**BudgetMate App**

**Final Year Project – Mid Report**

**Session 2022-2026**

**A 4th Year Student**

A project submitted in partial fulfilment of the

COMSATS University Degree

of

BSc. (Hons.)BS in Computer Science / Software Engineering (CUI)



Department of Computer Science

COMSATS University Islamabad, Lahore Campus

20 April 2025

# **Project Detail**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Type (Nature of project) | | | [ ✔️ ] **Development**  [ ] **Research** & **Development** | | |
| Area of specialization | | | Mobile Application Development | | |
| **Project Group Members** | | | | | |
| Sr.# | Reg. # | Student Name | | Email ID | \*Signature |
| (i) | SP22-BCS-122 | Maryam Tahir | | SP22-BCS-122@cuilahore.edu.pk | ***Maryam Tahir*** |
| (ii) |  | - | | - | - |
| (iii) |  | - | | - | - |

\*The candidates confirm that the work submitted is their own and appropriate credit has been given where reference has been made to the work of others

# **Plagiarism Free Certificate**

This is to certify that I am Maryam Tahir S/D/o Andleeb Ahmed Tahir, group leader of FYP under registration no CUI/SP22-BCS-122/LHR at Computer Science Department, COMSATS University Islamabad, Lahore Campus. I declare that my FYP report is checked by my supervisor and the similarity index is 11% that is less than 20%, an acceptable limit by HEC. Report is attached herewith as Appendix A.

Date: 04/28/2025 Name of Group Leader: Maryam Tahir Signature: ***Maryam Tahir***

Name of Supervisor: Mahwish Waqas Co-Supervisor (if any):\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Designation: Lecturer Designation: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature: ***Mahwish Waqas*** Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

HoD: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Abstract

**BudgetMate** is a full-fledged mobile app that is designed to assist the user in managing their finances efficiently and accurately. The app encourages financial responsibility in this fast-paced era by enabling the user to record daily expenditures, allocate daily allowance, and monitor expenditure using easy-to-understand graphs and charts. BugdetMate prioritize confidentiality and authentication so that user’s sensitive information is protected. It gives charts and graphs from user’s data and helps identify their spendings and avoid overspending.

Moreover, **BudgetMate** allows for saving goal setting on intended purchases, promoting better financial habits. Having an easy-to-use interface and an emphasis on simplicity, BugdetMate provides smart money management to everyone.

# Acknowledgement

Table of Contents

[**Project Detail** 2](#_Toc216024681)

[**Plagiarism Free Certificate** 2](#_Toc216024682)

[Abstract 3](#_Toc216024683)

[Acknowledgement 4](#_Toc216024684)

[Chapter 01: Introduction 10](#_Toc216024685)

[1.1 Introduction 10](#_Toc216024686)

[1.2 Problem Statement 10](#_Toc216024687)

[1.3 Proposed Solution 10](#_Toc216024688)

[1.4 Main Objectives 10](#_Toc216024689)

[1.5 Assumptions and Constraints 11](#_Toc216024690)

[1. 6 Project Scope 11](#_Toc216024691)

[1. 7 11](#_Toc216024692)

[Chapter 2: Requirements Analysis 11](#_Toc216024693)

[2.1 Literature review / Existing system study 11](#_Toc216024694)

[2.2 Stakeholders List (Actors) 13](#_Toc216024695)

[2.3 Requirements elicitation 13](#_Toc216024696)

[2.3.1 Functional Requirements 13](#_Toc216024697)

[2.3.2 Non-Functional Requirements 17](#_Toc216024698)

[2.3.3 Requirements Traceability Matrix 20](#_Toc216024699)

[2.4 Use Case Descriptions 21](#_Toc216024700)

[Use Case: UC-01 – User Registration 21](#_Toc216024701)

[Use Case: UC-02 – User Login 23](#_Toc216024702)

[Use Case: UC-03 – Create and Manage Budget 24](#_Toc216024703)

[Use Case: UC-04 – Manage Expenses 25](#_Toc216024704)

[Use Case: UC-05 – Manage Income Sources 26](#_Toc216024705)

[Use Case: UC-06 – Set and Track Saving Goals 27](#_Toc216024706)

[Use Case: UC-07 – View Financial Summary/Reports 28](#_Toc216024707)

[Use Case: UC-08 – Manage Notifications and Alerts 29](#_Toc216024708)

[Use Case: UC-09 – Manage User Profile 30](#_Toc216024709)

[2.5 Use Case Design 31](#_Toc216024710)

[UC-01 – Register and Login 31](#_Toc216024711)

[UC-02 – Create and Manage Budget 31](#_Toc216024712)

[UC-03 – Manage Expenses 32](#_Toc216024713)

[UC-04 – Manage Income Sources 32](#_Toc216024714)

[UC-05 – Set and Track Saving Goals 33](#_Toc216024715)

[UC-06 – View Financial Summary/Reports 33](#_Toc216024716)

[UC-07 – Manage Notifications and Alerts 33](#_Toc216024717)

[UC-08 – Manage User Profile 34](#_Toc216024718)

[2.6 Software Development Life Cycle Model 35](#_Toc216024719)

[Chapter 03: System Design 36](#_Toc216024720)

[3.1 Work Breakdown Structure 36](#_Toc216024721)

[3.2 Activity Diagrams 38](#_Toc216024722)

[Activity Diagram ID #1 – User Registration 38](#_Toc216024723)

[Activity Diagram ID #2 – User Login 39](#_Toc216024724)

[Activity Diagram ID #3 – Create and Manage Budget 40](#_Toc216024725)

[Activity Diagram ID #4 – Manage Expenses 41](#_Toc216024726)

[Activity Diagram ID #5 – Manage Income Sources 42](#_Toc216024727)

[Activity Diagram ID #6 – Set and Track Saving Goals 43](#_Toc216024728)

[Activity Diagram ID #7 – View Financial Summary/Reports 44](#_Toc216024729)

[Activity Diagram ID #8 – Manage Notifications and Alerts 45](#_Toc216024730)

[Activity Diagram ID #9 – Manage User Profile 46](#_Toc216024731)

[3.3 Sequence Diagram 47](#_Toc216024732)

[Sequence Diagram #1 – Register and Login 47](#_Toc216024733)

[Sequence Diagram #2 – Create and Manage Budget 48](#_Toc216024734)

[Sequence Diagram #3 – Manage Expenses 49](#_Toc216024735)

[Sequence Diagram #4 – Manage Income Sources 49](#_Toc216024736)

[Sequence Diagram #5 – Set and Track Saving Goals 50](#_Toc216024737)

[Sequence Diagram #6 – View Financial Summary/Reports 51](#_Toc216024738)

[Sequence Diagram #7 – Manage Notifications and Alerts 52](#_Toc216024739)

[Sequence Diagram #8 – Manage User Profile 53](#_Toc216024740)

[3.4 Software Architecture 54](#_Toc216024741)

[3.5 Class Diagram 55](#_Toc216024742)

[3.6 Database Diagram 56](#_Toc216024743)

[3.7 Network Diagram / Gantt Chart 57](#_Toc216024744)

[3.8 Collaboration Diagram 58](#_Toc216024745)

[Collaboration Diagram #1: Register and Login 58](#_Toc216024746)

[Collaboration Diagram #2: Create and Manage Budget 59](#_Toc216024747)

[Collaboration Diagram #3: Manage Expenses 59](#_Toc216024748)

[Collaboration Diagram #4: Manage Income Sources 60](#_Toc216024749)

[Collaboration Diagram #5: Set and Track Saving Goals 60](#_Toc216024750)

[Collaboration Diagram #6: View Financial Summary/Reports 61](#_Toc216024751)

[Collaboration Diagram #7: Manage Notifications and Alerts 62](#_Toc216024752)

[Collaboration Diagram #8: Manage User Profile 62](#_Toc216024753)

[Chapter 4: System Testing 63](#_Toc216024754)

[4.1 Test Cases 63](#_Toc216024755)

[Test Case: TC-01 – User Registration 63](#_Toc216024756)

[Test Case: TC-02 – User Login 64](#_Toc216024757)

[Test Case: TC-03 – Dashboard and Budget Management 65](#_Toc216024758)

[Test Case: TC-04 – Expense (Daily) Tracking 66](#_Toc216024759)

[Test Case: TC-05 – Managing Income 67](#_Toc216024760)

[Test Case: TC-06 – Managing Savings Goals 68](#_Toc216024761)

[Test Case: TC-07 – Generating Reports and Analyzing Finances 69](#_Toc216024762)

[Test Case: TC-08 – Securing Data and Protecting Privacy 70](#_Toc216024763)

[Test Case: TC-09 – Edit Profile 71](#_Toc216024764)

[Test Case: TC-10 – Notifications on Budget Exceed 72](#_Toc216024765)

[4.2 Unit / Integration / Acceptance Testing 73](#_Toc216024766)

[Unit Testing 73](#_Toc216024767)

[Integration Testing 73](#_Toc216024768)

[Acceptance Testing 73](#_Toc216024769)

[Chapter 5: Conclusion 73](#_Toc216024770)

[5.1 Problems Faced and Lessons Learned 73](#_Toc216024771)

[5.2 Project Summary 74](#_Toc216024772)

[5.3 Future Work 74](#_Toc216024773)

[5.4 Screenshots 75](#_Toc216024774)

[**References** 79](#_Toc216024775)

List of Tables

[Table 1-User Authentication and Registration 13](#_Toc198245012)

[Table 2-Dashboard and Budget Management 14](#_Toc198245013)

[Table 3-Expense (Daily) Tracking 14](#_Toc198245014)

[Table 4-Managing Income 14](#_Toc198245015)

[Table 5-Managing Savings Goals 15](#_Toc198245016)

[Table 6-Generating Reports and Analyzing Finances 15](#_Toc198245017)

[Table 7-Securing Data and Protecting Privacy 16](#_Toc198245018)

[Table 8-Edit Profile 16](#_Toc198245019)

[Table 9-Performance Requirements 17](#_Toc198245020)

[Table 10-Scalability and Reliability 17](#_Toc198245021)

[Table 11-Usability and User Experience 17](#_Toc198245022)

[Table 12-Security and Data Privacy 18](#_Toc198245023)

[Table 13-Maintainability 18](#_Toc198245024)

[Table 14-Compatibility Requirements 18](#_Toc198245025)

[Table 15-Data Integrity and Backup 19](#_Toc198245026)

[Table 16- Requirements Traceability Matrix 20](#_Toc198245027)

[Table 17 - User Registration 21](#_Toc198245028)

[Table 18- User Login 22](#_Toc198245029)

[Table 19- Create and Manage Budget 23](#_Toc198245030)

[Table 20- Manage Expenses 24](#_Toc198245031)

[Table 21- Manage Income Sources 25](#_Toc198245032)

[Table 22- Set and Track Saving Goals 26](#_Toc198245033)

[Table 23- View Financial Summary/Reports 27](#_Toc198245034)

[Table 24- Manage Notifications and Alerts 28](#_Toc198245035)

[Table 25- Manage User Profile 29](#_Toc198245036)

[Table 26- User Registration 62](#_Toc198245037)

[Table 27- User Login 63](#_Toc198245038)

[Table 28- Dashboard and Budget Management 64](#_Toc198245039)

[Table 29- Expense (Daily) Tracking 65](#_Toc198245040)

[Table 30- Managing Income 66](#_Toc198245041)

[Table 31- Managing Savings Goals 67](#_Toc198245042)

[Table 32- Generating Reports and Analyzing Finances 68](#_Toc198245043)

[Table 33- Securing Data and Protecting Privacy 69](#_Toc198245044)

[Table 34- Edit Profile 70](#_Toc198245045)

[Table 35- Notifications on Budget Exceed 71](#_Toc198245046)

List of Figures

[Figure 1- Register and Login 30](#_Toc198245047)

[Figure 2- Create and Manage Budget 30](#_Toc198245048)

[Figure 3- Manage Expenses 31](#_Toc198245049)

