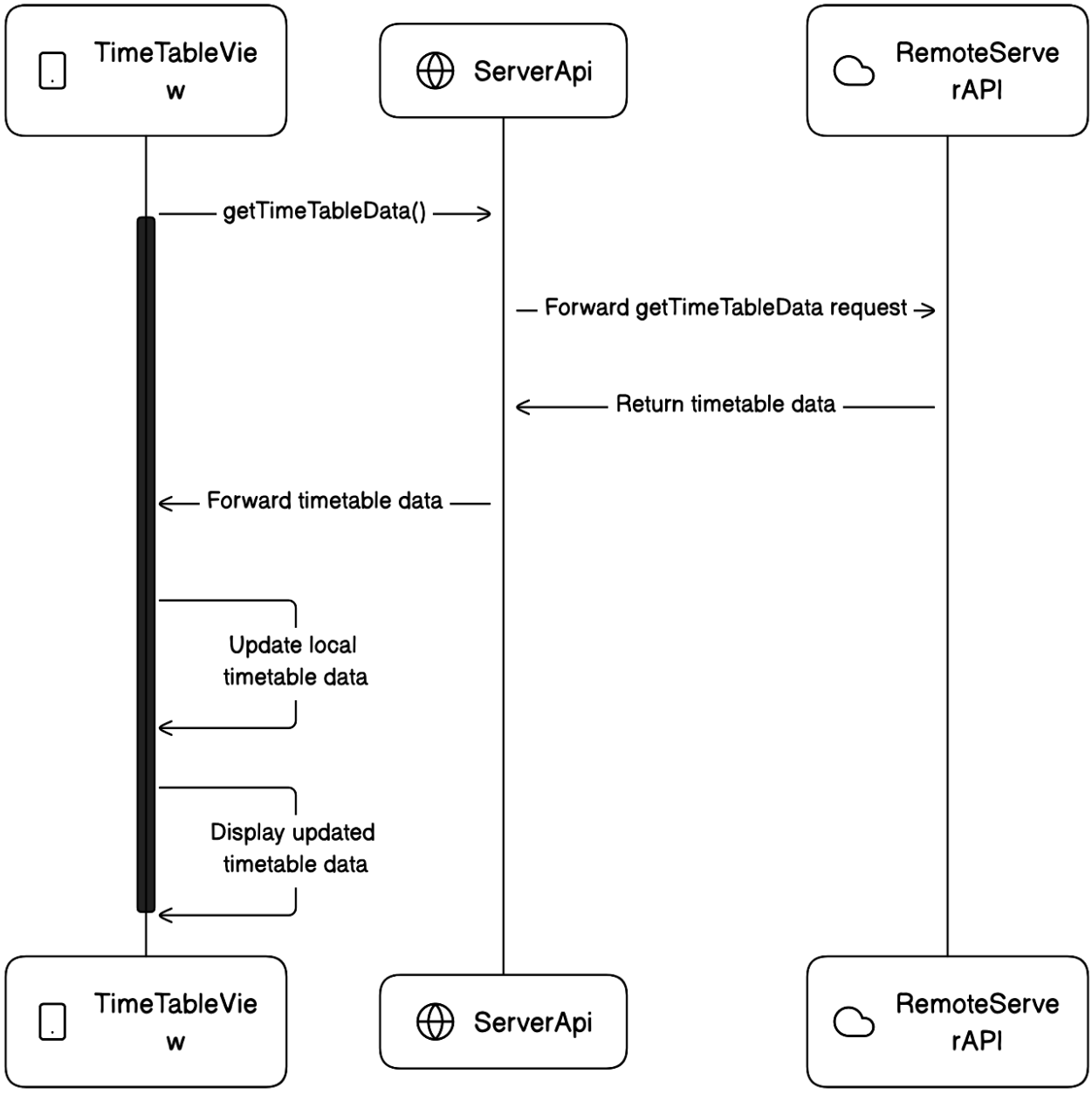
### 2.7.9 Get Expense Sequence



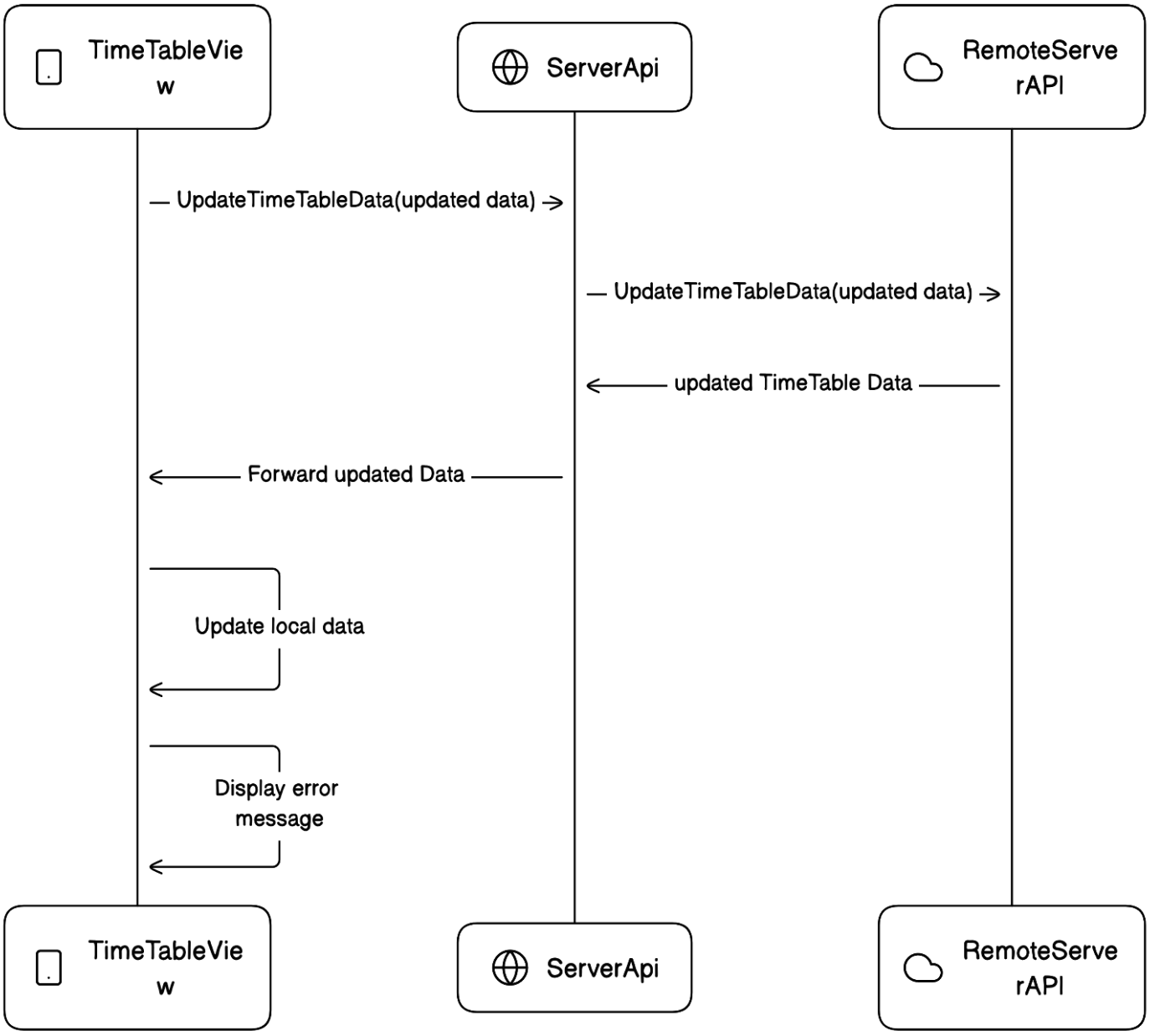
### 2.7.10 Update Expense Sequence



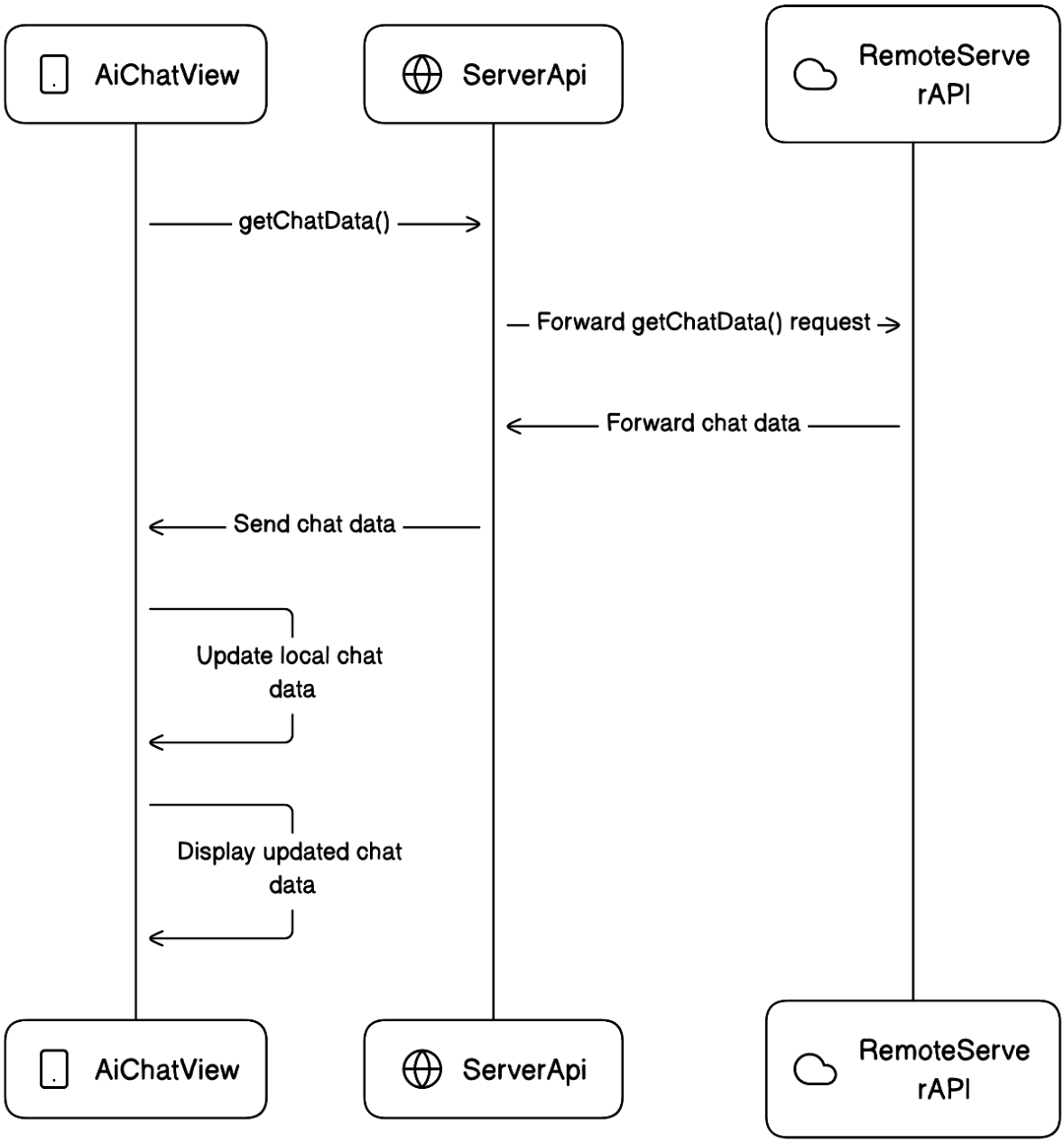
### 2.7.11 Get Timetable Sequence



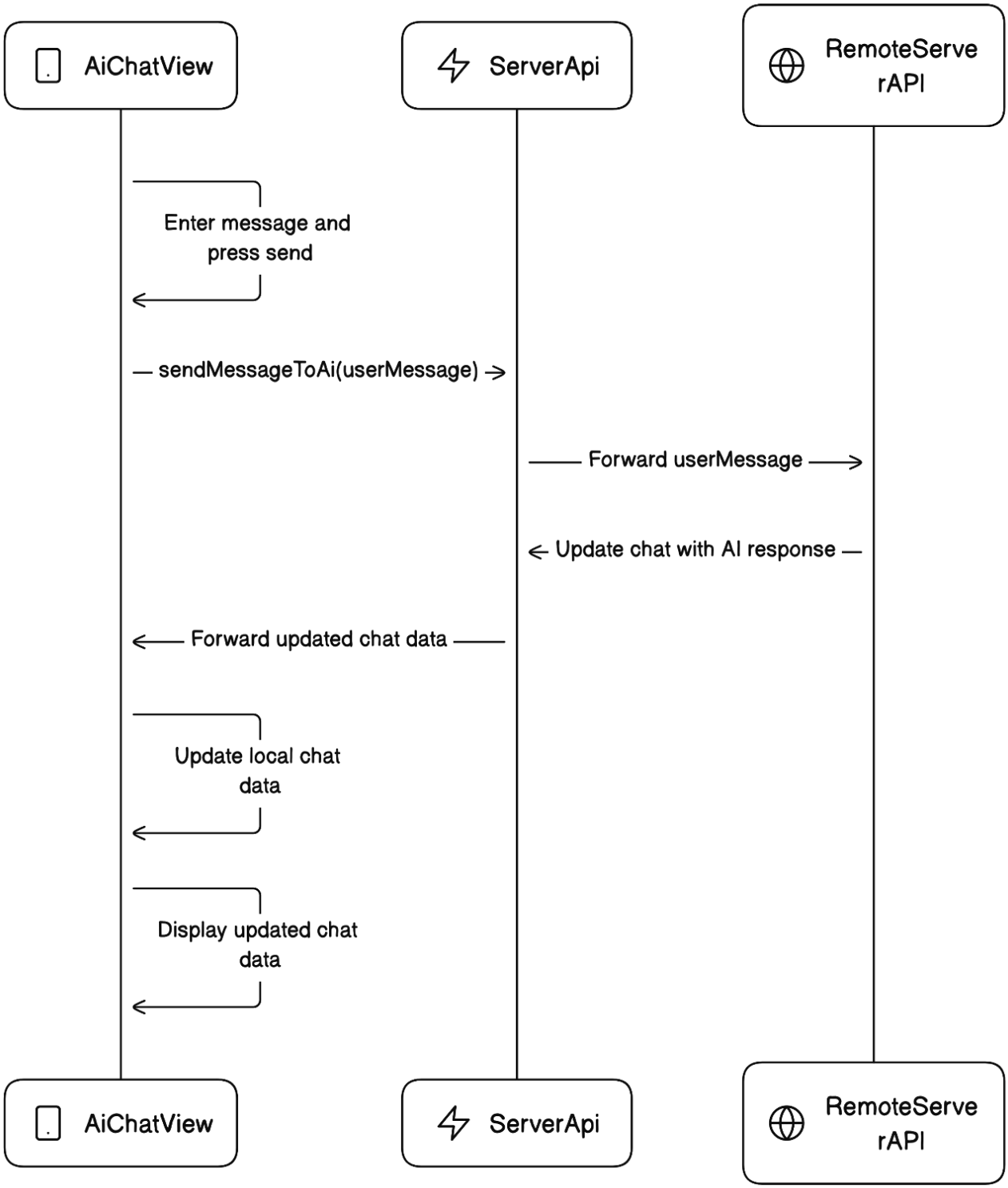
### 2.7.12 Update Timetable Sequence



### 2.7.13 