Software Requirements Specification

for

Expense Manager

Version 1.0 approved

Prepared by Sabil Mann

SUNY Farmingdale

June 2024

Table of Contents

Revision History

Name	**Date**	**Reason For Changes**	**Version**
Expense-Manager	06/12/24	Initial Commit	0.9 Alpha
	6/17/24	Frontend Integration	1.0 Tango
	6/19/24	Backend Integration/Deploy	1.1 Zed

1. Introduction

1.1 Purpose

The purpose of this document is to present a detailed description of the Expense Manager application. It will explain the purpose and features of the software, the interfaces of the software, what the software will do, and the constraints under which it must operate. This document is intended for users of the software and potential developers.

1.2 Document Conventions

This Document was created based on the IEEE template for System Requirement

1.3 Intended Audience and Reading Suggestions

- Typical Users: Individuals, such as students or professionals, who want to use the Expense Manager to track and manage their expenses and subscriptions.
- Advanced/Professional Users: Financial planners or advisors who want to use the Expense Manager for more comprehensive financial analysis and management.
- Programmers: Developers who are interested in contributing to the project by further developing it or fixing existing bugs.

1.4 Product Scope

The Expense Manager is a desktop application that helps users efficiently track and manage their expenses and subscriptions. Users can categorize their expenses, set budgets, and receive notifications for upcoming payments. The application aims to provide users with a comprehensive tool for monitoring their financial activities, making informed decisions, and achieving their financial goals. It offers robust features, a secure platform, and an intuitive interface to ensure a seamless user experience.

1.5 References

- GitHub Page: https://github.com/CSC325-Summer24/Expenses-Manager.

2. Overall Description

2.1 Product Perspective

The Expense Manager was developed for individuals who want to efficiently track and manage their financial activities, including expenses and subscriptions. It supports various types of expenses such as car payments, bills, and other recurring costs. The application is designed to run on Windows, Mac OS X, and Linux platforms, providing a cross-platform solution for users to manage their finances. The Expense Manager is intended to be a robust, user-friendly, and secure application that integrates seamlessly into users' daily financial management routines.

2.2 Product Functions

File

- New Project: Create a new expense tracking project.
- Open: Load an existing project or file.
- Open Recent: Load one of the recently opened projects.
- Close Project: Close the currently open project.
- Save: Save the project without changing its name or directory.
- Save As: Save the project with a new name or in a different directory.
- Export: Export the project data in various formats (e.g., CSV, PDF).
- Import: Import data from various sources (e.g., CSV files).
- Exit: Shut down the Expense Manager application.

Expense Management:

- Add Expense: Add a new expense entry, including details such as name, amount, category, and due date.
- Edit Expense: Edit the details of an existing expense entry.
- Delete Expense: Remove an expense entry from the project.
- View Expenses: Display a list of all expenses with filtering and sorting options.
- Categorize Expenses: Categorize expenses into predefined or custom categories (e.g., bills, subscriptions, car payments).
- Recurring Expenses: Manage recurring expenses with options to set frequency (e.g., weekly, monthly).

Budget Management:

- Create Budget: Set up a budget for different categories or specific expenses.
- Edit Budget: Modify an existing budget.
- Delete Budget: Remove a budget from the project.
- View Budget: Display budget summaries and progress.

Notifications:

- Due Date Alerts: Send notifications for upcoming due dates.
- Budget Alerts: Notify users when they are nearing or exceeding their budget limits.

Reporting:

- Expense Reports: Generate visual reports and summaries of expenses over time.
- Budget Reports: Generate reports on budget usage and remaining budget.

Window:

- Dashboard: Display an overview of the user's financial status, including total expenses, budgets, and upcoming notifications.
- Expense Table: Display the details of all expenses in a table format.
- Budget Table: Display budget details in a table format.
- Notifications Panel: Show upcoming notifications and alerts.
- Reports Panel: Display generated reports and visualizations.

2.3 User Classes and Characteristics

- Typical Users: Individuals, such as students or professionals, who want to use the Expense Manager to track and manage their expenses and subscriptions.
- Advanced/Professional Users: Financial planners or advisors who want to use the Expense Manager for more comprehensive financial analysis and management.
- Programmers: Developers who are interested in contributing to the project by further developing it or fixing existing bugs.