[Figure 4- Manage Income Sources 31](#_Toc198245050)

[Figure 5- Set and Track Saving Goals 32](#_Toc198245051)

[Figure 6- View Financial Summary/Reports 32](#_Toc198245052)

[Figure 7- Manage Notifications and Alerts 32](#_Toc198245053)

[Figure 8- Manage User Profile 33](#_Toc198245054)

[Figure 9- Iterative and Incremental Model 34](#_Toc198245055)

[Figure 10- Work Breakdown Structure (a) 35](#_Toc198245056)

[Figure 11- Work Breakdown Structure (b) 36](#_Toc198245057)

[Figure 12- User Registration 37](#_Toc198245058)

[Figure 13- User Login 38](#_Toc198245059)

[Figure 14- Create and Manage Budget 39](#_Toc198245060)

[Figure 15- Manage Expenses 40](#_Toc198245061)

[Figure 16- Manage Income Sources 41](#_Toc198245062)

[Figure 17- Set and Track Saving Goals 42](#_Toc198245063)

[Figure 18- View Financial Summary/Reports 43](#_Toc198245064)

[Figure 19- Manage Notifications and Alerts 44](#_Toc198245065)

[Figure 20- Manage User Profile 45](#_Toc198245066)

[Figure 21- Register and Login 46](#_Toc198245067)

[Figure 22- Create and Manage Budget 47](#_Toc198245068)

[Figure 23- Manage Expenses 48](#_Toc198245069)

[Figure 24- Manage Income Sources 48](#_Toc198245070)

[Figure 25- Set and Track Saving Goals 49](#_Toc198245071)

[Figure 26- View Financial Summary/Reports 50](#_Toc198245072)

[Figure 27- Manage Notifications and Alerts 51](#_Toc198245073)

[Figure 28- Manage User Profile 52](#_Toc198245074)

[Figure 29- Software Architecture 53](#_Toc198245075)

[Figure 30- Class Diagram 54](#_Toc198245076)

[Figure 31- Database Diagram 55](#_Toc198245077)

[Figure 32- Network Diagram / Gantt Chart 56](#_Toc198245078)

[Figure 33- Register and Login 57](#_Toc198245079)

[Figure 34- Create and Manage Budget 58](#_Toc198245080)

[Figure 35- Manage Expenses 58](#_Toc198245081)

[Figure 36- Manage Income Sources 59](#_Toc198245082)

[Figure 37- Set and Track Saving Goals 59](#_Toc198245083)

[Figure 38- View Financial Summary/Reports 60](#_Toc198245084)

[Figure 39- Manage Notifications and Alerts 61](#_Toc198245085)

[Figure 40- Manage User Profile 61](#_Toc198245086)

# Chapter 01: Introduction

## 1.1 Introduction

The BudgetMate App mobile application assists the user in efficient management of personal finances. The application allows the user to track expenses on a daily basis, set budgets, monitor budgets, and view expenditures in the form of graphs and charts and become aware of their spending habits. This application comes with secure authentication and data encryption, which makes sure that it protects confidential data about its user. This tool gives its user the power to exercise financial discipline, avoid overspending and saving towards its goal.



## 1.2 Problem Statement

Most people find it difficult to manage their finances due to a lack of visibility and control over their daily expense. There is a need for an easy-to-use digital solution that helps users monitor, plan, and organize their financial habits. Without such tools, people often rely on memory or scattered records, which can lead to overspending, missed savings goals, and poor financial decisions. A mobile-based application that simplifies expense tracking, budgeting, and savings in one place can play a key role in helping users develop more mindful spending patterns.

## 1.3 Proposed Solution

## 1.4 Main Objectives

To achieve this aim, the project has the following specific objectives:

* Enables users to log and categorize daily expenses.
* Users must be able to control category budget and monitor budgets. It must also provide notifications and alerts when they exceed the budget.
* Develop a facility that enables individuals to set goals, such as saving money to purchase something. Include tools to determine how much they should save each day, week, or month.
* Use graphs and charts to assist users in analyzing spending patterns, budget usage, and goal progress.
* User information must be protected through secure authentication and encryption processes to maintain privacy and trustworthiness.
* This makes it a better interface for all types of users with a clear and easy-to-use interface.

## 1.5 Assumptions and Constraints

**Assumptions:**

* Users will have access to a smartphone and an active internet connection.
* Users will be willing to manually enter their financial data (expenses, income, etc.).
* Users are expected to have basic digital literacy to navigate the application interface.

**Constraints:**

* The app will not integrate with external banking systems or fetch transaction data automatically.
* The application is intended for individual use and does not support multi-user or shared accounts.
* The scope of the app is limited to personal finance management and does not cover investments or loans.

## 1. 6 Project Scope

**In Scope:**

Expense and income tracking: User will be able to log their income and expenses manually. Budget setup and category: User can create custom budget categories and set monthly limits. Monthly reports and visualizations: Graphs and charts will provide information on spending patterns. Goal Tracking: User can set savings goals (e.g., saving for a trip) and monitor progress.

**Out of Scope:**

Integration with bank accounts or credit cards: This system will not support automatic syncing with banks and cards. Investment portfolio management: the system will not handle investment such as stocks, mutual funds, or crypto.

## 1. 7 Software Development Life Cycle Model

For the development of the BudgetMate application, the **Iterative and Incremental Model** of Software Development Life Cycle (SDLC) is selected. This model is appropriate as it allows the project to be divided into small manageable modules, which were developed, tested, and refined through multiple iterations. Each iteration provides a working version of the app, incorporating feedback and improvements in subsequent cycles.

The reasons for selecting this model include:

* **Early Delivery of Core Features:** This basic functionality, such as user registration, login, and expense tracking, was delivered early in the development process to ensure quick usability.
* **Flexible design:** This model allowed the continuous improvement of the app ,models like budget creation, saving goals management, and financial reporting based on testing and stakeholder feedback.
* **Risk management:** By testing each model early, risks and criminal issues were identified and resolved before the final product was completed.
* **User Feedback Integration:** Feedback from prototype testing phase was incorporated into the shop to enhance usability and functionality, ensuring user satisfaction.
* **Continuous Testing and Validation:** Every iteration included through unit, integration, and acceptable testing to ensure stability and reliability.

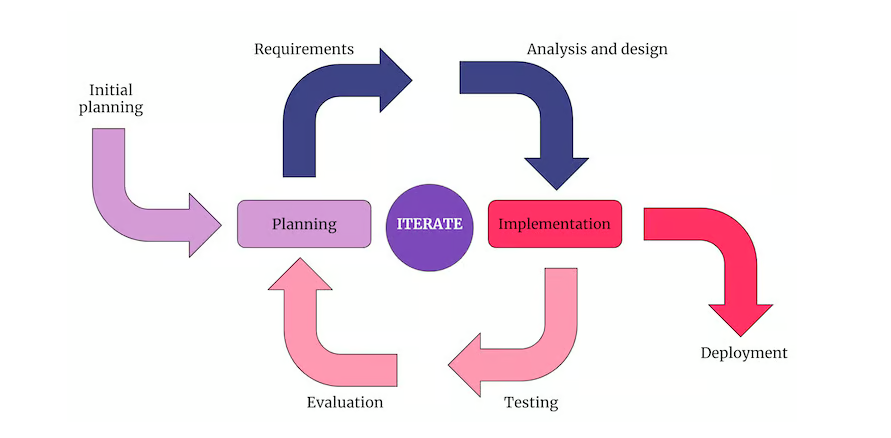


Figure 9- Iterative and Incremental Model

# Chapter 2: Requirements Analysis

## 2.1 Literature review / Existing system study

Several applications exist for financial tracking such as Quicken, Money Manager, and Wallet. These systems offer comprehensive features but often come with premium costs or a complex interface. BudgetMate aims to provide a fee, lightweight, and simplified alternative version.

Designed for local users with essential features and easy usability.

1. **Simplifi by Quicken**

Simplifi is a personal finance app that can be tailored to fit your needs. It connects to more than 14,000 banks and financial services to give users a complete view of their money. It also helps the user manage earnings, track bills, and save money effectively with its spending tool. The app further identifies recurrent payments or costs that the users might overlook and gives opportunities for their improvements in terms of finance. Its user-friendly interface and automation features make it a convenient tool for individuals seeking better control over their finances. This app stands out for its emphasis on real-time updates, customizable categorization of transactions, and holistic financial tracking. [1]

1. **Wallet: Budget and Money Manager**

Wallet is an application that aids users in the management of daily expenses, putting control limits and putting control measures towards achieving target budgets. The wallet provides a secure set of tools for users to achieve their financial objectives. The wallet facilitates the international currency base and maintains association with bank accounts to facilitate real-time active currency exchanges. Users also speak active role in setting up financial targets and can reach their savings goal and share accounts for budget planning or tracking finances. The app is intuitively simple and has everything needed to understand the finances throughout life without having to worry about being a financial expert. [2]

1. **Money Manager: Expense and Budget**

This app helps users track and manage their expenses in a budget friendly manner. Users can enter their expenditures in a day-to-day manner and subsequently classify them, making it easier for them to obtain comprehensive insights into their spending. Additionally, it offers a budgeting tool that allows the users to plan how much they intend to spend on the given categories. Notably, a unique double-entry accounting system has the capability to enhance tracking precision for keen users of the app. It also has different graphing tools that enable users to see their money management and spend less. [3]

1. **Money Keeper: Expense and Budget**

Money Keeper is one of the expense tracking and budgeting manager apps prepared for the user to ensure they are totally in control of their personal finances. It lets them use the app for setting up budgets, tracking expenses, and monitoring the flow of cash. Users can edit categories and use more than one account to track the income and expenses from cash as well as from accounts. Money Keeper also alerts users about upcoming bills so that they do not miss the due dates. The app provides visual reports and charts so that users understand the spending habits and make some good choices about saving money. Money Keeper is a special expense tracking and budgeting application that gives users strong control over their money. [4]

1. **Money Pocket: Expense and Budget**

Help user to create budgets, log their expenses, and check their cash flow easily. This is possible because of the customizable categories and multi-account features that allow tracking of income and expenditure from various sources: cash as well as different bank accounts. Last but not the least, the app provides reminder functions on upcoming bills, thus removing a chance that users forget about the payments. The visual reports and charts available within the app are also useful in portraying the spending so that it meets the user's own targets for savings. [5]

## 2.2 Stakeholders List (Actors)

**End User:**

Individuals utilize the BudgetMate system to monitor expenses, create budgets, and manage their financial goals.

**Administrator:**

Responsible for managing user accounts, maintaining system security, and ensuring smooth app functioning.