Get Chat Data Sequence



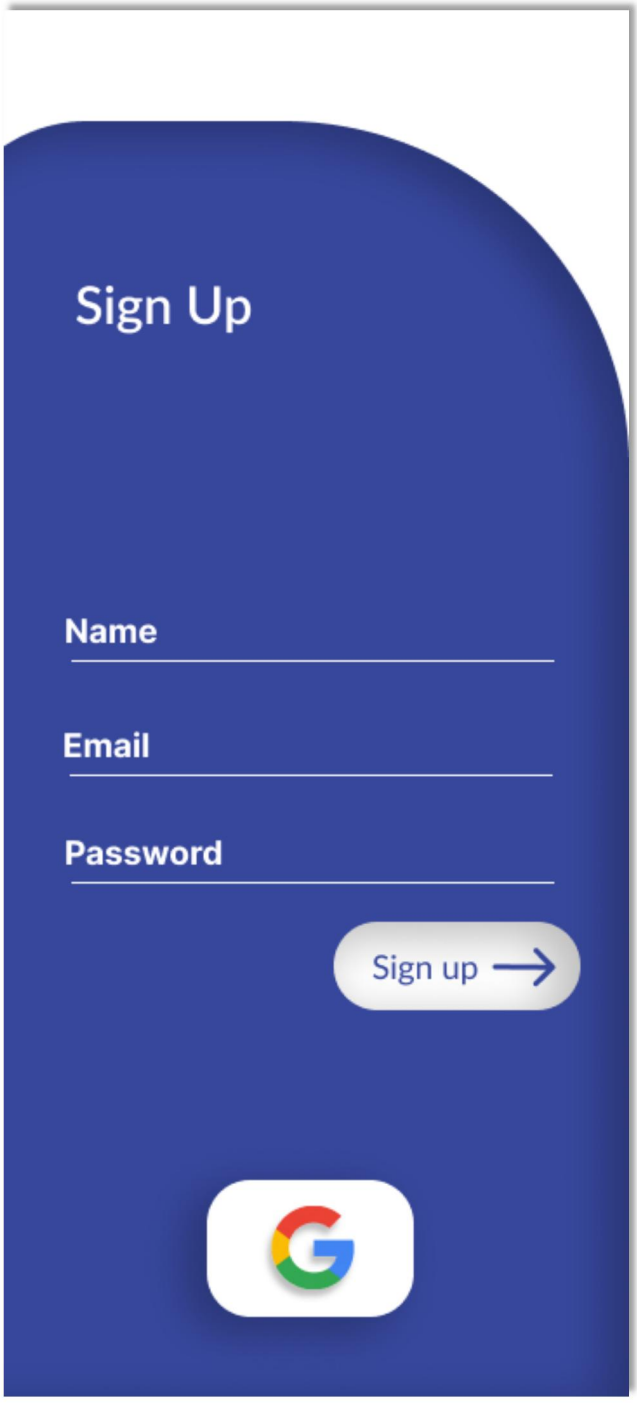
### 2.7.14 Send Message Sequence



# 3. Interface Designs

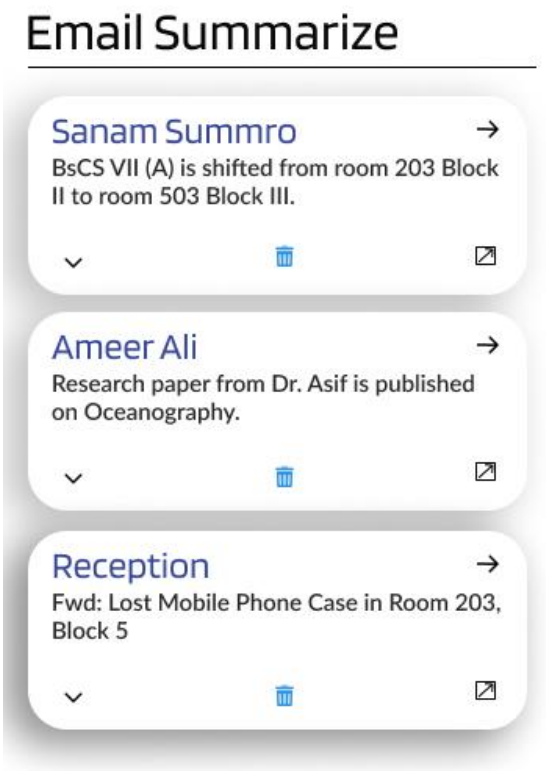
## 3.1 Login

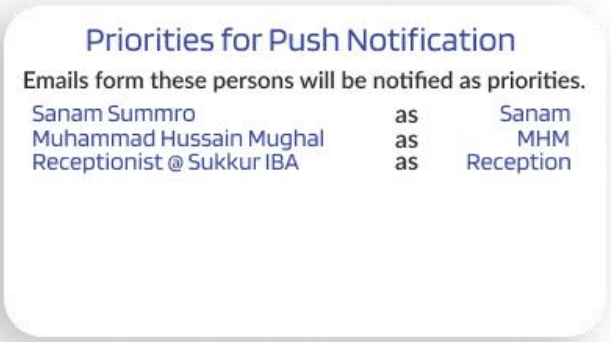
## 3.2 Sign Up



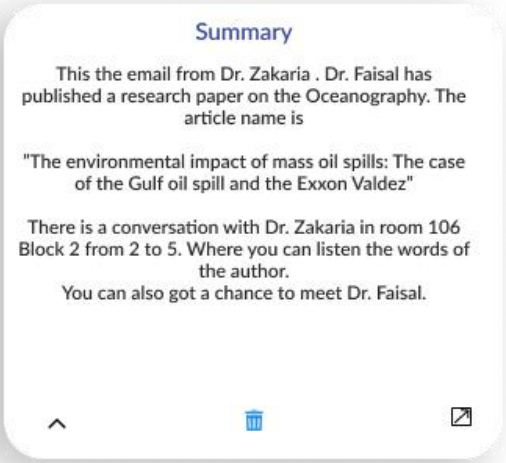