2.4 Operating Environment

- Windows 7

- Windows 8
- Windows 10
- Mac OS X
- Linux

2.5 Design and Implementation Constraints

The Expense Manager is developed in Java, using JavaFX for its user interface. It utilizes Firebase for data storage and authentication, ensuring secure handling of user data. The application follows a modular design where each feature is encapsulated in separate modules that interact through well-defined APIs, making it easy to maintain and extend.

2.6 User Documentation

There will be a quick start guide available on the project's GitHub page:
https://github.com/CSC325-Summer24/Expenses-Manager
Additional help and information can be found on the project's wiki page, also hosted on GitHub

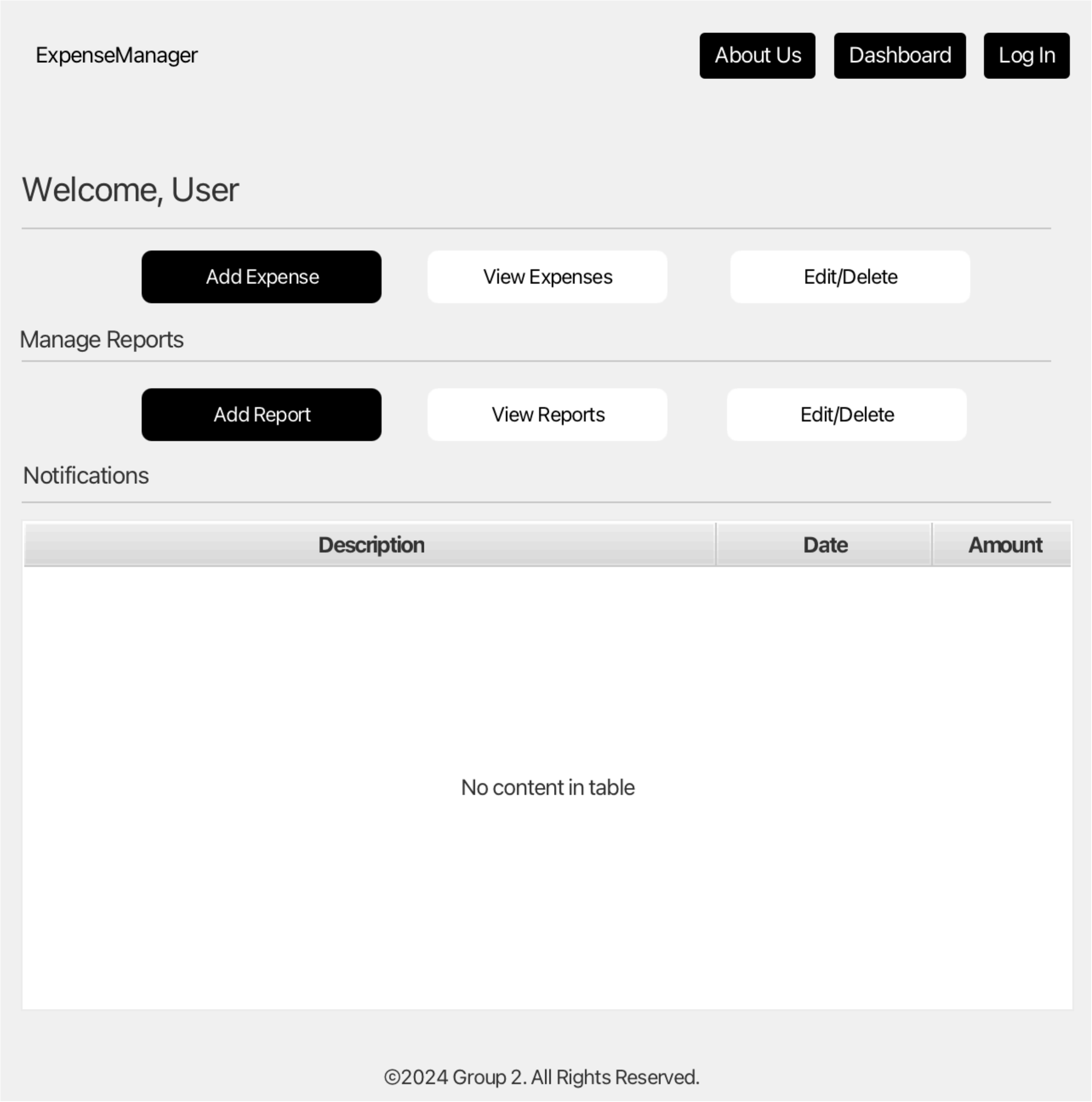
2.7 Assumptions and Dependencies

The Expense Manager is developed in Java and requires Java to be installed on the user's system. The application will work with Java version 8 or higher. It also requires an internet connection for data synchronization and user authentication via Firebase.

3. External Interface Requirements

3.1 User Interfaces

Dashboard Page

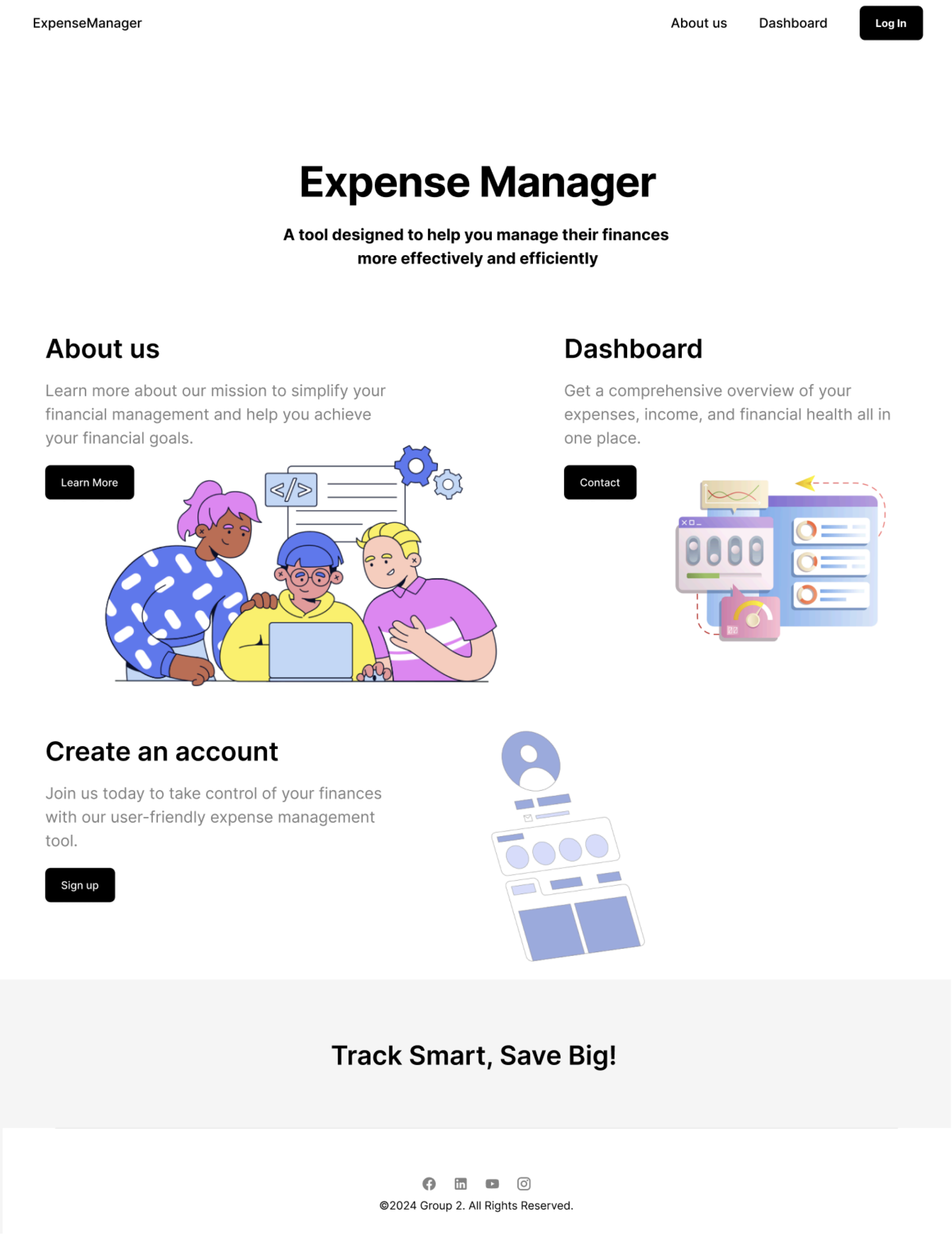


Description	Date	Amount

No content in table

©2024 Group 2. All Rights Reserved.