## 2.3 Requirements elicitation

### 2.3.1 Functional Requirements

#### 2.3.1.1 User Authentication and Registration

|  |  |
| --- | --- |
| **ID** | **Requirement Description** |
| FR 1.1 | User will be able to register with their email, username, and password. |
| FR 1.2 | User will be able to log in with their credentials (email & password). |
| FR 1.3 | The system will validate user credentials before granting access. |
| FR 1.4 | User will be able to reset their password through a "Forgot Password" feature. |

Table 1-User Authentication and Registration

#### 2.3.1.2 Dashboard and Budget Management

|  |  |
| --- | --- |
| **ID** | **Requirement Description** |
| FR 2.1 | User will be able to create multiple budgets (e.g., monthly, weekly, yearly). |
| FR 2.2 | User will be able to set budget limits for different categories (food, rent, shopping, etc.). |
| FR 2.3 | User will be able to see their remaining budget and spending analysis in a graphical format. |
| FR 2.4 | User will be shown pop-up messages when their spending crosses a predefined budget limit. |
| FR 2.5 | Users will be able to edit their active budgets. Completed budgets will not be deleted but instead saved or deleted for further access. |

Table 2-Dashboard and Budget Management

#### 2.3.1.3 Expense (Daily) Tracking

|  |  |
| --- | --- |
| **ID** | **Requirement Description** |
| FR 3.1 | User will be able to enter their expenses by entering details (amount, category, date, description). |
| FR 3.2 | User will be able to classify expenses under predefined or custom categories. |
| FR 3.3 | The system will produce real-time reports summarizing expenses by category. |
| FR 3.4 | The system will offer spending insights in graphical form using charts and graphs. |

Table 3-Expense (Daily) Tracking

#### 2.3.1.4 Managing Income

|  |  |
| --- | --- |
| **ID** | **Requirement Description** |
| FR 4.1 | User will be able to add multiple sources of income (salary, freelance work, investments, monthly salary). |
| FR 4.2 | The system will allow user to specify recurring income. |
| FR 4.3 | User will be able to track total earnings over a given period. |
| FR 4.4 | The system will produce a balance report (income vs. expenses). |

Table 4-Managing Income

#### 2.3.1.5 Managing Savings Goals

|  |  |
| --- | --- |
| **ID** | **Requirement Description** |
| FR 5.1 | User will be able to create savings goals (e.g., for a gadget, vacation, emergency fund). |
| FR 5.2 | User will input the target amount that they want to save. |
| FR 5.3 | User will specify a timeframe (weeks, months, or years) for achieving the goal. |
| FR 5.4 | The system will calculate and show the savings needed on a daily, weekly, or monthly basis. |
| FR 5.5 | Users will select an income source from which a percentage of their income will be allocated to their savings goal. |
| FR 5.6 | The system will provide a progress tracker and display a visual representation (e.g., a progress bar) of savings completed. |
| FR 5.7 | User can edit or achieve savings goals at any time in the system. |

Table 5-Managing Savings Goals

#### 2.3.1.6 Generating Reports and Analyzing Finances

|  |  |
| --- | --- |
| **ID** | **Requirement Description** |
| FR 6.1 | User will be able to produce financial reports for different timeframes (weekly, monthly, yearly). |
| FR 6.2 | The system will give graphical reports on income and expenses. |

Table 6-Generating Reports and Analyzing Finances

#### 2.3.1.7 Securing Data and Protecting Privacy

|  |  |
| --- | --- |
| **ID** | **Requirement Description** |
| FR 7.1 | The system will encrypt user sensitive information to ensure privacy. |
| FR 7.2 | The system will log and inform user about unauthorized login attempts. |

Table 7-Securing Data and Protecting Privacy

#### 2.3.1.8 Edit Profile

|  |  |
| --- | --- |
| **ID** | **Requirement Description** |
| FR 8.1 | User will be able to update and view their profile information (name, email, profile picture, etc.). |
| FR 8.2 | User will be able to reset their password by entering the existing and new password. |
| FR 8.3 | User will have the option to log out from their account securely. |
| FR 8.4 | User will be able to deactivate their account after confirmation. |
| FR 8.5 | User will be able to maintain their Wishlist, where they can add or delete items they plan to purchase later. |
| FR 8.6 | User will be able to set their preferred currency (e.g., PKR, USD, EUR, etc.) for financial tracking. |
| FR 8.7 | The system will save and show the selected currency format across all budget and expense-related calculations. |
| FR 8.8 | User will receive a confirmation prompt before making critical modifications such as resetting password or deleting the account. |

Table 8-Edit Profile

### 2.3.2 Non-Functional Requirements

#### 2.3.2.1 Performance Requirements

|  |  |
| --- | --- |
| **ID** | **Requirement Description** |
| NFR 1.1 | The system will respond to user actions within 2 seconds. |
| NFR 1.2 | The system will process expense logging, budget updates, and savings calculations efficiently without delay. |
| NFR 1.3 | The app will maintain optimal performance, even when processing a high number of financial transactions per user. |

Table 9-Performance Requirements

#### 2.3.2.2 Scalability and Reliability

|  |  |
| --- | --- |
| **ID** | **Requirement Description** |
| NFR 2.1 | The system will handle more users and financial transactions without a decline in performance. |
| NFR 2.2 | The system will maintain 99.9% uptime, ensuring continuous availability of financial data. |

Table 10-Scalability and Reliability

#### 2.3.2.3 Usability and User Experience

|  |  |
| --- | --- |
| **ID** | **Requirement Description** |
| NFR 3.1 | The system will have a user-friendly interface with simple navigation for all users. |
| NFR 3.2 | The system will provide clear graphical representations (graphs, charts) of spending and savings data. |
| NFR 3.3 | All features will be understandable by users within 5 minutes of using the system. |

Table 11-Usability and User Experience

#### 2.3.2.4 Security and Data Privacy

|  |  |
| --- | --- |
| **ID** | **Requirement Description** |
| NFR 4.1 | All sensitive financial information will be encrypted to avoid unauthorized access. |
| NFR 4.2 | Unauthorized access to user financial information will be strictly avoided. |

Table 12-Security and Data Privacy

#### 2.3.2.5 Maintainability

|  |  |
| --- | --- |
| **ID** | **Requirement Description** |
| NFR 5.1 | The system architecture will be modular and well-structured, so updates and feature addition can be easily done. |
| NFR 5.2 | The system will support integration of new features without interfering with existing functions. |
| NFR 5.3 | The system will give error logs so that developers can fix issues easily and quickly. |

Table 13-Maintainability

#### 2.3.2.6 Compatibility Requirements

|  |  |
| --- | --- |
| **ID** | **Requirement Description** |
| NFR 6.1 | The system will be compatible with Android and iOS. |
| NFR 6.2 | The system will be capable of supporting different screen sizes, including smartphones and tablets. |
| NFR 6.3 | The system will function properly on both Wi-Fi and mobile data. |

Table 14-Compatibility Requirements

#### 2.3.2.7 Data Integrity and Backup

|  |  |
| --- | --- |
| **ID** | **Requirement Description** |
| NFR 7.1 | The system will ensure data consistency, avoiding duplicate or loss of financial information by mistake. |
| NFR 7.2 | The system will back up the user’s financial information to a secure database. |

Table 15-Data Integrity and Backup

### 2.3.3 Requirements Traceability Matrix

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Associate ID** | **Requirement Description** | **Use Case ID** | **Test Case ID** | **Activity Diagram**  **ID** | **Sequence Diagram**  **ID** | **Collaboration Diagram ID** |
| 1 | FR-01 | User Registration and Login | UC-01, UC-02 | TC-01, TC-02 | AD#1, AD#2 | SD#1 | CD#1 |
| 2 | FR-02 | Dashboard and Budget Management | UC-03 | TC-03 | AD#3 | SD#2 | CD#2 |
| 3 | FR-03 | Daily Expense Tracking | UC-04 | TC-04 | AD#4 | SD#3 | CD#3 |
| 4 | FR-04 | Managing Income | UC-05 | TC-05 | AD#5 | SD#4 | CD#4 |
| 5 | FR-05 | Managing Saving Goals | UC-06 | TC-06 | AD#6 | SD#5 | CD#5 |
| 6 | FR-06 | Generating Reports and Financial Analysis | UC-07 | TC-07 | AD#7 | SD#6 | CD#6 |
| 7 | FR-07 | Securing Data and Protecting Privacy | UC-01, UC-02,  UC-09 | TC-02, TC-08, TC-09 | AD#1, AD#2, AD#9 | SD#1, SD#8 | CD#1, CD#8 |
| 8 | FR-08 | Profile Management and Settings | UC-09 | TC-09 | AD#9 | SD#8 | CD#8 |
| 9 | FR-02, FR-05 | Notification on Budget Exceed | UC-08 | TC-10 | AD#8 | SD#7 | CD#7 |

Table 16- Requirements Traceability Matrix

## 2.4 Use Case Descriptions

### Use Case: UC-01 – User Registration

|  |  |  |  |
| --- | --- | --- | --- |
| ID: | 001 | Name: | User Registration |
| Priority: | High | | |
| Actor: | User | | |
| Summary: | Allow a new user to register using valid email, password, confirm password and Email verification. | | |
| Pre-conditions: | User is not already registered. | | |
| Normal Flow of Path | | Alternative Path | |
| 1. User opens the registration screen. | | **If Email Verification link is not received:** | |
| 1. User enters a valid email and password (minimum 6 characters). | | 1. User clicks on “Resend Verification Email.” | |
| 1. System creates a user in Firebase and sends an email verification link to the user. | | 1. The system triggers Firebase to send a new email verification link to the registered email. | |
| 1. User opens their email and click the verification link. | | 1. User clicks on the new verification link to complete verification. | |
| 1. The system verifies email through Firebase authentication. | |  | |
| 1. User returns to the app, enters email and password on the Login screen, and logs in. | |  | |
| 1. The system checks verification status and logs in to the user if the email is verified. | |  | |
| Exceptions/Alerts: | | | |
| 1. If the email is already registered, the system shows an error. 2. If the email/password format is invalid, the system prompts the user to correct it. 3. If user tries to log in without verification, the system shows alert message. | | | |
| Post-conditions: | | | |
| **Step number** | | **Description** | |
| 1. | | User account is created and redirected to login. | |
| Cross References: | | | |
| Includes: | | Email verification | |
| Extends: | | Two-Factor Authentication | |

Table 17 - User Registration

### Use Case: UC-02 – User Login

|  |  |  |  |
| --- | --- | --- | --- |
| ID: | 002 | Name: | User Login |
| Priority: | High | | |
| Actor: | User | | |
| Summary: | Allow a registered user to securely log in using credentials and Email verification. | | |
| Pre-conditions: | User has an active registered account. | | |
| Normal Flow of Path | | Alternative Path | |
| **Login:** | | **Forget Password Scenario:** | |
| 1. User opens the login screen. | | 1. User clicks “Forget Password.” | |
| 1. User enters registered email and correct password. | | 1. System prompts the user to enter their registered email. | |
| 1. The system verifies credentials via Firebase Authentication. | | 1. System sends a password reset email with a link to the provided email address. | |
| 1. If the email is verified and credentials are valid, the user is logged in and taken to the dashboard. | | 1. User opens the email and clicks reset link, which opens Firebase’s reset page. | |
|  | | 1. User sets a new password and logs in with updated credentials. | |
| Exceptions/Alerts: | | | |
| 1. Invalid credentials show an error message. 2. Email not registered. 3. Email not verified. 4. User does not receive reset email. | | | |
| Post-conditions: | | | |
| **Step number** | | **Description** | |
| 1. | | User is logged in and redirected to dashboard | |
| Cross References: | | | |
| Includes: | | Email verification | |
| Extends: | | Two-Factor Authentication | |

Table 18- User Login

### Use Case: UC-03 – Create and Manage Budget

|  |  |  |  |
| --- | --- | --- | --- |
| ID: | 003 | Name: | Create and Manage Budget |
| Priority: | High | | |
| Actor: | User | | |
| Summary: | Enables user to create monthly or custom budgets, define limits, and categorize expenses. | | |
| Pre-conditions: | The user must be logged in. | | |
| Normal Flow of Path | | Alternative Path | |
| 1. User navigates to the budget section. | | * User edits an existing budget or achieves it. | |
| 1. User select month or custom range. | | * System updates/ achieve the selected budget accordingly. | |
| 1. User sets spending limits and assigns categories (e.g., Food, Rent). | |  | |
| 1. The system saves budget details and provides visualization. | |  | |
| Exceptions/Alerts: | | | |
| 1. Invalid input (e.g., negative budget). 2. Budget already exists for selected period. | | | |
| Post-conditions: | | | |
| **Step number** | | **Description** | |
| 1. | | Budget is saved, updated, or achieved successfully. | |
| Cross References: | | | |
| Includes: | | View Dashboard | |
| Extends: | | Financial Reports | |