## 3.3 Landing Page

# 3.4 Email Summarizer



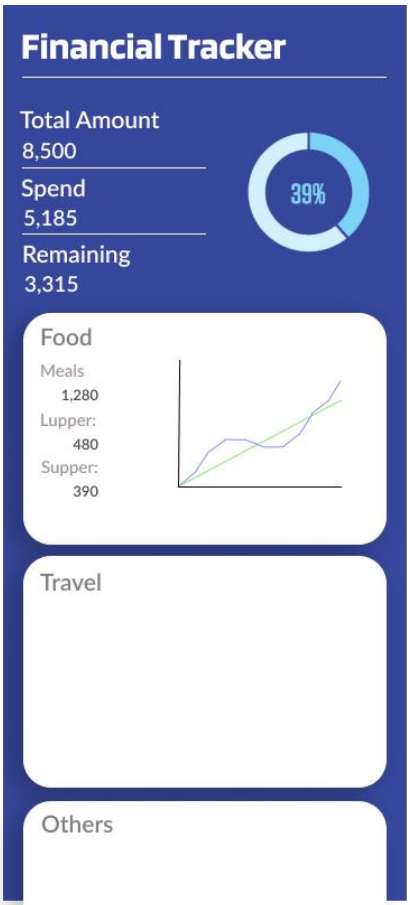




Goto Email to read full email

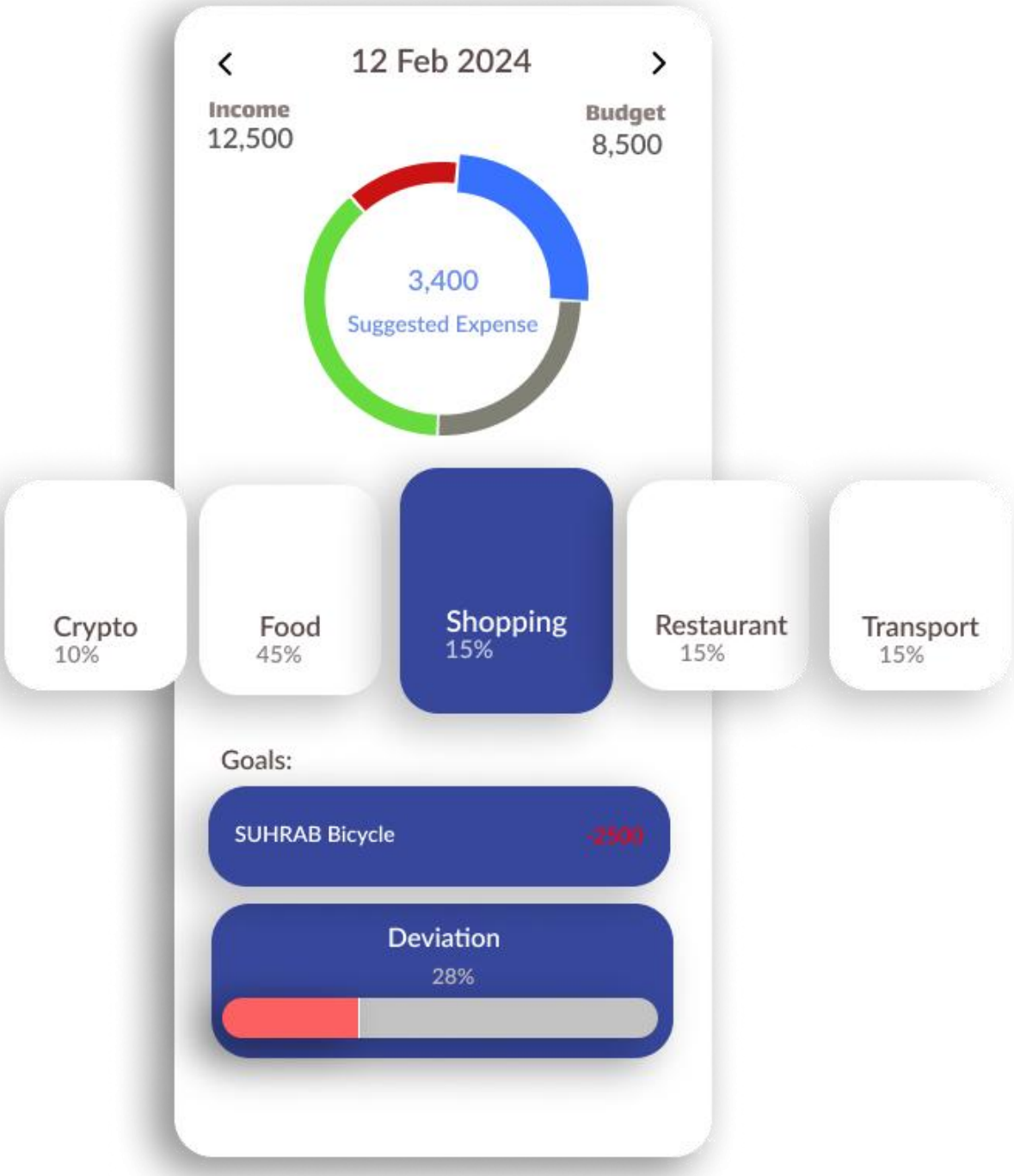


# 3.5 Financial Tracker

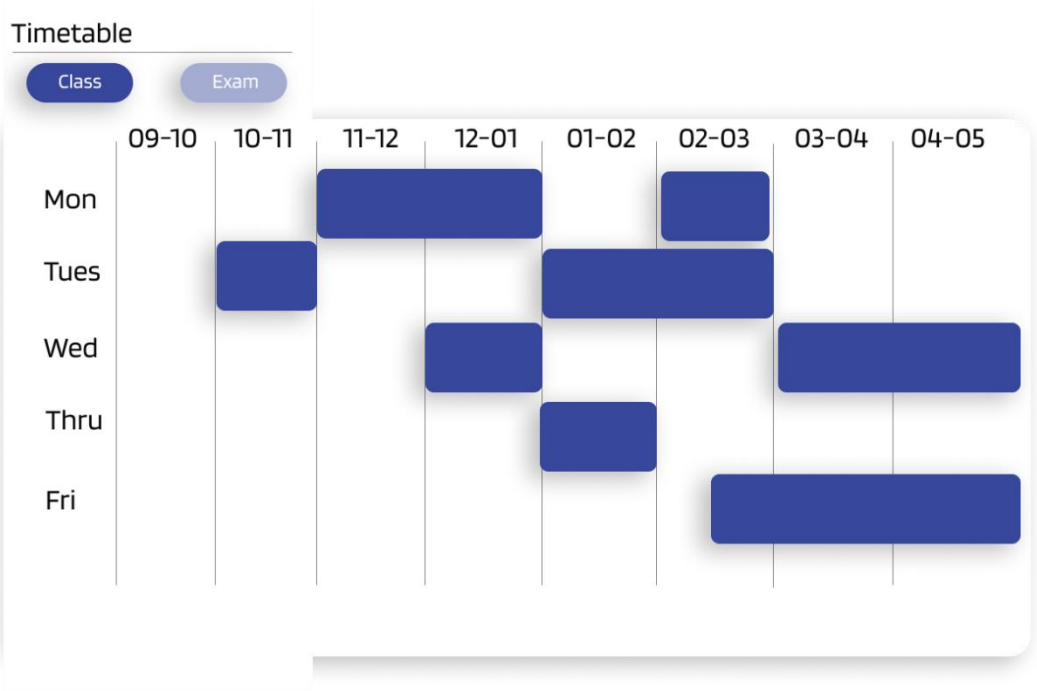
## 3.5.1 One View

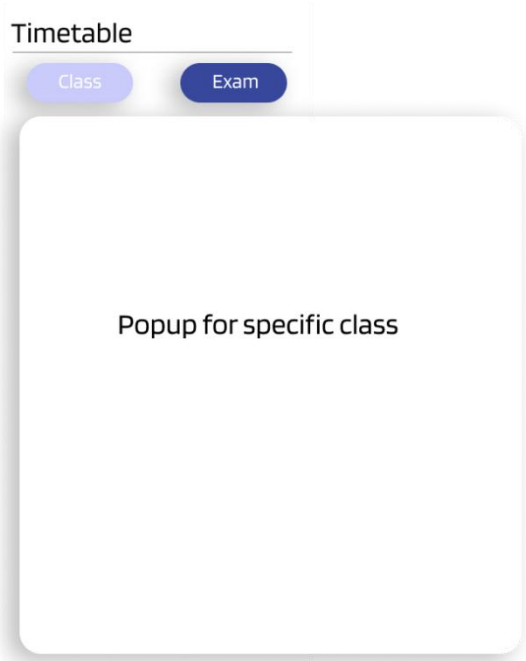



### 3.5.2 Second View

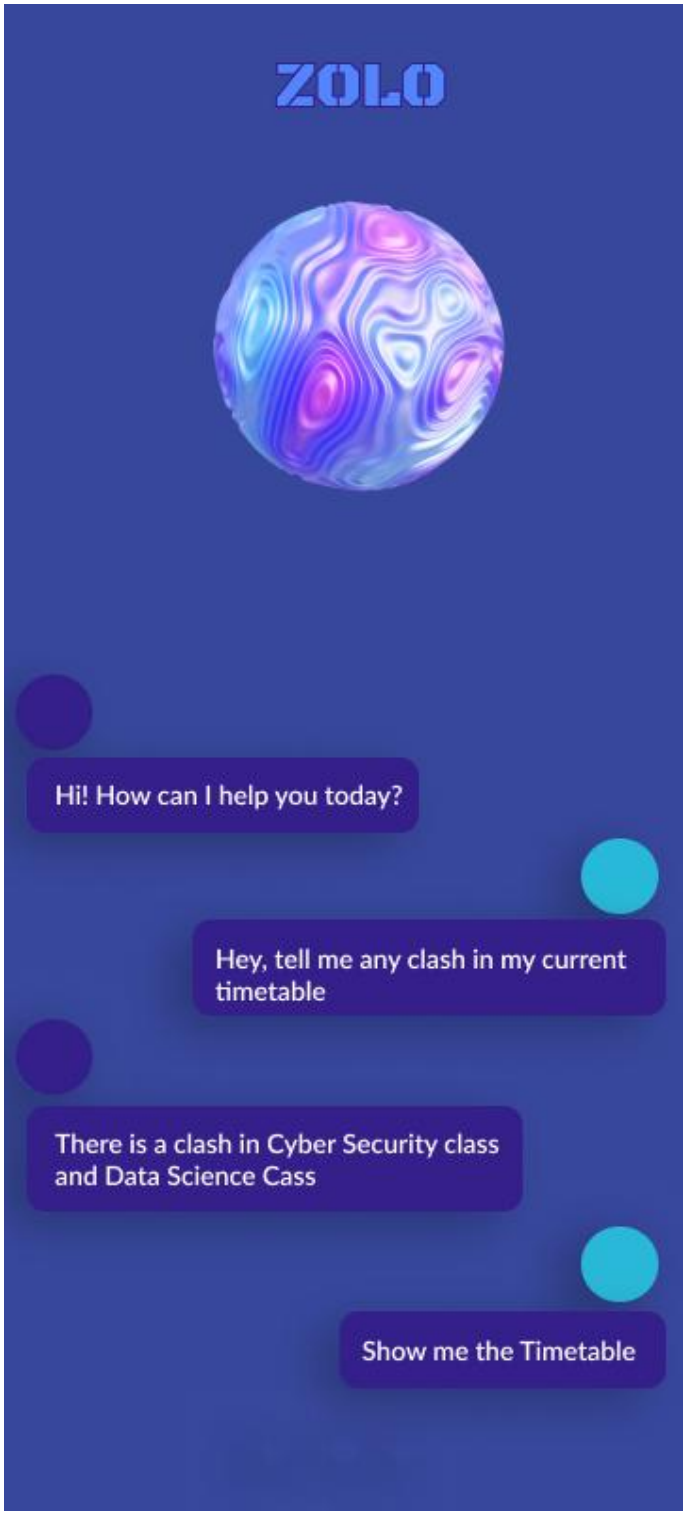


# 3.6 Timetable



## 3.7 Virtual AI Assitant : Zolo



# 4. Test Cases

## 4.1 User Registration

**ID:** UPM-TC-001

**Test Case Description:** Verify that users can successfully register with valid information.

**Steps to Execute:**

1. Navigate to the application's registration page.
2. Fill in the registration form with valid information, including:
3. First Name: John
4. Last Name: Doe
5. Email: johndoe@example.com
6. Password: P@ssw0rd
7. Confirm Password: P@ssw0rd
8. Click the "Sign Up" button.
9. Check for the registration confirmation message or redirection to the user profile page.

**Input Data:**

First Name: John
Last Name: Doe
Email: johndoe@example.com
Password: P@ssw0rd
Confirm Password: P@ssw0rd
**Expected Result:** The user should be successfully registered and directed to their user profile page with a confirmation message. The registered user's information, including their name and email, should be displayed correctly on their profile.