Landing Page



Expense Manager

A tool designed to help you manage their finances more effectively and efficiently

About us

Learn more about our mission to simplify your financial management and help you achieve your financial goals.

Dashboard

Get a comprehensive overview of your expenses, income, and financial health all in one place.

Create an account

Join us today to take control of your finances with our user-friendly expense management tool.

Track Smart, Save Big!

©2024 Group 2. All Rights Reserved.

3.2 Hardware Interfaces

The minimum hardware requirements for the Expense Manager application are as follows:

- CPU: 1.0 Gigahertz (GHz) processor or faster.
- RAM: 512 Megabytes (MB) of RAM or higher.
- Storage: 100 Megabytes (MB) of available disk space.
- Graphics: A basic graphics card that supports standard desktop applications.
- Display: Minimum screen resolution of 1024 x 768 pixels.

These specifications should be sufficient for handling typical expense management tasks and data entry operations. For optimal performance, particularly when dealing with extensive financial data and generating detailed reports, a system with higher specifications is recommended.

3.3 Software Interfaces

The Expense Manager application requires the following software components:

- Java Runtime Environment (JRE): The application is developed in Java and requires Java version 8 or higher to be installed on the user’s system.
- JavaFX: Included with Java 8 and higher versions for creating the graphical user interface.
- Firebase SDKs: For integration with Firebase services, including Firestore for data storage and Firebase Authentication for user login.
- Operating System: Compatible with Windows 7, Windows 8, Windows 10, Mac OS X, and Linux.

The application interfaces with Firebase Firestore for secure data storage and Firebase Authentication for user authentication. Detailed information about these integrations can be found in the respective Firebase documentation.

3.4 Communications Interfaces

The Expense Manager application requires an internet connection for the following functionalities:

- Data Synchronization: To synchronize expense data with Firebase Firestore, ensuring data is securely stored and accessible from any device.
- User Authentication: To enable user login and authentication through Firebase Authentication.
- Notifications: To send notifications about upcoming due dates and budget alerts.
- Updates: To check for and download updates for the application.

An active internet connection is necessary to utilize these features effectively and ensure the application operates seamlessly.

4. System Features

4.1 Expense Entry and Management

4.1.1 Description and Priority

The Expense Entry and Management feature allows users to add, edit, delete, and view their expenses. Users can categorize expenses, set due dates, and manage recurring expenses. This feature is of High priority as it constitutes the core functionality of the Expense Manager application.

4.1.2 Stimulus/Response Sequences

1. Stimulus: User selects "Add Expense" from the main menu.

- Response: System displays a form to enter expense details (name, amount, category, due date).

2. Stimulus: User fills out the form and clicks "Save."

- Response: System validates the input and adds the expense to the expense list, displaying a confirmation message.

3. Stimulus: User selects an existing expense and clicks "Edit."

- Response: System displays the expense details in an editable form.

4. Stimulus: User updates the expense details and clicks "Save."

- Response: System validates the input and updates the expense in the list, displaying a confirmation message.

5. Stimulus: User selects an expense and clicks "Delete."

- Response: System prompts for confirmation and, upon confirmation, removes the expense from the list.

4.1.3 Functional Requirements

- REQ-1: The system shall provide a form to enter new expense details including name, amount, category, and due date.
- REQ-2: The system shall validate the input data and display error messages for invalid entries.
- REQ-3: The system shall add a new expense to the expense list upon successful validation.

- REQ-4: The system shall allow users to edit existing expenses and save the changes.
- REQ-5: The system shall allow users to delete an expense after confirmation.
- REQ-6: The system shall support categorization of expenses into predefined or custom categories.
- REQ-7: The system shall allow users to set up and manage recurring expenses with specified frequencies (e.g., weekly, monthly).
- REQ-8: The system shall display a confirmation message after adding, editing, or deleting an expense.