Table 19- Create and Manage Budget

### Use Case: UC-04 – Manage Expenses

|  |  |  |  |
| --- | --- | --- | --- |
| ID: | 004 | Name: | Manage Expenses |
| Priority: | High | | |
| Actor: | User | | |
| Summary: | Allow user to add, update, delete, and categorize expenses linked to a specific budget. | | |
| Pre-conditions: | * User must be logged in. * A budget must exist. | | |
| Normal Flow of Path | | Alternative Path | |
| 1. User navigates to an expense tracker. | | * User edits or achieve existing expense | |
| 1. User clicks “Add Expense” and fills details (amount, category, date). | | * The system reflects updated data in the budget summary. | |
| 1. Expense is added and shown in real-time charts. | |  | |
| Exceptions/Alerts: | | | |
| 1. Invalid amount or date. 2. Adding expense without an active budget. | | | |
| Post-conditions: | | | |
| **Step number** | | **Description** | |
| 1. | | Expense is stored and affects related budget category. | |
| Cross References: | | | |
| Includes: | | Create and Manage Budget | |
| Extends: | | Notifications and Reports | |

Table 20- Manage Expenses

### Use Case: UC-05 – Manage Income Sources

|  |  |  |  |
| --- | --- | --- | --- |
| ID: | 005 | Name: | Manage Income Sources |
| Priority: | Medium | | |
| Actor: | User | | |
| Summary: | User can record, update, and achieve their income streams (salary, freelance, etc.). | | |
| Pre-conditions: | The user must be logged in. | | |
| Normal Flow of Path | | Alternative Path | |
| 1. User access Income Source section. | | * User update or delete and income entry. | |
| 1. Clicks “Add Income,” provides source name, amount, and date. | |  | |
| 1. System stores the income record and links it to budget calculations. | |  | |
| Exceptions/Alerts: | | | |
| 1. Invalid income entry. 2. Duplicate income source name. | | | |
| Post-conditions: | | | |
| **Step number** | | **Description** | |
| 1. | | Income data is updated and affects saving/budget plans. | |
| Cross References: | | | |
| Includes: | | View Financial Reports | |
| Extends: | | Budget Calculation | |

Table 21- Manage Income Sources

### Use Case: UC-06 – Set and Track Saving Goals

|  |  |  |  |
| --- | --- | --- | --- |
| ID: | 006 | Name: | Set and Track Saving Goals |
| Priority: | Medium | | |
| Actor: | User | | |
| Summary: | Allows user to define savings target and track their progress over time. | | |
| Pre-conditions: | The user must be logged in. | | |
| Normal Flow of Path | | Alternative Path | |
| 1. User selects “Saving Goals” section. | | * User update a goal. | |
| 1. Enter goal name, target amount, and timeframe. | |  | |
| 1. System tracks contributions and visualizes progress. | |  | |
| Exceptions/Alerts: | | | |
| 1. Target date in the past. 2. Navigate goal amount. | | | |
| Post-conditions: | | | |
| **Step number** | | **Description** | |
| 1. | | Goals are tracked and progress updated in dashboard | |
| Cross References: | | | |
| Includes: | | Manage Income Sources | |
| Extends: | | Reports | |

Table 22- Set and Track Saving Goals

### Use Case: UC-07 – View Financial Summary/Reports

|  |  |  |  |
| --- | --- | --- | --- |
| ID: | 007 | Name: | View Financial Summary/Reports |
| Priority: | High | | |
| Actor: | User | | |
| Summary: | Generates detailed reports on income, expenses, budgets, and saving in graphs/charts | | |
| Pre-conditions: | * The user must be logged in. * Financial data must exist. | | |
| Normal Flow of Path | | Alternative Path | |
| 1. User selects “Reports” from the menu. | | * User filter data by category or date. | |
| 1. The system generates monthly/weekly/yearly summaries and visualizes with graph. | |  | |
| Exceptions/Alerts: | | | |
| 1. No financial data available. 2. Report generation failure. | | | |
| Post-conditions: | | | |
| **Step number** | | **Description** | |
| 1. | | Reports are displayed in-app through interactive graphs and summaries | |
| Cross References: | | | |
| Includes: | | Mange Expense, Manage Income | |
| Extends: | | Dashboard | |

Table 23- View Financial Summary/Reports

### Use Case: UC-08 – Manage Notifications and Alerts

|  |  |  |  |
| --- | --- | --- | --- |
| ID: | 008 | Name: | Manage Notifications and Alerts |
| Priority: | Low | | |
| Actor: | User | | |
| Summary: | User can set remainders for budget threshold, bill payments, or saving goals. | | |
| Pre-conditions: | The user must be logged in and have at least one active budget with a set limit. | | |
| Normal Flow of Path | | Alternative Path | |
| 1. User sets a budget with a spending limit. | | * Notification appears as a pop-up or in dashboard message inside the app. | |
| 1. User logs expense. | |  | |
| 1. The system checks if the budget exceeds the limit. | |  | |
| 1. Is exceeded, system shows an in-app alert | |  | |
| Exceptions/Alerts: | | | |
| 1. Notification may not appear due to app errors or if the user has dismissed previous alerts. | | | |
| Post-conditions: | | | |
| **Step number** | | **Description** | |
| 1. | | User is alerted in-app and can take corrective action such as editing the budget or stopping further spending. | |
| Cross References: | | | |
| Includes: | | Create and Manage Budget | |
| Extends: | | User Profile | |

Table 24- Manage Notifications and Alerts

### Use Case: UC-09 – Manage User Profile

|  |  |  |  |
| --- | --- | --- | --- |
| ID: | 009 | Name: | Manage User Profile |
| Priority: | Medium | | |
| Actor: | User | | |
| Summary: | Allows user to update personal information, security settings, and view activity logs. | | |
| Pre-conditions: | The user must be logged in. | | |
| Normal Flow of Path | | Alternative Path | |
| 1. User accesses profile section. | | * User review recent login activity or deactivate account. | |
| 1. Edit name, email, password, or security settings. | |  | |
| 1. Saves changes; system validates and updates profile. | |  | |
| Exceptions/Alerts: | | | |
| 1. Invalid data format. 2. Security setting conflict. | | | |
| Post-conditions: | | | |
| **Step number** | | **Description** | |
| 1. | | Profile is updated and saved and reflected in system. | |
| Cross References: | | | |
| Includes: | | Two-Factor Authentication | |
| Extends: | | Logout | |

Table 25- Manage User Profile

## 2.5 Use Case Design

### UC-01 – Register and Login

A diagram of a login and password recovery

AI-generated content may be incorrect.

Figure 1- Register and Login

### UC-02 – Create and Manage Budget

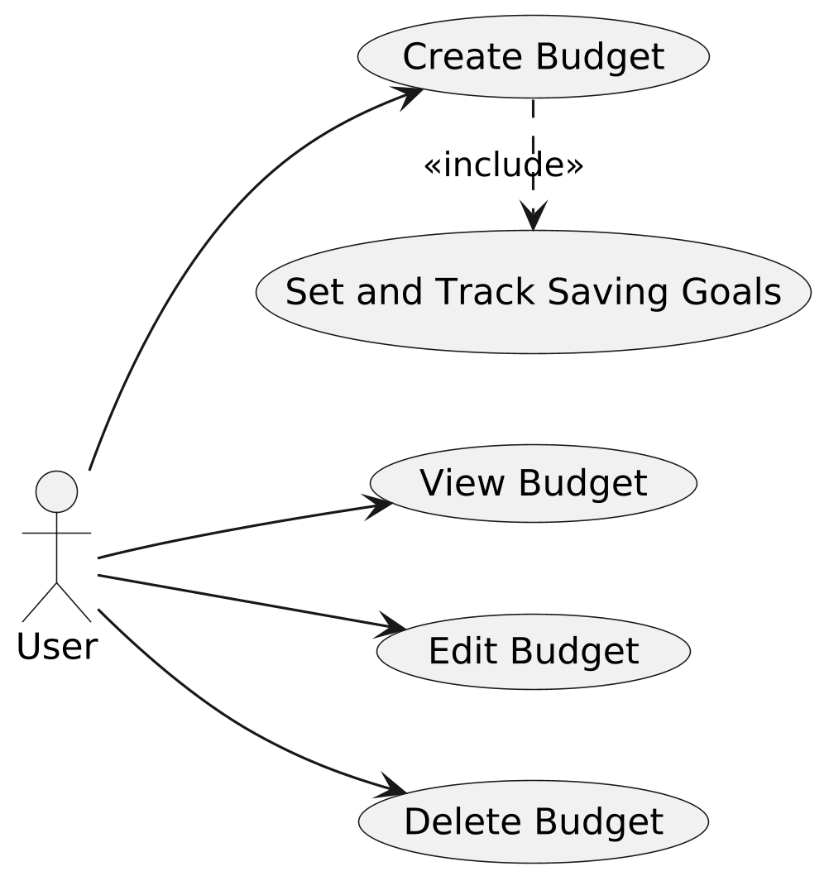


Figure 2- Create and Manage Budget

### UC-03 – Manage Expenses

A diagram of a company

AI-generated content may be incorrect.

Figure 3- Manage Expenses

### UC-04 – Manage Income Sources

A diagram of a flowchart

AI-generated content may be incorrect.

Figure 4- Manage Income Sources

### UC-05 – Set and Track Saving Goals

A diagram of a business process

AI-generated content may be incorrect.

Figure 5- Set and Track Saving Goals

### UC-06 – View Financial Summary/Reports

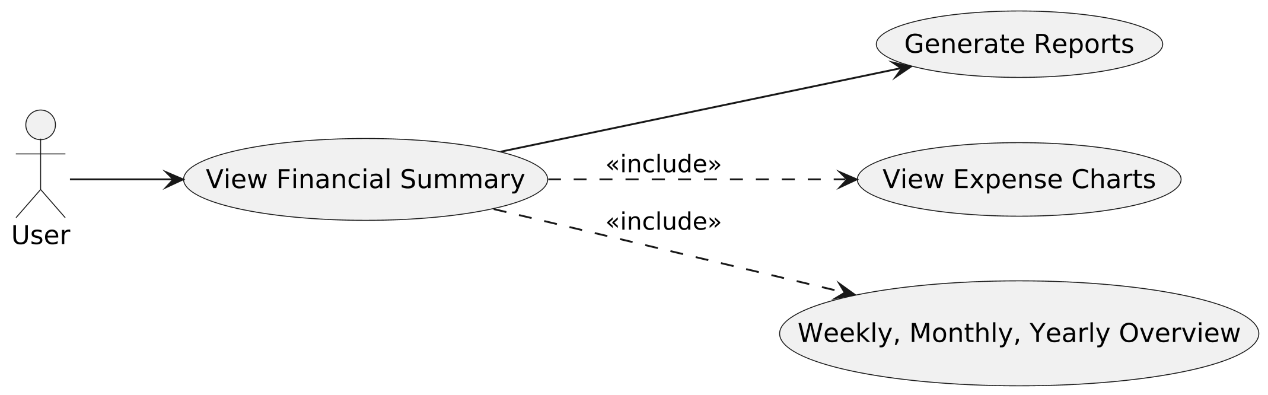


Figure 6- View Financial Summary/Reports

### UC-07 – Manage Notifications and Alerts

A diagram of a diagram

AI-generated content may be incorrect.

