# Software Requirements Specification

for

# LOONIE

Version 3.4 approved

Prepared by:

Kajivan Thiruchelvam (301280496), Marianna McCue (301481506) &
Marlon Haynes (301111380)

COMP225 Software Requirements Engineering, Section 403

April 9, 2025

## Table of Contents

## Revision History

| Name | Date | Reason For Changes | Version |
|---|---|---|---|
| Marianna McCue | Jan 16 | Creation of Section 1 | Version 1.0 |
| Kajivan Thiruchelvam | Jan 17 | Creation of Section 2 | Version 1.1 |
| Marianna McCue | Jan 17 | Creation of Section 3 | Version 1.2 |
| Marlon Haynes | Jan 22 | Creation of Stakeholder Register | Version 1.3 |

| | | | |
|---|---|---|---|
| Kajivan Thiruchelvam | Jan 23 | Creation of Interview Question | Version 1.4 |
| Marianna McCue | Jan 24 | Creation of color code and Interview questions edits | Version 1.5 |
| Marlon Haynes | Jan 24 | Rearrange Interview Questions | Version 1.6 |
| Kajivan Thiruchelvam | Jan 27 | Creation of Functional & Non-Functional Requirements | Version 1.7 |
| Marianna McCue | Jan 28 | FR & NFR Formatting, update users, update table of contents pages | Version 1.8 |
| Marianna McCue | Jan 31 | Edits to FR & NFR after team review | Version 1.9 |
| Marlon Haynes | Feb 4 | Creation of Use Cases | Version 2.0 |
| Marianna McCue | Feb 7 | Use Cases edits and formatting | Version 2.1 |
| Kajivan Thiruchelvam | Feb 11 | Addition of Use Case Description & Activity Swim Lane Diagram | Version 2.2 |
| Marianna McCue | Feb 12 | Addition of Use Case Diagram & formatting edits | Version 2.3 |
| Marianna McCue | Feb 14 | Addition of Domain Class Diagram | Version 2.4 |
| Kajivan Thiruchelvam | March 1 | CRC index | Version 2.5 |
| Marianna McCue | March 7 | Update Domain class diagram and CRC cards | Version 2.6 |
| Marlon Haynes | Mar 11 | Update Class Diagram | Version 2.7 |
| Marlon Haynes | Mar 13 | Gen Ai Class Diagram | Version 2.8 |
| Marianna McCue | Mar 14 | Update Domain class diagram and CRC cards based on feedback | Version 2.9 |
| Kajivan Thiruchelvam | March 27 | UML Sequence Diagram | Version 3.0 |
| Marlon Haynes | Mar 28 | State Diagrams | Version 3.1 |
| Marlon Haynes | Mar 29 | State Diagrams Update After Prof Review | Version 3.2 |
| Marianna McCue | April 2 | Party Analysis Pattern updates | Version 3.3 |
| Marianna McCue | April 8 | Final formatting | Version 3.4 |

## 1.1 Purpose

The purpose of this Software Requirements Specification (SRS) document is to define the requirements for "Loonie," a personal finance tracking application designed to address the growing need for effective personal finance management. As individuals increasingly navigate complex financial landscapes, they often struggle with budgeting, expense tracking, and overall financial awareness. Loonie aims to solve these challenges by providing users with a comprehensive tool to monitor their financial activities, set budgets, and achieve financial goals.

This document details the core functionalities and user interactions required for the successful implementation and operation of the Loonie software. By delivering an intuitive user interface and robust features, Loonie empowers users to take control of their finances, promoting informed decision-making and enhancing financial literacy.

## 1.2 Document Conventions

This Software Requirements Specification (SRS) document adheres to specific conventions and formatting standards to ensure clarity and consistency throughout. The following conventions have been used:

1. **Formatting Styles:**
    - **Bold Text**: Used to highlight key terms, section headings, and important concepts throughout the document. For instance, terms like “Budgeting Tools” and “Expense Tracking” will be in bold to distinguish them as significant features of the Loonie app.
    - **Italics:** Utilized for illustrative examples, such as user interactions or descriptions of user interface elements. This helps distinguish between general text and specific references to functionality.
2. **Acronyms:**
    - All acronyms are defined at their first usage in the document. For example, "User Interface (UI)" is presented in full first, with the acronym in parentheses. A Table of Acronyms is included to list and define frequently used terms for quick reference.

| Acronym | Description |
|---|---|
| API | Application Programming Interface |
| DB | Database |
| DBMS | Database Management System |
| FTP | File Transfer Protocol |
| HTTPS | Hypertext Transfer Protocol Secure |
| IT | Information Technology |

| | |
|---|---|
| PC | Personal Computer |
| ROI | Return on Investment |
| SDK | Software Development Kit |
| SRS | Software Requirements Specification |
| SQL | Sequential Query Language |
| UI | User Interface |
| UX | User Experience |
| FR | Functional Requirement |
| NFR | Nonfunctional Requirement |

3. **Color code for stakeholders and actors:**

| | |
|---|---|
| 🟥 | Investor |
| 🟧 | Marketing |
| 🟨 | System Administrator |
| 🟩 | Developer |
| 🟦 | Project Manager |
| 🟪 | User |
| ▨ | Tester |
| 🟦 | APIs |

4. **Use Case Numbering:**

   - Use cases are numbered according to the section they appear in for easier reference. For instance, if the use case relates to the external interface that allows users to connect their bank accounts, it may be numbered as UC-03.1 (User Account Linking) to indicate its association with the requirements outlined in Section 3.1 (User Interfaces).

5. **Priorities:**

   - Priorities for higher-level requirements are assumed to be inherited by the detailed requirements. For instance, if a primary feature like “Expense Tracking” is specified, all related detailed functionalities will follow that priority.

By adhering to these conventions, the document aims to provide a clear and organized framework for discussing the requirements and functionalities of the Loonie personal finance tracking app, making it accessible and understandable to all stakeholders involved in the project.

## 1.3 Intended Audience and Reading Suggestions

This document is intended for a diverse audience, including:

a. **Developers:** Responsible for implementing the technical requirements needed for the app's development.

b. **Project Managers:** Key stakeholders overseeing timelines and resources, using this document to track progress and align project goals with user needs.

c. **Testers:** Ensuring that the app's functionalities meet requirements and provide a smooth user experience.

d. **Documentation Writers:** Creating manuals and guides to help users navigate and utilize the Loonie app effectively.

e. **Investors:** Stakeholders in the company who are interested in understanding the performance and growth potential of the Loonie app. They require insights into how the app meets market needs, its user engagement levels, and overall financial viability to assess their investment strategies.

f. **Marketing Staff:** Professionals focused on understanding user needs and market strategies, gaining insights into how Loonie addresses personal finance challenges.

g. **Sales staff:** Team members who promote and sell the Loonie app to potential users. They need to understand its features and benefits to effectively communicate how it can help users with their personal finance needs.

## 1.4 Product Scope

Loonie is a personal finance tracking application specifically designed to address the widespread challenges individuals face in managing their finances effectively. By enabling users to link their bank accounts, Loonie provides a comprehensive view of their financial health, allowing for real-time expense tracking and budgeting.

Key features of Loonie include:

- **Expense Tracking**: Users can automatically categorize and monitor their spending, making it easier to identify spending habits and areas for improvement.
- **Budgeting Tools**: The app allows users to set personalized budgets tailored to their financial goals, promoting disciplined spending and savings.
- **Financial Goal Setting**: Users can establish and track progress toward specific financial goals, such as saving for a major purchase or paying off debt.
- **Bill Alerts**: Loonie sends real-time notifications about upcoming bills, helping users avoid late payments and better manage their cash flow.

The primary goal of Loonie is to empower individuals to achieve financial literacy and stability, enhancing their ability to make informed financial decisions. By providing intuitive tools and insights, Loonie supports users in navigating their financial journeys and aligns with broader goals of promoting financial wellness and encouraging digital engagement in personal finance management.