## 4.2 User Login

**ID:** UPM-TC-002

**Test Case Description:** Verify that registered users can log in successfully using their credentials.

**Steps to Execute:**

1. Navigate to the application's login page.
2. Enter the user's registered email and password:
3. Email: johndoe@example.com
4. Password: P@ssw0rd
5. Click the "Log In" or "Sign In" button.
6. Check for successful login by verifying that the user is redirected to their user dashboard or profile.

**Input Data:**

Email: johndoe@example.com
Password: P@ssw0rd

**Expected Result:** The user should be able to log in successfully and be redirected to their user dashboard or profile page. The application should recognize the user's credentials and grant access to the corresponding user account.

## 4.3 Profile Update

**ID:** UPM-TC-003

**Test Case Description:** Verify that registered users can update their profile information successfully.

**Steps to Execute:**

1. Log in to the user account using valid credentials.
2. Navigate to the "Profile" or "Account Settings" section.
3. Select the "Edit Profile" or similar option.
4. Modify one or more user profile fields (e.g., name, profile picture, password).
5. Click the "Save" or "Update" button.
6. Check for a success message or confirmation.

**Input Data:**

Original Name: John Doe
Updated Name: John Smith

Original Profile Picture: [Path to original image]
Updated Profile Picture: [Path to updated image]

Original Password: [Original Password]
Updated Password: [New Password]

**Expected Result:** The user's profile information should be updated with the modified data. A success message or confirmation should be displayed. The updated information should be reflected in the user's profile.

## 4.4 Email Integration

**ID:** EM-TC-001

**Test Case Description:** Verify that the application can successfully integrate with the user's email account to fetch and process emails.

**Steps to Execute:**

1. Log in to the application using valid credentials.
2. Navigate to the "Email" or "Inbox" section.
3. Click on the "Integrate Email Account" button.
4. Provide email account details, including email address and password.
5. Click the "Authenticate" or "Connect" button.
6. Wait for the system to authenticate and integrate with the email account.

**Input Data:**

Email Address: user@example.com

Password: [User's email account password]

**Expected Result:** The application should integrate with the user's email account successfully. It should be able to fetch email messages from the connected account and display them in the user's inbox. The user should be able to see their emails within the application.

## 4.5 Email Summarization

**ID:** EM-TC-002

**Test Case Description:** Verify that the application can accurately summarize an email message.

**Steps to Execute:**

1. Log in to the application using valid credentials.
2. Navigate to the "Email" section.
3. The system should automatically generate a summary of emails.

**Input Data:** N/A

**Expected Result:** The application should generate a concise and accurate summary of email messages. The summary should capture the key points and content of the email message, making it easier for the user to understand the message's contents.

## 4.6 Email Updates

**ID:** EM-TC-003

**Test Case Description:** Verify that the application can detect and process email updates, request user confirmation, and update the timetable when given permission.

**Steps to Execute:**

1. Log in to the application using valid credentials.
2. Navigate to the "Email" or "Inbox" section.
3. Click on an email message that contains an update related to the user's timetable.
4. The system should automatically detect the update and generate a summary.
5. Review the summary and any proposed changes to the timetable.

6. The summary should include a request for user confirmation regarding the proposed timetable changes.
7. Provide user confirmation by clicking "Accept."
8. The application should update the timetable as per the confirmed changes.
9. Check if the updated timetable reflects the changes accurately.

**Input Data:** An email containing a timetable update.

**Expected Result:** The application should correctly detect the email update, generate a summary, request user confirmation for timetable changes, and apply the changes when the user accepts. The updated timetable should accurately reflect the changes made in the email.

## 4.7 Expense Entry

**ID:** FT-TC-001

**Test Case Description:** Verify that the application can successfully record and save user-entered expenses.

**Steps to Execute:**

1. Log in to the application using valid credentials.
2. Navigate to the "Financial Tracker" section.
3. Click on the "Expense Entry" or "Add Expense" option.
4. Fill in the required fields: Expense category, date, amount, payment method, and any optional notes.
5. Click the "Save" or "Submit" button.
6. Verify if the entered expense is saved successfully.

**Input Data:**

Expense category: "Food"
Date: [Current date]
Amount: Rs 150
Notes: "Cafeteria Lunch"

**Expected Result:** The application should successfully record and save the entered expense. The saved expense should be visible in the expense history or listings, with all the details accurately reflected.

## 4.8 Budget Management

**ID:** FT-TC-002

**Test Case Description:** Verify that the application can successfully set and manage budgets for different expense categories.

**Steps to Execute:**

1. Log in to the application using valid credentials.
2. Navigate to the "Financial Tracker" section.
3. Click on the "Budget Management" or "Set Budget" option.
4. Choose an budget time category to set a budget for.
5. Enter a budget amount for the selected category.
6. Click the "Save" or "Submit" button.
7. Verify if the budget is set successfully.
8. Perform similar steps to update or change an existing budget.

**Input Data:**

Budget time category: "Weekly"
Budget amount: Rs 2000

**Expected Result:** The application should successfully set and save the budget for the selected category. If an existing budget is updated, the application should reflect the changes accurately.

## 4.9 Expense Reports

**ID:** FT-TC-003

**Test Case Description:** Verify that the application generates accurate expense reports for a specified time period.

**Steps to Execute:**

1. Log in to the application using valid credentials.
2. Navigate to the "Financial Tracker" section.
3. Click on the "Expense Reports".
4. Select the desired time period or date range for the report (e.g., "Last Month" or "Custom Date Range").
5. Choose the specific expense categories to include in the report.
6. Click the "View Report" button.
7. Review the generated report for accuracy.
8. Verify that the total expenses match the sum of expenses within the selected time frame.

**Input Data:**

Time period: "Last Month"
Expense categories: "Groceries," "Utilities," "Transportation", “Lunch”, “Dinner”, “Breakfast”, “Brunch”

**Expected Result:** The application should generate a detailed report that accurately summarizes the expenses within the specified time period and the selected categories. The total expenses in the report should match the sum of expenses for the chosen categories during the given time frame.