Figure 7- Manage Notifications and Alerts

### UC-08 – Manage User Profile



Figure 8- Manage User Profile

# Chapter 03: System Design

## 3.1 Work Breakdown Structure

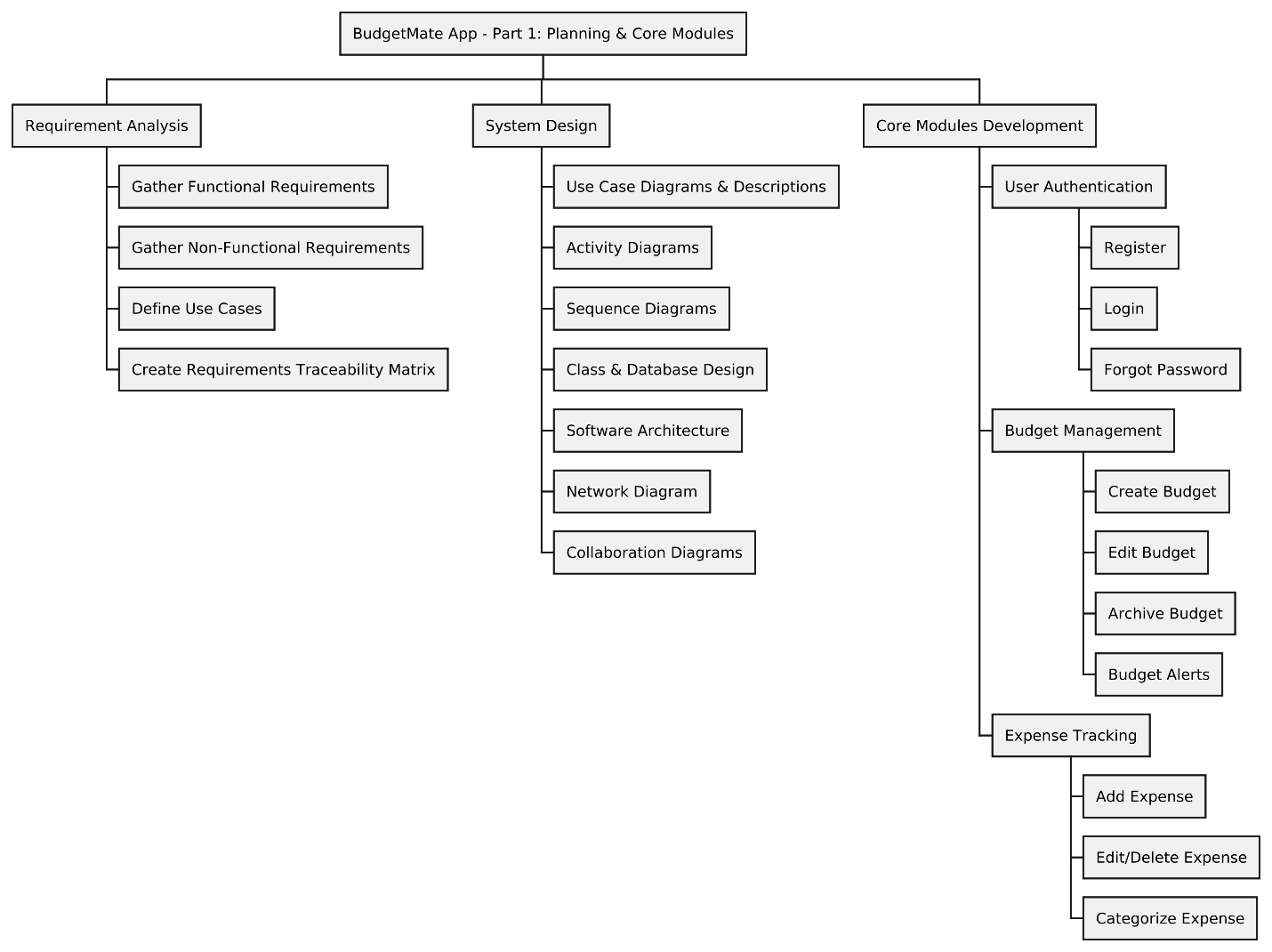


Figure 10- Work Breakdown Structure (a)

A diagram of a company

AI-generated content may be incorrect.

Figure 11- Work Breakdown Structure (b)

## 3.2 Activity Diagrams

### Activity Diagram ID #1 – User Registration

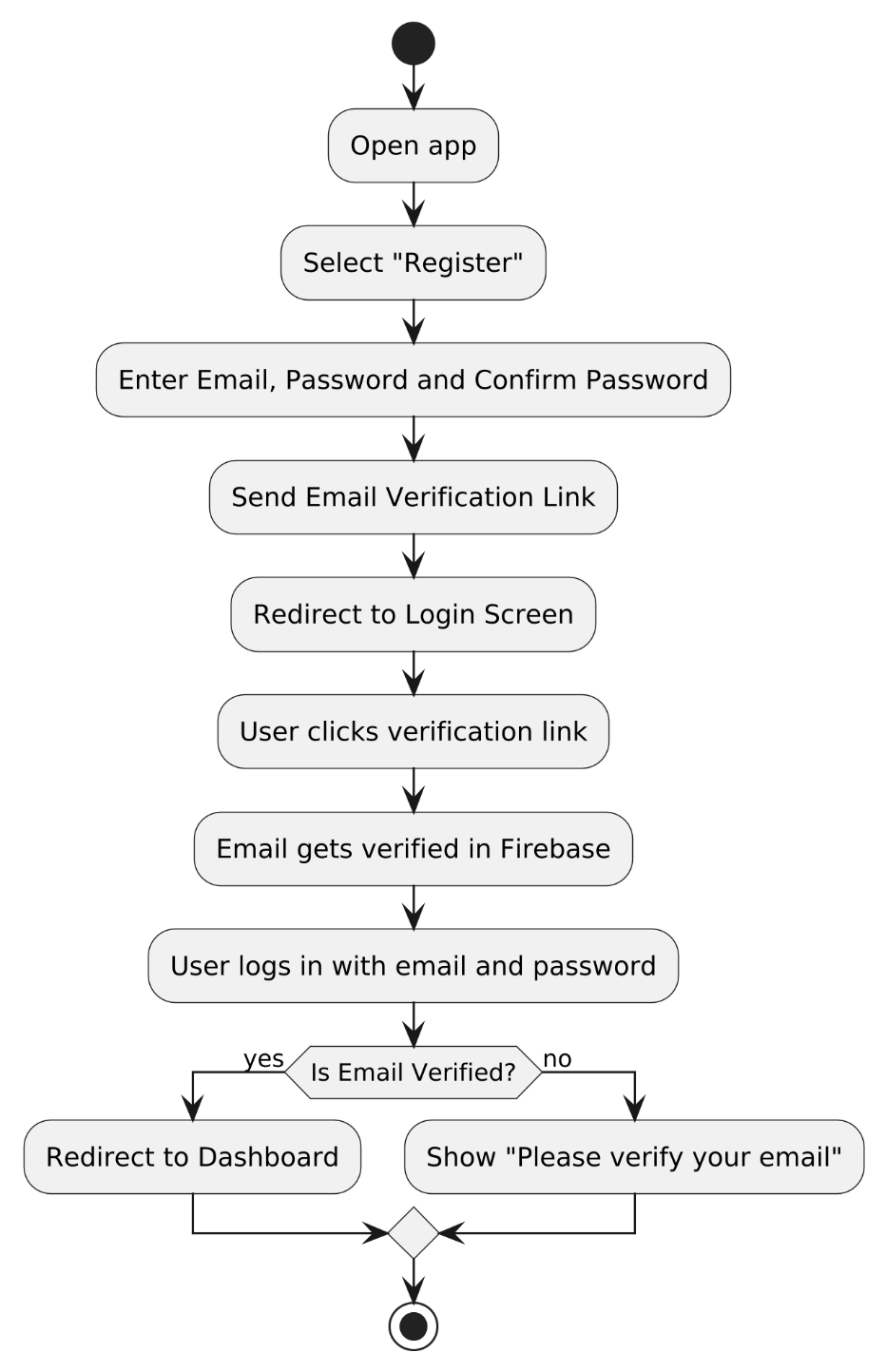


Figure 12- User Registration

### Activity Diagram ID #2 – User Login

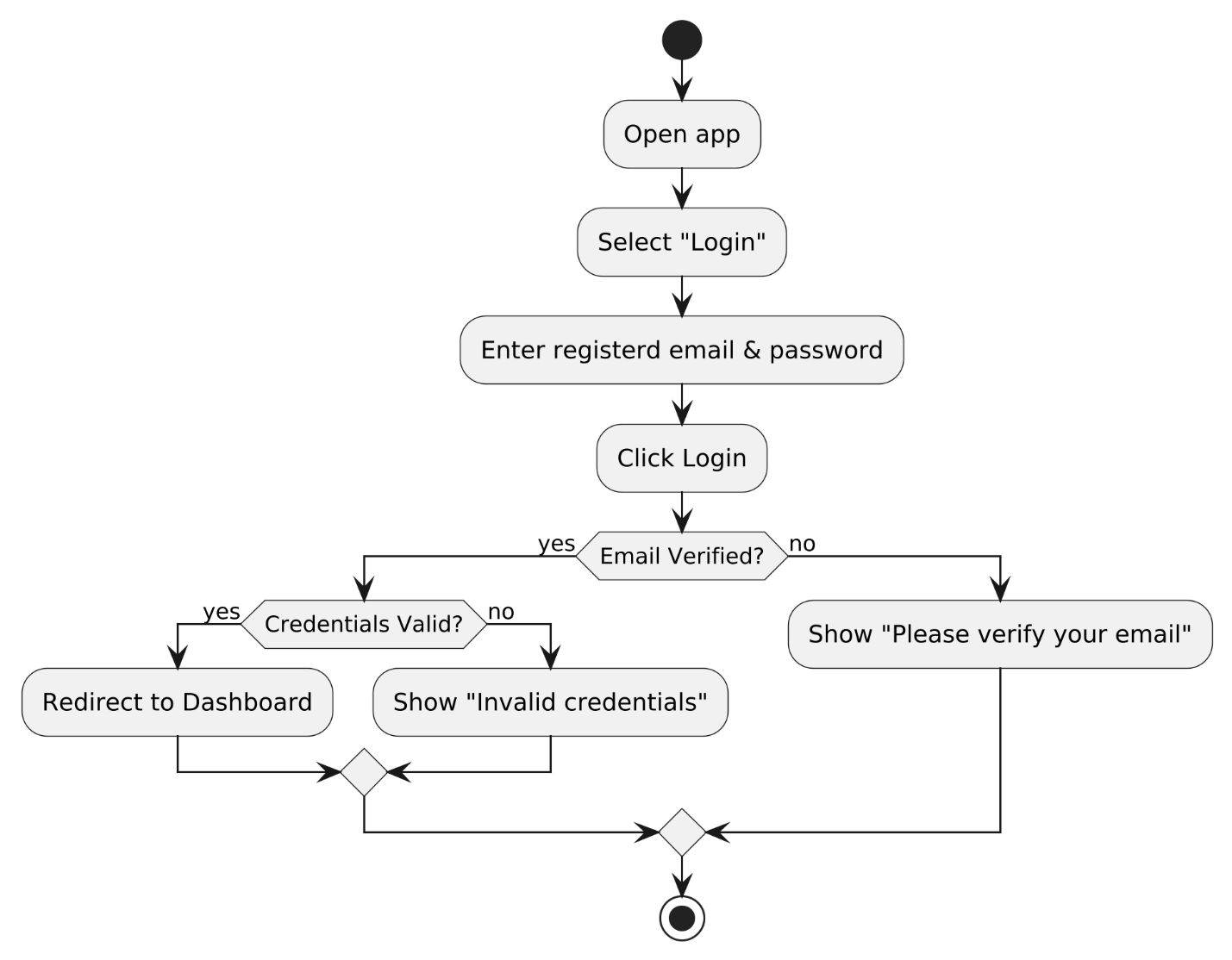


Figure 13- User Login

### Activity Diagram ID #3 – Create and Manage Budget

A diagram of a budget

AI-generated content may be incorrect.

Figure 14- Create and Manage Budget

### Activity Diagram ID #4 – Manage Expenses

A diagram of a flowchart

AI-generated content may be incorrect.

Figure 15- Manage Expenses

### Activity Diagram ID #5 – Manage Income Sources

A diagram of a software application

AI-generated content may be incorrect.

Figure 16- Manage Income Sources

### Activity Diagram ID #6 – Set and Track Saving Goals

A diagram of a goal

AI-generated content may be incorrect.

Figure 17- Set and Track Saving Goals

### Activity Diagram ID #7 – View Financial Summary/Reports

A diagram of a workflow

AI-generated content may be incorrect.