## 1.5 References

Cadabra Studio. (2023, December 20). *All about software requirements specification (SRS) + bonus template*, from https://cadabra.studio/what-is-software-requirements-specification-and-how-to-write-it-with-example/

Coursera. (2023, April 13). *5 types of programming languages,* from https://www.coursera.org/articles/types-programming-language

Figma. (n.d.). *How to create a context diagram*, from https://www.figma.com/resource-library/context-diagram/

Intuit. (n.d.). *Loonie: Personal finance & budgeting app*, from https://Loonie.intuit.com

Keith, C. (2010, August 23). *Defining system timing requirements*. EET Times, from https://www.eetimes.com/defining-system-timing-requirements/

Krüger, G., & Lane, C. (2023, January 17). *How to write an SRS document (Software Requirements Specification document)*, from https://www.perforce.com/blog/alm/how-write-software-requirements-specification-srs-document

Mitchell, G. (2022, September 5). *What is File Transfer Protocol (FTP) and what is it used for?* Investopedia, from https://www.investopedia.com/terms/f/ftp-file-transfer-protocol.asp

MySQL. (n.d.). *MySQL documentation*, from https://dev.mysql.com/doc/

Process Street. (n.d.). *How to use Visio to create a system context diagram*, from https://www.process.st/how-to/use-visio-to-create-a-system-context-diagram/

QAT Global. (2012-2025). *Detailing the external interfaces in the software requirements*, from https://qat.com/external-interfaces-software-requirements/#:~:text=Provide%20detailed%20and%20unambiguous%20specifications,and%20minimizes%20misinterpretations%20or%20misunderstandings.

## 2.1 Product Perspective

**Overview:** Loonie is a new, self-contained personal finance tracking application designed specifically to help users manage their financial activities effectively. It draws inspiration from existing financial management solutions, particularly the Mint app, which has established a strong foothold in the market. However, Loonie aims to provide a unique user experience tailored to Canadian users by integrating specific features and functionalities that cater to local financial practices.

**Context:** Loonie serves as an innovative entry into the personal finance software market, addressing the growing need for effective financial management tools. In contrast to Mint, which is geared primarily toward the U.S. market, Loonie focuses on the Canadian demographic, offering relevant functionalities such as integration with Canadian banks through the Flinks API and synchronized calendar features with both Google Calendar and iCal.

**Positioning:**

- **New Product:** Loonie is a new software application that does not serve as a replacement for any existing system. Instead, it is designed to stand alone, leveraging modern technology and user-centered design principles to deliver a robust personal finance management experience.
- **Relation to Larger Systems:** While Loonie operates independently, it interfaces with external systems through APIs (e.g., Flinks for bank integration and Firebase Cloud Messaging for notifications) to enhance user interaction with financial data and alerts.

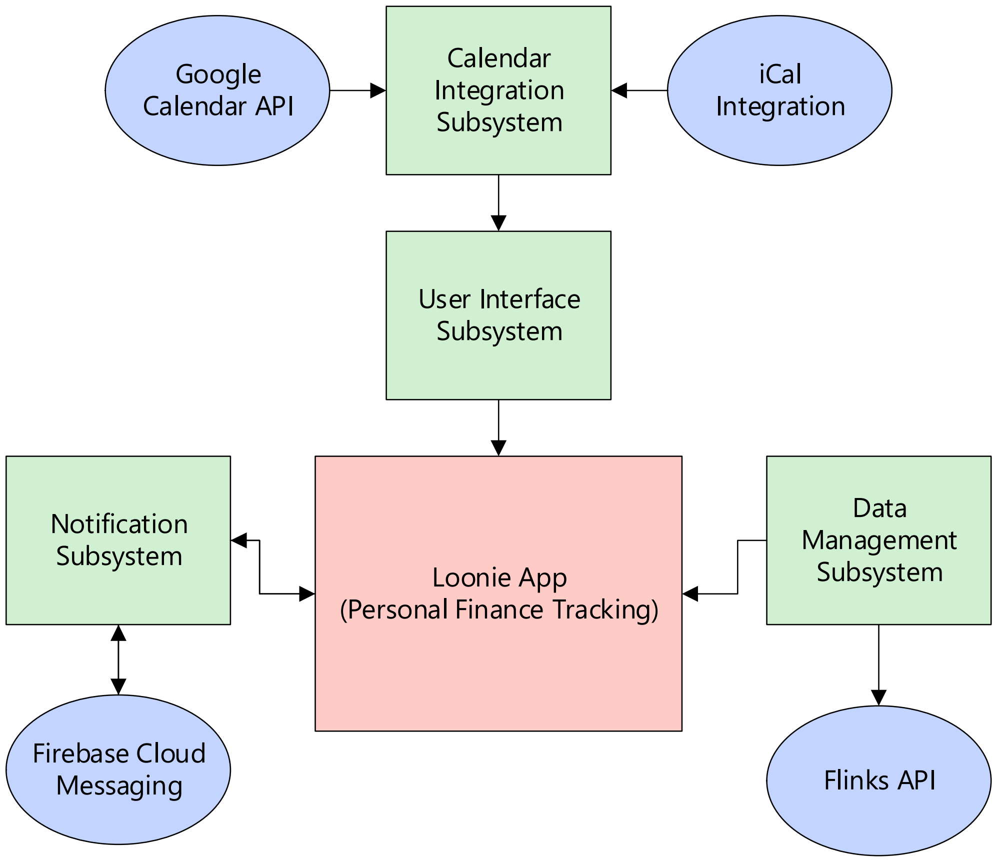
**Subsystem Interconnections:** Loonie consists of several core subsystems that work together to provide a seamless user experience:

- **User Interface Subsystem:** This includes both the mobile and web interface that users interact with.
- **Data Management Subsystem:** This handles data collection, storage, and integration with the Flinks API for bank transactions.
- **Notification Subsystem:** This is responsible for sending out alerts and notifications to users using Firebase Cloud Messaging and email services.
- **Calendar Integration Subsystem:** This manages the synchronization of financial events with Google Calendar and iCal.

**External Interfaces:** Loonie also interacts with various external components such as:

- **Flinks API:** For accessing user financial data securely.
- **Google Calendar API:** To sync important dates and reminders with users' calendars.
- **Firebase Cloud Messaging:** For sending real-time notifications to users.

**System Context Diagram:**


Google Calendar API
Calendar Integration Subsystem
iCal Integration
User Interface Subsystem
Notification Subsystem
Loonie App (Personal Finance Tracking)
Data Management Subsystem
Firebase Cloud Messaging
Flinks API

## 2.2 Product Functions

The Loonie app will provide the following core functionalities:

- **User Account Management**:
    - Allow users to create, manage, and delete their accounts.
    - Provide features for secure login and authentication, including options for single sign-on (SSO).
- **Expense Tracking**:
    - Enable users to automatically categorize and monitor their spending in real-time.
    - Allow users to manually add and categorize expenses.
- **Budgeting Tools**:
    - Facilitate the creation of personalized budgets based on user-defined categories.
    - Provide visual representations of budget adherence and spending trends.
- **Financial Goal Setting**:
    - Allow users to set specific financial goals (e.g., saving for a vacation) and track their progress toward achieving these goals.
- **Bill Alerts and Notifications**:
    - Send real-time notifications about upcoming bills and budget limits to help users stay on track.
- **Bank Account Integration**:
    - Enable users to securely link their bank accounts using the Flinks API to pull transaction and balance data.
- **Calendar Integration**:
    - Allow users to sync important financial events (such as bill due dates) with Google Calendar and iCal.
- **Reporting and Insights**:
    - Generate visual reports showing spending habits, budget adherence, and financial progress over time.

## 2.3 User Classes and Characteristics

The Loonie app is designed to cater to various user classes, each with distinct needs and characteristics. The anticipated user classes are as follows:

1. **Users**:

   - **Description**: Users are the primary users of Loonie, leveraging the product to fulfill its intended purpose. For example, they may use Loonie for financial tracking, analytics, or transactions, depending on the product's scope.
   - **Characteristics**:
     - Varied skill levels, from tech-savvy to novice.
     - Highly goal-oriented and outcome-focused.
     - Need an intuitive and user-friendly interface.
     - Expect reliability, performance, and security.

