
Final Year Project Report

for

The Student Assistant: AI-Enabled Academic and Financial Management

Submitted to The Department of Computer Science in
Partial Fulfillment of The Requirements For The Degree
of Bachelor of Science In Computer Science / Software
Engineering



SUKKUR IBA UNIVERSITY

Prepared by 20F-33

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

by

Agha Kaleemullah Khan

Asghar Ali Shah

M. Aizazullah Khan

SUBMITTED TO THE DEPARTMENT OF
COMPUTER SCIENCE IN PARTIAL
FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF

**BACHELOR OF SCIENCE IN
COMPUTER SCIENCE / SOFTWARE
ENGINEERING**

at

the

SUKKUR IBA University

April, 2024

© 2024 Agha Kaleemullah Khan, Asghar Ali

Shah, M. Aizazullah Khan. All rights

reserved.

The author hereby grants permission to Sukkur IBA to reproduce and distribute publicly paper and electronic copies of this thesis and grant others the right to do so.

Signature of Authors

Certified by:

Accepted by:

DECLARATION

We hereby declare that this project report entitled “THE STUDENT ASSISTANT: AI-ENABLED ACADEMIC AND FINANCIAL MANAGEMENT” submitted to the “DEPARTMENT OF COMPUTER SCIENCE”, is a record of an original work done by us under the guidance of Supervisor “DR. ASIF KHAN” and that no part has been plagiarized without citations. Also, this project work is submitted in the partial fulfillment of the requirements for the degree of Bachelor of Computer Science.

Team Members

Signature

Asghar Ali Shah

Agha Kaleemullah Khan

M. Aizazullah Khan

Supervisor:

Signature

Dr. Asif Khan

Date:

Place:

Table of Contents

Cover Page.....	i
Declaration.....	ii
Table of Contents	iii
List of Figures	v
1. Introduction	1
1.1 Abstract.....	2
1.2 Project Overview	2
1.3 Objectives.....	2
1.4 Scope	3
1.5 Approach.....	3
1.6 Styles.....	3
1.6.1 Typeface.....	3
1.6.2 Margins.....	3
1.6.3 Headings.....	4
1.6 Figures	4
1.7 References	4
2. Literature Review	5
3. Problem Definition	10
4. Solution Statement	12
5. Detailed Design & Architecture	14
5.1 System Architecture	14
5.1.1 Architecture Design Approach.....	14
5.1.2 Architecture Design	15
5.1.3 Subsystem Architecture.....	21
5.2 Detailed System Design	26
5.2.1 Chatbot Module.....	26
5.2.2 Financial Management Module.....	30
5.2.3 Timetable Management Module.....	36
5.2.4 Email Management Module.....	40
5.3 Class Diagram	43
5.4 Entity Relationship Diagram	44
6. Implementation and Testing	45
6.1 Application Scenario	45
6.2 Development Process	47
6.3 Core Functionalities	48
6.4 Controlled Libraries and Templates	50
6.5 Code Walkthroughs	50
6.6 Testing Methodologies	51
6.7 Evaluation and Comparison	51
7. Results and Discussion	52

7.1 Evaluation of the Solution	52
7.1.1 Quantitative Metrics.....	52
7.1.2 Qualitative Assessments.....	52
7.2 Results	53
7.3 Discussion of Results	60
7.3.1 Performance Evaluation.....	60
7.3.2 Usability Assessment.....	60
7.3.3 Functionality Validation.....	60
7.4 System Testing	61
7.4.1 Unit Testing.....	61
7.4.2 Acceptance Testing.....	61
8. Conclusion and Future Work.....	62
Appendix A: References.....	65

List of Figures

1.	5.1	Context Diagram of the System.....	15
2.	5.2	System Architectural Diagram.....	17
3.	5.3	The Student Assistant Functional Flow Diagram.....	19
4.	5.4	Phase 1 Figma Workspace Snapshot.....	20
5.	5.5	Phase 2 and 3 Figma Workspace Snapshot.....	20
6.	5.6	Login/Signup Data Flow Diagram.....	21
7.	5.7	Email Management Data Flow Diagram.....	22
8.	5.8	Financial Management Data Flow Diagram.....	23
9.	5.9	Timetable Management Data Flow Diagram.....	24
10.	5.10	Virtual Assistant Data Flow Diagram.....	25
11.	5.11	getChatData Sequence Diagram.....	28
12.	5.12	sendMessageToAi Sequence Diagram.....	29
13.	5.13	GetBudgetData Sequence Diagram.....	33
14.	5.14	UpdateBudgetData Sequence Diagram	34
15.	5.15	UpdateExpenseData Sequence Diagram.....	35
16.	5.16	GetTimetableData Sequence Diagram.....	39
17.	5.17	GetAlertsData Sequence Diagram.....	42
18.	5.18	Student Assistant Class Diagram.....	43
19.	5.19	Student Assistant Entity Relationship Diagram.....	44
20.	6.1	Timetable Screen Architecture Diagram.....	45
21.	6.2	Finance Screen Architecture Diagram.....	46
22.	6.3	Chatbot Screen Architecture Diagram.....	46
23.	6.4	Email Alerts Screen Architecture Diagram.....	47
24.	6.5	EmailAlert Database Snapshot.....	49
25.	6.6	ChatMessage Database Snapshot.....	47
26.	6.7	Timetable Database Snapshot.....	49
27.	6.8	User Database Snapshot.....	49
28.	6.9	UserBudget Database Snapshot.....	49
29.	6.10	UserExpense Database Snapshot.....	50
30.	7.1	Student Assistant Landing Screen.....	54
31.	7.2	Student Assistance Finance Screen.....	55
32.	7.3	Student Assistant Finance Detail Screen.....	56
33.	7.4	Student Assistant AI Chatbot Screen.....	57
34.	7.5	Student Assistant Timetable Screen.....	58
35.	7.6	Student Assistant Email Alerts Screen.....	59

Chapter 1

INTRODUCTION

1.1 Abstract

Students presently face numerous difficulties in focusing on the academics in complement to other dubious financial and daily life inconveniences. Hence, final year project entails the creation of an integrated student utility app enriched with artificial intelligence (AI) capabilities. This comprehensive app is designed to significantly improve students' academic management and financial control within the university environment. The project's primary focus includes the development of a robust finance/budget tracking feature, enabling students to efficiently manage their expenses and enhance financial literacy. Leveraging AI, the application also automates the extraction of timetable information from university emails, streamlining schedule organization. Furthermore, the integration of a virtual AI assistant provides personalized guidance to users. This project aims to create a practical solution that empowers students to navigate their academic journey more effectively and underscore the value of AI integration in educational technology. Throughout the development process, user feedback was integrated to ensure an intuitive and easy-to-use interface. Evaluations conducted through testing and user feedback sessions have provided valuable insights into the application's performance and usability. The findings contribute to ongoing improvements and future enhancements to better meet the needs of students. In conclusion, the Student Assistant Application offers a valuable tool to help students navigate their academic journey and manage their busy lives more effectively.

1.2 Project Overview

In today's fast-paced educational environment, students often struggle to juggle multiple responsibilities such as attending classes, completing assignments, managing extracurricular activities, and maintaining a healthy work-life balance. The Student Assistant Project seeks to alleviate these challenges by providing a user-friendly application that integrates various features to streamline academic and personal tasks.

1.3 Objectives

The primary objective of the Student Assistant Project is to develop a robust and user-friendly application that assists students in managing their academic and personal tasks effectively. Specifically, the project aims to achieve the following objectives:

1. Provide a centralized platform for students to access and manage their academic schedule, including class timetables, assignment deadlines, and exam dates.
2. Facilitate efficient communication among students, teachers, and administrators through features such as email management and notifications.
3. Offer tools for financial management, including expense tracking and budget management, to help students maintain financial responsibility.
4. Integrate a virtual AI assistant to provide personalized assistance and information retrieval services to students.
5. Enhance user experience through intuitive interface design, seamless navigation, and responsive performance.

1.4 Scope

The scope of the Student Assistant Project encompasses the development of a comprehensive software application tailored to the needs of students in educational institutions. The application will include modules for email management, financial tracking, timetable management, and virtual AI assistance. While the initial version of the application will focus on core functionalities, future iterations may incorporate additional features based on user feedback and requirements.

1.5 Approach

The approach to developing the Student Assistant Project involves a systematic and iterative process, starting with requirements analysis and culminating in software deployment and user feedback collection. The project will follow the agile methodology, allowing for flexibility and adaptation to changing requirements throughout the development life cycle. Collaboration with stakeholders, including students, teachers, and administrators, was integral to ensuring that the application meets the needs and expectations of its users.

1.6 Styles

1.5.1 Typeface

Space of the text line is 1.5 inches, Font. 12 Times New Roman (TNR), text is justified. The first line of the paragraph is indented and single line space be given between paragraphs.

1.5.2 Margins

Left Margin is 1.5 inches. Right, Top, and Lower Margin is 1.2 inches.

1.5.3 Headings

Chapter Number 16 TNR (Italics, Bold, Justified to the right, first letter in capital i.e. “C”). Chapter Heading 16 TNR (All capital, bold, adjusted in the center)

The following headings all are left aligned and text begins from the nextline with indentation.

1. First Level Heading 14 TNR (All capital, bold)
2. Second Level Heading 12 TNR (Bold, First letter of each main word in capital) ThirdLevel Heading 12 TNR (Bold, Only first letter of first word in capital)
3. Figure captions are 10 TNR (Blue)

1.6 FIGURES

Titles of the figures are written under the figures, along with the figure number (1.1: Xyz)

1.7 REFERENCES

All of the references are alphabetically ordered and APA style formatted.

Web References have name of the Author/s (If Known), Title of the topic followed by complete web address.

Chapter 2

LITERATURE REVIEW

In recent years, educational institutions worldwide have increasingly integrated technology into their operations to enhance teaching, learning, and administrative processes. Educational technology, or EdTech, encompasses a broad spectrum of digital tools and platforms designed to support educators, students, and administrators in various aspects of teaching and learning. These include Learning Management Systems (LMS), Student Information Systems (SIS), Virtual Learning Environments (VLE), and educational apps, among others. The proliferation of EdTech solutions reflects a growing recognition of the potential of technology to transform educational practices and improve student outcomes (Donahoe, B., Rickard, D., Holden, H., Blackwell, K., & Caukin, N. (2019).

Despite the benefits of technology integration in education, students face numerous challenges in effectively implementing and leveraging EdTech solutions. Some of those challenges include ensuring equitable access to technology, Difficulty in managing timetables, assignments, and deadlines. Inefficient email management leading to information overload, Lack of financial literacy and budgeting skills, Limited access to centralized academic resources and support services and navigating data privacy and security concerns. Additionally, there is a need to balance the integration of technology with pedagogical best practices to promote meaningful learning experiences and student engagement.

It is observed that students have difficulty in their financial management due to

various reasons and that majority of students do not even have budget (Thobejane, K., & Fatoki, O., 2017). More over, studies suggest that students have tendencies to make unplanned and unnecessary spending that may reflect on their academic procrastination (Parfenova, A. and Romashova, S., 2020). Another study showed that the significant portions of students' budget are used to spend on shopping, dining out, mobile phone expenditures, attending training for their career, travelling on trips with their friend and transit. It was seen that the majority of students lacked proper budgeting habits. The study recommended that the students should have knowledge how much they can spend (Nwe Ni Aung and Hla Hla Mon, 2020). Hence, it can be inferred that Students, who practice good financial management, will tend to receive less debt and show better financial well-being Chan, S. F., Chau, A. W.-L., & Chan, K. Y.-K., 2012). There have been practices to help with the students financial management problems. One study intended to implement a budgeting course in "Personal Financial Planning" to students that would help them in their financial planning (Farahdita Dyah Susanto and Ria Sandra Alimbudiono, 2019). But there is still a gap in helping students through an application that would help them in their financial management. Our application fills that gap with the use of AI to suggest the user on their spending.

It is said that on average, we get 20 emails per day. The time spent per email is 2 minutes. Hence, 80 minutes of the student's time are spent by reading the emails. Additionally, most of the email could not be important to students. For Example, some emails are shared to students that does not help the them. Therefore, there should be a email summarizing feature that saves the student's precious time by big margin. The ShortMail system (Mahira Kirmani, Gagandeep Kaur, Mudasir Mohd, 2023) is one of the email summarization system made using semantic models and deep-learning technologies to summarize the incoming emails and leave the important part to the user. However this system is made independent and made using deep learning technology. Our application is domain specific and is made integrating ChatGPT API for students usage. Personalization is also a factor in emails as study conducted at

King Abdulaziz University showed that only 23% of students were utilizing email where in contrast, 80% of the faculty members were utilizing the emails (Alwagdani, B., & Alomar, K., 2020). The results show that there is personalization problem with the emails that students receive. Our application tackles this problem by providing emails that are only related to the particular student.

It is observed that large amount of time daily is wasted by scrolling through emails and gallery to find the appropriate timetable for a student's course. More over, timetable tends to change frequently. Hence our timetable module automates this process by extracting the timetable automatically through email and update it on the app. The timetable that is updated on the app is relevant to the student and their semester. The timetable could be manually updated later if user wants to make any change in it. Kanagaraj, E., Arshad, N. S., & Santiagoo, R. solve the timetable problem similarly where University Academic Portal is designed to create a dynamic timetable for students and staff, tailored to their specific courses and fields. It features an automated system that ensures users stay organized with an up-to-date schedule of classes and university events. Our application has similar goal with the additional features like postponed or extra class update that is automatically fetched and updated through email.

The Student Assistant further provides the chat-bot functionality powered by ChatGPT API that is trained to give answer to any query from students. Some similar work done in educational field is by Lappalainen, Y., & Narayanan, N. in 2023 where they built a chat-bot named Aisha using ChatGPT API to provide quick and efficient reference and support services to students and faculty outside the library's regular operating hours. However our implementation would be for any general students' query.

There are various applications related to Student Assistant Application. Most of those are individual tools and not a collective suite for students. However, most related application to this project is "Studo - University Student App" available in Google

Play Store. It has the following features relating to our project:

1. **Mail:** Students have all important e-mails at hand.
2. **Courses:** In the Courses tab you can find all courses grouped by semester. There is also a menu with an overview and dates for each course.
3. **Calendar:** The calendar is filled automatically and without manual effort with the timetable imported from the university system.
4. **To Do list**

The project is developed in Europe and has been going since 2016.

However, the Student Assistant Application contrasts the Studo App in following ways:

1. **Scalability:** Although The Studo App is independent provider it still should be used by students enrolled in universities supporting the Studo App. For the Student Assistant application, you would need any university email access to use it. However, the email policy of that university must align with Sukkur IBA University in terms that the timetable must be shared through email, class postponement, extra class, exam schedule, all should be shared in email. The format of timetable must also align with the timetable of Sukkur IBA University.
2. **Use of AI:** The Student Assistant Application uses AI to streamline tasks of students. When it comes to summarizing emails, class postponements, extra classes, or budget suggestions. This feature is not found in any application related to student assistance.
3. **Student Assistant Chatbot:** Although the Studo does have chatting feature but it is only for the contact with university administration. The Student Assistant Application automates this process and introduces AI into this. Any query of

student is answered by the AI Assistant Chatbot. The AI-driven features of the Student Assistant Application offer real-time assistance to students. Whether it's providing quick answers to queries, offering study tips, or suggesting budget-saving strategies, the application provides immediate support to users, enhancing their overall experience.

Chapter 3

PROBLEM DEFINITION

University student life encompasses a myriad of challenges that can impede academic success, personal development, and overall well-being. These challenges span various facets of student life, including academic responsibilities, extracurricular engagements, financial management, and access to essential resources and support services.

1. Timetable Management Challenges:

University students often grapple with the complexities of managing their class schedules, assignment deadlines, and exam dates. Juggling multiple courses with varying timetables each semester poses a significant organizational challenge. Inconsistent course schedules, last-minute changes in class timings, and overlapping commitments further exacerbate the situation. Without a centralized system for managing timetables and academic deadlines, students risk missing classes, submitting assignments late, and experiencing unnecessary stress and anxiety.

2. Email Overload and Information Overwhelm:

The influx of emails inundating students' inboxes from various sources, including professors, administrative departments, student organizations, and peers, presents a significant challenge. Managing the deluge of emails, discerning critical information amidst clutter, and prioritizing actions based on email content prove daunting. Without efficient email management tools and strategies, students may inadvertently overlook important communications, miss essential deadlines, and struggle to stay informed about academic and extracurricular activities.

