Savetify

Software Requirements Specification

09.04.2024

Mustafa Emir Uyar - 150120007

Muhammed Hayta - 150121068

Efe Özgen - 150121077

Necati Koçak - 150120053

Prepared for

CSE3044 Software Engineering Term Project

Table of Contents

1. Introduction

1.1 Purpose

Savetify is a cross-platform solution designed to help individuals take control of their finances. It provides a secure platform for tracking diverse assets (cash, investments, precious metals, cryptocurrency), visualizing spending patterns, generating insightful financial reports, and receiving personalized recommendations tailored to the user's financial behavior.

1.2 Scope

(1) Savetify will be a cross-platform mobile, desktop and web application

(2) Savetify's core focus is to empower users to make informed financial decisions. This includes:

- Consolidating financial information across multiple asset types.
- Providing clear visual representations of income and spending trends.
- Offering data-driven insights and personalized financial insight.
- Facilitating the setting and tracking of financial goals.

(3) Savetify will make users more financially aware and make it easy to track their investments. It will show their monthly report and will encourage the user to improve their financial management.

1.3 Definitions, Acronyms, and Abbreviations

Definitions:

- Asset: Any item of economic value owned by the user (e.g., cash, stocks, bonds, real estate, precious metals, cryptocurrency).
- Investment: Allocating money with the expectation of generating income or profit.
- Portfolio: A collection of assets held by the user.
- Diversification: The strategy of spreading investments across different asset classes to reduce risk.
- Financial Goal: A specific target the user aims to achieve financially (e.g., saving for a down payment, building an emergency fund, retiring early).
- Insight: A data-driven observation about the user's financial behavior that provides helpful information (e.g., identifying a recurring expense that could be reduced).
- ROI (Return on Investment): A percentage measuring the profitability of an investment over time, taking into account gains and losses.
- Cloud: The way to store your data (like files, photos, and software) and access powerful computing resources over the internet.
- Transaction: An instance of buying or selling something; a business deal.
- Category: A group of items sharing common characteristics, used for organizing transactions.
- Balance: The remaining amount of money in an account after all credits and debits have been accounted for.
- Dividend: The portion of a company's profits that is distributed to its shareholders.
- Stock Exchange: A marketplace where shares of stock and other assets are bought and sold.

Acronyms:

- ROI: Return on Investment
- UI: User Interface
- UX: User Experience
- API: Application Programming Interface

1.4 References

Flutter Development Guidelines: https://docs.flutter.dev

Firebase Best Practices: https://firebase.google.com/docs

1.5 Overview

Savetify is a cross-platform application designed to improve users personal financial management skills. It offers tools for tracking diverse assets (including traditional savings, investments, and cryptocurrency), visualizing spending patterns to gain insights, and receiving personalized insights for user's financial behavior. The app aims to increase financial awareness and promote healthy spending habits.

2. General Description

2.1 Product Perspective

Savetify shares similarities with "Fintables" in terms of functionality but distinguishes itself through a minimalist design for the general public usage. Unlike "Fintables", It does not include specialized features requiring expertise in finance.

In terms of user interface, Savetify draws inspiration from the app "Midas" but does not provide any means to invest, it only tracks your personal finance.

Unlike "Fintables" and "Midas." Users can access and use Savetify without an internet connection, with data syncing to the database whenever possible. Additionally, Savetify not only tracks investments but also monitors income and daily spendings, offering valuable insights to users throughout their financial journey.

2.2 Product Functions

Key elements of the functionality include:

- Savetify will offer an overall view of the user's financial holdings, traditional assets (cash, investments), precious metals, and cryptocurrencies.
- The application will provide visual tools to illustrate spending patterns and income trends. It will generate insights from this data, offering personalized financial insight to the user's behavior.
- Savetify will assist users in establishing specific financial goals and provide mechanisms to track progress towards these objectives.
- The aim of the app is to increase financial awareness and encourage healthy financial habits. It will present regular reports and use positive reinforcement to promote continued investment and responsible spending.

2.3 User Characteristics

Savetify aims to appeal to individuals from various backgrounds who seek to manage their personal finances effectively. It targets users with a basic understanding of personal economics, regardless of their profession or domain expertise. Savetify is designed to be accessible to users with minimal to no experience in economics. A simple understanding of modern user interfaces and digital tools is sufficient to begin using Savetify's features and functionalities. The platform prioritizes user-friendly design and intuitive navigation, ensuring that even novice users can easily navigate through the application and leverage its capabilities for financial management.

2.4 General Constraints

- The system should not give any binding financial advice.
- Compatibility with a wide range of platforms, including web browsers, Android devices, and Windows systems, to reach a broader user base.
- Resource limitations such as device memory, processing power, and network bandwidth to optimize performance and ensure smooth operation across various devices and network conditions.
- Scalable architecture design , allowing it to accommodate a growing user base and increasing data volumes without significant performance degradation.
- Consider limitations imposed by the chosen development framework (Flutter) and ensure compatibility with desired features and functionalities.

2.5 Assumptions and Dependencies

Assumptions:

User Internet Connectivity: It is assumed that users will have reliable internet connectivity to access and synchronize data with the Savetify application. However, offline functionality with data caching will be provided for seamless user experience in case of intermittent or no internet access.

User Device Compatibility: It is assumed that users will have devices compatible with the Savetify application, including web browsers, Android devices, and Windows systems. The application will be optimized for various screen sizes and resolutions to ensure compatibility across different devices.

Dependencies:

Flutter SDK and Firebase Integration: The development of Savetify depends on the integration of the Flutter SDK for cross-platform app development and Firebase for backend services such as user authentication and data storage.