2. **Investor**
   - **Description:** Investors are stakeholders interested in the financial performance and growth potential of Loonie. They analyze data to assess return on investment and long-term viability.
   - **Characteristics:**
     - Strong analytical skills and business acumen.
     - Focused on financial metrics, growth potential, and market trends.
     - Interested in user engagement and satisfaction metrics.
3. **Project Manager**:
   - **Description**: Project managers use Loonie to oversee the development lifecycle, monitor progress, assign tasks, and ensure the project is delivered on time and within scope. Loonie might also serve as a hub for managing timelines, budgets, and stakeholder communication.
   - **Characteristics**:
     - Strong organizational and leadership skills.
     - Proficient in task management and resource allocation.
     - Effective communicators and decision-makers.
     - Highly adaptable and results driven.
4. **Tester**:
   - **Description** Testers use Loonie to execute manual and automated tests, document bugs, and validate fixes. They ensure the product meets quality standards by identifying and reporting inconsistencies or performance issues.
   - **Characteristics**:
     - Detail-oriented with a focus on quality.
     - Skilled in debugging and test automation.

    - Persistent and curious.
    - Proficient in problem identification and resolution.

5. **System Administrator**:
    - **Description**: System administrators use Loonie to manage system configurations, monitor performance, and ensure uptime. They handle tasks like access control, troubleshooting, and optimizing backend processes.
    - **Characteristics**:
        - Proficient in system management tools and technologies.
        - Strong troubleshooting and problem-solving skills.
        - Highly detail-oriented with a focus on system stability.
        - Skilled at multitasking and working under pressure.
6. **Developer**:
    - **Description**: Developers work on creating, maintaining, and improving the Loonie app, including coding, integrating APIs, and implementing new features. They also address user support issues and troubleshooting tasks to ensure smooth application performance.
    - **Characteristics**:
        - Patient and customer focused.
        - Calm under pressure with a proactive approach.
        - Strong analytical and problem-solving skills.
        - Proficient in various programming languages and frameworks.
        - Skilled in diagnosing and resolving technical issues.
        - Detail-oriented and focused on optimizing performance.
        - Collaborative and capable of working independently.
7. **Marketing**:
    - **Description**: The marketing team utilizes Loonie to plan, execute, and analyze marketing campaigns aimed at increasing user acquisition and engagement. They are also responsible for creating and managing content such as user guides, tutorials, FAQs, and promotional materials.
    - **Characteristics**:
        - Creative thinkers with strong analytical skills.
        - Proficient in digital marketing tools and analytics.

        - Audience-focused with a strong understanding of trends.
        - Collaborative and results oriented.
        - Skilled in written and visual communication for content creation.

## 2.4 Operating Environment

The Loonie app will operate within the following environments and utilize the specified technologies:

- **Hardware Platform**:
    - The application will be deployed on cloud-based servers to ensure scalability and high availability.
    - Mobile versions will be designed for iOS and Android smartphones and tablets.
- **Operating Systems**:
    - **Mobile**:
        - iOS 14 and later for Apple devices, supporting iPhones and iPads.
        - Android 10 and later for Android devices.
    - **Web**:
        - Compatible with Windows and macOS operating systems for desktop browser access.
- **Supported Browsers**:
    - The web version of the app will support the latest versions of:
        - Chrome
        - Firefox
        - Safari
        - Microsoft Edge
- **Database**:
    - Loonie will utilize a relational database system, such as PostgreSQL or MySQL, to store user data, transaction history, budgeting information, and settings securely.
- **Development Technologies**:
    - The mobile app will be developed using React Native for cross-platform compatibility, allowing for streamlined updates across iOS and Android.
    - The web app will be developed using HTML, CSS, and JavaScript, potentially utilizing frameworks like React or Angular for a responsive user interface.

- **Third-Party Services**:
    - The app will integrate with:
        - **Flinks API** for bank account connectivity and transaction data retrieval.
        - **Firebase Cloud Messaging** for push notifications.
        - **Google Calendar API** and **iCal** for financial event synchronization.

## 2.5 Design and Implementation Constraints

## 2.6 User Documentation

## 2.7 Assumptions and Dependencies

The development and functionality of the Loonie app are based on several key assumptions and dependencies that may impact the requirements outlined in this SRS:

1. **Assumptions:**
2. **User Adoption**:
    - It is assumed that users will have a basic level of familiarity with mobile and web applications, enabling them to navigate and use the Loonie app effectively.
3. **Bank API Availability**:
    - The availability and reliability of the **Flinks API** are assumed for seamless integration with Canadian banks. If there are changes in API terms, access issues, or service disruptions, it could hinder data retrieval.
4. **Third-Party Services**:
    - It is assumed that **Firebase Cloud Messaging** and calendar services (Google Calendar and iCal) will remain stable and well-supported throughout the development and deployment phases. Changes in these services could require adjustments to application functionality.
5. **Regulatory Changes**:
    - It is assumed that there will be no significant changes in regulations regarding data privacy and financial services that could impact the app’s compliance requirements.
6. **Performance Expectations**:
    - It is assumed that the app will be able to handle a reasonable number of concurrent users without performance degradation, based on expected usage patterns.
7. **Dependencies:**

1. **Third-Party APIs**:

  - The functionality of the app heavily depends on the **Flinks API** for bank account integration. Any changes, limitations, or downtimes in this API could directly affect the app's ability to retrieve financial data.

2. **Cloud Hosting Services**:
   - The app is dependent on the chosen cloud hosting provider (e.g., AWS, Azure, or Google Cloud) for database and application hosting. Any outages or changes in service agreements could impact app availability and performance.
3. **Development Frameworks**:
   - The usage of frameworks for the mobile app (e.g., React Native) and the web app (e.g., React or Angular) introduces dependencies on those frameworks staying updated and supported. If critical vulnerabilities arise or if support ceases, the app's development may be impacted.
4. **User Feedback**:
   - The success of the app's features is somewhat dependent on user feedback during testing phases. This feedback will be crucial to refining features and addressing any unmet user needs.
5. **Market Research**:
   - The app's functionality and features are influenced by market research and trends in personal finance management. A significant shift in user preferences could require a re-evaluation of features planned for future releases.

## 3.1 User Interfaces

Loonie will be accessible to users through both a mobile app and a web interface, ensuring a flexible and user-friendly experience across multiple platforms.

**Mobile Interface:**

**Supported Platforms:**

- iOS: The app will be available on the Apple App Store for iPhones and iPads.
- Android: The app will be available on the Google Play Store for Android smartphones and tablets.

**User Interface Characteristics:**

- **Login Page:**
  - Users will first encounter a login page where they can enter their credentials (username and password).
  - Options for single sign-on (SSO) may be provided through Google and Facebook accounts for ease of access.

        - A "Forgot Password?" link will be available for users to recover their accounts.

    - **Main Page:**
        - Upon successful login, users will be directed to the main page, which provides an overview of their financial health.
        - The main page will prominently display the user's current budget status, recent transactions, and any notifications (e.g., bill reminders).
        - Key navigation links to other functionalities like Expense Tracking, Budgeting Tools, and Financial Goals will be accessible from the main page.
    - **Screen Layout:** Each screen will follow a standard layout featuring a navigation bar at the bottom, providing access to key functionalities such as Home, Budgeting, Expenses, and Goals.
    - **Error Messages:** All error messages will follow a standard format, displaying in red text with clear explanations and suggested corrective actions.
    - **Standard Buttons:** Every screen will include standard buttons such as:
        - 'Help' for accessing support documentation
        - 'Settings' for user customization options
        - 'Submit' or 'Save' buttons for confirming actions

- **Web Interface:**
    - **Supported Browsers:**
        - Compatible with the latest versions of Chrome, Firefox, Safari, and Edge to ensure a seamless user experience on desktop and laptop computers.
- **User Interface Characteristics:**
    - **Login Page:**
        - The web interface will also include a login page similar to the mobile app, allowing users to enter their credentials, with options for SSO and password recovery.
    - **Main Page:**
        - After logging in, users will land on the main page, similar to the mobile app, where vital financial information is displayed and accessible.
    - **Responsive Design:** The web interface will be responsive, ensuring usability on varying screen sizes, adjusting elements dynamically based on the device used.
    - **Accessibility Standards:** The web app will comply with Web Content Accessibility Guidelines (WCAG) to ensure that users with disabilities can effectively interact with the application.