3. Financial Management Dilemmas:

Many students grapple with financial management challenges, including tracking expenses, budgeting for tuition fees and living expenses, and planning for future financial obligations. Limited financial literacy, inadequate budgeting skills, and a lack of awareness about available resources and support services contribute to financial stress and uncertainty among students. Without access to reliable financial management tools and resources, students may find themselves overspending, accruing debt, or facing financial hardship, which can adversely affect their academic performance and overall well-being.

4. Access to Information and Support Services:

Students frequently encounter obstacles in accessing relevant information and support services related to academics, campus life, health and wellness, and career development. Navigating university websites, locating essential resources, and understanding administrative procedures can be challenging. Limited availability of support services, prolonged wait times for appointments, and insufficient communication channels exacerbate feelings of frustration, isolation, and disconnection from the university community.

5. Communication and Collaboration Barriers:

Effective communication with professors, administrative staff, and fellow students is essential for academic success. However, students often face barriers in communicating effectively, particularly in large lecture-based courses or online learning environments. Inadequate communication channels, limited opportunities for collaboration, and a lack of personalized feedback and support hinder students' academic engagement, participation, and sense of belonging. Without access to effective communication tools and platforms, students may struggle to seek academic guidance, clarify doubts, and engage meaningfully with peers and faculty members.

Chapter 4

SOLUTION STATEMENT

The Student Assistant Application aims to address the multifaceted challenges faced by university students by providing a comprehensive, user-centric, and technologically advanced solution. Leveraging cutting-edge technologies, innovative features, and user-centered design principles, the application offers a holistic platform to support students' academic endeavors, enhance their personal well-being, and foster a thriving university experience.

Following are the key features and functionalities:

1. Timetable Management:

The application provides students with a centralized platform to manage their class schedules, assignment deadlines, exam dates, and other academic commitments. Through intuitive calendar interfaces, personalized notifications, and automated updates, students can stay organized, track their academic progress, and plan their study routines effectively. The system allows for easy customization of timetables, integration with university systems, and seamless synchronization across devices.

2. Email Management:

To streamline email communication and reduce information overload, the application offers advanced email management capabilities. ChatGPT API, the system automatically categorizes and prioritizes incoming emails, highlights important messages, and generates summaries for quick review. Additionally, students can set preferences, create filters, and customize email settings to tailor their communication

experience.

3. Financial Tracker:

To promote financial literacy and responsible money management, the application includes a robust financial tracker module. Students can track their expenses, create budgets, set financial goals, and receive personalized recommendations based on their spending habits and financial goals. The system aggregates financial data from multiple sources, analyzes spending patterns, identifies areas for cost optimization, and provides actionable insights to help students make informed financial decisions.

4. AI Virtual Assistant:

A central component of the application is the AI-powered virtual assistant, designed to provide personalized assistance, answer queries, and offer proactive support to students. The virtual assistant can assist students with a wide range of tasks, including timetable updates, email summaries, financial queries, academic advice, and campus navigation. The virtual assistant learns from user interactions, adapts to individual preferences, and continuously improves its performance over time.

Chapter 5

DETAILED DESIGN AND ARCHITECTURE

5.1 System Architecture

5.1.1 Architecture Design Approach

The architectural design approach adopted for the Student Assistant Application is based on the Model-View-ViewModel (MVVM) pattern, which promotes separation of concerns and facilitates better organization of code. In MVVM, the View layer is responsible for displaying data to the user, the ViewModel layer acts as an intermediary between the View and the Model, and the Model layer represents the underlying data and business logic.

By adhering to the MVVM pattern, the Student Assistant Application benefits from several advantages:

1. **Separation of Concerns:** MVVM promotes a clear separation of concerns between the UI logic (View), application logic (ViewModel), and data logic (Model), making the codebase easier to understand, maintain, and test.
2. **Testability:** Each component in the MVVM architecture can be independently unit tested, enabling developers to validate the behavior of individual components in isolation.
3. **Reusability:** MVVM encourages the reuse of ViewModels and Models across different UI components, facilitating code reuse and minimizing code duplication.
4. **Flexibility:** The MVVM pattern allows developers to adapt the application's UI and business logic independently, making it easier to accommodate changes and enhancements over time.

5.1.2 Architecture Design

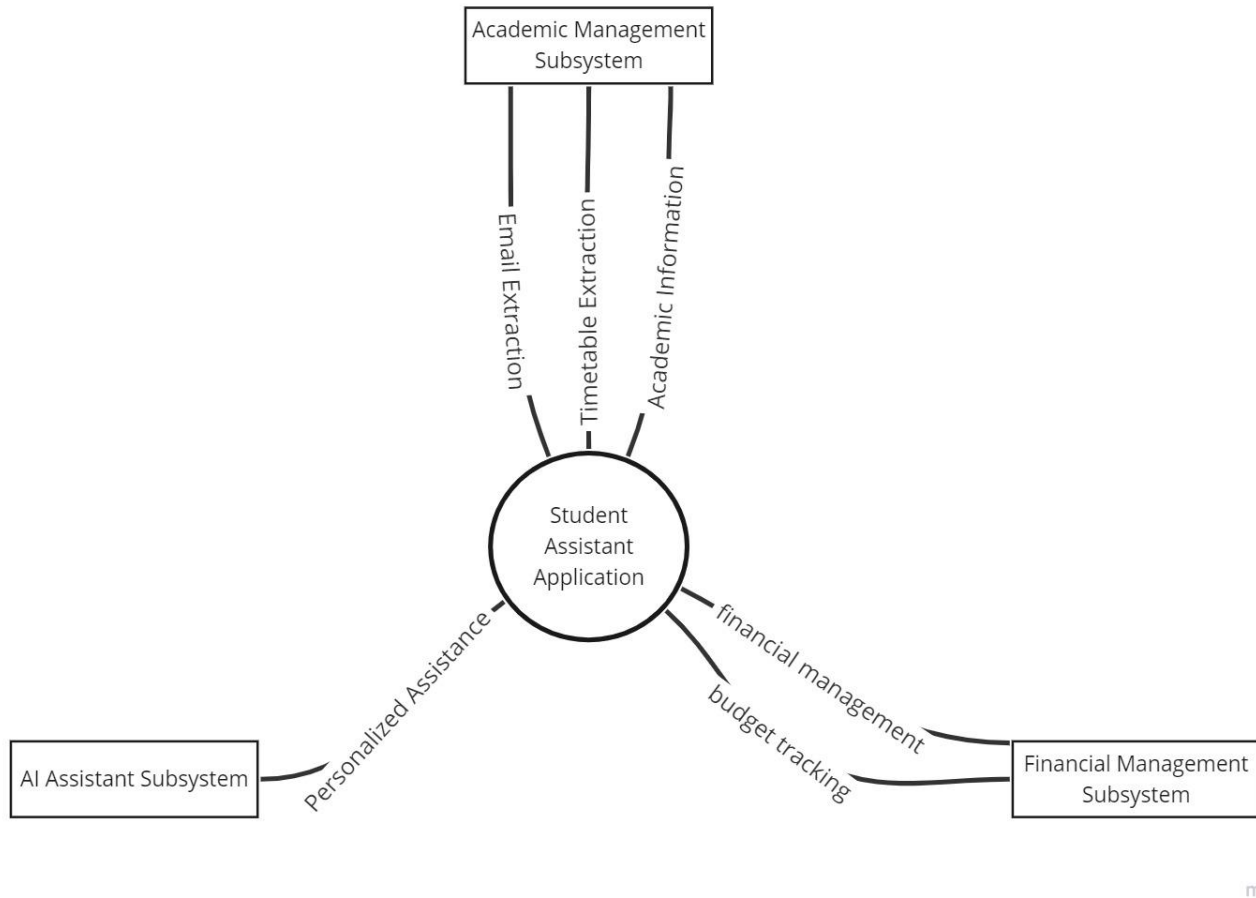


Figure 5.1 Context Diagram of the System

The Figure 5.1 shows following modules:

1. **Academic Management Subsystem:** This subsystem provides academic information to the Student Assistant Application, including course schedules, and exam dates.
2. **Financial Management Subsystem:** This subsystem is responsible for providing financial data and suggestions to the Student Assistant Application. It helps track

students budget and spending.

3. AI Assistant Subsystem: This subsystem provides artificial intelligence (AI) assistance to the Student Assistant Application.

The arrows in the diagram indicate a two-way flow of data between the Student Assistant Application and each external system. This means the application can both receive and send data to these systems.

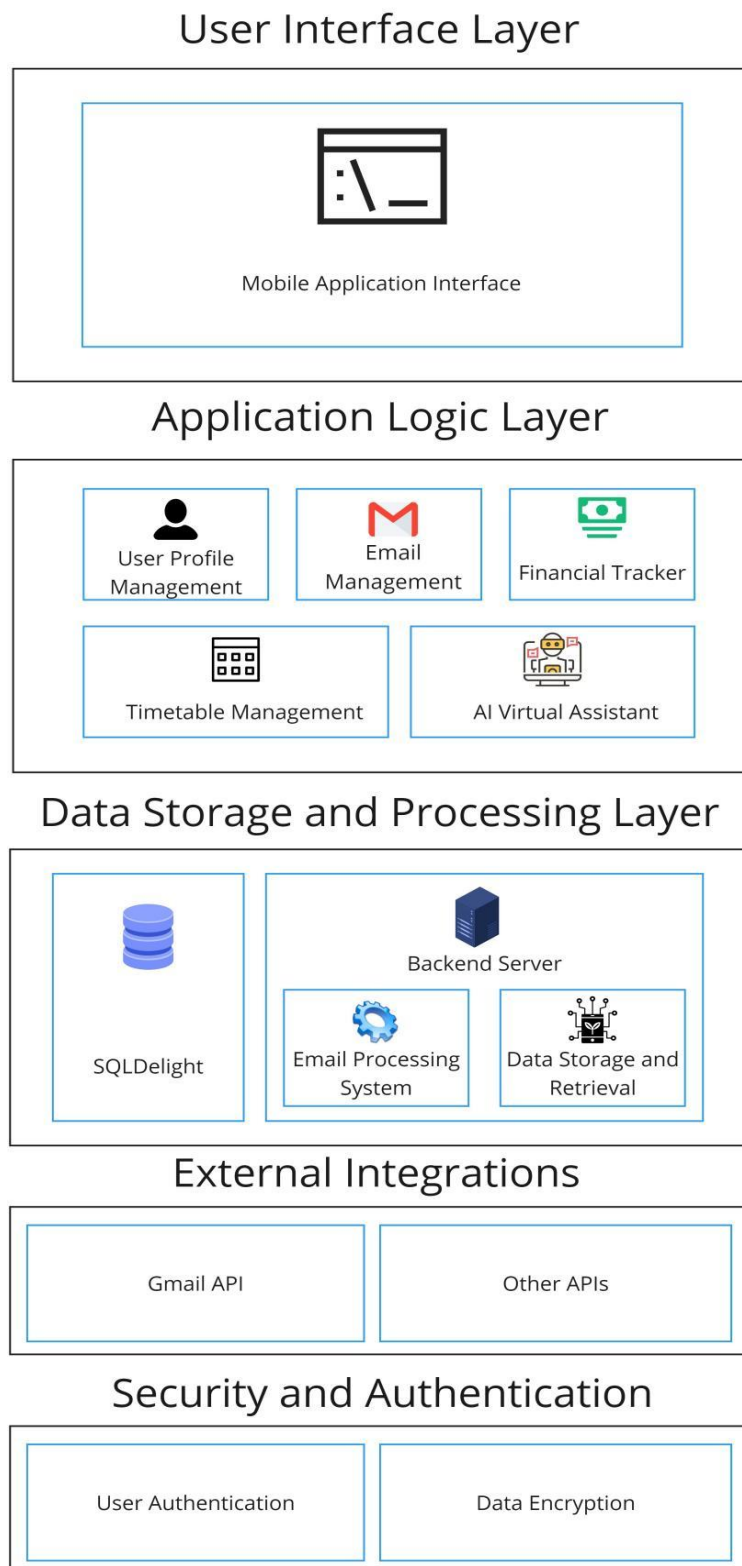


Figure 5.2 System Layered Architectural Diagram

The above Architectural Diagram (Figure 5.2) shows that the Application Logic Layer will include:

1. User Profile Management: This module manages the profile preferences and settings of the user.
2. Email Management: This manages the email part of the system including reading the emails, parsing them to ChatGPT API to get the meaningful information and then making appropriate action according to that.
3. Financial Tracker: This module tracks the budget and spending of the user and give suggestions according to it. Moreover, it will also provide visualization and reports.
4. Timetable Management: This module shows Timetable to the users.
5. AI Virtual Assistant: This module serves as a guidance for the students that have any general query.

The system will use the SQL Delight Database for its database operations. Other APIs like ChatGPT and GmailAPI are being used for external integration.

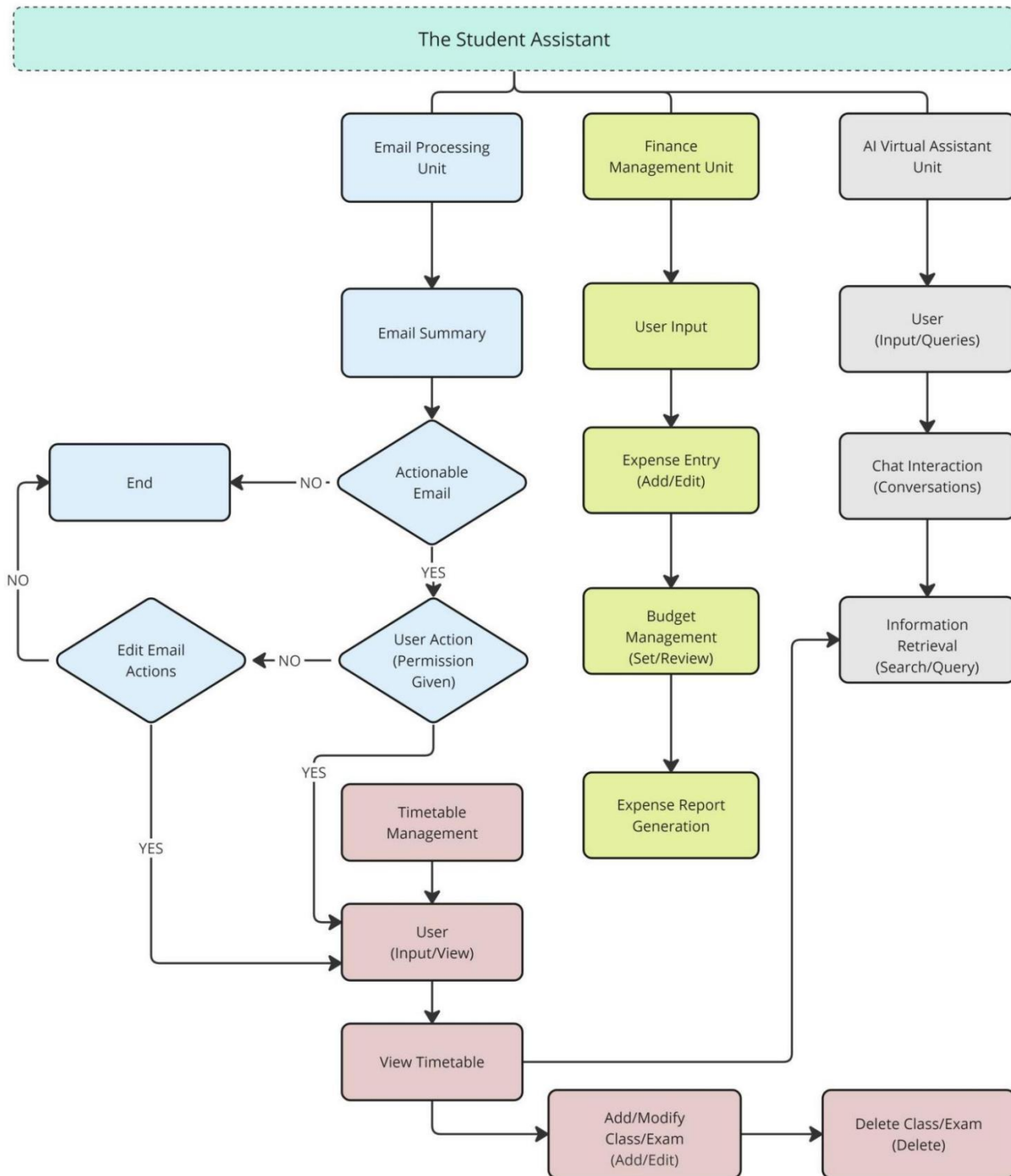


Figure 5.3 The Student Assistant Functional Flow Diagram describes the whole functional flow of the system including all the modules.