## 4.10 View Timetable

**ID:** TT-TC-001

**Test Case Description:** Verify that users can view their class timetable.

**Steps to Execute:**

1. Log in to the application with valid credentials.
2. Navigate to the "Timetable" or "Class Schedule" section.
3. Select the option to "View Timetable."
4. Choose the date or time period for which you want to view the timetable (e.g., "Today," "This Week").
5. Click the "View" or "Display" button.
6. Check the displayed timetable for correctness.

**Input Data:**

Date/Time Period: "Weekly"

**Expected Result:** The application should display the user's class timetable for the selected date or time period. All classes should be listed in chronological order, including details like course name, instructor, venue, and time. The timetable displayed should be accurate and correspond to the user's enrolled classes.

## 4.11 Add/Modify Class/Exam

**ID:** TT-TC-002

**Test Case Description:** Verify that users can add or modify classes/exams to their timetable.

**Steps to Execute:**

1. Log in to the application with valid credentials.
2. Navigate to the "Timetable" or "Class Schedule" section.
3. Select the option to "Add/Modify Class/Exam."
4. Choose the action you want to perform: "Add" or "Modify."
5. Fill in the necessary details, such as course name, instructor, venue, date, and time.
6. Click the "Save" or "Update" button.
7. Check the timetable to ensure that the class/exam has been added/modified correctly.

**Input Data:**

Action: "Add"
Course Name: "ICT Theory"
Instructor: "Dr. XYZ"
Venue: "Room 107, Block II"
Date: [Select a valid date]
Time: [Select a valid time]

**Expected Result:** The application should add or modify the class/exam as per the user's input. The updated timetable should reflect the changes made, and the class/exam details should be displayed accurately.

## 4.12 Information Retrieval

**ID:** AI-TC-001

**Test Case Description:** Verify that the Virtual AI Assistant can correctly retrieve information in response to user queries.

**Steps to Execute:**

1. Open the Virtual AI Assistant Module.
2. Enter a user query in natural language, such as "When is my next ICT Theory Class?"
3. Initiate the query by pressing the button.
4. Observe the response provided by the AI Assistant.

**Input Data:** User Query: "When is my next ICT Theory Class?"

**Expected Result:** The Virtual AI Assistant should respond with accurate information related to the user query, including the next ICT Class.

## 4.13 Chat Interaction

**ID:** AI-TC-002

**Test Case Description:** Verify that the Virtual AI Assistant can engage in a conversation and provide relevant responses to user input.

**Steps to Execute:**

1. Open the Virtual AI Assistant module.
2. Initiate a conversation by typing a greeting message, such as "Hello."
3. Observe the response provided by the AI Assistant.
4. Continue the conversation by asking a question, such as "What date was Quaid-e-Azam born?"
5. Observe the response to the question.
6. Ask another question or provide a statement to continue the conversation.
7. Observe the AI Assistant's responses to maintain a coherent and informative conversation.

**Input Data:** User Input: "Hello," "What date was Quaid-e-Azam born?"

**Expected Result:** The Virtual AI Assistant should provide coherent and contextually relevant responses during the conversation, demonstrating its ability to interact naturally with the user.

## 4.14 Performance Testing

**ID:** APP-PERF-001

**Test Case Description:** Evaluate the application's performance under heavy load to ensure it meets specified performance requirements.

**Steps to Execute:**

1. Prepare a test environment to simulate heavy user load.
2. Configure the testing tool to generate a high number of concurrent user requests.
3. Initiate the performance testing tool to simulate peak user load.
4. Monitor and record system performance metrics, including response time, resource utilization, and error rates.
5. Gradually increase the user load until it reaches the defined maximum capacity.
6. Analyze the test results for any performance bottlenecks, degradation, or failures.
7. Determine if the system meets or exceeds the predefined performance requirements.

**Input Data:** Performance testing tool, test environment, simulated user load.

**Expected Result:** The application should withstand the peak user load without critical performance degradation, and response times should remain within acceptable limits, as defined in the performance requirements.

## 4.15 Usability Testing

**ID:** APP-USABILITY-001

**Test Case Description:** Evaluate the application's usability by assessing how easily users can interact with and navigate the system.

**Steps to Execute:**

1. Select a group of representative users, including both experienced and inexperienced users.
2. Provide the users with a set of tasks to perform within the application.
3. Observe and record users' interactions and their ability to complete tasks.
4. Collect feedback from users about their experiences, including any difficulties encountered and suggestions for improvement.
5. Analyze the results and feedback to identify usability issues.
6. Prioritize and categorize usability issues into critical, major, and minor problems.
7. Provide recommendations for improving usability based on the test results.

**Input Data:** Test users, usability test scenarios, usability testing guidelines.

**Expected Result:** Usability issues are identified and categorized, and recommendations for improving the application's usability are provided, helping to enhance the overall user experience.

## 4.16 Error Handling

**ID:** APP-ERR-001

**Test Case Description:** Verify that the application handles errors gracefully and provides appropriate feedback to users.

**Steps to Execute:**

1. Identify the types of errors to be tested (e.g., input validation errors, server errors, database errors).
2. Simulate error conditions based on identified error types.
3. Perform actions that trigger the identified error conditions (e.g., submitting invalid data, disconnecting from the server).
4. Observe the application's response and error messages.
5. Verify that error messages are clear, informative, and user-friendly.
6. Ensure that the application offers options for users to recover from errors when possible.
7. Confirm that error conditions do not crash the application and that it continues to function correctly.
8. For any unhandled exceptions or unexpected errors, document the issue with details.

**Input Data:**

- ❖ List of identified error types
- ❖ Test scenarios that trigger each error type

**Expected Result:** The application correctly identifies and handles errors by providing clear and informative error messages to the users. It continues to function correctly even when errors occur. Any unhandled exceptions or unexpected errors are documented for resolution.