Third-Party APIs and Services: The functionality of Savetify may depend on third-party APIs and services for features such as real-time stock market data, cryptocurrency prices, or exchange rates. Dependencies on these external services require proper integration and reliance on their availability and reliability.

3. Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

Easy-to-use interfaces with visually appealing designs for seamless navigation.

3.1.2 Hardware Interfaces

Support for commonly used hardware setups on Windows, Android, IOS, and Web.

3.1.3 Software Interfaces

Integration with Flutter SDK for cross-platform development and Firebase for backend database management. Additionally, integration with Third Party API's for accessing investment, currency, and cryptocurrency data for analysis and visualization within the application.

3.1.4 Communications Interfaces

Support for data synchronization and communication between client applications and the backend server. Integration with Third Party API's for data exchange and communication between the application and services, specifically for retrieving investment, currency, precious metals and cryptocurrency data, etc..

3.2 Functional Requirements

3.2.1 Expense Tracking

3.2.1.1 Introduction / Description: This feature enables users to record, categorize, and track their expenses over time within Savetify.

3.2.1.2 Inputs / Display: Users input details such as expense date, amount, category, and description. Savetify provides a user-friendly interface for easy expense entry.

3.2.1.3 Processing: Savetify processes entered expense data and securely stores it in the database.

3.2.1.4 Outputs: Users can view their expense history, categorized by date, amount, and category, within Savetify. Additionally, they can generate reports to analyze spending habits.

3.2.1.5 Constraints: Savetify should support various currencies and allow users to customize expense categories.

3.2.1.6 Error/Data Handling: Savetify should handle errors such as invalid inputs and provide appropriate feedback to users.

3.2.2 Investment Tracking

3.2.2.1 Introduction / Description: This feature allows users to record their investments, including purchase details, investment type, and performance, within Savetify.

3.2.2.2 Inputs / Display: Users input information such as investment type, amount, purchase date, and performance metrics within Savetify. The application displays investment summaries and performance charts.

3.2.2.3 Processing: Savetify calculates investment returns, tracks changes in investment value, and securely stores investment data.

3.2.2.4 Outputs: Users can view their investment portfolio, including performance metrics such as ROI, asset allocation, and historical returns, within Savetify.

3.2.2.5 Constraints: Savetify should support various investment types, including stocks, bonds, mutual funds, cryptocurrencies, etc.

3.2.2.6 Error/Data Handling: Savetify should handle errors such as incorrect investment data entry and provide users with clear instructions for resolving issues.

3.2.3 Personalized Spending Profiles

3.2.3.1 Introduction / Description: This feature provides users with personalized spending profiles based on their financial habits and predefined financial goals.

3.2.3.2 Inputs / Display: Users input their financial goals, such as saving for a down payment, building an emergency fund, or retiring early, as well as their income and expenses. Savetify presents users with recommended spending profiles based on their inputs.

3.2.3.3 Processing: Savetify categorizes users into predefined spending profiles based on their financial goals and habits, without utilizing machine learning algorithms. Instead, it relies on the user's stated financial objectives and insights derived from their financial behavior.
3.2.3.4 Outputs: Users receive personalized spending profile recommendations tailored to their financial situation and goals, along with insights derived from their financial behavior.
3.2.3.5 Constraints: Savetify should offer a variety of predefined spending profiles that users can choose from based on their preferences and financial objectives. Additionally, insights provided to users should be relevant and actionable, such as identifying recurring expenses that could be reduced to help achieve their financial goals.
3.2.3.6 Error/Data Handling: Savetify should handle errors in profile recommendation generation and insight delivery, ensuring users receive accurate and helpful information to guide their financial decisions.

3.2.4 Monthly/Weekly Summaries and Reports

3.2.4.1 Introduction / Description: This feature provides users with summarized reports of their financial activities, including expenses, income, savings, and investments, within Savetify.
3.2.4.2 Inputs / Display: Users can select monthly or weekly summaries and reports from the Savetify dashboard. The reports display key financial metrics and trends.
3.2.4.3 Processing: Savetify aggregates and analyzes users' financial data to generate summarized reports.
3.2.4.4 Outputs: Users receive visual summaries and detailed reports of their financial activities within Savetify, helping them track progress towards their financial goals.
3.2.4.5 Constraints: Savetify should provide customizable reporting options to cater to users' preferences and needs.
3.2.4.6 Error/Data Handling: Savetify should handle data inconsistencies and discrepancies in reports, providing users with accurate and reliable information.

3.2.5 Income Management

3.2.5.1 Introduction / Description: Savetify enables users to record, track, and analyze their regular income sources.

3.2.5.2 Inputs / Display: Users input their income amount, date, and source within Savetify. The application displays income categorized by source and date.

3.2.5.3 Processing: Savetify processes entered income data and securely stores it in the database.

3.2.5.4 Outputs: Users can view their income history and sources, as well as visualize income trends over time within Savetify.

3.2.5.5 Constraints: Savetify should support various income sources and allow users to customize income categories.

3.2.5.6 Error/Data Handling: Savetify should handle errors such as invalid inputs and provide appropriate feedback to users.

3.2.6 Budget Planning and Tracking

3.2.6.1 Introduction / Description: Savetify assists users in creating budgets and managing their expenses within budget limits.

3.2.6.2 Inputs / Display: Users define budget categories and spending limits within Savetify. The application displays budget summaries and alerts users of overspending.

3.2.6.3 Processing: Savetify processes budget categories and tracks spending against set limits.

3.2.6.4 Outputs: Users receive visual representations of their budget performance and spending habits, along with alerts for exceeding budget limits within Savetify.

3.2.6.5 Constraints: Savetify should offer flexible budget management options and provide notifications to prevent budget overruns.