**System Context Diagram:**

- A system context diagram will illustrate the interactions between users and the Loonie application, depicting the mobile and web interfaces, along with the backend services they communicate with.

## 3.2 Hardware Interfaces

Loonie primarily operates as a software application and does not require direct interaction with specialized hardware. However, it may utilize the following standard hardware components:

- **Mobile Devices:**
    - **Smartphones/Tablets:** Users will access Loonie on their mobile devices. The application will leverage the device's features, including GPS for location-based financial insights and notifications, and the camera for scanning receipts.
- **Web Browsers:**
    - **Personal Computers/Laptops:** Users can access the web interface via standard PCs and laptops, utilizing traditional input devices such as keyboards and mice.

**Supported Device Characteristics:**

- **Communication Protocols:**
    - HTTPS will be employed for secure communication between the application and the server, ensuring data privacy and protection during transmission.

**Data Interaction:**

- Loonie will collect user data through input forms (e.g., adding expenses, setting budgets) and synchronize with bank accounts using APIs, without the need for a physical interface with hardware like scanners or barcode readers.

## 3.3 Software Interfaces

Loonie will require integration with several third-party software components to enhance its functionality and provide users with a comprehensive personal finance management experience without direct payment processing.

**Required Third-Party Software Interfaces:**

1. **Bank Account Integration:**
    - **Flinks API:** Loonie will use the Flinks API to securely connect and access users' bank account information. This integration allows users to sync their transactions from various bank accounts and credit cards for real-time expense tracking.
    - **Data In/Out:** User financial data, including transaction histories and balances, will be pulled from their bank accounts and securely stored in Loonie's database.

2. **Calendar Integration:**
    - **Google Calendar API:** Loonie will integrate with Google Calendar to allow users to sync important financial events, such as bill due dates and budget reminders. This will help users stay organized and ensure they never miss a payment.
    - **iCal Integration:** Loonie will also integrate with iCal for Apple users to sync important financial events in their Apple calendar applications.
    - **Data In/Out:** Event details (such as due dates and reminders) will be sent to and accessed from users' iCal calendars.
3. **Notifications Service:**
    - **Firebase Cloud Messaging (FCM):** Loonie will utilize FCM for sending push notifications to users about important updates, such as alerts for upcoming bills or reminders.
    - **Data In/Out:** Notification payloads will be sent from Loonie's backend to FCM, which will then forward them to users' mobile devices.

**Shared Data Items:**

- User profiles, transaction details, and reminders will be shared between these third-party APIs and Loonie, ensuring seamless functionality without the need for payment processing.

## 3.4 Communications Interfaces

## Analysis modelling scenario & class based

| **Use Cases** | | | |
|---|---|---|---|
| **Use case name** | **List of related Requirements ID** | **Actor(s)** | **Brief Description** |
| **Create service request** | FR01 | User | The user navigates to the “Support” or “Service Request” section within the Loonie application.<br><br>The user clicks on the "Create Service Request" button.<br><br>The system presents a service request form that prompts the user to select a service type and sub-type from predefined options. |

| | | | |
|---|---|---|---|
| | | | The user fills out the necessary details within the form, including a description of the issue or request.<br><br>Upon submitting the form, the system generates a unique service request number for tracking purposes.<br><br>The system sends a confirmation email to the user, including the service request number and details of the request.<br><br>The request is logged in the system for further processing by customer support. |
| **User Login and Authentication** | FR01, NFR09 | System Administrator<br><br>User | The user opens the Loonie application.<br><br>The user clicks on the login button.<br><br>The system prompts the user to enter The user opens the Loonie application.<br><br>The user clicks on the login button.<br><br>The system prompts the user to enter their username and password.<br><br>The system ensures secure authentication through a username/password system, supporting single sign-on (SSO) through Google and Facebook.<br><br>The user submits their credentials.<br><br>If two-factor authentication (2FA) is enabled, the system prompts the user to enter the verification code sent to their registered device.<br><br>The System Administrator monitors the authentication system to ensure security measures are properly enforced, including compliance with encryption standards and 2FA protocols. |

| | | | |
|---|---|---|---|
| | | | The system verifies the credentials and the 2FA code. If both are correct, it grants access to the user’s profile.<br><br>If authentication fails, the user is notified and given an option to reset their password. |
| **Track and Categorize Expenses** | FR02, NFR03 | Flinks API<br><br>User | The user navigates to the "Track Expenses" section of the app.<br><br>The user selects an option to log a new expense.<br><br>The system prompts the user to enter details such as amount, date, and category.<br><br>The user can categorize the expense using predefined categories or create a custom category.<br><br>The user submits the expense.<br><br>Simultaneously, the Flinks API accesses the user’s linked bank accounts to pull relevant transaction data.<br><br>The system integrates the imported transactions from the Flinks API and automatically categorizes them based on the user’s custom rules or predefined categories.<br><br>The user’s expense tracking profile is updated to reflect both manually entered and imported transactions, providing a comprehensive overview of spending. |
| **Manage and Set Budgets** | FR03, NFR01 | User | The user accesses the budgeting feature within the app.<br><br>The user clicks on the "Set Budget" button. |

| | | | |
|---|---|---|---|
| | | | The system prompts the user to define their budget categories and amounts for the month.<br><br>The user enters the details and submits the budget.<br><br>The system dynamically adjusts the budget based on imported transaction data from linked accounts.<br><br>The user can view updates and status of their budget on their profile. |
| **Notifications and Reminders for Bill Payments and Budgeting Alerts** | FR04, NFR04 | User | The user opts into notifications during account setup.<br><br>The system tracks due dates and budget limits.<br><br>At appropriate times, the system sends push notifications to the user's device regarding upcoming bill payments, budget limits, and financial milestones.<br><br>The user receives these alerts to help manage their spending effectively. |
| **Integrate Financial Events with Google Calendar** | FR05, NFR08 | Yahoo Finance API | The user navigates to the calendar integration feature within the Loonie application.<br><br>The user selects financial events (e.g., bill due dates and budget reminders) that they wish to sync with their Google Calendar.<br><br>The system connects with the user's Google Calendar and prompts for permissions if necessary. |
| | | User | The user confirms which events to sync.<br><br>The system processes the request, and any relevant financial data is |

| | | | |
|---|---|---|---|
| | | | retrieved from the Yahoo Finance API, ensuring the user has the latest financial market updates or related events when syncing.<br><br>The system updates the Google Calendar with the selected financial events, including any data enhanced by the Yahoo Finance API if applicable. |
| **Generate Financial Insights and Recommendations** | FR06, NFR02 | User<br><br>Yahoo Finance API | The user navigates to the insights section of the app.<br><br>The system analyzes the user's spending patterns based on logged expenses and bank data.<br><br>The Yahoo Finance API is accessed to gather external financial data trends, such as market conditions or investment opportunities relevant to the user's financial behavior.<br><br>The system processes the user's data alongside the data retrieved from the Yahoo Finance API.<br><br>Based on this analysis, the system generates personalized insights and recommendations for the user, such as tips for saving money or improving financial habits.<br><br>The user reviews the insights provided and can act on suggestions to enhance their financial management. |
| **Search and Filter Transaction History** | FR07, NFR10 | User | The user accesses the transaction history feature from the profile.<br><br>The user selects options to search or filter transactions by various criteria (date range, amount, category).<br><br>The system processes the user's request and displays matching transactions. |

| | | | |
|---|---|---|---|
| | | | The user reviews filtered results to analyze past spending. |
| **Visualize Financial Data with Graphs and Charts** | FR08, NFR06 | User | The user navigates to the visualization feature within the application.<br><br>The system retrieves financial data, including expenses and budget status.<br><br>The user selects which types of graphs or charts to view (e.g., income vs. expenses, spending by category).<br><br>The system generates and displays the selected graphs or charts, providing visual insights into the user's financial trends.<br><br>The user can interact with the visual data, such as hovering over elements to see detailed descriptions or selecting specific date ranges for analysis.<br><br>The user can save or share these visualizations as reports or screenshots for future reference. |
| **Set and Monitor Financial Goals** | FR09, NFR03 | User | The user navigates to the financial goals setting feature in the app.<br><br>The user selects the option to create a new financial goal (e.g., saving for a vacation).<br><br>The system prompts the user to enter goal details, such as the goal amount, target date, and category.<br><br>Upon submission, the system saves the goal and sets reminders for progress tracking.<br><br>The user can view their current progress towards the goal on their profile, which updates dynamically as they log expenses and savings. |

| | | | |
|---|---|---|---|
| **Customize Profile to Display Relevant Financial Metrics** | FR10, NFR07 | System Administrator | The user accesses the profile customization feature within the app.<br>The system presents available widgets and metrics that can be displayed on the user's profile.<br>The user selects their preferred metrics, such as recent transactions, budget status, and goal progress. |
| | | User | The user arranges the widgets in their desired order on the profile.<br>The system saves the customization settings.<br>The System Administrator ensures that the profile customization functions efficiently by managing the backend configuration and ensuring data consistency.<br>The system updates the user's profile to reflect their choices, maintaining proper access control and performance standards, including backups and multi-region deployment, to ensure data safety and reliability. |