Phase-1:

5.1.3 Subsystem Architecture

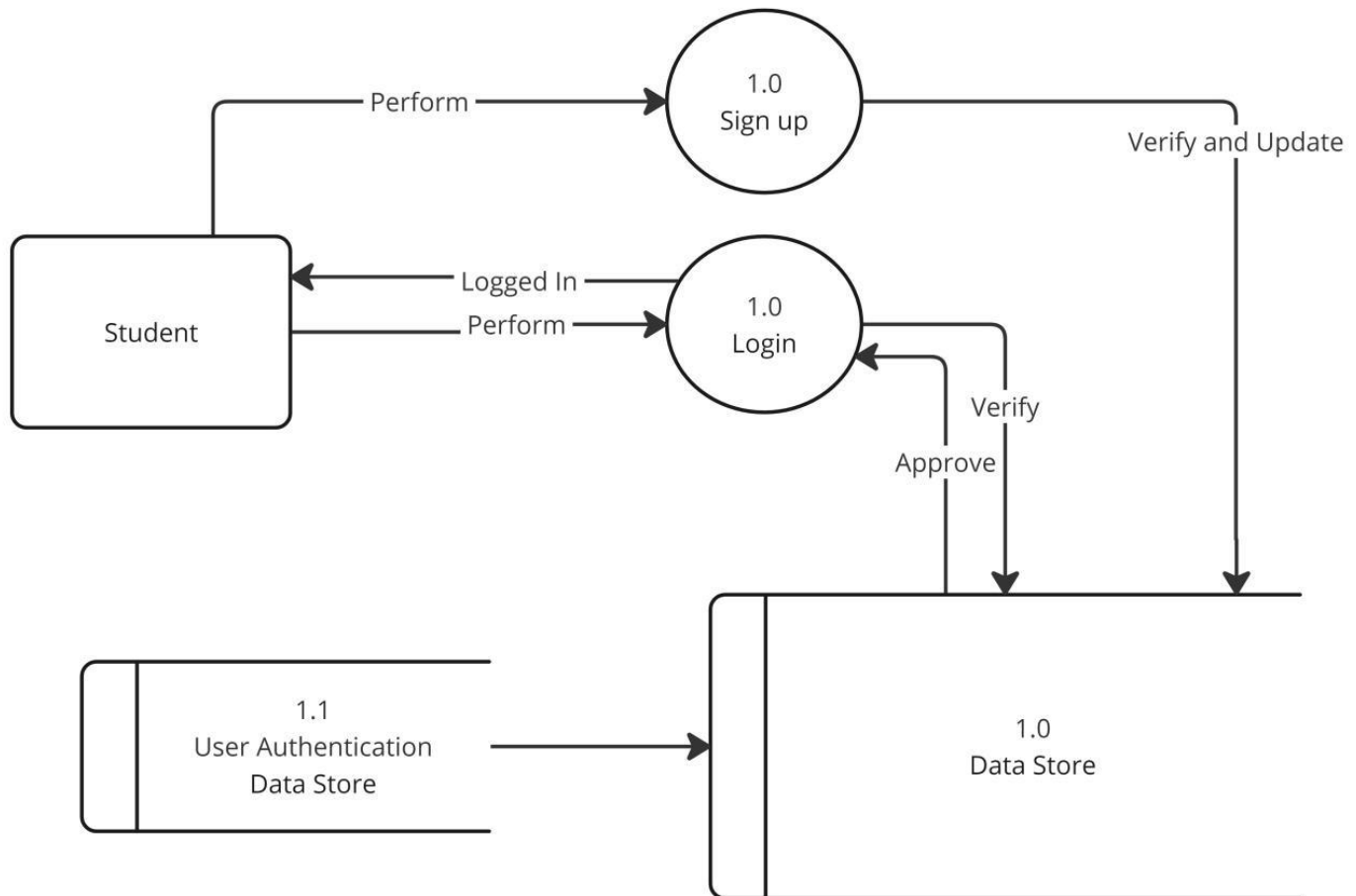


Figure 5.6 Login/Signup Data Flow Diagram

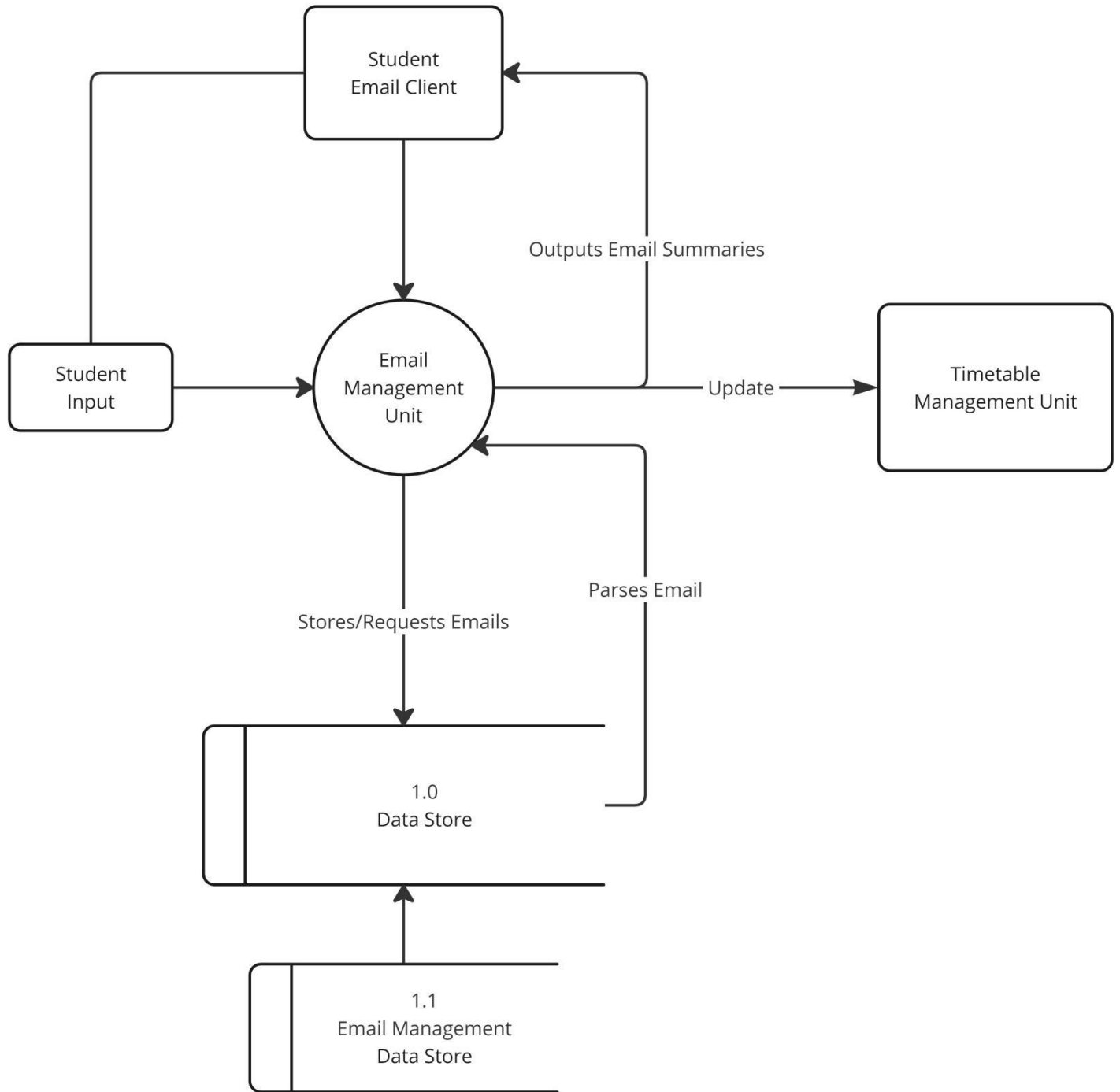


Figure 5.7 Email Management Data Flow Diagram

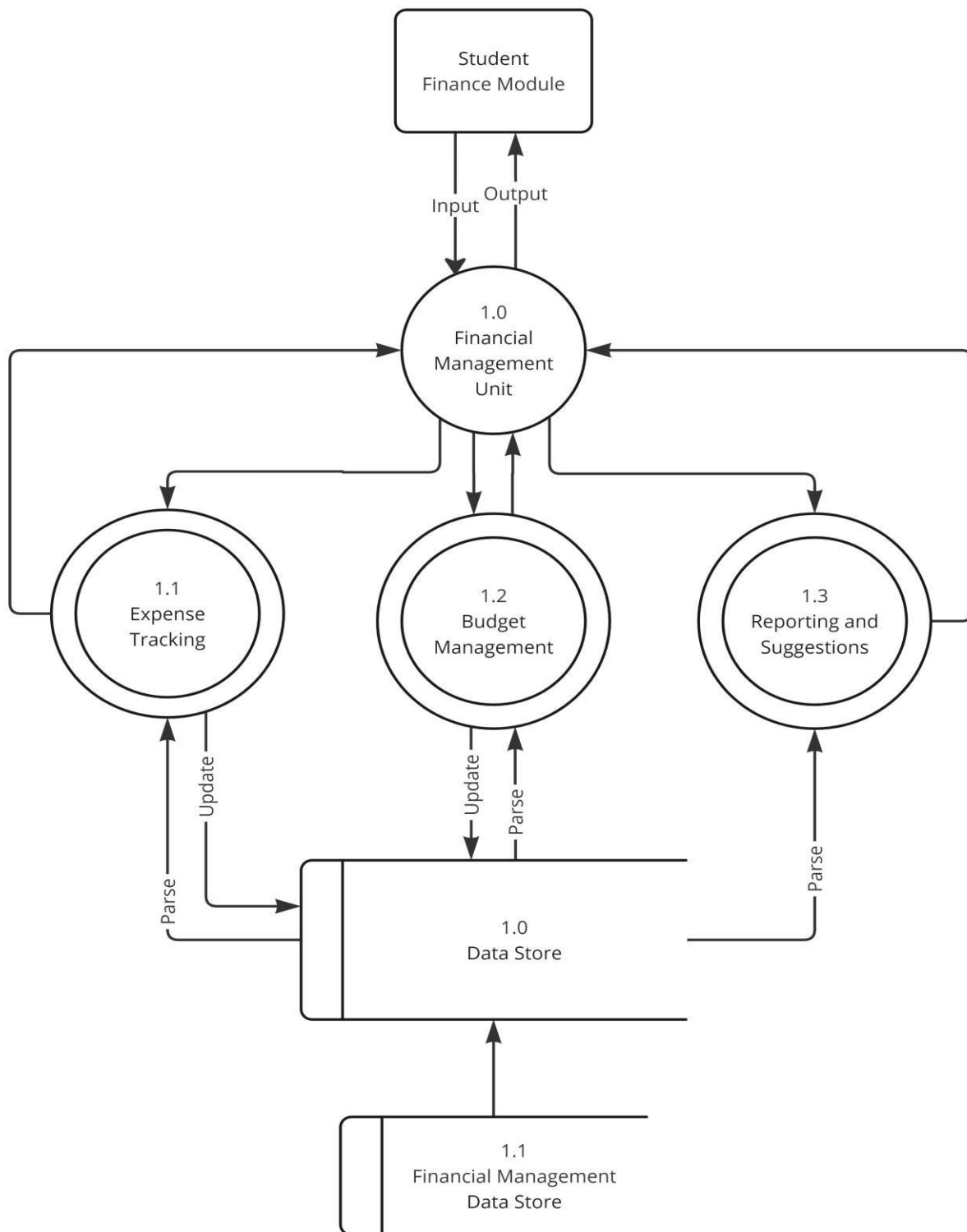


Figure 5.8 Financial Management Data Flow Diagram

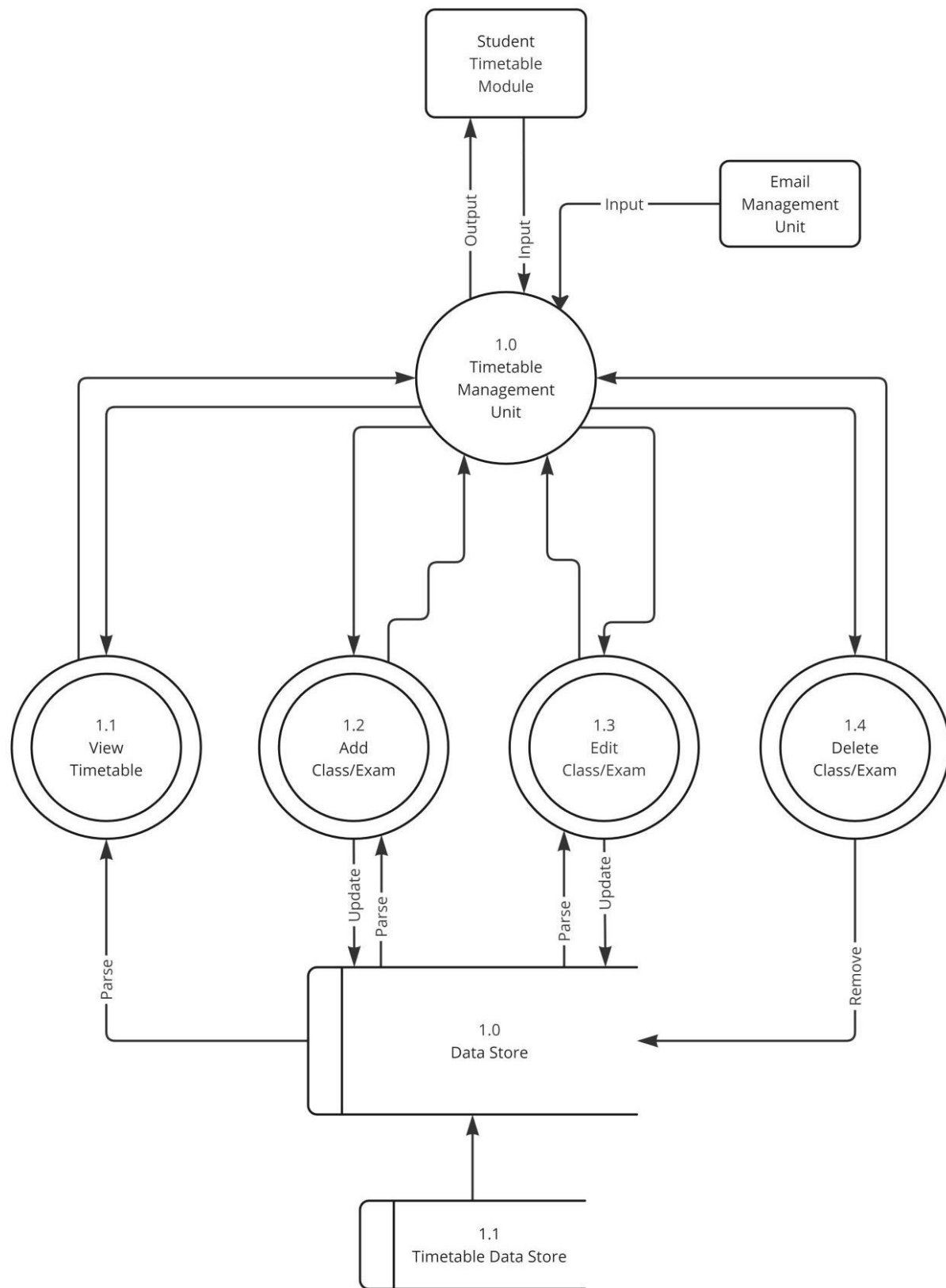


Figure 5.9 Timetable Management Data Flow Diagram

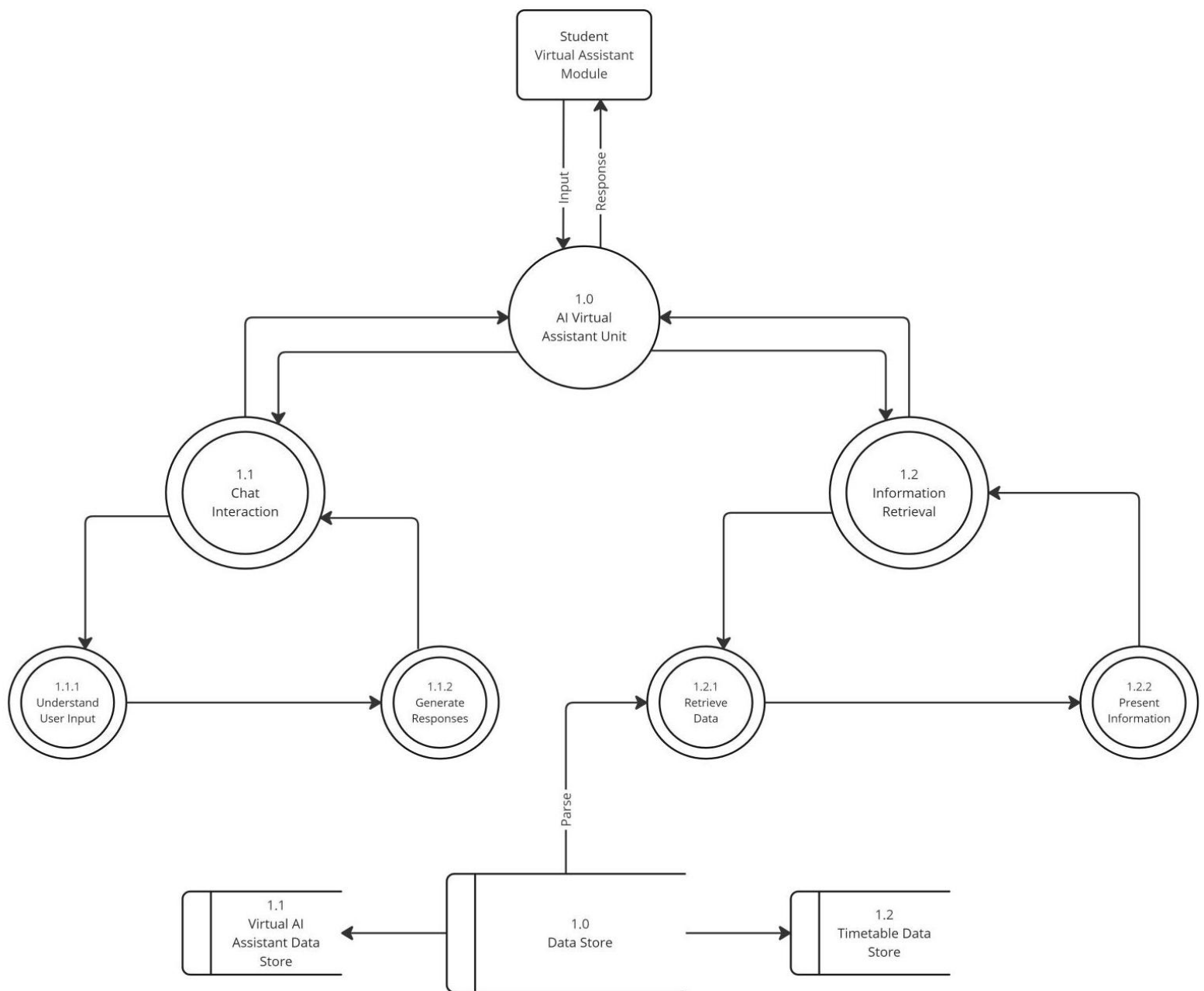


Figure 5.10 Virtual Assistant Data Flow Diagram

5.2 Detailed System Design

5.2.1 Chatbot Module

The ChatBotScreen.kt, ChatBotViewModel.kt, ChatMessage.kt, ChatMessageField.kt, MessageResponse.kt, and ChatMessage.sq are Kotlin and SQLDelight files respectively. They are part of the com.lumins.sua.android.views.chatbot, com.lumins.sua.data.model, and com.lumins.sua.data.local.db packages.

These files are part of the chatbot module of the Student Assistant Application. They are responsible for managing the chat interface, handling user input, sending messages to the AI assistant, displaying the responses, and managing the state of the chat.

Each file has a specific responsibility related to the chatbot feature of the application. For example, ChatBotScreen.kt is responsible for displaying the chat interface, ChatBotViewModel.kt is responsible for managing the state of the chat and handling sending messages to the AI assistant, ChatMessage.kt and ChatMessageField.kt are responsible for displaying a single chat message and providing a text field for the user to enter their message respectively. MessageResponse.kt is a data class that represents the response from the AI chatbot. ChatMessage.sq is a SQLDelight file that defines the ChatMessage table in the local database and its associated SQL operations.

These files must be used within the context of an Android application and require the Android SDK and runtime environment. They also depend on various Android and Jetpack Compose libraries. The classes must be used in accordance with the lifecycle of Android components (such as activities and fragments) and the state management principles of Jetpack Compose.

The chatbot module is composed of several classes and a SQLDelight file:

1. **ChatBotScreen.kt:** This class is a Composable function that displays the chat

interface.

2. **ChatBotViewModel.kt:** This class manages the state of the chat and handles sending messages to the AI assistant.
3. **ChatMessage.kt:** This class is a Composable function that displays a single chat message.
4. **ChatMessageField.kt:** This class is a Composable function that provides a text field for the user to enter their message.
5. **MessageResponse.kt:** This is a data class that represents the response from the AI chatbot.
6. **ChatMessage.sql:** This is a SQLDelight file that defines the ChatMessage table in the local database and its associated SQL operations.

The chatbot module interacts with other components of the application. For example, the ChatBotScreen.kt class uses the ChatBotViewModel.kt class to get the chat messages and send new messages. The ChatBotViewModel.kt class uses the SuaRepository class to interact with the database.

The chatbot module manages and interacts with various resources such as UI components and databases. It also manages the state of these resources and handles any potential concurrency issues. For example, the ChatBotViewModel.kt class manages the chat messages in the database and ensures that the database operations are performed in the correct order.