3.2.6.6 Error/Data Handling: Savetify should handle errors related to budget category entries and provide users with clear instructions for corrective actions.

3.3 Non-Functional Requirements

3.3.1 Performance

- The application should respond to user actions or events within 200 milliseconds to maintain responsiveness.
- The screen refresh time should not exceed 100 milliseconds to provide users with a seamless interaction experience.

3.3.2 Reliability

- Savetify should have a MTTF of at least 7 days, indicating its reliability and stability.
- The probability of Savetify being unavailable at any given time should be less than 1%.
- The rate of failure occurrence should not exceed 1% per day, demonstrating the application's reliability.

3.3.3 Availability

- The core features such as entering an expense or seeing a monthly report must remain usable for a minimum of 24 hours without an internet connection.
- Changes made offline must be synchronized with the online database within 5 seconds of re-establishing an internet connection.
- The online application will maintain a 99.9% uptime per month

3.3.4 Security

- Firebase Authentication will be the sole method of user access, with 100% of login attempts handled by the service.

3.3.5 Maintainability

- Code must be maintainable, with well-organized and modular components based on object-oriented programming principles.
- In order to merge the code to the production, at least 2 people must review the code.

3.3.6 Portability

- Savetify should be compatible with at least three different target systems (Web, Android, Windows) to maximize its reach and availability.

3.4 Inverse Requirements

- Savetify will not grant investment incentives.
- Shares will not be traded through Savetify.
- The user will have to enter the purchase/sale data themselves, Savetify will not do this automatically.
- Savetify will not arrange or access any data of any other financial platforms such as open-banking.

3.5 Design Constraints

3.5.1 Modular Code and Testing: Write the code in small, independent modules that can be easily tested. This approach ensures that each part of the application works correctly and can be modified or updated without affecting other parts.

3.5.2 Cross-Platform Compatibility: "Savetify" needs to run smoothly on Web, Android, IOS and Windows, keeping the user experience consistent.

3.5.3 Lightweight: Make sure "Savetify" runs well on different devices without consuming resources.

3.5.4 Flexible External API Usage: The application will rely on external APIs for certain functionalities like fetching financial data. These APIs should be chosen for their flexibility and reliability, allowing us to integrate them seamlessly into "Savetify" without becoming overly dependent on any single provider. This flexibility ensures that we can adapt to changes in the API or switch to alternative providers if needed, without disrupting the application's functionality.

3.5.5 Grows with Users: Design "Savetify" to handle more users and data without slowing down.

3.5.6 Security First: Protect user data with encryption and strong authentication methods provided by firebase.

3.5.7 Easy for Everyone: Make "Savetify" usable and accessible to everyone with an easy to use UI.

3.6 Logical Database Requirements

Savetify will use Firebase to store user data. The Firebase's firestore database uses a NoSQL database approach. This way, the data retrieval and data writing is very fast in the cloud server. Also, Firebase supports ACID transactions. We can store the data more reliably using the ACID transactions.

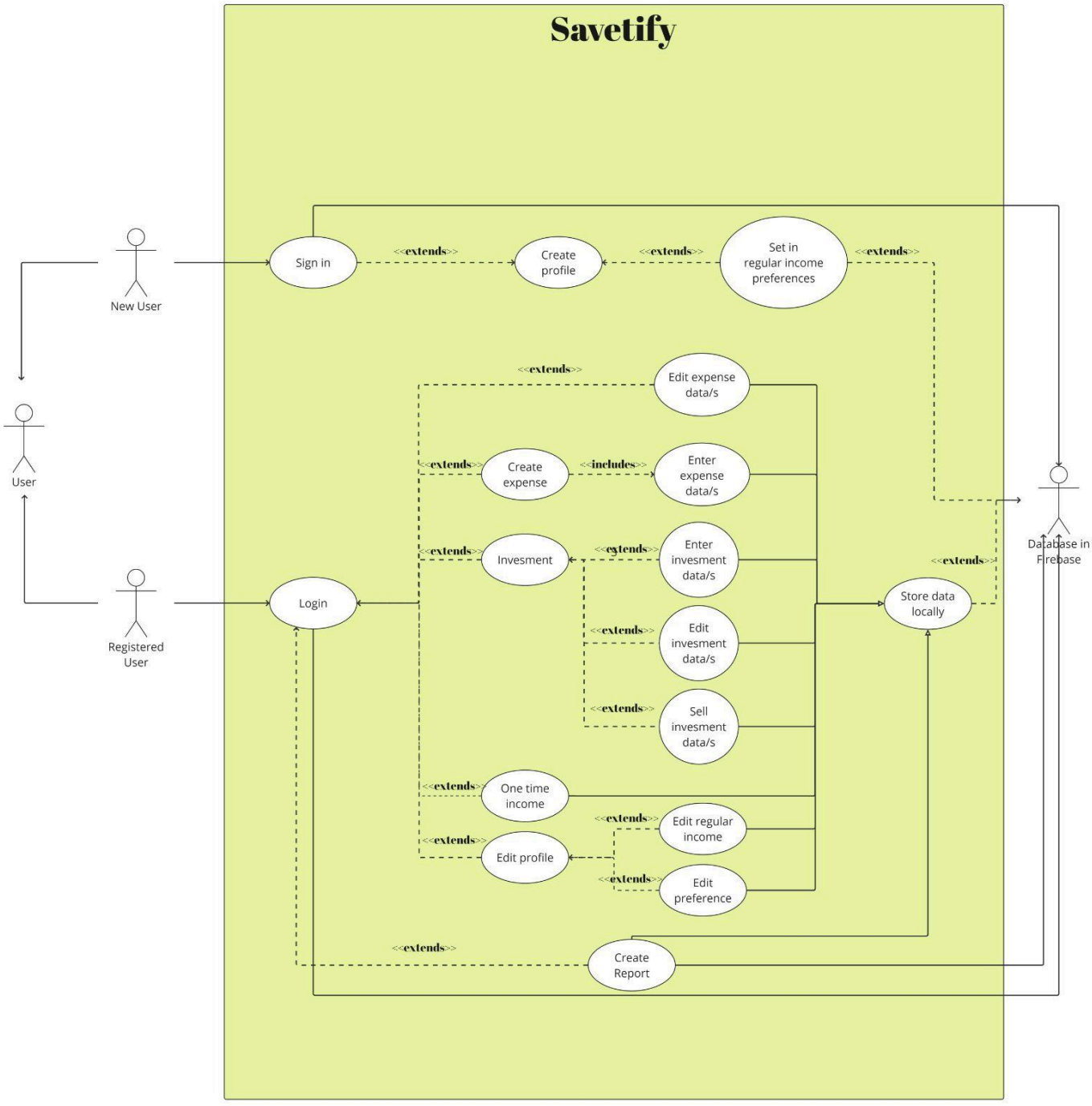
- We'll use JSON to organize financial information (like spending, investments, etc.) in a way Firebase understands.
- The amount of storage needed will grow as Savetify has more users. Firebase can easily scale to handle this.
- We need rules to make sure the user data is correct and private. Firebase has tools to manage this.
- Savetify will regularly back up data to Firebase. This way, if something goes wrong, we can restore the user's information.
- We'll organize the data in Firebase carefully so the app is fast and responsive for users.