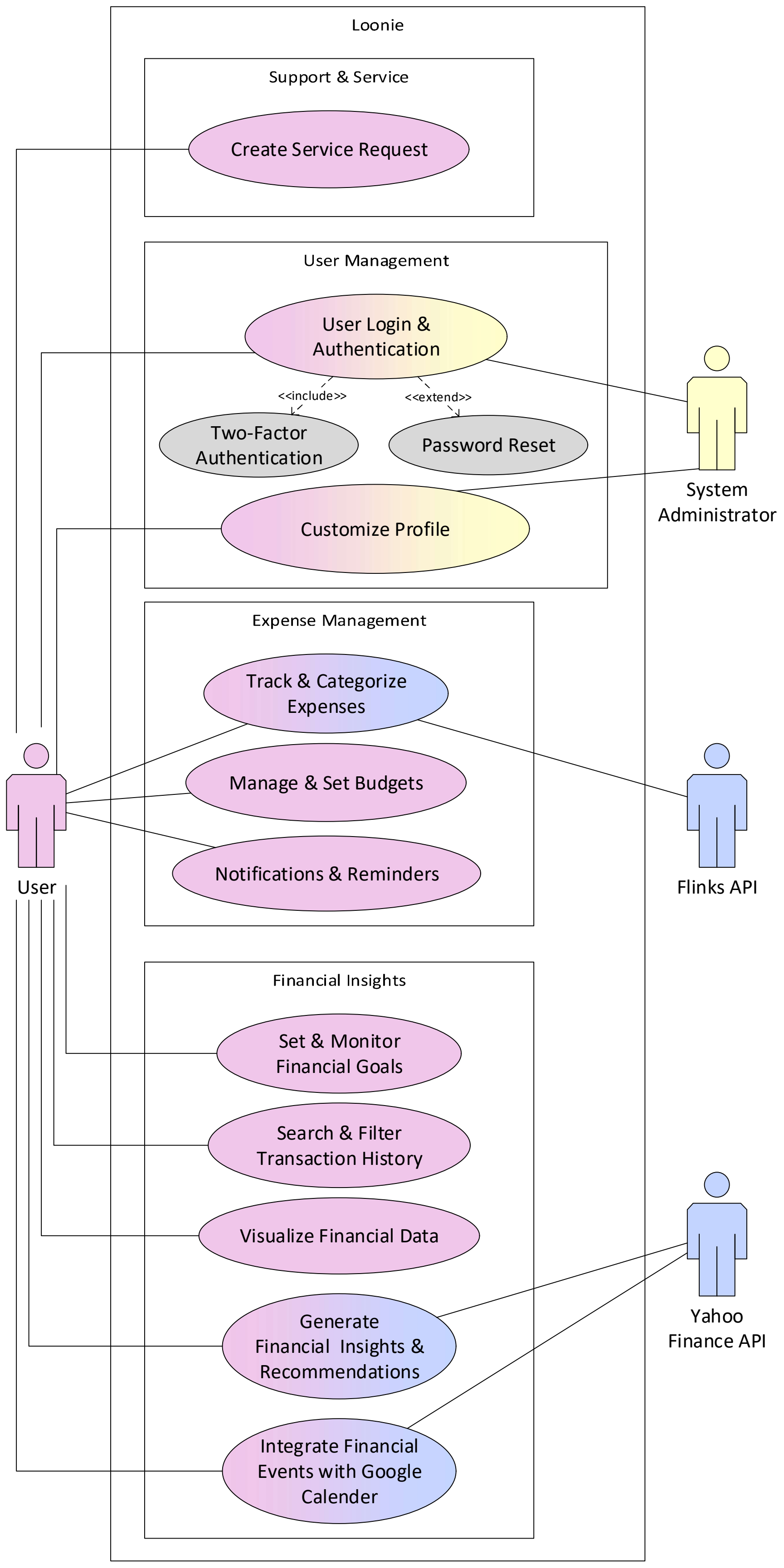
## 4.1 Functional Requirements

| **Functional Requirements List** | | | | |
|---|---|---|---|---|
| **Requirement ID** | **Requirement Title** | **Short Description** | **Priority** | **Requestor** |
| FR01 | **User Authentication** | Loonie must include a secure login system that supports username/password authentication, single sign-on (SSO) through Google and Facebook, and account recovery options such as a "Forgot Password?" link. | Expected | Developer |
| | | | | User |
| | | | | System Administrator |

| | | | | |
|---|---|---|---|---|
| FR02 | **Expense Tracking** | Users must be able to log their expenses manually and categorize them using predefined or custom categories, ensuring greater flexibility and personalization. The app should also support importing transaction data from bank accounts via Flinks API. These features align with user needs for effective tracking and the business goal of increasing user engagement. | Expected | User |
| | | | | Developer |
| | | | | Project Manager |
| | | | | Tester |
| FR03 | **Budget Management** | Users must be able to set, modify, and monitor monthly budgets. Real-time updates based on transaction data and visualizations like graphs or charts should provide clarity. Overseeing prioritization and alignment with the app's vision. | Expected | User |
| | | | | Project Manager |
| FR04 | **Notifications and Reminders** | The app should send push notifications and reminders for bill due dates, budgeting alerts (e.g., reminders when users are approaching their budget limits), and financial milestones using Firebase Cloud Messaging. This feature will help users stay on track with their budgeting goals and avoid overspending. | Exciting | Developer |
| | | | | User |
| | | | | Tester |
| FR05 | **Calendar Integration** | The app must integrate with Google Calendar and iCal to sync important financial events, such as bill due dates and budget reminders. | Expected | Developer |
| FR06 | **Insights and Recommendations** | Loonie must analyze user spending patterns and provide personalized insights, tips for saving, and | Exciting | User |
| | | | | Marketing |

| | | | | |
|---|---|---|---|---|
| | | suggestions to improve financial habits. | | |
| FR07 | **Transaction Search and Filtering** | Users must be able to search for specific transactions using various filters such as date range, category, amount, and keywords. This feature will enable users to quickly locate and review past expenses, making financial management more efficient. | Expected | User<br>Tester<br>Developer |
| FR08 | **Financial Data Visualization** | Users must be able to view financial data through dynamic graphs and charts to track their progress and trends effectively. | Expected | User<br>Developer<br>Tester |
| FR09 | **Goal Setting** | Users must be able to set financial goals (e.g., saving for a vacation, paying off debt) and track their progress towards these goals. The app should provide visual indicators and reminders to encourage continuity and engagement. | Expected | User<br>Marketing<br>Project Manager |
| FR10 | **Customizable Profiles** | Users must be able to customize their profile to display preferred financial metrics, such as recent transactions, budget status, expense summaries, and goal progress. This customization should allow users to prioritize the information most relevant to their financial management needs. | Expected | Developer<br>Project Manager<br>User |

## 4.2 Use Case Diagram


Loonie
Support & Service
Create Service Request
User Management
User Login & Authentication
<<include>>
<<extend>>
Two-Factor Authentication
Password Reset
Customize Profile
System Administrator
Expense Management
Track & Categorize Expenses
Manage & Set Budgets
Notifications & Reminders
User
Flinks API
Financial Insights
Set & Monitor Financial Goals
Search & Filter Transaction History
Visualize Financial Data
Generate Financial Insights & Recommendations
Integrate Financial Events with Google Calender
Yahoo Finance API

## 4.3 Detailed Use Case Description

**Use Case:** Manage and Set Budgets

**Iteration:** 1

**Last Modification**: n/a

**Primary actor:** User

**Goal in context:** Enable users to set, modify, and monitor monthly budgets efficiently within the Loonie app.

**Preconditions:**

1. The user must be logged in into Loonie application.
2. The user must have an active account with at least one linked financial source (bank account or manual transactions).