The chatbot module performs its duties by implementing methods that are called in response to user interactions or system events. It uses various algorithms and data structures to manage its state and perform its tasks. It also handles initialization and cleanup of resources, and handles exceptional conditions such as errors or unexpected inputs.

The chatbot module provides a set of methods and properties that can be used by other

classes or components. These methods and properties form the interface of the classes. For example, the `ChatBotViewModel.kt` class provides a `sendMessageToAI` method that can be used to send a new message to the AI, and a `messages` property that can be used to get the current list of chat messages.

AiChatView `getChatData()` Flow

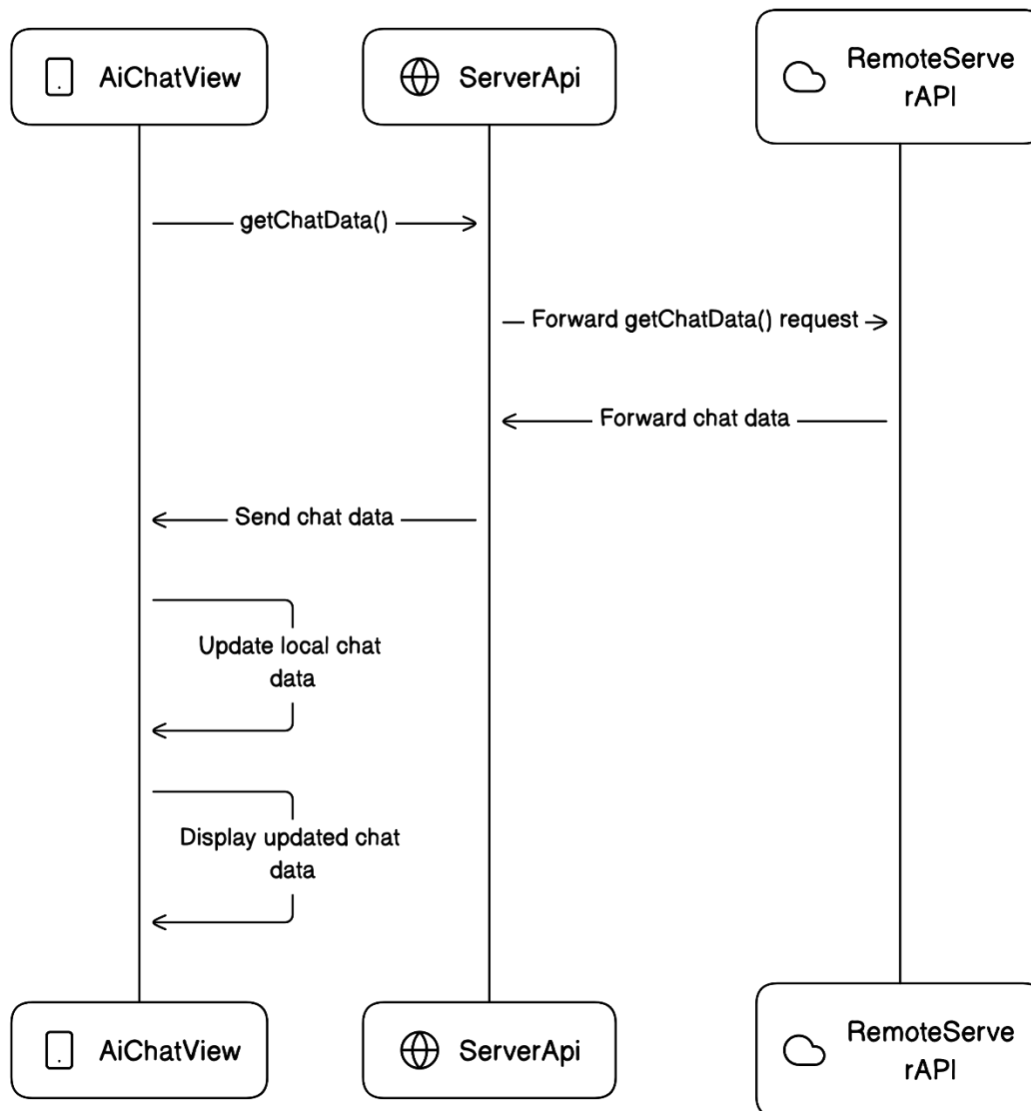


Figure 5.11 `getChatData` Sequence Diagram shows the sequence of fetching chat data from chatbot

AiChatView sendMessageToAi() Flow

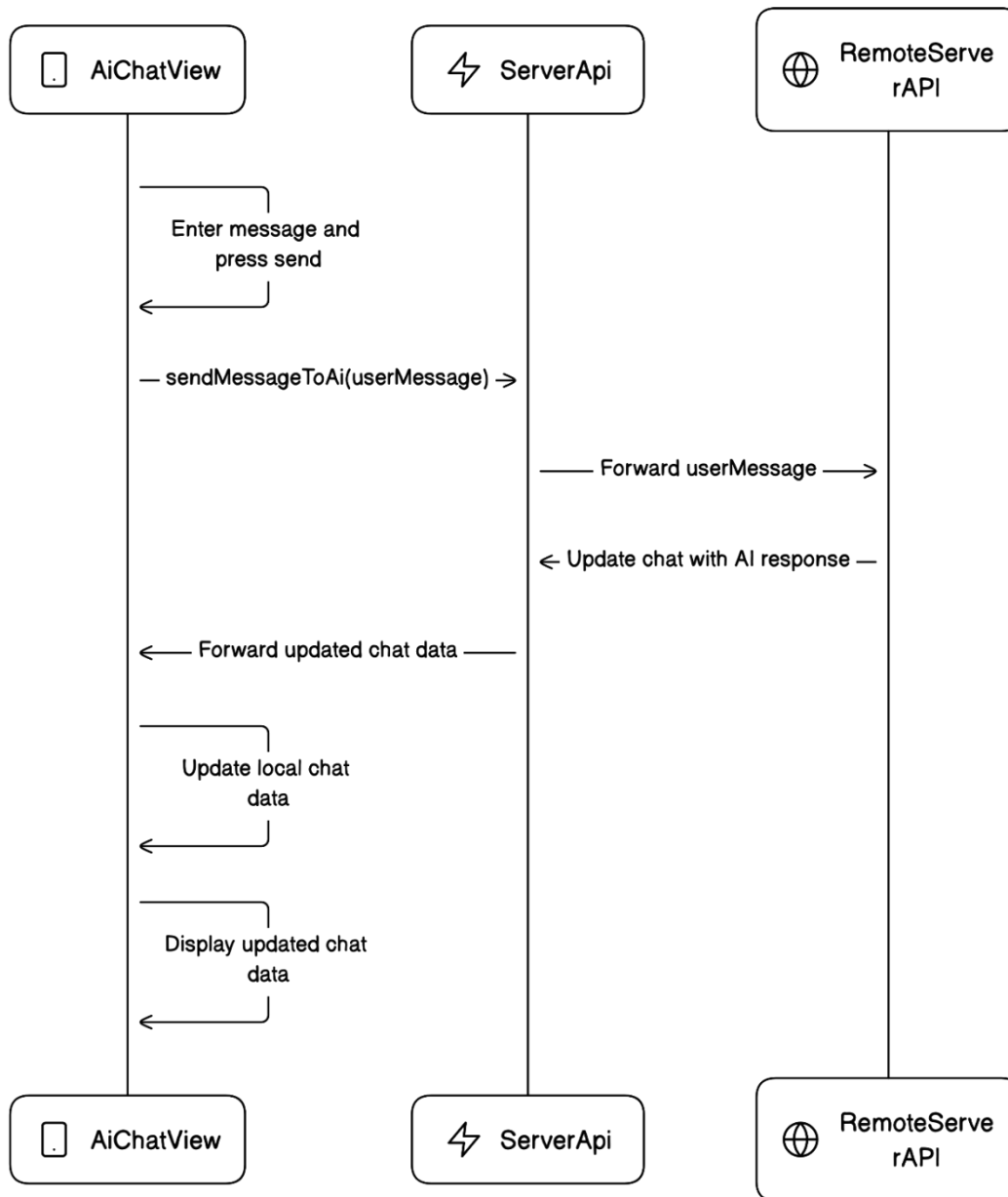


Figure 5.12 `sendMessageToAi` Sequence Diagram shows sequence of sending and receiving messages to AI Chatbot

5.2.2 Financial Management Module

The Financial Management module is composed of several Kotlin classes ('FinanceScreen.kt', 'FinanceViewModel.kt', 'WeekBudgetScreen.kt', 'BudgetCategoryCard.kt', 'ExpenseCategoryDetails.kt', 'FinanceReportCard.kt') and SQLDelight files ('Finance.sq'). These files are part of the 'com.lumins.sua.android.views.finance', 'com.lumins.sua.android.views.finance.components', 'com.lumins.sua.data.model', and 'com.lumins.sua.data.local.db' packages.

The Financial Management module is designed to provide financial management features for the Student Assistant Application. It allows users to track their income, expenses, and savings, and provides visualizations of their financial data.

Each file in the 'finance' module has a specific responsibility related to the financial management feature of the application. For example, 'FinanceScreen.kt' is responsible for displaying the main finance screen, 'FinanceViewModel.kt' is responsible for managing the state of the finance screen, 'WeekBudgetScreen.kt' is responsible for displaying the budget for a single week, 'BudgetCategoryCard.kt' is responsible for displaying a card for a budget category, 'ExpenseCategoryDetails.kt' is responsible for displaying the details of an expense category, and 'Finance.sq' is responsible for defining the SQL operations for the finance data.

The files in the Financial Management module must be used within the context of an Android application and require the Android SDK and runtime environment. They also depend on various Android and Jetpack Compose libraries. The classes must be used in accordance with the lifecycle of Android components (such as activities and fragments) and the state management principles of Jetpack Compose.

The Financial Management module is composed of several classes and a SQLDelight file:

1. **FinanceScreen.kt:** This class is a Composable function that displays the main finance screen.
2. **FinanceViewModel.kt:** This class manages the state of the finance screen.
3. **WeekBudgetScreen.kt:** This class is a Composable function that displays the budget for a single week.
4. **BudgetCategoryCard.kt:** This class is a Composable function that displays a card for a budget category.
5. **ExpenseCategoryDetails.kt:** This class is a Composable function that displays the details of an expense category.
6. **Finance.sq:** This is a SQLDelight file that defines the SQL operations for the finance data.