4.2 Budget Management

4.2.1 Description and Priority

The Budget Management feature enables users to set, edit, and track budgets for different expense categories. This feature is of **Medium** priority as it helps users plan and manage their finances more effectively.

4.2.2 Stimulus/Response Sequences

1. Stimulus: User selects "Create Budget" from the main menu.

- Response: System displays a form to enter budget details.

2. Stimulus: User fills out the form and clicks "Save."

- Response: System validates the input and adds the budget to the budget list, displaying a confirmation message.

3. Stimulus: User selects an existing budget and clicks "Edit."

- Response: System displays the budget details in an editable form.

4. Stimulus: User updates the budget details and clicks "Save."

- Response: System validates the input and updates the budget in the list, displaying a confirmation message.

5. Stimulus: User selects an budget and clicks "Delete."

- Response: System prompts for confirmation and, upon confirmation, removes the budget from the list.

4.2.3 Functional Requirements

- REQ-1: The system shall provide a form to enter new budget details including category, amount, and duration.
- REQ-2: The system shall validate the input data and display error messages for invalid entries.
- REQ-3: The system shall add a new budget to the budget list upon successful validation.
- REQ-4: The system shall allow users to edit existing budgets and save the changes.
- REQ-5: The system shall allow users to delete a budget after confirmation.
- REQ-6: The system shall track expenses against the set budgets and display remaining budget amounts.
- REQ-7: The system shall notify users when they are nearing or exceeding their budget limits.
- REQ-8: The system shall display a confirmation message after adding, editing, or deleting a budget.

4.3 Notifications

4.3.1 Description and Priority

The Notifications feature provides alerts and reminders to users about upcoming due dates for expenses and budget limits. This feature is of High priority as it helps users stay informed and avoid missing payments or exceeding budgets.

4.3.2 Stimulus/Response Sequences

1. Stimulus: Due date for an expense approaches.

   - Response: System sends a notification alert to the user

2. Stimulus: User nears or exceeds their budget limit.

   - Response: System sends a budget alert to the user.

4.3.3 Functional Requirements

- REQ-1: The system shall send notifications for upcoming due dates of expenses.
- REQ-2: The system shall send notifications when users are nearing their budget limits.
- REQ-3: The system shall allow users to configure notification preferences (e.g., notification method, timing).
- REQ-4: The system shall ensure notifications are sent in a timely manner based on user preferences.

- REQ-5: The system shall log sent notifications and provide a history of alerts

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The Expense Manager application is expected to perform efficiently under typical usage scenarios. Specific performance requirements include:

- Response Time: The application should respond to user inputs within 2 seconds for standard operations such as adding or viewing expenses.
- Data Synchronization: Data synchronization with Firebase should complete within 5 seconds for normal data volumes (up to 100 expenses).
- Scalability: The application should be able to handle up to 500 expenses without noticeable performance degradation.
- Resource Utilization: The application should utilize no more than 50% of CPU and 200 MB of RAM during normal operation on a system meeting the minimum hardware requirements.

5.2 Safety Requirements

The Expense Manager application must ensure the safety of user data and operation:

- Data Integrity: Regular data backups should be performed to prevent data loss.
- Error Handling: The application should gracefully handle errors and provide informative error messages to the user.
- Data Validation: All input data must be validated to prevent incorrect or harmful data entries.
- System Stability: The application should be thoroughly tested to minimize the risk of crashes or freezes.

5.3 Security Requirements

Security is a critical aspect of the Expense Manager application. The following requirements ensure that user data is protected:

- User Authentication: The application must use Firebase Authentication to securely manage user logins.
- Data Encryption: All sensitive data must be encrypted during storage and transmission.
- Access Control: Only authenticated users should be able to access their own data.
- Audit Trail: The application should maintain logs of user activities to monitor access and changes to data.

- Compliance: The application must comply with relevant data protection regulations, such as GDPR.