**Trigger:** The user decides to set or modify their monthly budgets.

**Scenario:**

1. The user accesses the budgeting feature within the app from the main profile.
2. The user clicks on the “Set Budget” button.
3. The system prompts the user to define budget categories and assign amounts.
4. The user enters budget details and submits the budget.
5. The system validates the entered budget data to ensure accuracy and consistency.
6. The system saves the budget and updates the profile to reflect the new budget plan.
7. The system dynamically adjusts the budget based on imported transaction data from linked accounts.
8. The system generates real-time financial insights and notifies the user any significant budget changes.
9. The user can view real-time updates and track budget performance through financial visualizations.

**Expectations:**

1. If the user is not logged in, the system prompts the user to login in before accessing the budgeting feature.
2. If no financial data sources are linked, the system prompts the user to add at least one account or enter manual transactions before setting a budget.
3. If the user enters an invalid budget amount (Ex., negative values or exceeding financial limits), the system displays an appropriate error message.
4. If there is an issue retrieving transaction data, the system notifies the user and allows manual adjustment.
5. If the system experiences a service disruption, it notifies the user and provides alternative manual adjustment.

**Priority:** High priority, as budgeting is a core feature of the application.

**When available:** First Increment.

**Frequency of Use:** Regularly, expected monthly or weekly.

**Channel to Actor:** Mobile apps (iOS & Android), Web application.

**Secondary Actors:**

1. Financial Data Source (Bank APIs) – Provides transaction data for dynamic budget adjustments.
2. System Administrator - Ensure system stability and data integrity.

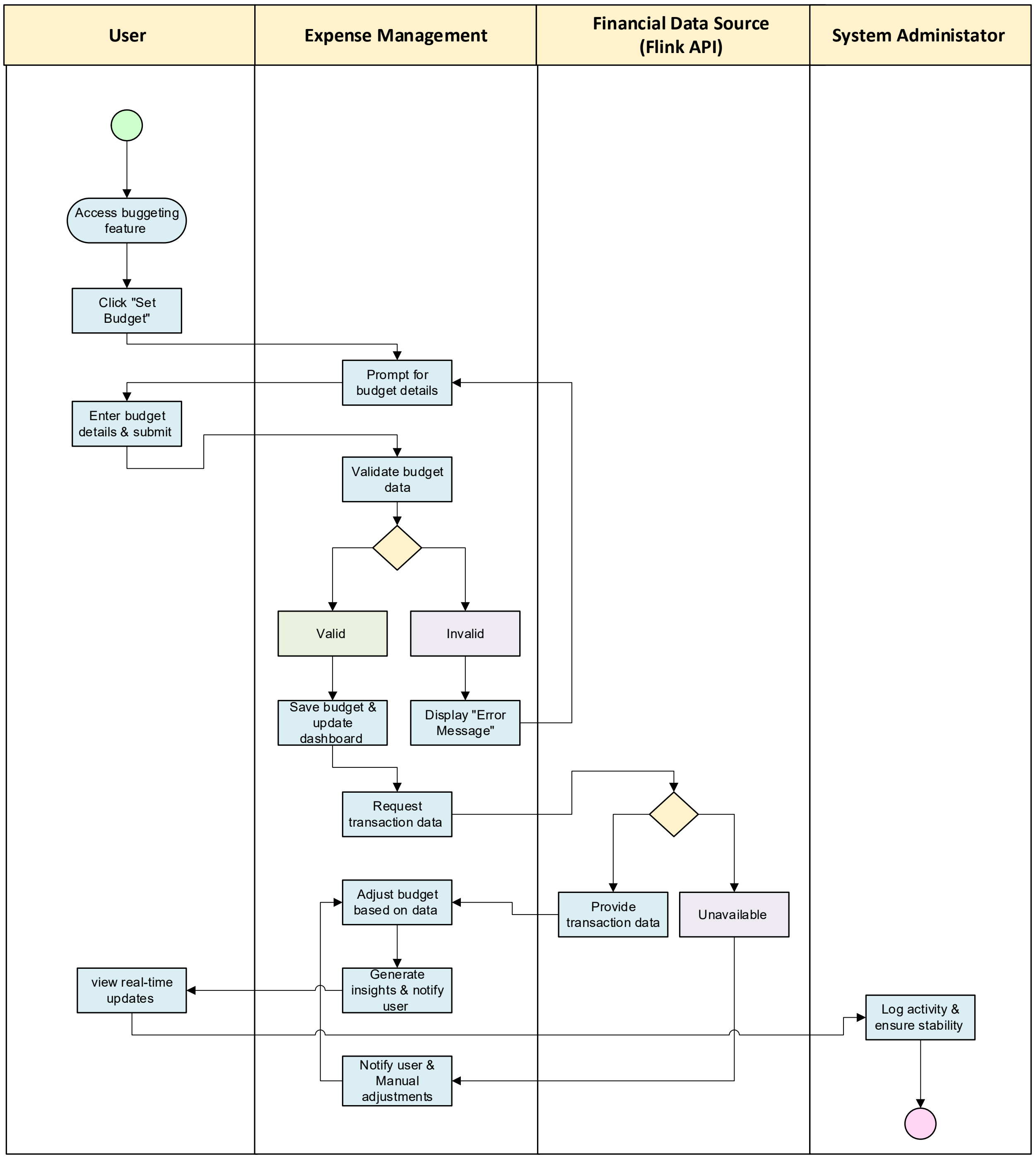
**Channels to Secondary Actors:**

1. Financial Data Source: Secure API communication (Ex., Flink API).
2. System Administrator: Internal database management and system maintenance.

**Open Issues:**

1. Should user be allowed to set budgets in multiple currencies?
2. Should the system support shared budgets for multiple users in a household?
3. Should AI-driven budget recommendations be introduced based on past spending habits?

## 4.4 Swim Lane Diagram


User
Expense Management
Financial Data Source (Flink API)
System Administator
Access buggeting feature
Click "Set Budget"
Prompt for budget details
Enter budget details & submit
Validate budget data
Valid
Invalid
Save budget & update dashboard
Display "Error Message"
Request transaction data
Provide transaction data
Unavailable
Adjust budget based on data
Generate insights & notify user
view real-time updates
Log activity & ensure stability
Notify user & Manual adjustments

## 5.1 Non-Functional Requirements

| Non-Functional Requirements List |
|---|

| Requirement ID | Requirement Title | Short Description | Priority | Requestor |
|---|---|---|---|---|
| NFR01 | **Scalability** | Loonie must support scaling resources dynamically based on user demand to ensure consistent performance under heavy loads. Scalability has been identified as a primary concern by stakeholders to accommodate growth and increasing user engagement without degradation in service quality. | Expected | System Administrator |
| NFR02 | **Data Privacy and Compliance** | The app must comply with GDPR, CCPA, and similar international data privacy regulations. This includes obtaining user consent, providing clear terms and conditions, and securing user data during storage and transmission. | Expected | Investor<br>System Administrator<br>Project Manager |
| NFR03 | **Uptime and Reliability** | The app must maintain an uptime of 99.9%, supported by a robust cloud hosting infrastructure with redundancy and failover mechanisms. | Expected | System Administrator<br>Project Manager |
| NFR04 | **Accessibility** | The web and mobile interfaces must meet Web Content Accessibility Guidelines (WCAG) to ensure usability for individuals with disabilities. | Expected | User<br>Marketing |
| NFR05 | **Performance Monitoring** | System performance must be monitored using tools like New Relic or Datadog to provide real-time insights and ensure quick responses to issues. | Expected | System Administrator |
| NFR06 | **Responsive Design** | The web interface must provide a seamless user experience across varying screen sizes, adjusting | Expected | Marketing |

| | | | | |
|---|---|---|---|---|
| | | dynamically to device dimensions. | | |
| NFR07 | **Disaster Recovery** | The system must include daily backups, multi-region deployment, and automated failover mechanisms to recover from outages efficiently. | Expected | System Administrator |
| NFR08 | **Multi-Platform Support** | The app should have a mobile interface for iOS and Android and a responsive web interface compatible with the latest versions of major browsers (Chrome, Firefox, Safari, Edge). | Expected | Marketing |
| | | | | User |
| NFR09 | **Security Measures** | The app must implement end-to-end encryption, two-factor authentication (2FA), and token-based authentication for secure API connections | Expected | Developer |
| | | | | System Administrator |
| NFR10 | **Error Handling** | The system must display clear, user-friendly error messages in red text with suggested corrective actions for all errors encountered. | Expected | Developer |
| | | | | User |