The Financial Management module interacts with other components of the application. For example, the `FinanceScreen.kt` class uses the `FinanceViewModel.kt` class to get the financial data and update the UI. The `FinanceViewModel.kt` class uses the `SuaRepository` class to interact with the database.

The Financial Management module manages and interacts with various resources such as UI components and databases. It also manages the state of these resources and handles any potential concurrency issues. For example, the `FinanceViewModel.kt` class manages the financial data in the database and ensures that the database operations are performed in the correct order.

The Financial Management module performs its duties by implementing methods that are called in response to user interactions or system events. It uses various algorithms and data structures to manage its state and perform its tasks. It also handles

initialization and cleanup of resources, and handles exceptional conditions such as errors or unexpected inputs.

The Financial Management module provides a set of methods and properties that can be used by other classes or components. These methods and properties form the interface of the classes. For example, the `FinanceViewModel.kt` class provides a `multiDataSetChartEntryModelProducer` property that can be used to get the data for the finance chart.

FinanceView GetBudgetData Flow

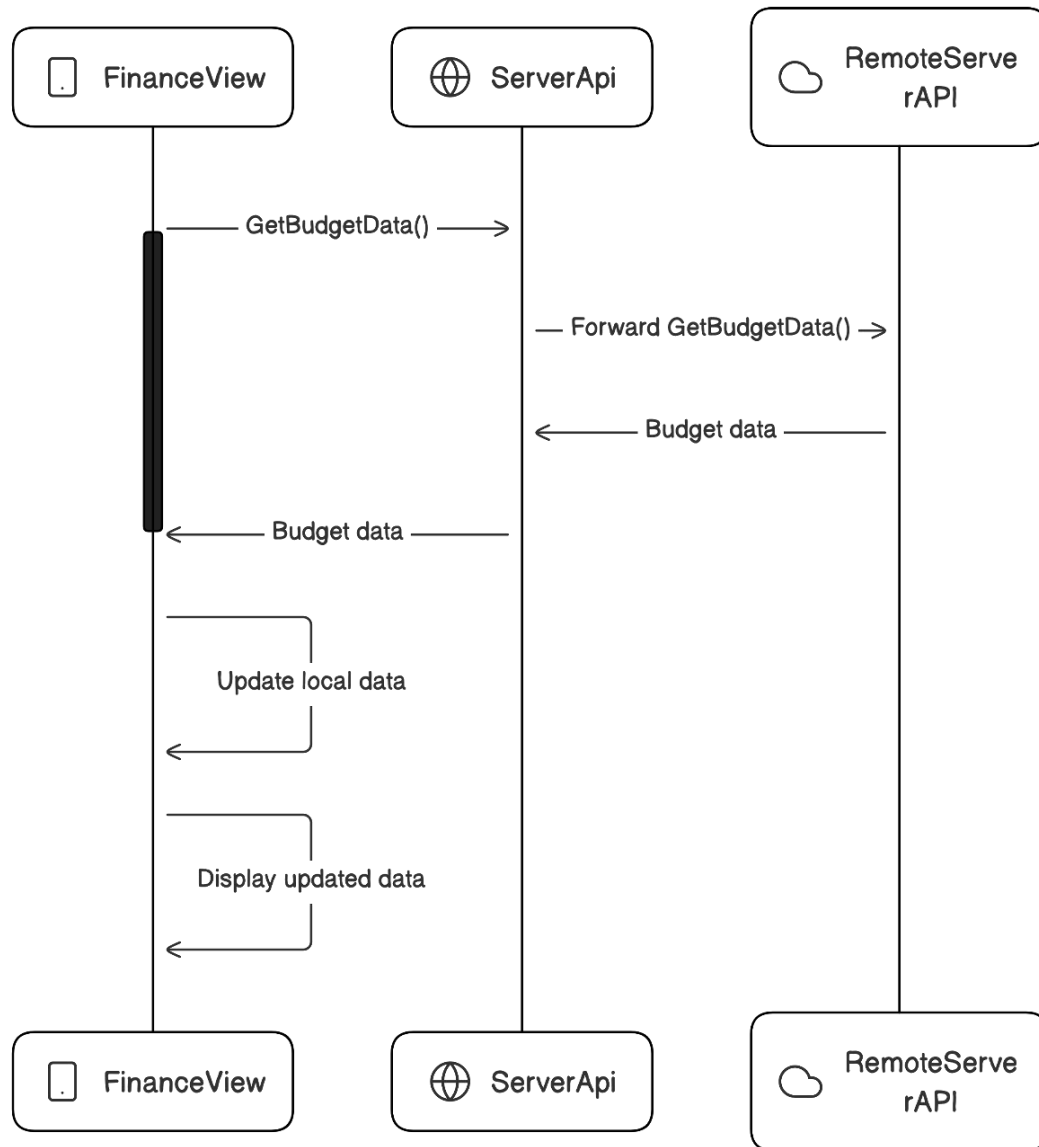


Figure 5.13 `GetBudgetData` Sequence Diagram shows how the system fetches the budget data

FinanceView's UpdateBudgetData() Flow

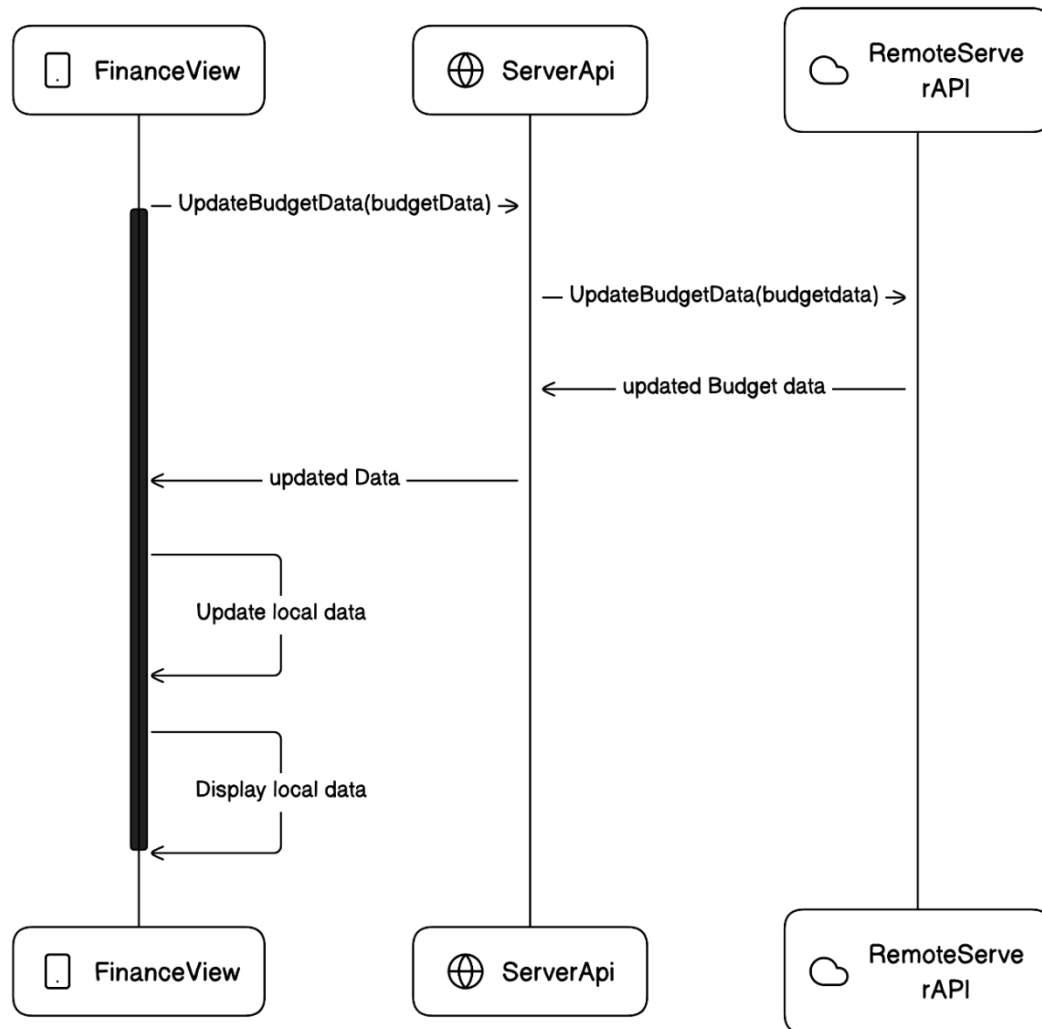


Figure 5.14 UpdateBudgetData Sequence Diagram shows how the system updates the budget data

FinanceView's UpdateExpenseData() Flow

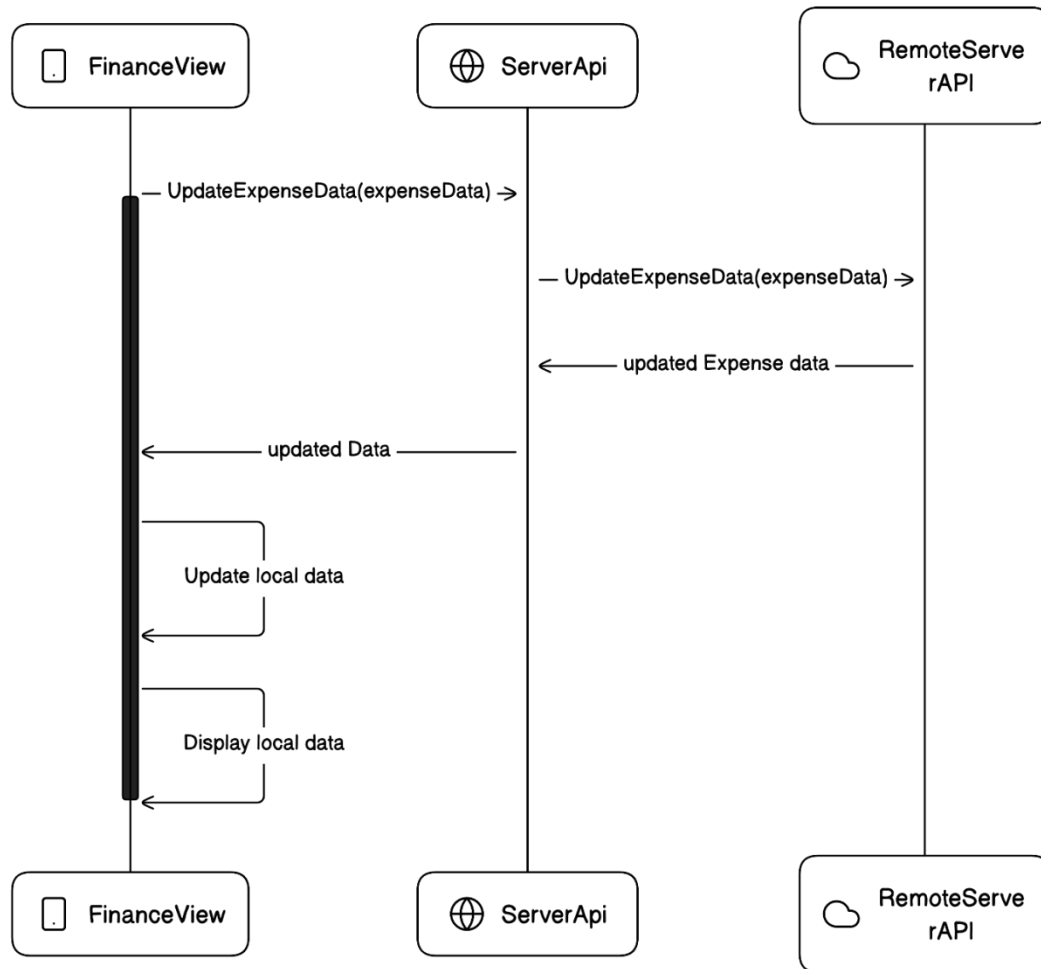


Figure 5.15 UpdateExpenseData Sequence Diagram shows how the system updates the expense data

5.2.3 Timetable Management Module

The `timetable` module is composed of several Kotlin classes (`TimetableViewModel.kt`, `TimetableScreen.kt`, `ClassesView.kt`, `ClassEventCard.kt`, `ClassSelection.kt`, `CalendarModeSelect.kt`, `WeeklyCalendar.kt`, `MonthlyCalendar.kt`) and SQLDelight files (`Timetable.sq`). These files are part of the `com.lumins.sua.android.views.timetable`, `com.lumins.sua.data.model`, and `com.lumins.sua.data.local.db` packages.

The `timetable` module is designed to provide timetable management features for the Student Assistant Application. It allows users to view and manage their class schedules, with options to view the timetable in weekly or monthly format.

Each file in the `timetable` module has a specific responsibility related to the timetable management feature of the application. For example, `TimetableViewModel.kt` is responsible for managing the state of the timetable screen, `TimetableScreen.kt` is responsible for displaying the main timetable screen, `ClassesView.kt` is responsible for displaying the list of classes, `ClassEventCard.kt` is responsible for displaying a single class event, `ClassSelection.kt` is responsible for selecting a class, `CalendarModeSelect.kt` is responsible for selecting the calendar mode (weekly or monthly), `WeeklyCalendar.kt` and `MonthlyCalendar.kt` are responsible for displaying the weekly and monthly calendars respectively, and `Timetable.sq` is responsible for defining the SQL operations for the timetable data.

The files in the `timetable` module must be used within the context of an Android application and require the Android SDK and runtime environment. They also depend

on various Android and Jetpack Compose libraries. The classes must be used in accordance with the lifecycle of Android components (such as activities and fragments) and the state management principles of Jetpack Compose.

The ``timetable`` module is composed of several classes and a SQLDelight file:

1. ``TimetableViewModel.kt``: This class manages the state of the timetable screen.
2. ``TimetableScreen.kt``: This class displays the main timetable screen.
3. ``ClassesView.kt``: This class displays the list of classes.
4. ``ClassEventCard.kt``: This class displays a single class event.
5. ``ClassSelection.kt``: This class is responsible for selecting a class.
6. ``CalendarModeSelect.kt``: This class is responsible for selecting the calendar mode (weekly or monthly).
7. ``WeeklyCalendar.kt``: This class displays the weekly calendar.
8. ``MonthlyCalendar.kt``: This class displays the monthly calendar.
9. ``Timetable.sql``: This is a SQLDelight file that defines the SQL operations for the timetable data.

The ``timetable`` module interacts with other components of the application. For example, the ``TimetableScreen.kt`` class uses the ``TimetableViewModel.kt`` class to get the timetable data and update the UI. The ``TimetableViewModel.kt`` class uses the ``SuaRepository`` class to interact with the database.

The ``timetable`` module manages and interacts with various resources such as UI components and databases. It also manages the state of these resources and handles any potential concurrency issues. For example, the ``TimetableViewModel.kt`` class manages the timetable data in the database and ensures that the database operations are performed in the correct order.

The ``timetable`` module performs its duties by implementing methods that are called in

response to user interactions or system events. It uses various algorithms and data structures to manage its state and perform its tasks. It also handles initialization and cleanup of resources, and handles exceptional conditions such as errors or unexpected inputs.

The ``timetable`` module provides a set of methods and properties that can be used by other classes or components. These methods and properties form the interface of the classes. For example, the ``TimetableViewModel.kt`` class provides a ``timetables`` property that can be used to get the current list of timetables, and a ``onEvent`` method that can be used to handle various events such as reloading the timetable, selecting a class name, and toggling the starred status of a timetable.