Figure 18- View Financial Summary/Reports

### Activity Diagram ID #8 – Manage Notifications and Alerts

A diagram of a workflow

AI-generated content may be incorrect.

Figure 19- Manage Notifications and Alerts

### Activity Diagram ID #9 – Manage User Profile

A diagram of a workflow

AI-generated content may be incorrect.

Figure 20- Manage User Profile

## 3.3 Sequence Diagram

### Sequence Diagram #1 – Register and Login

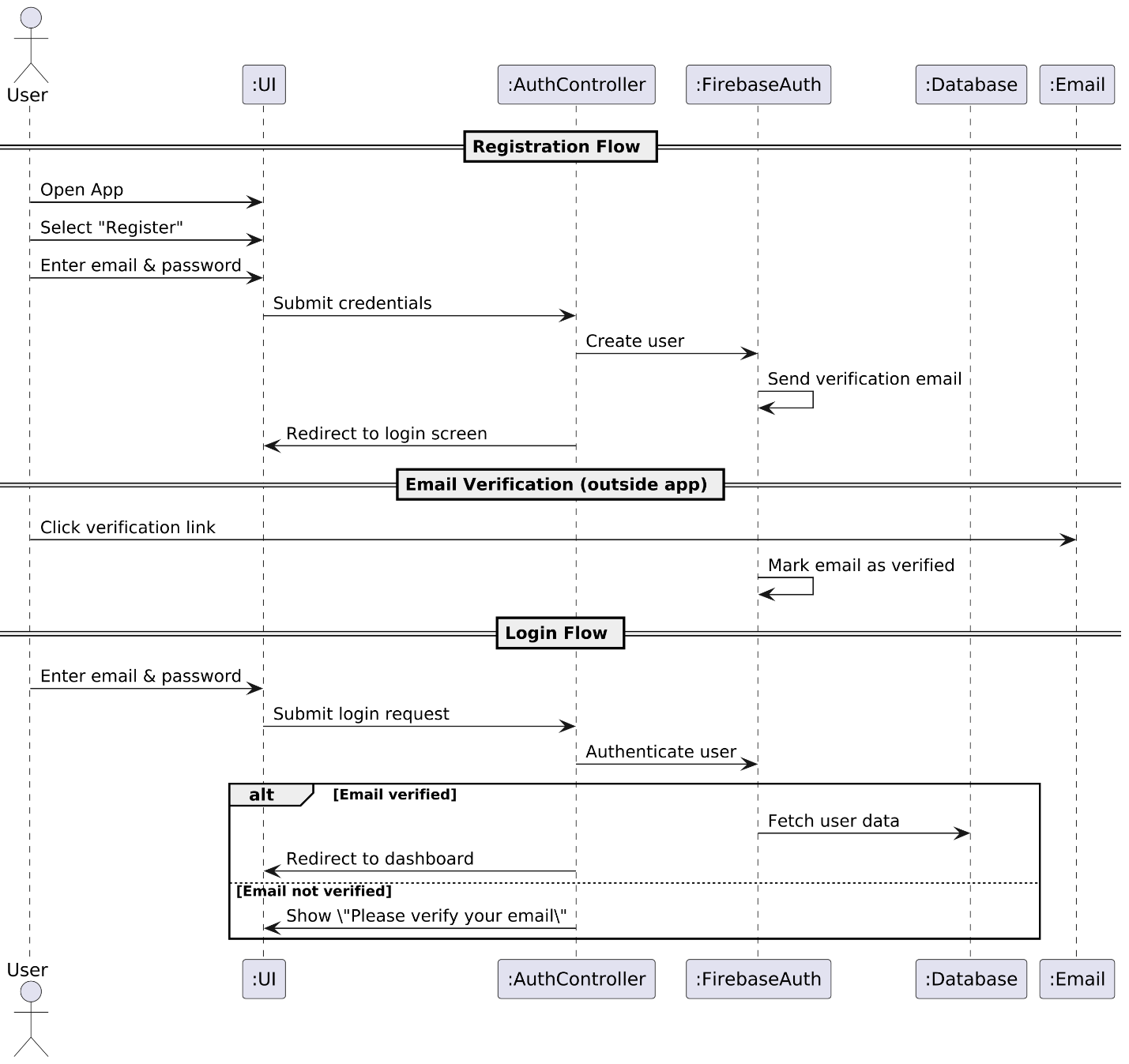


Figure 21- Register and Login

### Sequence Diagram #2 – Create and Manage Budget

A diagram of a project

AI-generated content may be incorrect.

Figure 22- Create and Manage Budget

### Sequence Diagram #3 – Manage Expenses

A diagram of a software system

AI-generated content may be incorrect.

Figure 23- Manage Expenses

### Sequence Diagram #4 – Manage Income Sources

A diagram of a software application

AI-generated content may be incorrect.

Figure 24- Manage Income Sources

### Sequence Diagram #5 – Set and Track Saving Goals

A diagram of a goal

AI-generated content may be incorrect.

Figure 25- Set and Track Saving Goals

### Sequence Diagram #6 – View Financial Summary/Reports

A diagram of a process

AI-generated content may be incorrect.

Figure 26- View Financial Summary/Reports

### 

### Sequence Diagram #7 – Manage Notifications and Alerts

A diagram of a flowchart

AI-generated content may be incorrect.

Figure 27- Manage Notifications and Alerts

### Sequence Diagram #8 – Manage User Profile

A diagram of a user interface

AI-generated content may be incorrect.

Figure 28- Manage User Profile

## 3.4 Software Architecture

The architecture of the BudgetMate application follows a modular, layered design, separating the user interface, business logic, and data management layers for maintainability and stability. The mobile frontend is developed using cross-platform frameworks, ensuring compatibility with both Android and IOS devices. It interacts with backend services via secure APIs to handle operations like authentication, expense tracker, budget creation and goal management. Data is stored in a structure related to database, designed to ensure consistency, security, and efficient retrieval. The architecture is designed to support future feature integration with minimal changes to the core system, promoting flexibility and long-term maintainability.