5.4 Software Quality Attributes

The Expense Manager application must exhibit the following quality attributes:

- Usability: The application should have an intuitive and user-friendly interface to facilitate ease of use.
- Reliability: The application should perform consistently under normal operating conditions.
- Maintainability: The application code should be modular and well-documented to facilitate maintenance and updates.
- Portability: The application should be compatible with Windows, Mac OS X, and Linux operating systems.
- Adaptability: The application should be able to adapt to different user requirements and preferences.

5.5 Business Rules

The following business rules apply to the Expense Manager application:

- User Roles: Only authenticated users can add, edit, or delete expenses and budgets.
- Data Ownership: Users own their data and can export it at any time.
- Budget Limits: Users can set budget limits for different categories, and the application will enforce these limits through notifications and alerts.
- Subscription Management: Users can manage recurring subscriptions, and the application will automatically track and notify them about upcoming payments.

6. Other Requirements

- Database Requirements: The application will use Firebase Firestore for data storage, ensuring real-time synchronization and secure data handling.
- Internationalization Requirements: The application should support multiple languages to cater to a diverse user base.
- Legal Requirements: The application must comply with all applicable laws and regulations regarding data privacy and security.

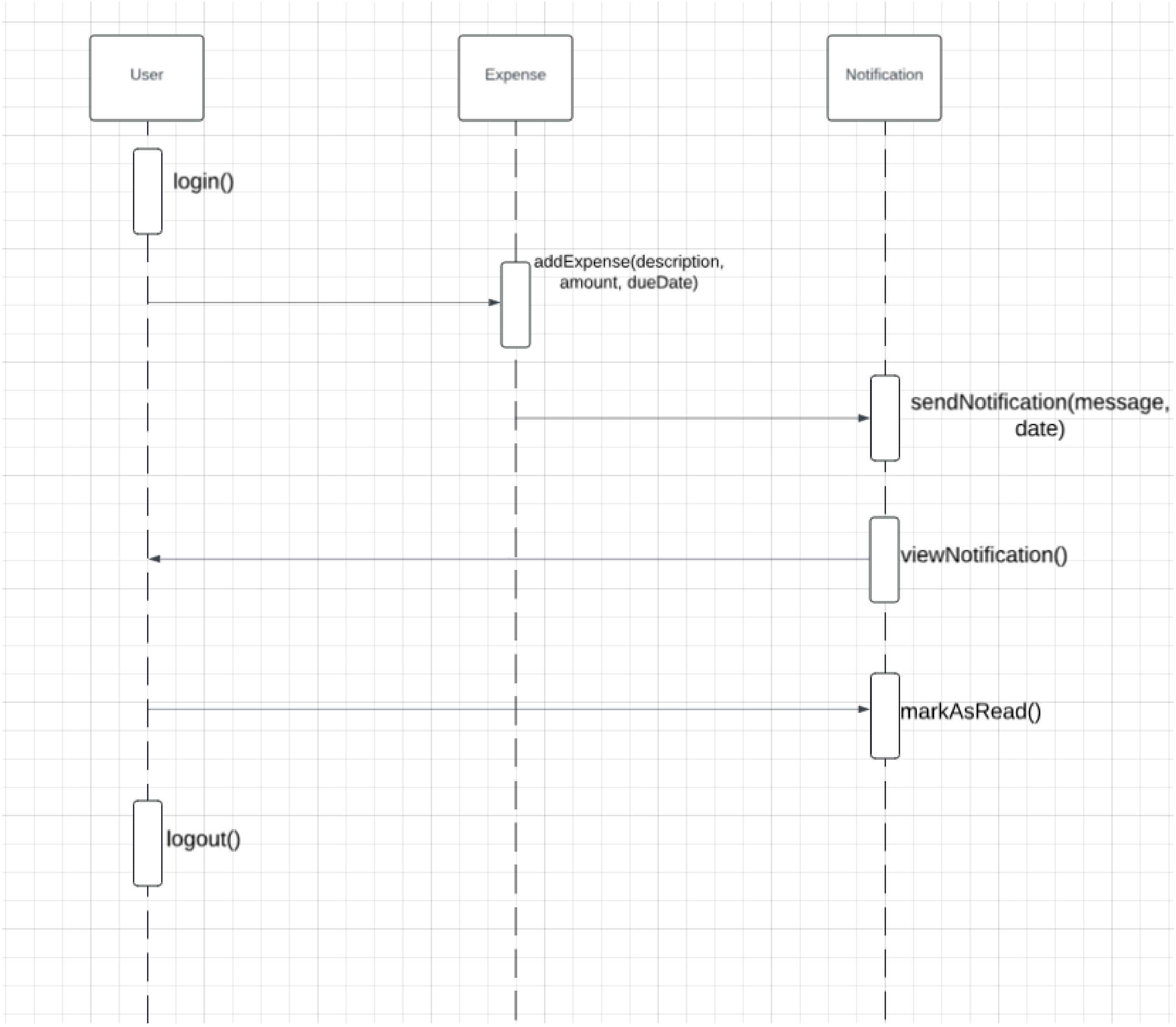
Appendix A: Glossary

- Expense: Any financial outlay or cost incurred by the user.
- Budget: A financial plan that limits the amount of money to be spent in a specific category or period.
- Firebase: A platform developed by Google for creating mobile and web applications, providing real-time database and authentication services.

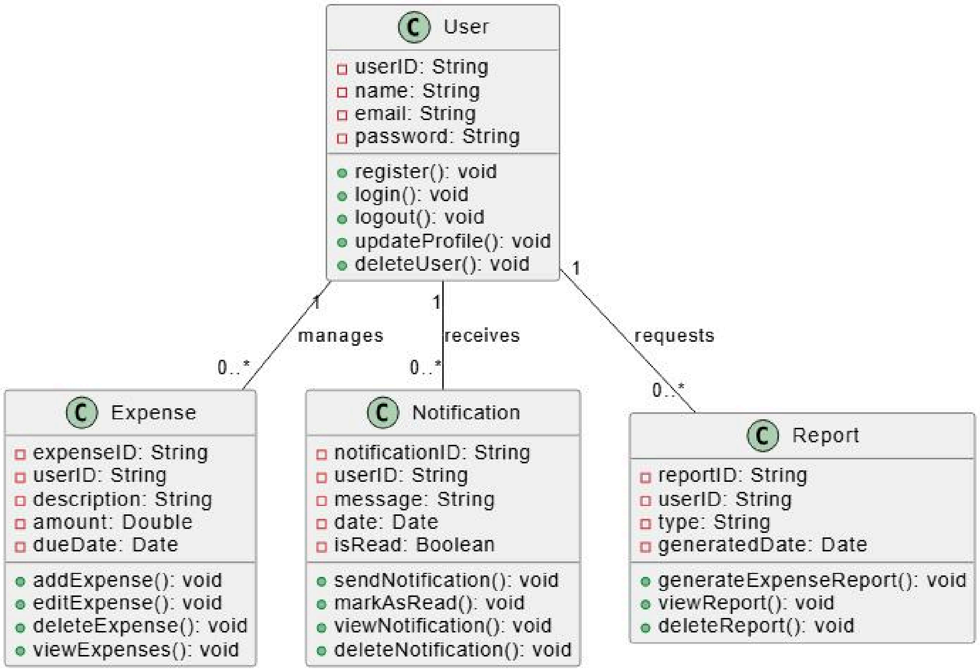
- JavaFX: A software platform used for creating and delivering desktop applications with rich user interfaces.
- Notification: A message sent to the user to inform them of important events, such as due dates or budget limits.

Appendix B: Analysis Models

Sequence Diagram


User
Expense
Notification
login()
addExpense(description, amount, dueDate)
sendNotification(message, date)
viewNotification()
markAsRead()
logout()

UML Diagram


User
userID: String
name: String
email: String
password: String
register(): void
login(): void
logout(): void
updateProfile(): void
deleteUser(): void
1
manages
0..*
1
receives
0..*
1
requests
0..*
Expense
expenseID: String
userID: String
description: String
amount: Double
dueDate: Date
addExpense(): void
editExpense(): void
deleteExpense(): void
viewExpenses(): void
Notification
notificationID: String
userID: String
message: String
date: Date
isRead: Boolean
sendNotification(): void
markAsRead(): void
viewNotification(): void
deleteNotification(): void
Report
reportID: String
userID: String
type: String
generatedDate: Date
generateExpenseReport(): void
viewReport(): void
deleteReport(): void