TimeTableView getTimeTableData() Flow

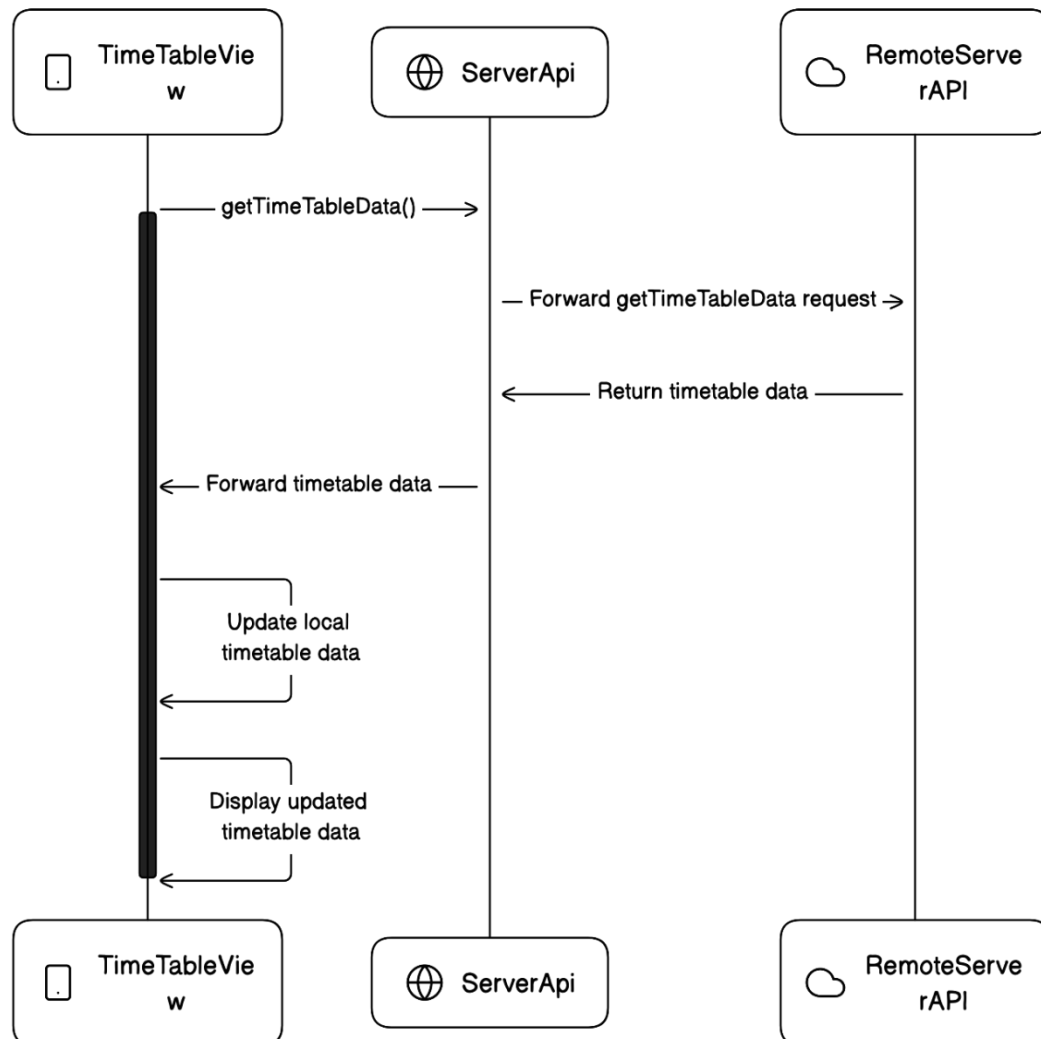


Figure 5.16 GetTimetableData Sequence Diagram shows how the system fetches timetable data

5.2.4 Email Management Module

The `email_alerts` module is composed of several Kotlin classes (`EmailAlert.kt`, `EmailAlertsScreen.kt`, `EmailAlertViewModel.kt`) and a SQLDelight file (`EmailAlert.sq`). These files are part of the `com.lumins.sua.android.views.email_alerts` package.

The `email_alerts` module is designed to provide email alert management features for the Student Assistant Application. It allows users to create, view, and delete email alerts.

Each file in the `email_alerts` module has a specific responsibility related to the email alert management feature of the application. For example, `EmailAlert.kt` is expected to define the data structure for an email alert, `EmailAlertsScreen.kt` is expected to display the email alerts screen, `EmailAlertViewModel.kt` is expected to manage the state of the email alerts screen, and `EmailAlert.sq` is responsible for defining the SQL operations for the email alert data.

The files in the `email_alerts` module must be used within the context of an Android application and require the Android SDK and runtime environment. They also depend on various Android and Jetpack Compose libraries. The classes must be used in accordance with the lifecycle of Android components.

The `email_alerts` module is composed of several classes and a SQLDelight file:

1. `EmailAlert.kt`: This class is expected to define the data structure for an email alert.
2. `EmailAlertsScreen.kt`: This class is expected to display the email alerts screen.
3. `EmailAlertViewModel.kt`: This class is expected to manage the state of the email alerts screen.
4. `EmailAlert.sq`: This is a SQLDelight file that defines the SQL operations for the email alert data.

The ``email_alerts`` module interacts with other components of the application. For example, the ``EmailAlertsScreen.kt`` class uses the ``EmailAlertViewModel.kt`` class to get the email alert data and update the UI. The ``EmailAlertViewModel.kt`` class uses the ``EmailAlert.sql`` to interact with the database.

The ``email_alerts`` module manages and interacts with various resources such as UI components and databases. It also manages the state of these resources and handles any potential concurrency issues. For example, the ``EmailAlertViewModel.kt`` class manages the email alert data in the database and ensures that the database operations are performed in the correct order.

The ``email_alerts`` module performs its duties by implementing methods that are called in response to user interactions or system events. It uses various algorithms and data structures to manage its state and perform its tasks. It also handles initialization and cleanup of resources, and handles exceptional conditions such as errors or unexpected inputs.

The ``email_alerts`` module provides a set of methods and properties that can be used by other classes or components. These methods and properties form the interface of the classes. For example, the ``EmailAlertViewModel.kt`` class provides a ``getAllEmailAlerts`` method that can be used to get all the email alerts.

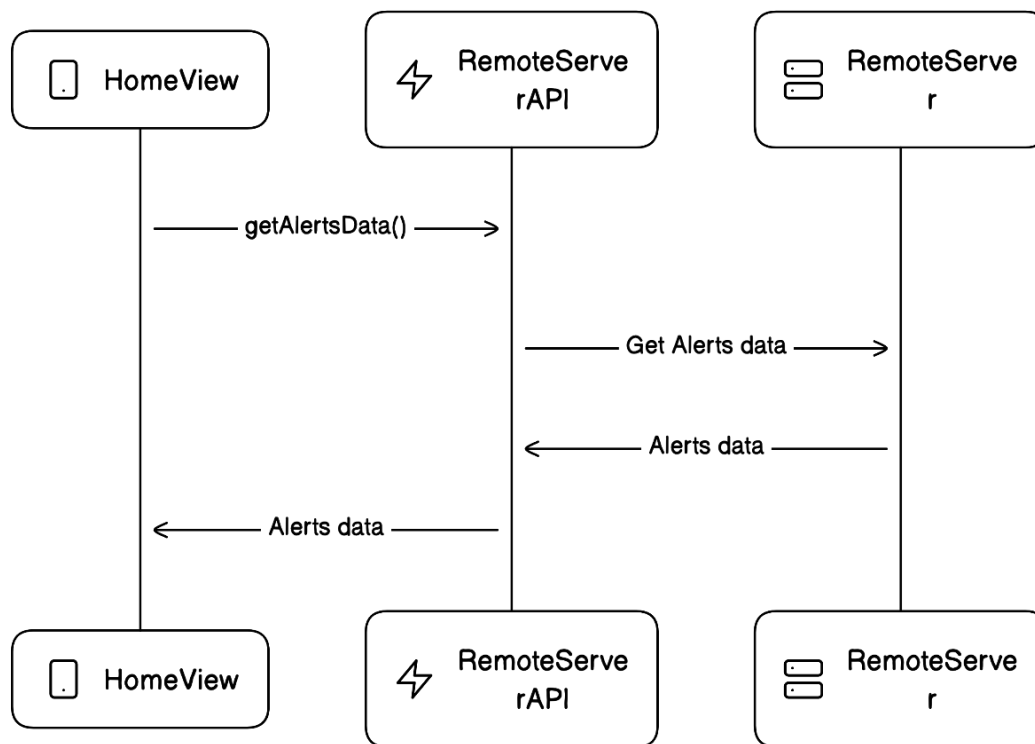


Figure 5.17 GetAlertsData Sequence Diagram show how the latest alerts are fetched

5.3 CLASS DIAGRAM

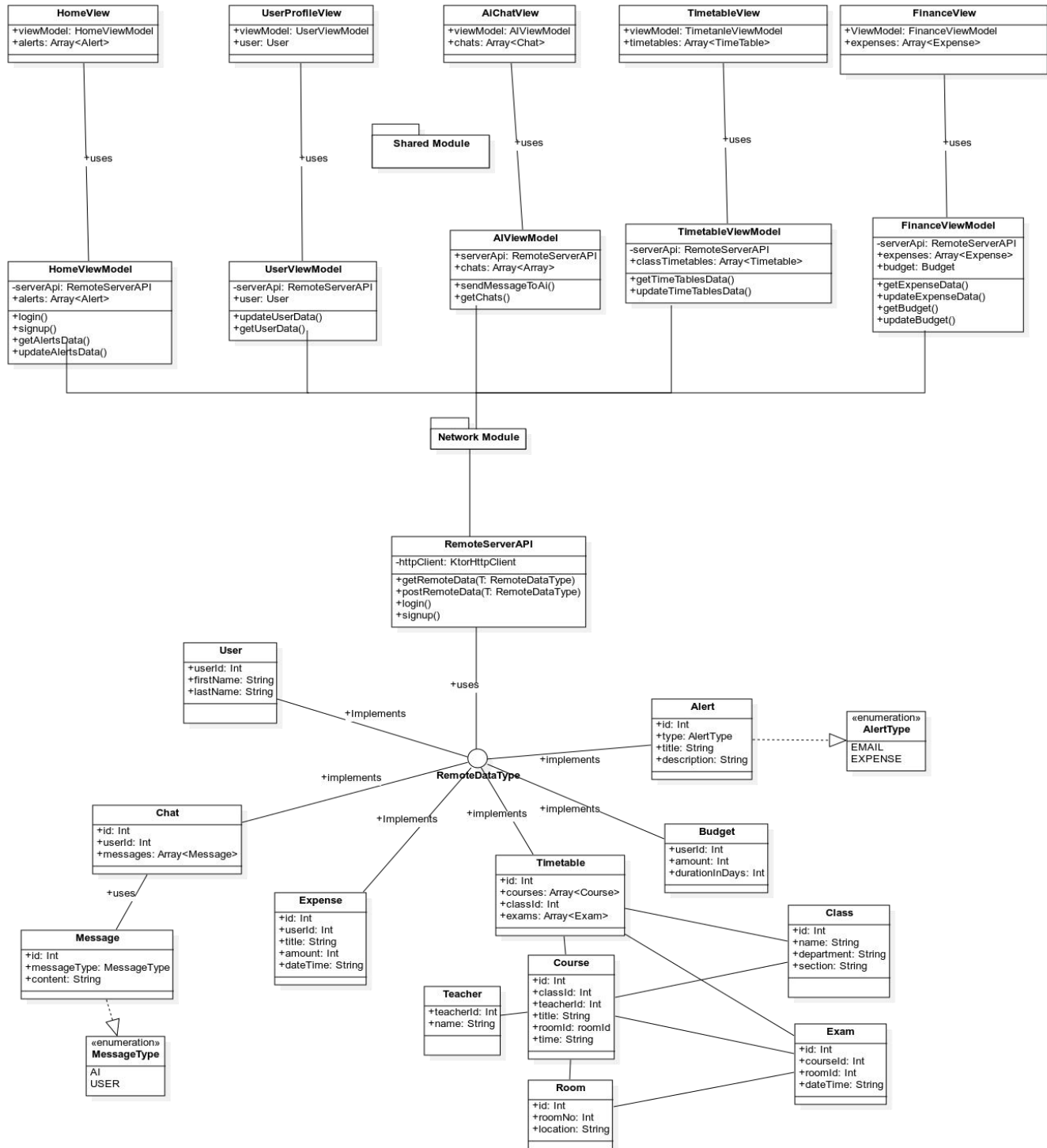


Figure 5.18 Student Assistant Class Diagram

5.4 Entity Relationship Diagram

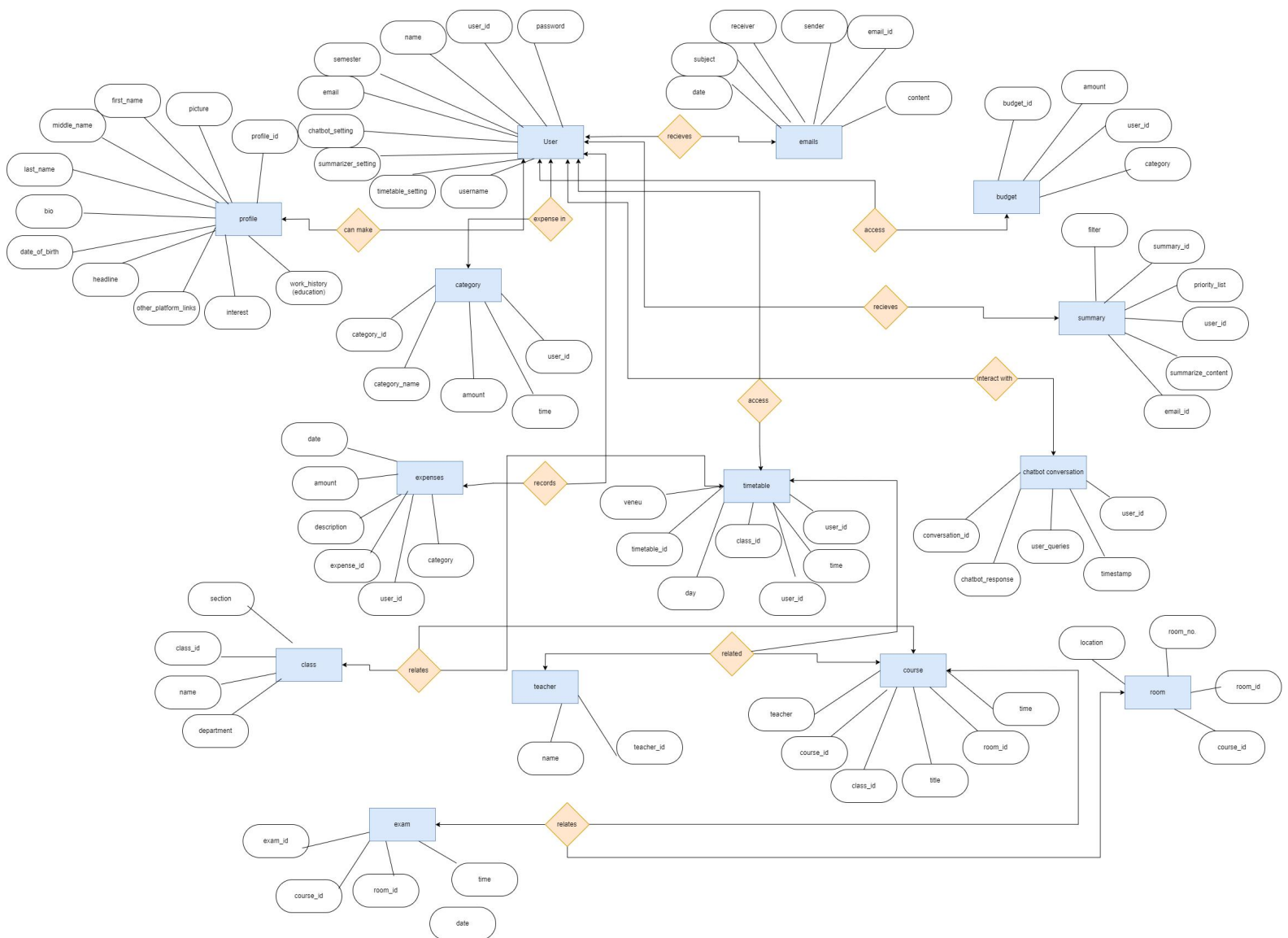


Figure 5.19 Student Assistant Entity Relationship Diagram

Chapter 6

IMPLEMENTATION AND TESTING

6.1 Application Scenario

Following module diagrams show the whole system functionality and the scenario of the application:

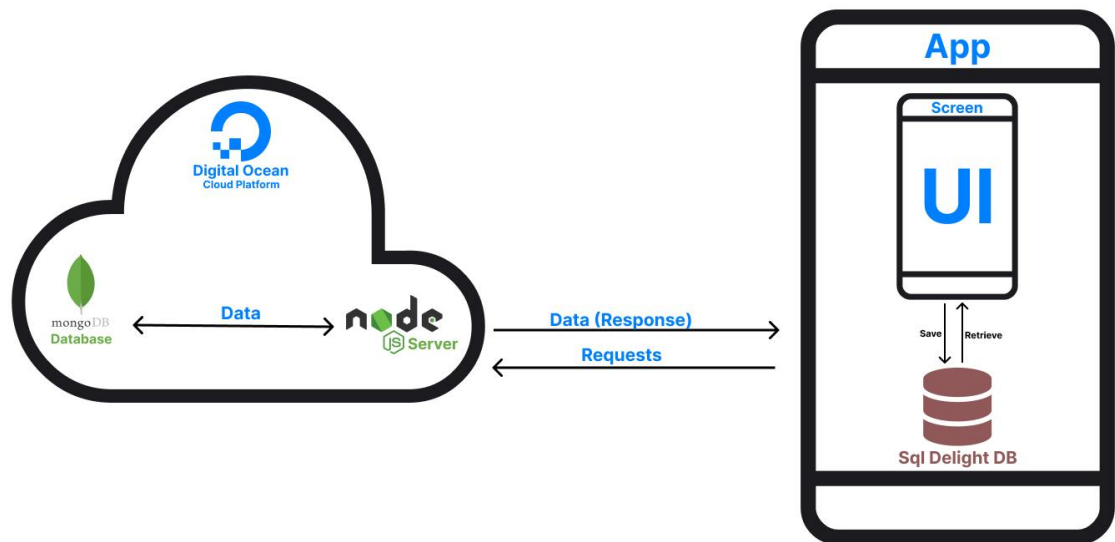
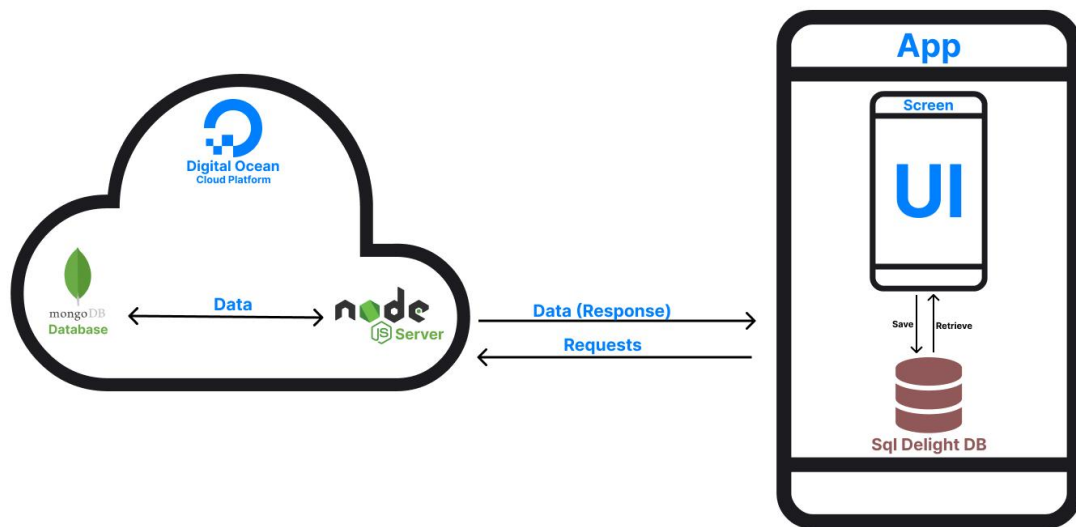


Figure 6.1 Timetable Screen Architecture Diagram



6.2 Finance Screen Architecture Diagram

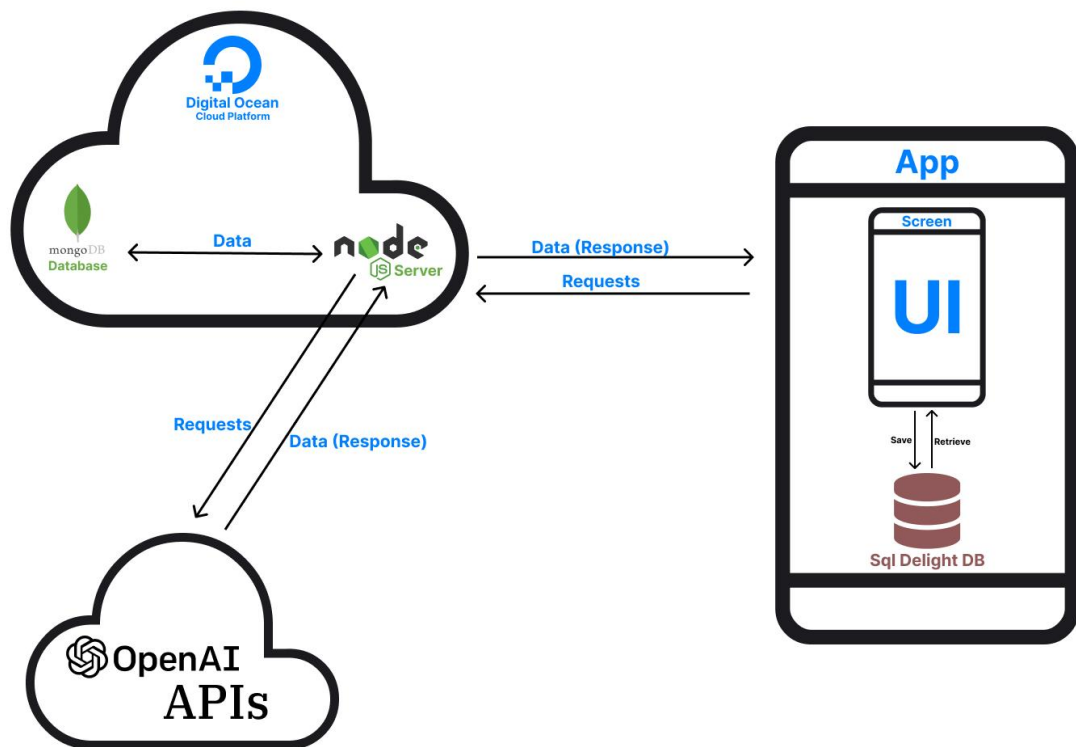


Figure 6.3 Chatbot Screen Architecture Diagram

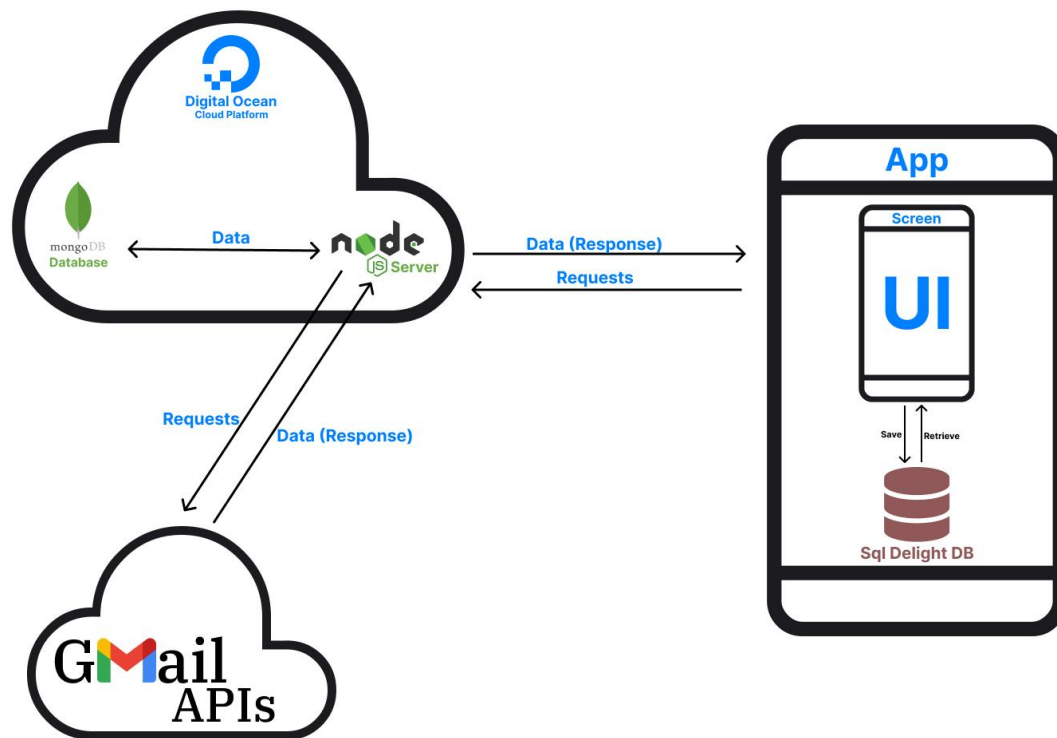


Figure 6.4 Email Alerts Screen Architecture Diagram

6.2 Development Process:

For the development of our student assistant application, we employed an Agile software development methodology, which facilitated iterative development and continuous feedback. We utilized Kotlin as the primary programming language due to its strong support for Android app development and interoperability with Java. Android Studio served as our integrated development environment (IDE), providing a comprehensive suite of tools for designing, coding, and debugging our application.

Throughout the development process, we adhered to best practices such as

modularization, encapsulation, and code re-usability to ensure maintainability and scalability. Version control was managed using Git, enabling collaboration among team members and facilitating code review processes. The communication was done through discord.

6.3 Core Functionalities:

The student assistant application encompasses a range of core functionalities designed to streamline various tasks and activities for university students. These functionalities include:

1. User registration and authentication: Allows students to create accounts and securely log in to the application.
2. Email management: Enables students to view, organize, and respond to emails directly within the application interface.
3. Financial tracker: Provides features for expense tracking, budget management, and generating financial reports.
4. Timetable management: Facilitates the organization of class schedules, exam dates, and assignment deadlines.
5. AI virtual assistant: Incorporates ChatGPT capabilities to provide personalized assistance and information retrieval services.

These functionalities are further tested in database using the dummy data and validated:

	id	email	alertType	alertValue	alertTime
1	1	test@email.com	Urgent	Today is Quantum Theory assignment's last date to submit.	2021-09-01
2	2	test@email.com	Normal	Tomorrow, Einstein will be heading the conference.	2021-09-02
3	3	test@email.com	Normal	Day after tomorrow, we have a quiz on Quantum Mechanics.	2021-09-03

Figure 6.5 EmailAlert Database Snapshot

	id	message	role
1	1	Hello	user
2	2	Hi	bot
3	3	How are you?	user
4	4	I'm fine, thank you	bot

Figure 6.6 ChatMessage Database Snapshot

	id	class	course	time	day	room	teacher	starred
1	1	Dummy Class	Nuclear Physic	9:00 - 10:00 AM	Monday	Room 101	Mr. John Doe	0
2	2	Dummy Class	Quantum Mechanics	10:00 - 11:00 AM	Tuesday	Room 102	Mr. John Doe	0

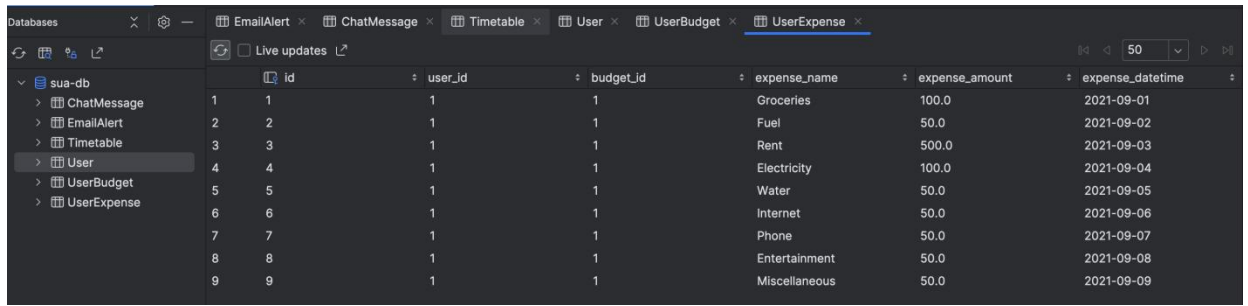
Figure 6.7 Timetable Database Snapshot

	id	name
1	0	John Doe

Figure 6.8 User Database Snapshot

	id	user_id	budget_type	budget_amount
1	0	1	FOOD	500.0

Figure 6.9 UserBudget Database Snapshot



id	user_id	budget_id	expense_name	expense_amount	expense_datetime
1	1	1	Groceries	100.0	2021-09-01
2	2	1	Fuel	50.0	2021-09-02
3	3	1	Rent	500.0	2021-09-03
4	4	1	Electricity	100.0	2021-09-04
5	5	1	Water	50.0	2021-09-05
6	6	1	Internet	50.0	2021-09-06
7	7	1	Phone	50.0	2021-09-07
8	8	1	Entertainment	50.0	2021-09-08
9	9	1	Miscellaneous	50.0	2021-09-09

Figure 6.10 UserExpense Database Snapshot

6.4 Controlled Libraries and Templates:

We used several controlled libraries and templates to expedite development and ensure consistency across the application. For user interface (UI) design, we utilized Material Design components provided by the Android Jetpack library. Additionally, we incorporated the SQLDelight Library for efficient data storage and management within the application's local database. More over, camelot library was used to extract timetable data from PDF files and organize it into tables.

6.5 Code Walkthroughs:

To maintain code quality and readability, we conducted code walkthroughs for critical components of the application. Notable areas of focus included the implementation of user authentication using SQLDelight for database management, integration of email management features using the Gmail API.

6.6 Testing Methodologies:

Our testing approach encompassed unit testing, integration testing, and end-to-end testing to ensure the reliability and functionality of the application. We employed JUnit and Mockito for unit testing, Espresso for UI testing, and Firebase Test Lab for device compatibility testing across multiple Android devices and versions.

6.7 Evaluation and Comparison:

The accuracy, performance, and scalability of the student assistant application were evaluated against the original specifications outlined in the requirements documentation. Quantitative metrics were used to measure response times, memory usage, and user satisfaction through user feedback and surveys. Overall, the application demonstrated robust performance and effectively addressed the identified problem statement, providing a valuable tool for university students to manage their academic and personal tasks efficiently.

Chapter 7

RESULTS AND DISCUSSION

7.1 Evaluation of the Solution

The evaluation of the student assistant application was conducted to assess its performance, functionality, and usability. The evaluation process involved both quantitative metrics and qualitative assessments.

7.1.1 Quantitative Metrics:

Accuracy: The accuracy of the application's core functionalities, such as user registration, email integration, financial tracking, and timetable management, was measured based on the correctness of results produced.

Performance: Performance metrics, including response time, latency, and system resource utilization, were monitored to gauge the efficiency of the application under various load conditions.

Scalability: The application's scalability was evaluated to determine its ability to handle an increasing number of users, data volumes, and concurrent transactions without compromising performance.

7.1.2 Qualitative Assessments:

Usability: User experience testing was conducted to assess the application's ease of

use, navigation, and intuitiveness. Feedback from users was collected to identify any usability issues or areas for improvement.

Reliability: The reliability of the application, including its stability, availability, and error-handling capabilities, was evaluated through rigorous testing scenarios and real-world usage simulations.

Functionality: The functionality of the application was assessed against the specified requirements and use cases to ensure that all intended features were implemented correctly and met user expectations.

7.2 Results

The application is built using the requirements from SRS and Design from SDS. The feedback of stakeholders was leveraged in incremental development of the application. The Dark Mode is also integrated to the application as per the requirements.

The following are the real-time snapshots of The Student Assistant Application:



Figure 7.1 Student Assistant Landing Screen

Figure 7.1 shows the home landing screen that shall be shown when the application starts. The upper left corner includes text to show that we are on home screen. The quote card includes text with quote of famous people. There are four buttons in the lower middle of the screen that navigate us to the other part of the Student Assistant Application.