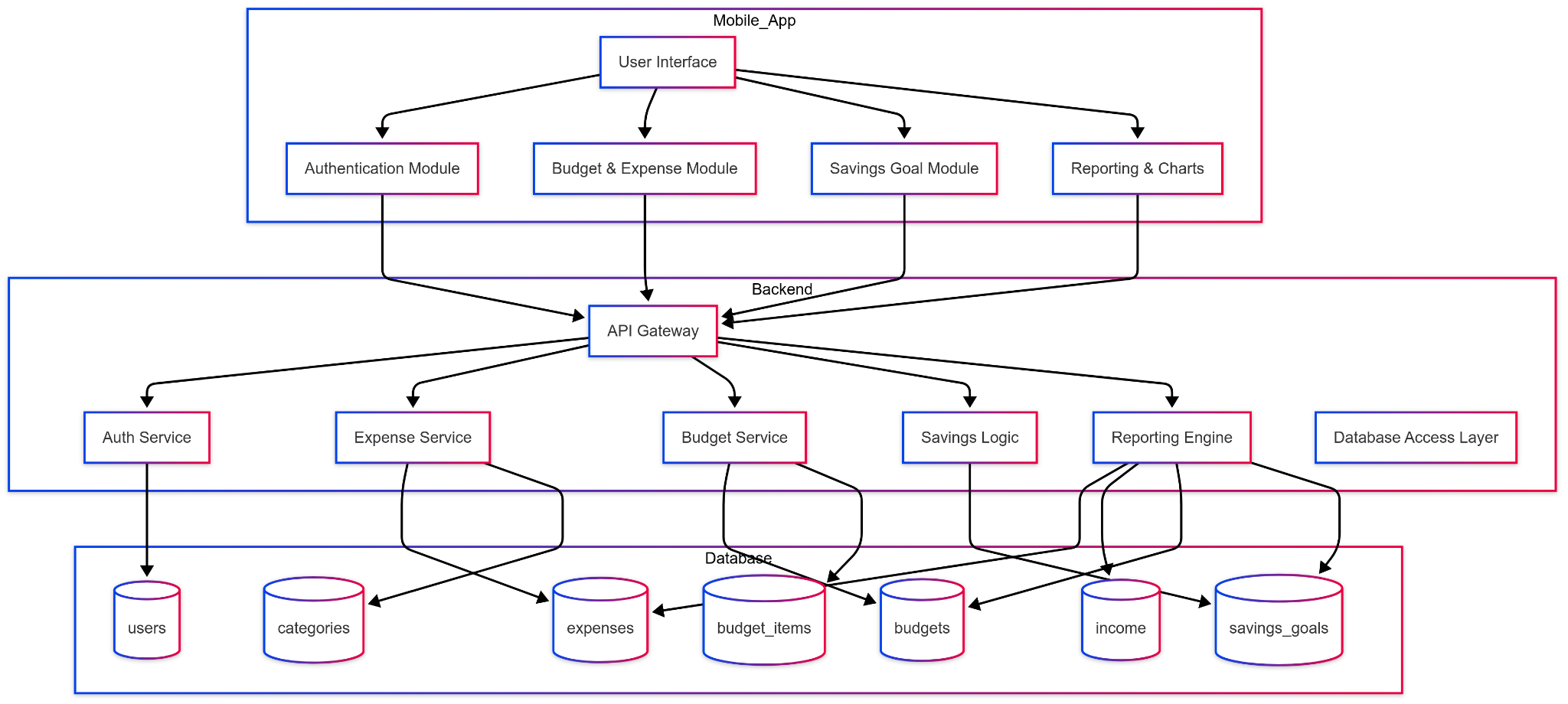


Figure 29- Software Architecture

## 3.5 Class Diagram

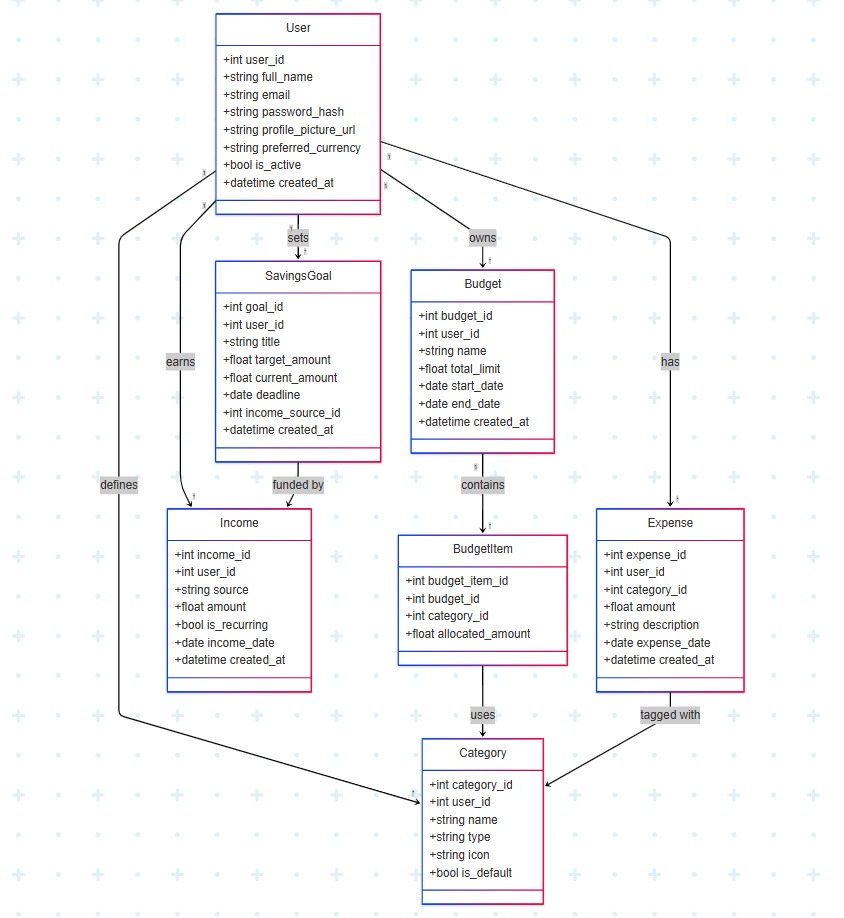


Figure 30- Class Diagram

## 3.6 Database Diagram

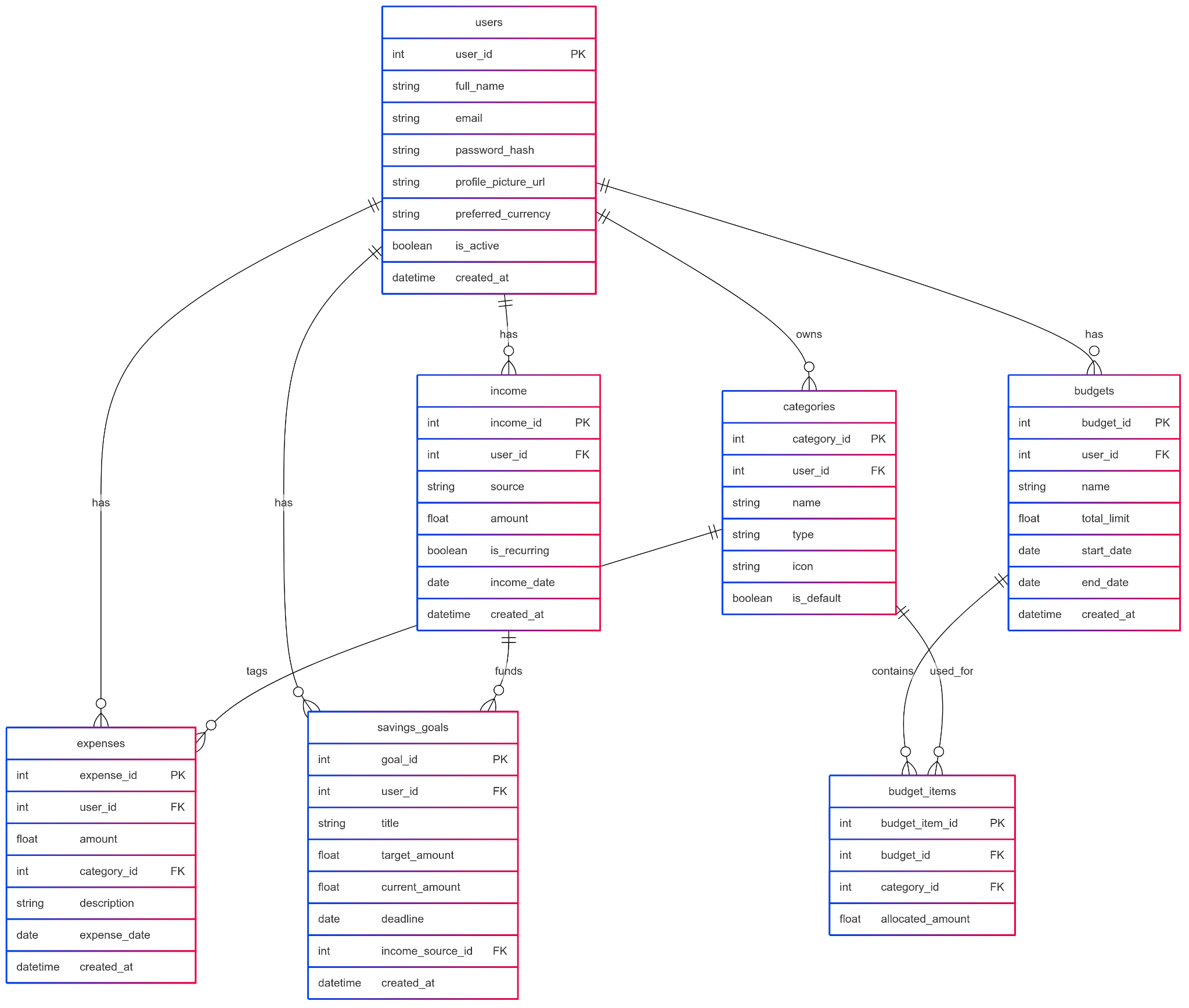


Figure 31- Database Diagram

## 

## 3.7 Network Diagram / Gantt Chart

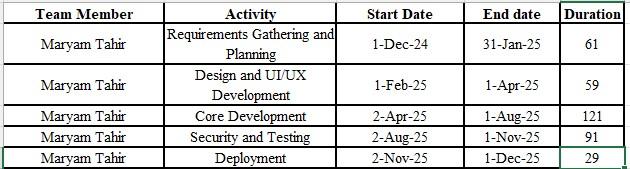




Figure 32- Network Diagram / Gantt Chart

## 3.8 Collaboration Diagram

### Collaboration Diagram #1: Register and Login

A diagram of a registration process

AI-generated content may be incorrect.

Figure 33- Register and Login

### Collaboration Diagram #2: Create and Manage Budget

A diagram of a diagram

AI-generated content may be incorrect.

Figure 34- Create and Manage Budget

### Collaboration Diagram #3: Manage Expenses

A diagram of a flowchart

AI-generated content may be incorrect.

Figure 35- Manage Expenses

### Collaboration Diagram #4: Manage Income Sources

A diagram of a software system

AI-generated content may be incorrect.

Figure 36- Manage Income Sources

### Collaboration Diagram #5: Set and Track Saving Goals

A diagram of a software development

AI-generated content may be incorrect.

Figure 37- Set and Track Saving Goals

### Collaboration Diagram #6: View Financial Summary/Reports

A diagram of a data flow

AI-generated content may be incorrect.

Figure 38- View Financial Summary/Reports

### Collaboration Diagram #7: Manage Notifications and Alerts

A diagram of a software system

AI-generated content may be incorrect.

Figure 39- Manage Notifications and Alerts

### Collaboration Diagram #8: Manage User Profile

A diagram of a process

AI-generated content may be incorrect.

Figure 40- Manage User Profile

# Chapter 4: System Testing

## 4.1 Test Cases

### Test Case: TC-01 – User Registration

|  |
| --- |
| **Test Case #:** 01 **Test Case Name:** User Registration  **System**: BudgetMate **Subsystem:** User  **Designed by:** Maryam Tahir **Design Date:** 11/05/2025  **Executed by:** **Execution Date:**  **Short Description:** Test the user registration  functionality with valid and invalid input data. |
| **Pre-Conditions** The application is launched, and the user is not already registered. |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Step** | **Action** | **Test Data** | **Expected System Response** | **Actual Response** | **Pass/Fail** | **Comments** |
| 1 | User opens the “Registration” screen. | N/A | System displays the registration form. | Registration form shown. |  |  |
| 2 | User enters valid email, password and confirms password. | Email: [user@example.com](mailto:user@example.com), Password: pass123 | System enables the “Register” button. | Register button is enabled. |  |  |
| 3 | User clicks “Register.” | Click on “Register” | System creates then account in Firebase and sends an email verification link. | Firebase email sent. |  |  |
| 4 | User opens email and clicks the verification link. | Clicked link from Gmail. | Firebase verifies the email. | Firebase shows “verified” |  |  |
| 5 | User logs in with email and password. | Entered same email and password. | System checks verification; if verified, redirects to dashboard. | Dashboard is shown. |  |  |

|  |
| --- |
| **Post-Conditions** |
| 1. User account is created successfully. 2. User is logged in and redirected to dashboard. |

Table 26- User Registration

### Test Case: TC-02 – User Login

|  |
| --- |
| **Test Case #:** 02 **Test Case Name:** User Login  **System**: BudgetMate **Subsystem:** User  **Designed by:** Maryam Tahir **Design Date:** 11/05/2025  **Executed by:** **Execution Date:**  **Short Description:** Test secure login with correct  credentials and Email verification. |
| **Pre-Conditions** User is already registered. |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Step** | **Action** | **Test Data** | **Expected System Response** | **Actual Response** | **Pass/Fail** | **Comments** |
| 1 | Open the “Login” screen. | N/A | Login fields for email and password are displayed. | Fields shown |  |  |
| 2 | Enter correct credentials | Valid email/password | System verifies credentials and check email verification status. | Dashboard shown. |  |  |
| 3 | Is verified, log in is successful. | Verified credentials. | User is redirected to dashboard. | Dashboard shown. |  |  |
| 4 | Enter invalid credentials or unverified email. | Invalid email/password | System displays error message. | Error shown. |  |  |
| 5 | Use “Forget Password.” | Registered email | System sends a password reset email with a link to the registered email address. | Reset email sent. |  |  |

|  |
| --- |
| **Post-Conditions** |
| 1. User is authenticated and taken to dashboard. 2. Errors are displayed for invalid credentials or unverified email. |

Table 27- User Login

### Test Case: TC-03 – Dashboard and Budget Management

|  |
| --- |
| **Test Case #:** 03 **Test Case Name:** Dashboard and  Budget Management  **System**: BudgetMate **Subsystem:** User  **Designed by:** Maryam Tahir **Design Date:** 11/05/2025  **Executed by:** **Execution Date:**  **Short Description:** Test functionality to create, edit,  and view budget with categories and limit and alerts. |
| **Pre-Conditions** User is logged in. |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Step** | **Action** | **Test Data** | **Expected System Response** | **Actual Response** | **Pass/Fail** | **Comments** |
| 1 | Navigate to Budget section. | N/A | Budget creation options are shown. | Options shown |  |  |
| 2 | Create a new budget with limits, date, and categories. | Rent: PKR 10,000 | System validates and saves budget. | Budget saved |  |  |
| 3 | View dashboard. | Budget Created | Budget summary and remaining amount shown. | Summary shown |  |  |
| 4 | Overspend a category. | Expense: PKR 11,000 (Rent) | System shows popup alerts. | Alert triggered |  |  |
| 5 | Edit or delete a budget. | Change limit to PKR 12,000 | System updates value or deletes the selected budget and displays updated graph. | Graph updated |  |  |

|  |
| --- |
| **Post-Conditions** |
| 1. Budget is created and update correctly and visible in dashboard. 2. Alerts are shown when limit are exceeded. |

Table 28- Dashboard and Budget Management

### Test Case: TC-04 – Expense (Daily) Tracking

|  |
| --- |
| **Test Case #:** 04 **Test Case Name:** Expense (Daily)  Tracking  **System**: BudgetMate **Subsystem:** User  **Designed by:** Maryam Tahir **Design Date:** 11/05/2025  **Executed by:** **Execution Date:**  **Short Description:** Test the ability to add, edit, and  delete expense records under categories. |
| **Pre-Conditions** User is logged in and has at least one active budget. |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Step** | **Action** | **Test data** | **Expected System Response** | **Actual Response** | **Pass/Fail** | **Comments** |
| 1 | Open “Expense Tracker.” | N/A | Expense form is displayed. | Form shown |  |  |
| 2 | Enter valid expense amount, date, and description. | PKR 500, 2025-05-01, Snacks | System validates fields and saves data. | Expense saved |  |  |
| 3 | Classify expense under a category. | Category: Food | Expense appears under correct category. | Categorized |  |  |
| 4 | View daily summary | Expanse logged | Real-time summary and graph shown. | Summary updated |  |  |
| 5 | Edit and delete expense. | Edit amount to PKR 450 | System updated or delete the expense. | Expense updated |  |  |

|  |
| --- |
| **Post-Conditions** |
| 1. Expense is updated in dashboard and linked to category. 2. Budget and dashboard are updated in real time. 3. Visual summaries reflect current data. |