3.7 Other Requirements

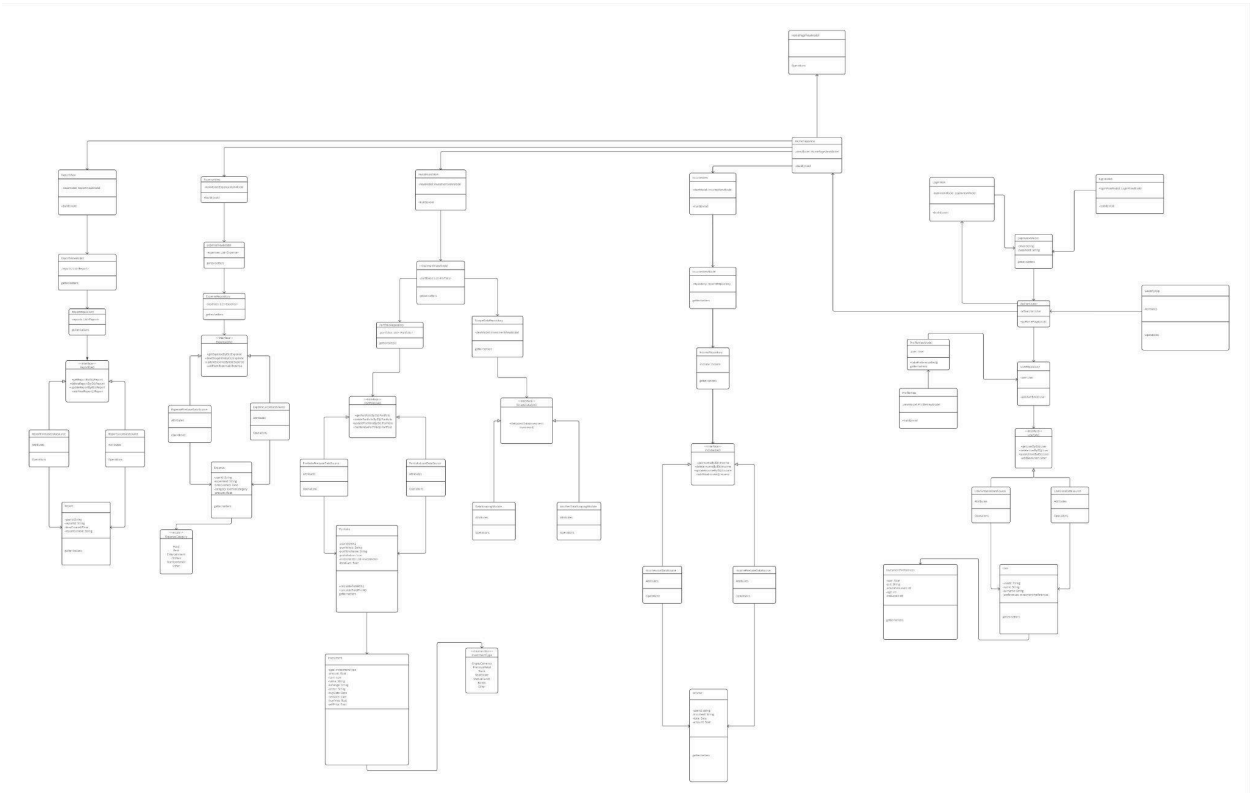
There are no other requirements for our project.