## Appendix A: Glossary

## Appendix B: Analysis Models

## Appendix C: Stakeholders Register

| Stakeholder Name | Stakeholder Position | External/ Internal | Stakeholders contact details | Operational/ Executive | Interest (high, medium, low) |
|---|---|---|---|---|---|
| Jason Carter | Developer | Internal | jasoncarter@hotmail.com | Operational | High |

| | | | | | |
|---|---|---|---|---|---|
| Emily Sanders | Project Manager | Internal | emilysanders@outlook.com | Executive | High |
| Liam Bennett | Tester | Internal | liambennett@gmail.com | Operational | High |
| Sophia Ramirez | System Administrator | Internal | sophiaramirez@messenger.com | Operational | High |
| Noah Brooks | Investor | External | noahbrooks@aol.com | Executive | High |
| Olivia Parker | Marketing | Internal | oliviaparker@hotmail.com | Operational | High |
| Ava Mitchell | User | External | avamitchell@outlook.com | Operational | High |

## Appendix D: Interview Questions

| Question | Stakeholder position | Answer |
|---|---|---|
| What security measures should be implemented to protect sensitive financial data? | Developer | End-to-end encryption, secure authentication (e.g., 2FA), and regular security audits should be implemented. |
| How do we secure API connections with external financial service providers? | Developer | Use HTTPS for data transfer, token-based authentication, and enforce strict API access controls. |
| What measures would you recommend to prevent unauthorized access or breaches? | Developer | Implement intrusion detection systems (IDS), monitor activity logs, and enforce role-based access control (RBAC). |
| How can we educate end users on maintaining their own security while using the app? | Developer | Provide in-app prompts on creating strong passwords and avoiding suspicious links, as well as a security FAQ section. |
| What hosting service plan would you recommend for a startup like ours? |  | Start with a pay-as-you-go plan to minimize upfront costs and enable scalability as demand grows. AWS, Azure, or Google Cloud are suitable options. |

| Question | Stakeholder position | Answer |
|---|---|---|
| What testing frameworks or tools do you recommend for this type of application? | Tester | Use Selenium for UI testing, JUnit for backend testing, and Postman for API testing. |
| How will you handle edge cases, like incorrect data entries or unexpected API failures? | Tester | We'll develop test cases for invalid inputs and simulate API downtime to verify the app's resilience. |
| What would your approach be to test the app's performance under heavy user load? | Tester | Use tools like JMeter to simulate high traffic and measure response times under various loads. |
| How do you propose handling user feedback during the beta testing phase? | Tester | Feedback will be categorized by severity and addressed in iterative development cycles before release. |
| What is the testing timeline in the project, and what milestones should we aim for? | Tester | Functional testing after the first prototype, integration testing mid-development, and final testing post-beta. |

| Question | Stakeholder position | Answer |
|---|---|---|
| What are the critical milestones for the "Loonie" project, including budget management features, and how do we prioritize them? | Project Manager | Key milestones include requirement gathering, which will encompass both expense tracking and budget management features, prototype development of the app, beta testing to gather user feedback on financial features, and the final launch. We should prioritize these milestones by focusing on dependencies, especially ensuring that the budget management features are developed in tandem with expense tracking to provide a cohesive user experience. |
| Are there resource constraints that may impact the project timeline or scope? | Project Manager | Yes, resource availability for testing and marketing might impact timelines. A detailed resource allocation plan is critical. |
| What success metrics will be used to determine if the application meets user needs? | Project Manager | Metrics like user retention, app store ratings, and monthly active users will measure success. |
| Are there any anticipated risks that could delay or derail the project? | Project Manager | API reliability, budget overruns, and potential delays in regulatory compliance are key risks. |
| What is the expected budget allocation for different phases of development? | Project Manager | Development (50%), testing (20%), marketing (20%), and miscellaneous (10%). |

| Question | Stakeholder position | Answer |
|---|---|---|
| Are there any additional features you believe would increase the app's marketability? | Investor | Incorporating premium features like financial coaching or integrations with tax software. |
| What is your perspective on the current budget allocation? | Investor | It’s reasonable, but more emphasis should be placed on post-launch marketing efforts. |
| What is the acceptable timeline for profitability? | Investor | Break-even within 18–24 months, with steady growth afterward. |
| If you have the resources required for the application, how much will you allocate to finish the application? | Investor | Allocating 40% of the budget to development, 20% to testing, 20% to marketing, and 10% to infrastructure, with the remaining 10% for legal and operational support. |
| How are we ensuring the app complies with legal and privacy regulations to protect the company and maintain user trust? | Investor | We are conducting a thorough legal review to ensure compliance with international privacy regulations such as GDPR and CCPA. This includes obtaining user consent for data collection, implementing robust data security measures, and creating clear terms and conditions to protect the company from liability. By prioritizing compliance and transparency, we build trust with users, which directly supports the app's marketability and long-term success. |