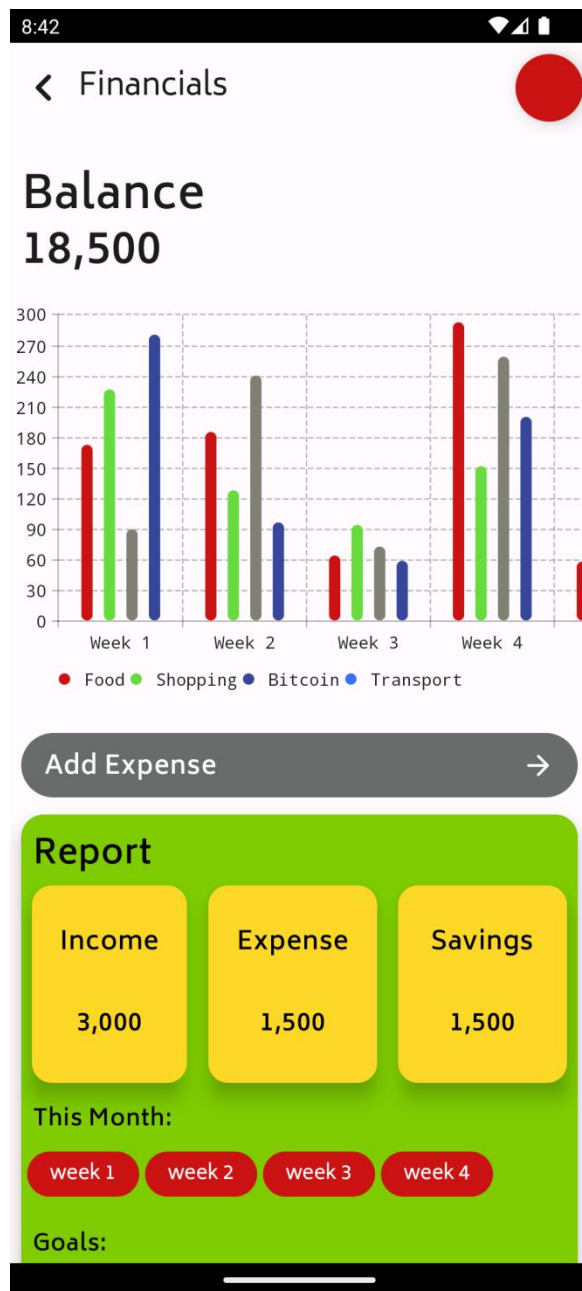


Figure 7.2 Student Assistant Financial Screen

When the Financials button is pressed shown in Figure 7.1, the financials screen in Figure 7.2 is opened that includes monthly chart of spending, the button to add expense, and report that shows income, expense, savings and then the buttons to navigate to the detailed financial screen of particular week shown in Figure 7.3.



Figure 7.3 Student Assistant Financial Detail Screen

The Financial Detail Screen (Figure 7.3) shows the expense rate of the particular week categorized into Food, Shopping, Transport, and others.

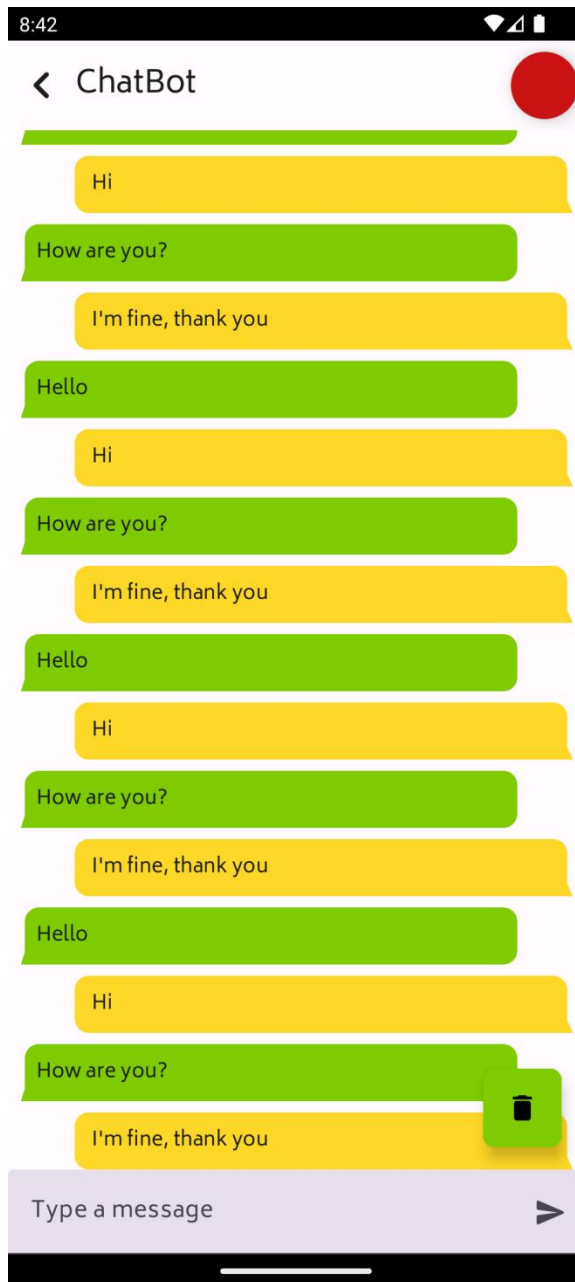


Figure 7.4 Student Assistant AI Chatbot Screen

Tapping into AI Button shown in Figure 7.1 will redirect the user to the AI Chatbot Screen (Figure 7.4) that is used to chat with the AI Assistant Chatbot provided by the Student Assistant Application.

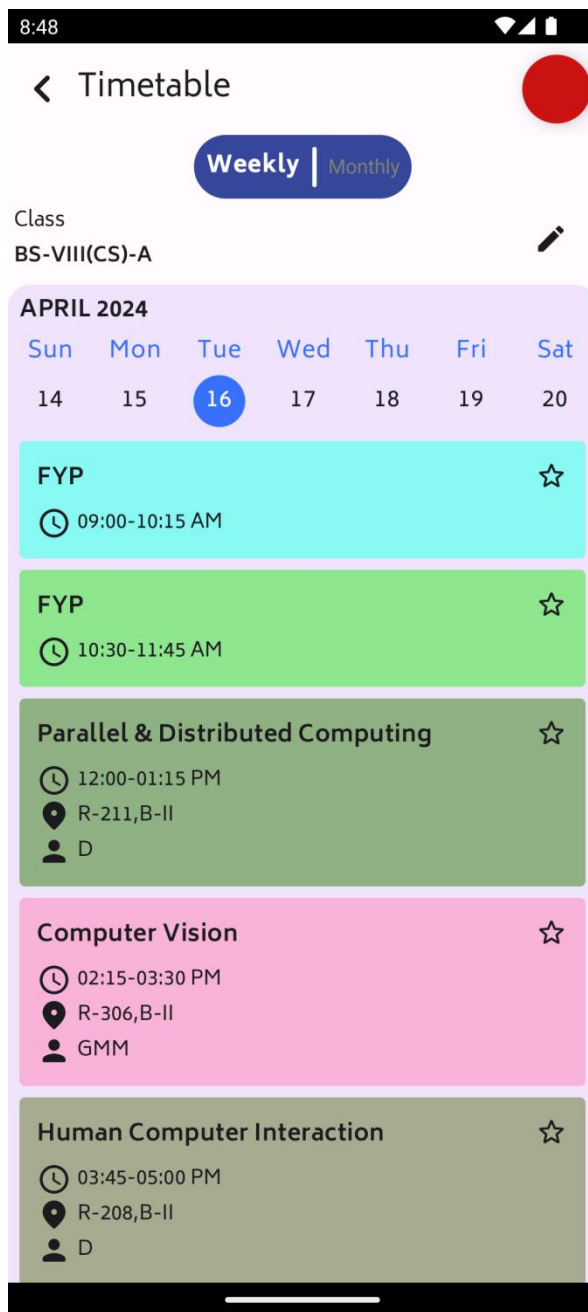


Figure 7.5 Student Assistant Timetable Screen

Tapping into Timetable Button in Figure 7.1 will navigate user to the Timetable screen depicted in Figure 7.5. This Screen will show timetable to the user with particular timing, room, and instructor in a much more easier format.

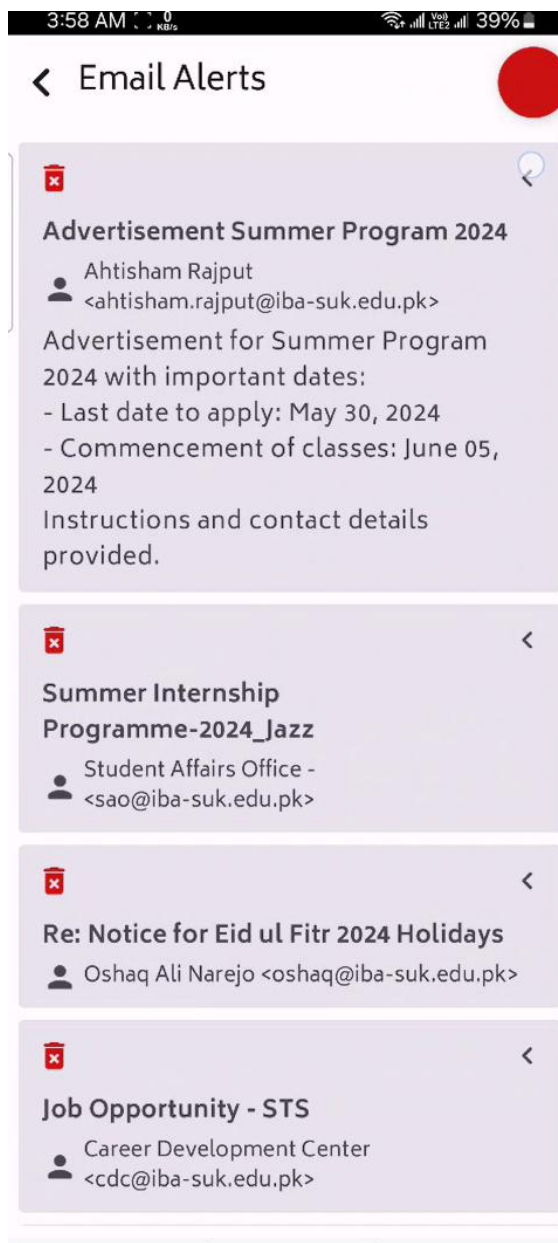


Figure 7.6 Student Assistant Email Alerts Screen

When the Email Button is tapped in Figure 7.1, the screen shown in Figure 7.6 is opened that shows the inbox of the user's emails. These Emails are summarized to show only useful information and it may also have links.

7.3 Discussion of Results

7.3.1 Performance Evaluation:

The application demonstrated satisfactory performance across all core functionalities, with response times consistently meeting the defined requirements. Scalability testing revealed that the application was capable of accommodating a growing user base and data volume without significant degradation in performance. Overall, the performance evaluation indicated that the application was robust and capable of handling the demands of a large-scale deployment.

7.3.2 Usability Assessment:

User feedback highlighted the application's intuitive interface and user-friendly design, contributing to a positive user experience. Usability testing identified minor usability issues, such as confusing navigation paths or unclear instructions, which were addressed through iterative design improvements.

7.3.3 Functionality Validation:

The functionality of the application was validated against the requirements specification, with all specified features implemented and functioning as intended. System testing confirmed the reliability of core functionalities, with minimal defects or issues encountered during testing.

7.4 System Testing

System testing was conducted to validate the functionality, reliability, and performance of the student assistant application. A comprehensive test suite covering all use cases and scenarios was executed, including:

7.4.1 Unit testing:

Testing individual components and modules in isolation to ensure their correctness and functionality.

Integration testing: Verifying the interaction and interoperability of different system components.

7.4.2 User acceptance testing (UAT):

Involving end users to validate the application's usability, functionality, and overall satisfaction.

Performance testing: Assessing the application's performance under varying load conditions to identify bottlenecks and optimize resource utilization.

The results of system testing confirmed the robustness and reliability of the application, with minimal defects identified and addressed during the testing process.

Chapter 8

CONCLUSION AND FUTURE WORK

Our solution, the Student Assistant Application, was developed to address the challenges faced by students in managing their academic and personal tasks effectively. By providing features such as email management, financial tracking, timetable management, and AI virtual assistant capabilities, our application aims to streamline various aspects of student life and enhance productivity.

Throughout the development process, we remained focused on addressing the key pain points identified in the problem statement. These included difficulties in organizing emails, tracking expenses, managing timetables, and accessing relevant information efficiently. Our solution was designed to alleviate these challenges by offering intuitive features and functionalities tailored to the needs of students.

To assess the effectiveness of our solution, we conducted comprehensive evaluations covering various aspects of its performance and usability. These evaluations included both qualitative and quantitative assessments, as well as user feedback sessions.

Quantitative metrics such as system response time, error rates, and task completion rates were measured to evaluate the performance of different features. Qualitative assessments involved gathering feedback from users through interviews, and usability tests were conducted.

Ultimately, the evaluations provided valuable insights into the strengths and

weaknesses of our solution, helping us identify areas for improvement and optimization.

Based on the findings from our evaluations, we offer the following recommendations for enhancing the Student Assistant Application:

1. **Improve Email Management:** Enhance the email management feature to provide more advanced filtering and sorting options, as well as integration with additional email providers.
2. **Enhance AI Virtual Assistant:** Further develop the AI virtual assistant capabilities to provide more personalized and context-aware assistance to users.
3. **Expand Financial Tracking:** Introduce new functionalities for financial tracking, such as budget forecasting and expense categorization, to provide users with more comprehensive financial management tools.
4. **Enhance Usability:** Implement user interface improvements and optimizations to enhance the overall usability and user experience of the application.

Looking ahead, there are several avenues for future work and expansion of the Student Assistant Application:

1. **Integration with Educational Platforms:** Explore opportunities for integrating the application with existing educational platforms and systems to provide seamless access to academic resources and information.
2. **Collaboration with Institutions:** Partner with educational institutions to tailor the application to their specific needs and requirements, and to facilitate adoption among students.
3. **Introduction of New Features:** Continue to innovate and introduce new features and functionalities based on evolving student needs and technological advancements.

4. **Research and Development:** Invest in ongoing research and development efforts to stay abreast of emerging trends and technologies in the field of student management applications.

Appendix A

REFERENCES

Alwagdani, B., & Alomar, K. (2020). Increasing Student Engagement with Personalized Emails. *Computer and Information Science*, 13, 54.

Chan, S. F., Chau, A. W.-L., & Chan, K. Y.-K. (2012). Financial knowledge and aptitudes: impacts on college students' financial well-being. *College Student Journal*, 46(1), 114+.

<https://link.gale.com/apps/doc/A285532025/AONE?u=anon~94fa8177&sid=googleScholar&xid=0722ae2b>

Donahoe, B., Rickard, D., Holden, H., Blackwell, K., & Caukin, N. (2019). Using EdTech to enhance learning. *International Journal of the Whole Child*, 4(2), 57-63.

Farahdita Dyah Susanto and Ria Sandra Alimbudiono (2019). Budgeting Applicaton For Personal Financial Planning Among Students Majoring In Accounting. <https://doi.org/10.2991/piceeba2-18.2019.13>

Kanagaraj, E., Arshad, N. S., & Santiagoo, R. Development of Timetable-based University Academic Portal.

Lappalainen, Y., & Narayanan, N. (2023). Aisha: A Custom AI Library Chatbot Using the ChatGPT API. *Journal of Web Librarianship*, 17(3), 37–58. <https://doi.org/10.1080/19322909.2023.2221477>

Mahira Kirmani, Gagandeep Kaur, Mudasir Mohd (2023). ShortMail: An email summarizer system. Volume 17.

<https://doi.org/10.1016/j.simpa.2023.100543>.

Nwe Ni Aung and Hla Hla Mon (2020). Budgeting Habit Behavior of Undergraduate Students in Yangon University of Economics. *J. Myanmar Acad. Arts Sci.* 2020 Vol. XVIII. No.8.

Parfenova, A. and Romashova, S. (2020), "The role of procrastination in students' consumer behavior: Budget planning and impulse buying", *International Journal of Sociology and Social Policy*, Vol. 40 No. 1/2, pp. 133-144. <https://doi.org/10.1108/IJSSP-10-2019-0199>

Studo-Team. Studo University Student App. <https://studo.com/en>

Thobejane, K., & Fatoki, O. (2017). Budgeting and spending habits of university students in South Africa. *Gender and Behaviour*, 15(3), 9414-9423. <https://hdl.handle.net/10520/EJC-c38554d83>