4. UML Diagrams

4.1 Use Cases

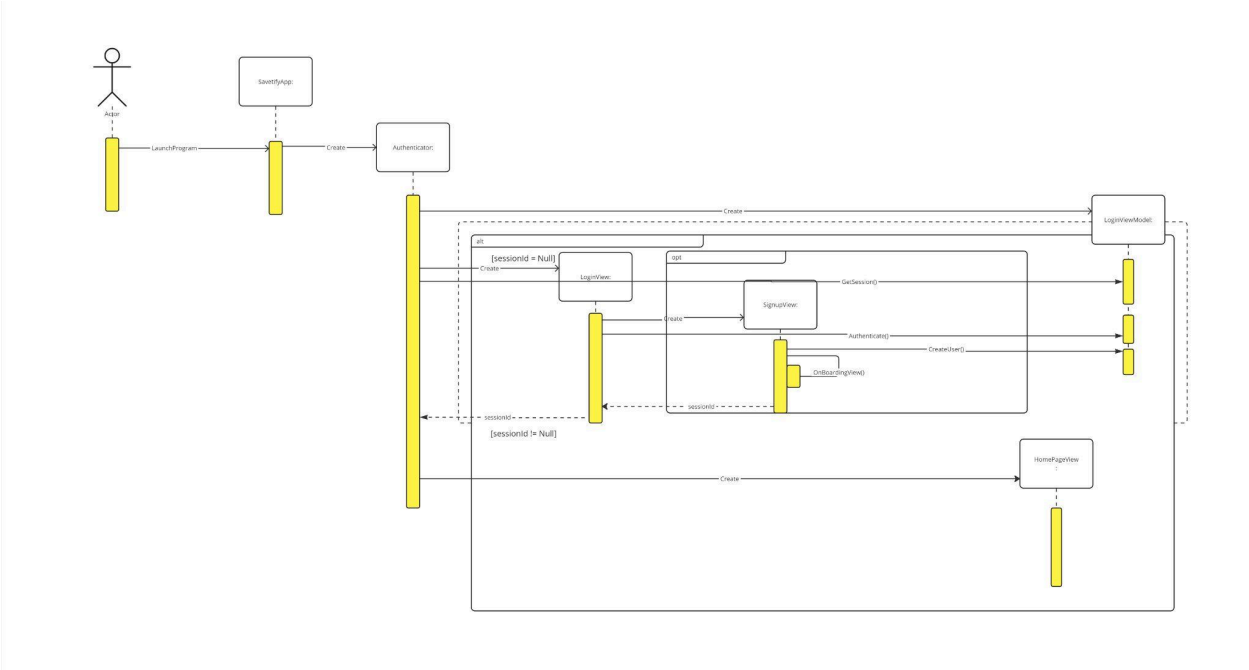


4.2 Classes / Objects

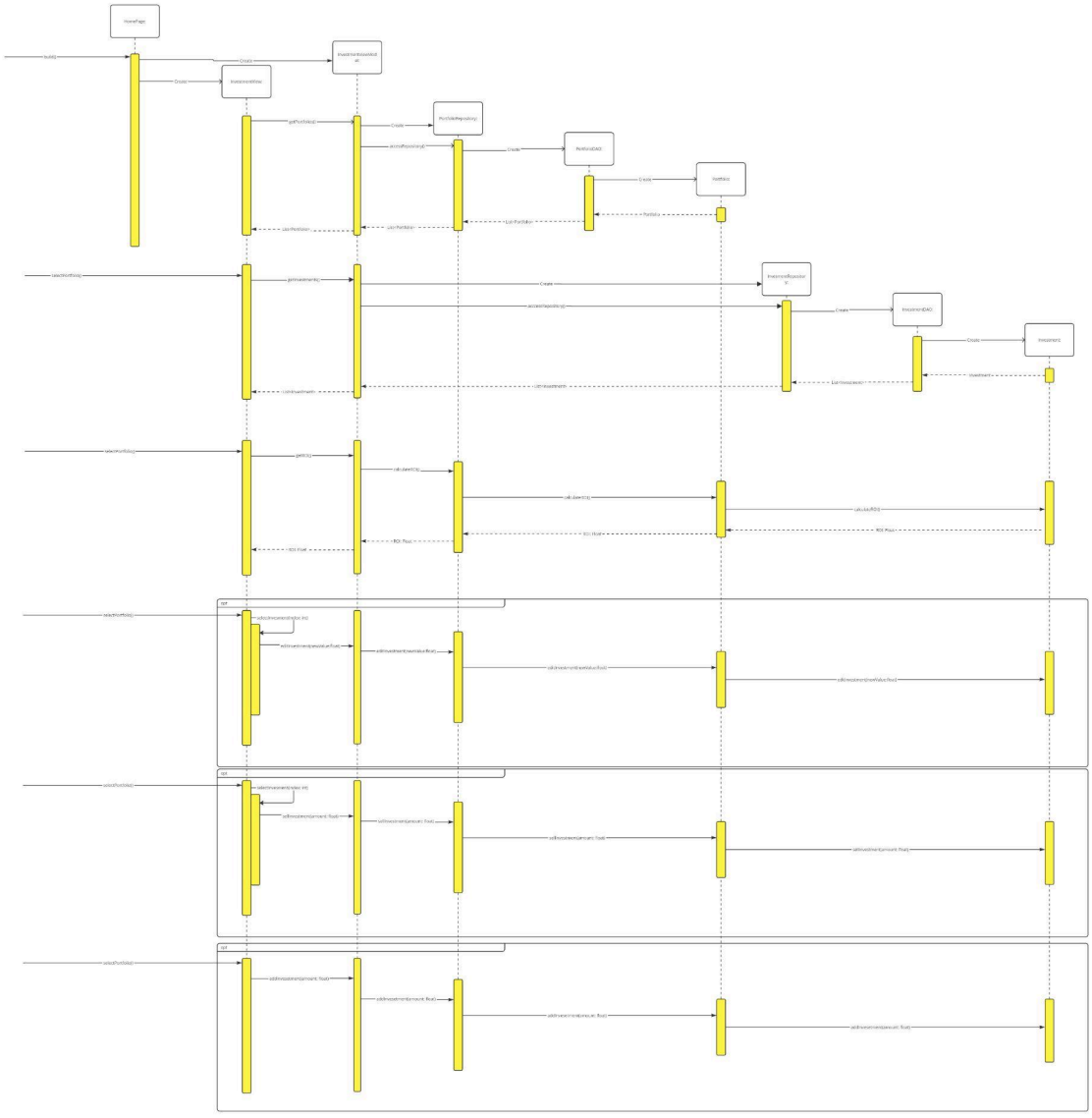


4.3 Sequence Diagrams

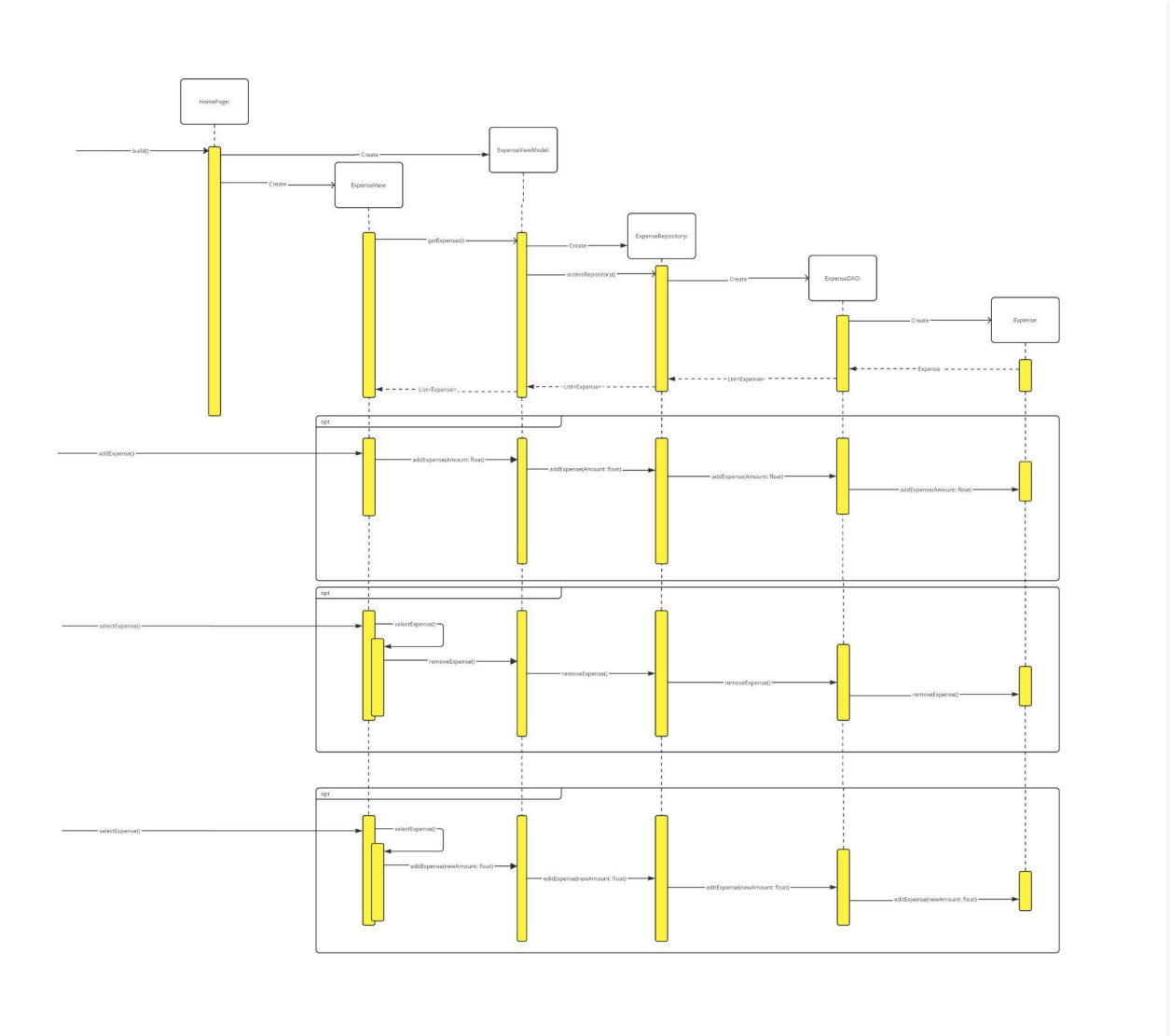
1) Login - Signup Sequence diagram



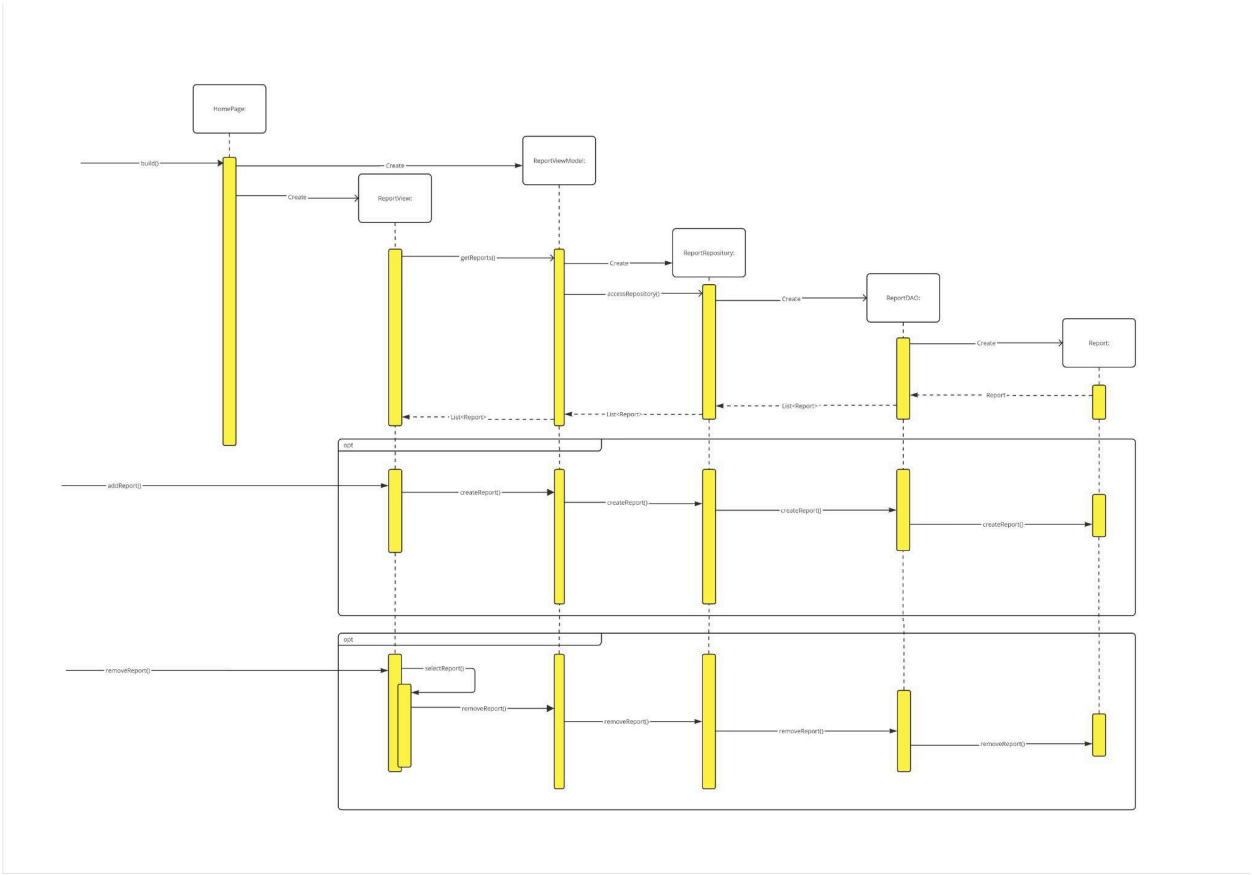
2) Investment Sequence diagram



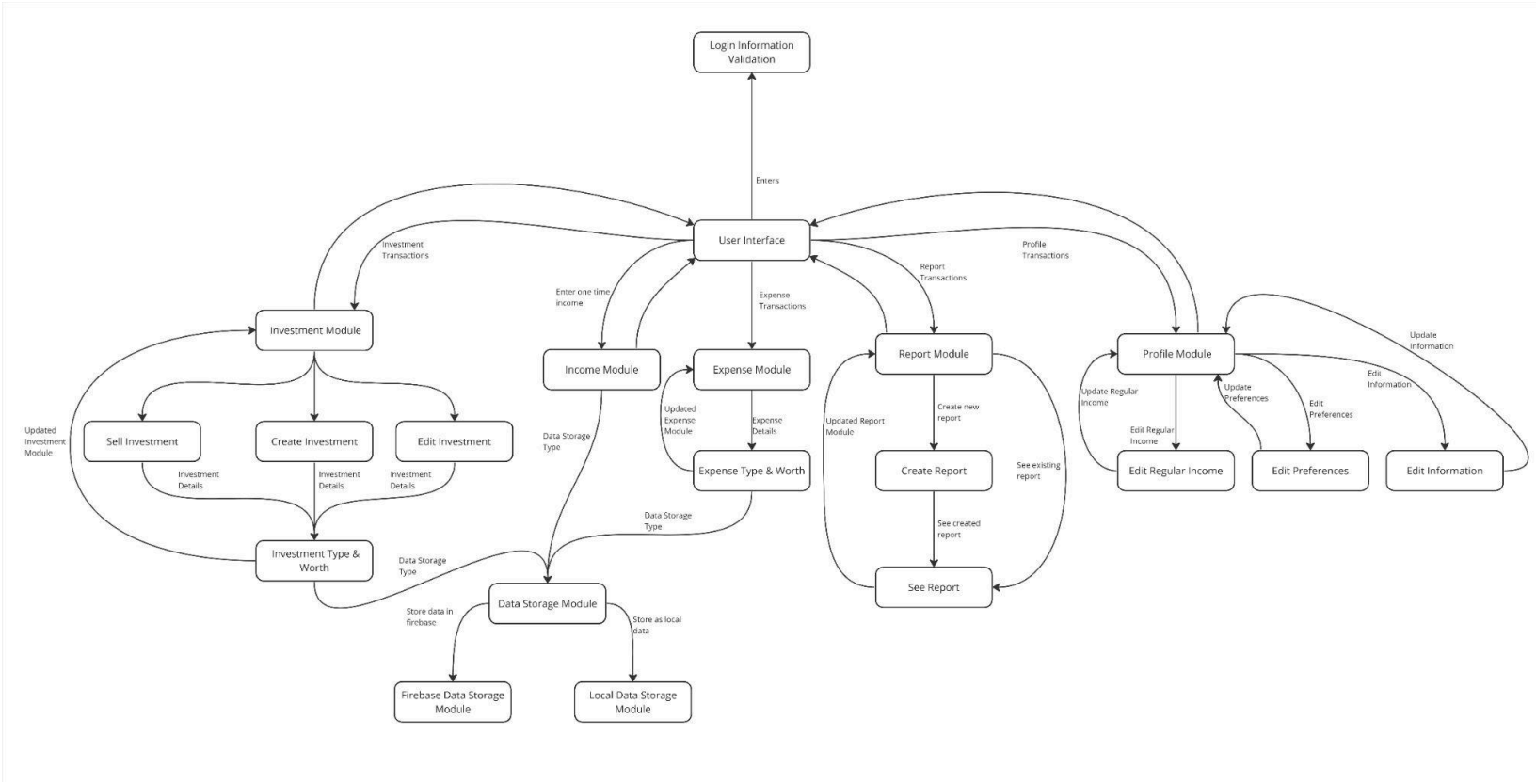
3) Expense Sequence diagram



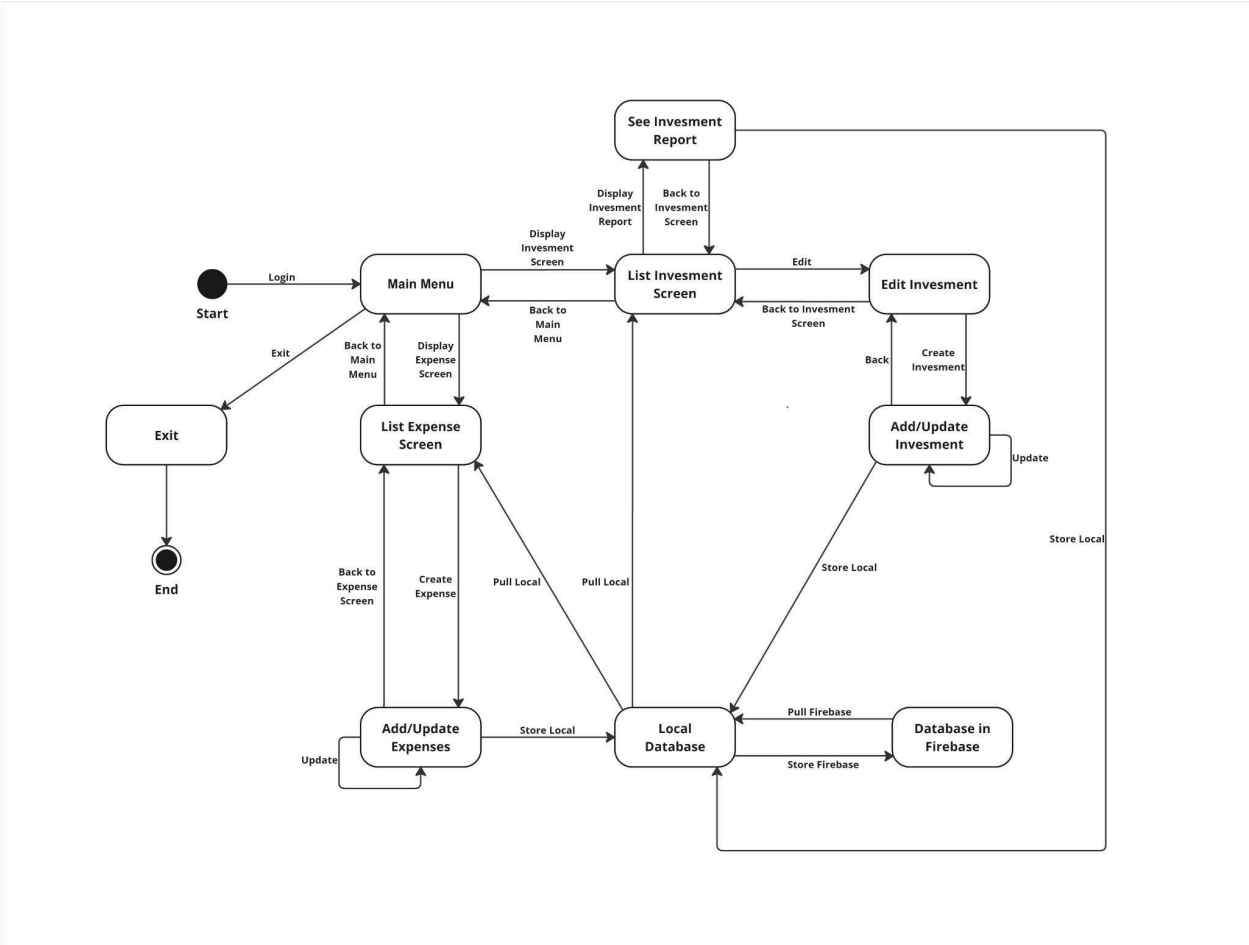