Table 29- Expense (Daily) Tracking

### Test Case: TC-05 – Managing Income

|  |
| --- |
| **Test Case #:** 05 **Test Case Name:** Managing Income  **System**: BudgetMate **Subsystem:** User  **Designed by:** Maryam Tahir **Design Date:** 11/05/2025  **Executed by:** **Execution Date:**  **Short Description:** Test adding income from different  sources and tracking it. |
| **Pre-Conditions** Uset is logged in. |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Step** | **Action** | **Test Data** | **Expected System Response** | **Actual Response** | **Pass/Fail** | **Comments** |
| 1 | Navigate to “Income” section. | N/A | System shows “Add Income” form. | Form displayed |  |  |
| 2 | Add income source, amount, and date. | Freelance, PKR 15,000, 2025-05-01 | System saves data and updates summary. | Income saved |  |  |
| 3 | Save entry. | Click “Save” | Income is saved and reflected in dashboard. | Dashboard updated |  |  |
| 4 | Edit or delete an income source. | Edit amount to PKR 16,000 | System updates or deletes it correctly. | Updated |  |  |

|  |
| --- |
| **Post-Conditions** |
| 1. Income is saved, visible in reports and used in financial calculations. 2. Dashboard reflects updated earnings. |

Table 30- Managing Income

### Test Case: TC-06 – Managing Savings Goals

|  |
| --- |
| **Test Case #:** 06 **Test Case Name:** Managing Savings  Goals  **System**: BudgetMate **Subsystem:** User  **Designed by:** Maryam Tahir **Design Date:** 11/05/2025  **Executed by:** **Execution Date:**  **Short Description:** Test creation and editing of savings  goals and visual tracking progress. |
| **Pre-Conditions** User is logged in. |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Step** | **Action** | **Test Data** | **Expected System Response** | **Actual Response** | **Pass/Fail** | **Comments** |
| 1 | Go to “Saving Goals” section. | N/A | System displays “Add Goal” button. | Button shown |  |  |
| 2 | Enter goal name, target amount, and timeframe. | Trip, PKR 50,000, 5 months | System calculates required savings frequency. | Calculated |  |  |
| 3 | Assign income percentage to the goal. | 10% from Salary | Amont is deducted from income accordingly toward goal. | Assigned |  |  |
| 4 | Save and track progress. | Click “Save” | Goal is visualized with a progress bar. | Progress shown |  |  |
| 5 | Edit goal. | Change to 6 months | System updated goal details and calculations. | Updated |  |  |

|  |
| --- |
| **Post-Conditions** |
| 1. Saving goals is tracked and shown in dashboard. 2. Progress updates as user adds savings. |

Table 31- Managing Savings Goals

### Test Case: TC-07 – Generating Reports and Analyzing Finances

|  |
| --- |
| **Test Case #:** 07 **Test Case Name:** Generating Reports  and Analyzing Finances  **System**: BudgetMate **Subsystem:** User  **Designed by:** Maryam Tahir **Design Date:** 11/05/2025  **Executed by:** **Execution Date:**  **Short Description:** Test generation of summary of  visual reports for financial data i.e., income, expenses,  and budgets. |
| **Pre-Conditions** User has income, expenses, and budget data entered. |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Step** | **Action** | **Test Data** | **Expected System Response** | **Actual Response** | **Pass/Fail** | **Comments** |
| 1 | Navigate to “Report” section. | N/A | System shows time filters (weekly, monthly, yearly) | Filters shown |  |  |
| 2 | Select report type and timeframe. | Monthly | System loads corresponding data. | Data loaded |  |  |
| 3 | View graphical representation. | Click “View Graph” | Pie/bar charts are displayed clearly. | Graphs shown |  |  |
| 4 | Drill down by category. | Select “Food” | Report details for selected category appear. | Details shown |  |  |

|  |
| --- |
| **Post-Conditions** |
| 1. User views accurate reports. 2. Graphs support clear decision-making. |

Table 32- Generating Reports and Analyzing Finances

### Test Case: TC-08 – Securing Data and Protecting Privacy

|  |
| --- |
| **Test Case #:** 08 **Test Case Name:** Securing Data and  Protecting Privacy  **System**: BudgetMate **Subsystem:** User  **Designed by:** Maryam Tahir **Design Date:** 11/05/2025  **Executed by:** **Execution Date:**  **Short Description:** View encryption of data and  unauthorized login alerts. |
| **Pre-Conditions** User account exists. |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Step** | **Action** | **Test Data** | **Expected System Response** | **Actual Response** | **Pass/Fail** | **Comments** |
| 1 | Attempt login with invalid credentials. | Wrong password | Systems show login error message. | Error shown |  |  |
| 2 | Repeated failed attempts. | 3 wrong tries | System logs attempts and alerts user via email. | Alert sent |  |  |
| 3 | View account info in DB. | View Firebase data | Sensitive data appears encrypted. | Encrypted |  |  |
| 4 | Access app over secure network. | HTTPS enabled | Data is transmitted via HTTPS. | Encrypted transfer |  |  |

|  |
| --- |
| **Post-Conditions** |
| 1. User data remains secure and encrypted. 2. Unauthorized access attempts are flagged. |

Table 33- Securing Data and Protecting Privacy

### Test Case: TC-09 – Edit Profile

|  |
| --- |
| **Test Case #:** 09 **Test Case Name:** Edit Profile  **System**: BudgetMate **Subsystem:** User  **Designed by:** Maryam Tahir **Design Date:** 11/05/2025  **Executed by:** **Execution Date:**  **Short Description:** Test functionality to update user  info, preferences, and deactivate account. |
| **Pre-Conditions** User is logged in. |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Step** | **Action** | **Test Data** | **Expected System Response** | **Actual Response** | **Pass/Fail** | **Comments** |
| 1 | Navigate to “Profile” section. | N/A | System shows editable fields. | Fields shown |  |  |
| 2 | Update name, email. | Name: Maryam, Email: maryam@xyz.com | Changes are saved and applied. | Updated |  |  |
| 3 | Change currency preference. | USD | Currency updates throughout the app. | Currency changed |  |  |
| 4 | Change password. | Old: abc123 | System validates old password before update. | Password updated |  |  |
| 5 | Enable “Dark Mode” button. | Click toggle | UI changes to dark theme instantly. | Dark mode applied |  |  |
| 6 | Click on “Logout” button. | N/A | System logs out and returns to login screen. | Logged out |  |  |
| 7 | Click “Deactivate Account.” | Click confirm | Confirmation prompts appears and proceeds. | Account deactivated |  |  |

|  |
| --- |
| **Post-Conditions** |
| 1. Profile data is update securely. 2. System reflects changes across the app. |

Table 34- Edit Profile

### Test Case: TC-10 – Notifications on Budget Exceed

|  |
| --- |
| **Test Case #:** 10 **Test Case Name:** Notifications on  Budget Exceed  **System**: BudgetMate **Subsystem:** User  **Designed by:** Maryam Tahir **Design Date:** 11/05/2025  **Executed by:** **Execution Date:**  **Short Description:** Test that alerts are triggered on  overspending or goal milestones. |
| **Pre-Conditions** User is logged in, and budget or goal is active. |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Step** | **Action** | **Test Data** | **Expected System Response** | **Actual Response** | **Pass/Fail** | **Comments** |
| 1 | Set a budget with limit. | Food: PKR 5,000 | System saves threshold. | Threshold saved |  |  |
| 2 | Log expenses beyond limit. | Food: PKR 6,000 | Alert is triggered and displayed. | Alert shown |  |  |
| 3 | Reach milestone in savings. | 50% of goa; achieved | Notification is shown in-app. | Milestone sown |  |  |
| 4 | Dismiss or ignore alert. | Click “Dismiss” | System does not show the same alert again unless reset. | Dismissed |  |  |

|  |
| --- |
| **Post-Conditions** |
| 1. Alters functions as expected. 2. User informed of financial limits/milestones. |

Table 35- Notifications on Budget Exceed

## 4.2 Unit / Integration / Acceptance Testing

### Unit Testing

Unit testing involves testing individual modules or components of the BudgetMate application independently. Each of the functions, like user registration, login authentication, expense addition, or budget creation, were tested independently to ensure that it does its specific task properly. For instance, the "Register New User" functionality was tested independently to ensure that it properly validates user input, verifies existing email addresses, and saves user information securely into the database.

### Integration Testing

Integration testing was carried out to confirm that various BudgetMate components operate appropriately when put together. Upon passing unit testing by individual modules, their behavior was tested to verify smooth communication and data transfer among them. For example, upon successful login by the user, integration testing ensured the correct retrieval and presentation of the user's dashboard, budget, and expense summary by the system. This assisted in revealing data-transfer-related, API communication-related, and dependency issues among modules.

### Acceptance Testing

Acceptance testing was aimed at confirming the entire BudgetMate application from the end-user's point of view. It was performed to confirm that the system meets the requirements defined during planning. Testing was done to verify key user flows, including user registration, expense addition, budgeting, savings goal tracking, and producing financial reports. The application was tested in terms of usability, correctness, and performance, and it met the expectations as described in the functional requirements.

# Chapter 5: Conclusion

## 5.1 Problems Faced and Lessons Learned

During the documentation and planning phase of BudgetMate, the most notable challenge was refining the scope to keep the project realistic within the available time and resources. Initially, several ambitious features were considered, but through discussions with the supervisor and peers, the scope was streamlined to focus on core functionalities that address the real needs of users.

Another issue was translating user expectations into detailed functional and non-functional requirements. This required research into similar applications, usability principles, and data security standards. Through this, we learned the importance of clear requirement definitions, stakeholder identification, and early documentation structure, which serve as the foundation for a smooth development cycle.

## 5.2 Project Summary

The documentation prepared so far lays the groundwork for the development of BudgetMate — a personal finance mobile application focused on simplifying expense tracking, budgeting, and savings goals. The report outlines the project’s objectives, problem statement, functional and non-functional requirements, and proposed system structure.

At this stage, we have developed a comprehensive set of use cases, test scenarios, and a planned design strategy. The aim was to identify user requirements, ensuring usability, and establishing a clear path for future development and testing levels.

## 5.3 Future Work

This phase is based on planning and documentation of the core features of BudgetMate. Future stages of the project will consider a number of improvements and more modules to be investigated and built. These include:

* **Bank** **Integration**: Adding support for linking bank accounts or mobile wallets to synchronize transactions automatically and categorize them.
* **AI-Based** **Spending Insights**: Adding smart suggestions based on user spending patterns, like alerts on repeating charges or savings opportunities.
* **Shared** **Budgeting Features**: Adding support for joint accounts or shared expenses for couples, families, or roommates.
* **Multi-Currency** **and Localization** **Support**: Enabling users to easily switch between regional formats and currencies to serve a wider audience.
* **Dark** **Mode and Theme** **Customization**: Enhancing the user experience by providing interface customization features.
* **Data Export** **and Backup** **Options**: Providing users with the ability to download financial reports or export data securely.

These features are out of scope for the current version but are aligned with long-term goals for extending BudgetMate’s usability, reach, and impact. Their implementation will be based on feasibility, user feedback, and available resources in later project stages.

## 5.4 Screenshots

A screenshot of a sign up form

AI-generated content may be incorrect.A screenshot of a login page

AI-generated content may be incorrect.

A screenshot of a phone

AI-generated content may be incorrect.A screenshot of a phone

AI-generated content may be incorrect.

A screenshot of a phone

AI-generated content may be incorrect.A screenshot of a phone

AI-generated content may be incorrect.

A screenshot of a phone

AI-generated content may be incorrect.A screenshot of a website

AI-generated content may be incorrect.

# **References**

|  |  |
| --- | --- |
| [1] | "Quicken," Quicken Inc, [Online]. Available: https://www.quicken.com/products/simplifi/. |
| [2] | "Bugdet Bakers," Bugdet Bakers Inc.,, [Online]. Available: https://budgetbakers.com/. |
| [3] | "Money Manager," RealByte Inc, [Online]. Available: https://www.realbyteapps.com/. |
| [4] | "Money Keeper: Expense and Budget," MISA JSC, [Online]. Available: https://play.google.com/store/apps/details?id=vn.com.misa.sothuchi&hl=en. |
| [5] | "Money Pocket: Expense and Budget," AYA Tech International, [Online]. Available: https://play.google.com/store/apps/details?id=io.moneypocket.app&hl=en. |

Appendix A