4) Report Sequence diagram



4.4 Data Flow Diagrams (DFD)



4.5 State-Transition Diagrams (STD)



A. Appendices

A.1 References

https://www.stickyminds.com/article/state-transition-diagrams

https://www.geeksforgeeks.org/state-transition-diagram-for-an-atm-system/

Ps Slides

https://www.geeksforgeeks.org/repository-design-pattern/

https://dotnettutorials.net/lesson/repository-design-pattern-csharp/

https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93viewmodel

https://www.geeksforgeeks.org/introduction-to-model-view-view-model-mvvm/

https://developer.android.com/topic/architecture/

https://www.geeksforgeeks.org/repository-design-pattern/

https://medium.com/@codemax120/flutter-clean-architecture-mvvm-f8802e3df564

https://medium.com/@ximya/clean-your-ui-code-in-flutter-7c58bf3e267d

https://www.youtube.com/watch?v=TRGQHg8jae8&ab_channel=FenyxAcademy

A.2 Teamwork

Mustafa Emir Uyar	Introduction, Use Case Diagram, Class Diagram
Muhammed Hayta	General Description, Use Case Diagram, Sequence Diagrams
Efe Özgen	Non-Functional Requirements, Use Case Diagram, Data Flow Diagram
Necati Koçak	Functional Requirements, Use Case Diagram, State Transition Diagram

Note: Other parts that we didn't mentioned are written together (face-to-face)

A.3 All diagrams with vector format.