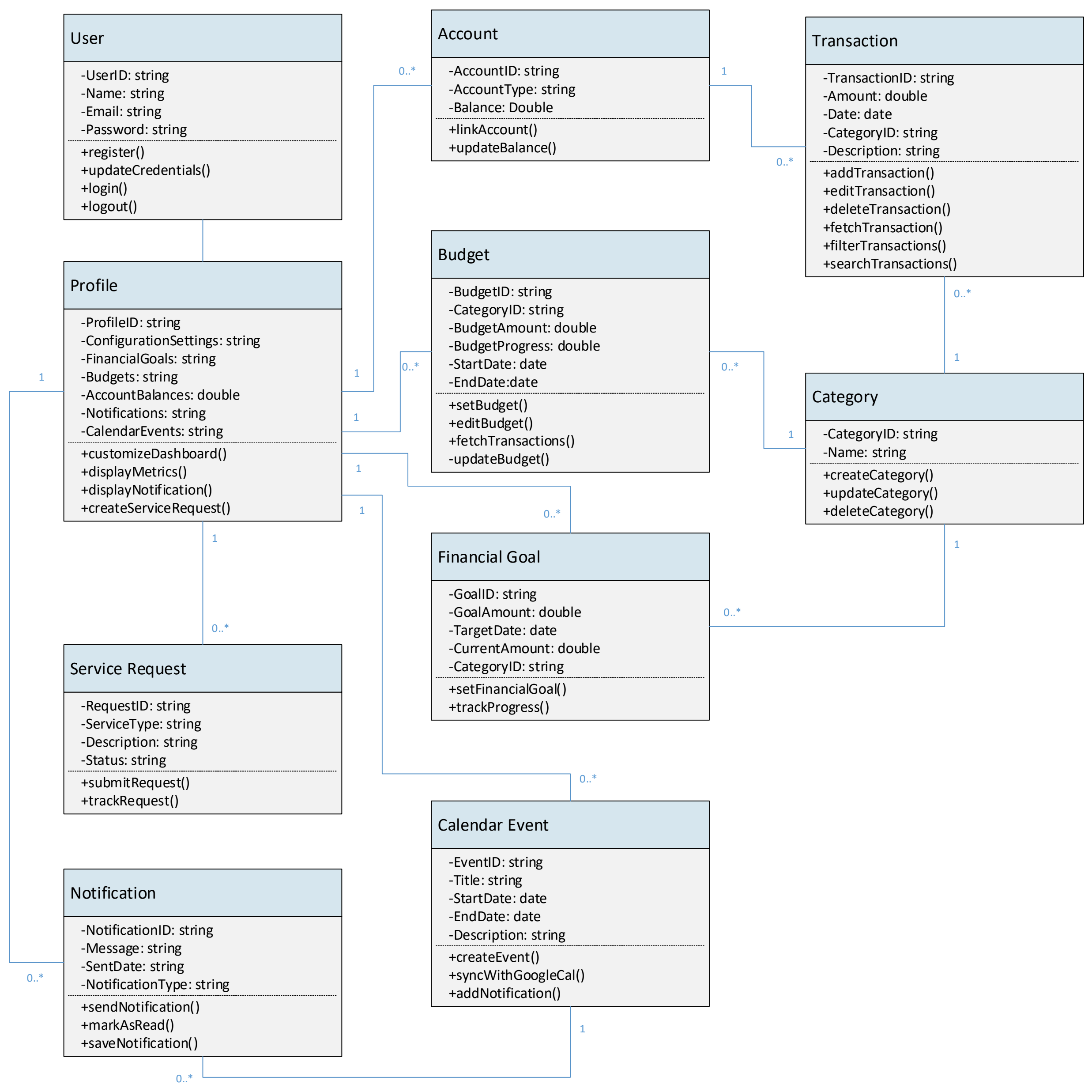
| Question | Stakeholder position | Answer |
|---|---|---|
| What features would be most valuable to you in a personal finance tracking application? | User | Expense tracking, budgeting, savings goals, and alerts for bill due dates. |
| How important is data visualization (e.g., graphs, charts) for tracking expenses and budgets? | User | Extremely important—it makes financial data easier to interpret and track progress. |
| How would you like to categorize your expenses in the app (e.g., groceries, entertainment, bills)? | User | I would like the ability to create custom categories to suit my personal spending habits and set predefined categories for common expenses. |
| How frequently do you think the app should update your financial data (e.g., daily, weekly)? | User | I'd like it to update daily, so I can stay on top of my finances regularly without having to check manually. |
| Would you like the app to offer insights or suggestions for improving your financial habits? | User | Yes, it would be great to receive insights based on my spending patterns, like suggestions to cut back in certain areas or tips on saving. |

| Question | Stakeholder position | Answer |
|---|---|---|
| What infrastructure will be used to host the app, and how do we ensure uptime? | System Administrator | Cloud hosting services like AWS or Azure with load balancers for redundancy. |
| How will system performance be monitored, and what tools will be used? | System Administrator | Tools like New Relic or Datadog for real-time performance monitoring. |
| What's your disaster recovery plan in case of outages or failures? | System Administrator | Daily backups, multi-region deployment, and rapid failover mechanisms. |
| How can the app's database be optimized for real-time operations? | System Administrator | Use indexed databases like PostgreSQL with caching via Redis. |
| Are there any scalability concerns with the planned infrastructure? | System Administrator | Not with proper resource planning and monitoring. |

| Question | Stakeholder position | Answer |
|---|---|---|
| What platforms should we prioritize to reach our target audience? | Marketing | Social media platforms like Instagram and Facebook are ideal for younger users, while LinkedIn can help with professional and older demographics. |
| What pricing model do you suggest for the application? | Marketing | A freemium model with a $4.99/month premium tier. |
| How can we effectively pitch the app to potential users? | Marketing | Highlight its simplicity, security, and ability to achieve financial goals. |
| What type of content would you suggest for educating users about personal finance? | Marketing | interactive tools, infographics, and articles on budgeting, saving, and credit management. |
| How will you ensure that the content remains relevant and up-to-date? | Marketing | By tracking financial trends and updating content quarterly. |

# Appendix E: Class Diagrams

## 7.1 Domain Class Diagram



## 7.2 Party Analysis Pattern

In the context of the "Loonie" personal finance tracking application, the Party analysis pattern cannot be effectively utilized due to the nature of the application's design requirements and its focus on user experience. Loonie is primarily tasked with providing a unified platform for managing personal finances rather than representing distinct roles or parties in a complex interaction. The primary interactions in Loonie revolve around individual users and their financial data, necessitating a straightforward user interface without the need for a complex network of interacting parties. Essentially, while the Party analysis pattern excels in scenarios where the relationships and collaborations among multiple stakeholders or systems are highlighted, Loonie's application structure focuses on direct user interactions and feature functionality, making this pattern less relevant and applicable to the project's goals.

## 7.3 Class-Responsibility-Collaborator Modeling

| **Class: User** | |
|---|---|
| Represents an individual utilizing the Loonie app for personal finance management. The user can track expenses, set budgets, and get insights into their financial activities to enhance financial literacy and decision-making. | |
| **Attributes:** | |
| **Name** | **Description** |
| UserID | ID for the user |
| NameID | Name for the user |
| Email | Email address of the user |
| Password | Password for account authentication |
| **Responsibilities:** | |
| **Name** | **Collaborator** |
| register() | login() |
| UpdateCredential() | login() |
| login() | User |
| logout() | login() |

| **Class: Account** | |
|---|---|
| Represents a user's linked bank account within the Loonie app. This class is responsible for managing account details, including balance updates and transaction histories, ensuring that users have a clear overview of their financial activity. | |
| **Attributes:** | |
| **Name** | **Description** |
| AccountID | ID for the account |
| AccountType | Type of account |
| Balance | Current balance in the account |
| **Responsibilities:** | |
| **Name** | **Collaborator** |
| linkedAccount() | User |
| updateBalance() | Transaction |

| **Class: Transaction** | |
|---|---|
| Represents individual financial transactions that occur within the user's linked accounts. The Transaction class is responsible for storing transaction details, managing transaction-related operations, and facilitating data integration with external services for analysis and insights. | |
| **Attributes:** | |
| **Name** | **Description** |
| TransactionID | ID for the transaction |
| Account | Monetary value of the transaction |
| Date | Date when the transaction occurred |
| CategoryID | Associated category for the transaction |
| Description | Details about the transaction |
| **Responsibilities:** | |
| **Name** | **Collaborator** |
| addTransaction() | Account |
| editTransaction() | Account |
| deleteTransaction() | Account |
| fetchTransaction() | Budget |
| filterTransaction() | Budget |
| searchTransaction() | Budget |

| **Class: Category** | |
|---|---|
| Represents financial categories used to organize transactions within the Loonie app. The Category class enables users to create, manage, and update categories for effective expense tracking and budgeting. | |
| **Attributes:** | |
| **Name** | **Description** |
| CategoryID | ID for the category |
| Name | Name of the category |
| **Responsibilities:** | |
| **Name** | **Collaborator** |
| createCategory() | Budget |
| updateCategory() | Budget |
| deleteCategory() | Budget |

| **Class: Budget** | |
|---|---|
| Represents the user's financial budget within the Loonie app. The Budget class helps users set spending limits for various categories, track their expenses against these limits, and manage their overall financial health. | |
| **Attributes:** | |
| **Name** | **Description** |
| BudgetID | ID for the budget (B###) |
| CategoryID | ID for category associated with budget |
| BudgetAmount | Total budget amount |
| BudgetProgress | Progress of spending within budget based on transactions within category |
| StartDate | Start of budget goal timeline |
| EndDate | End of budget goal timeline |
| **Responsibilities:** | |
| **Name** | **Collaborator** |
| setBudget() | Profile |
| editBudget() | Profile |
| fetchTransactions() | Category |
| updateBudget() | fetchTransactions() |

| **Class: Financial Goal** | |
|---|---|
| Represents the user-defined financial objectives within the Loonie app. The Financial Goal class allows users to set specific targets for saving or spending, track progress toward these goals, and link them to relevant spending categories for enhanced financial management. The purpose is to empower users to achieve their financial aspirations through structured planning and monitoring. | |
| **Attributes:** | |
| **Name** | **Description** |
| GoalID | ID for the financial Goal |
| GoalAmount | Target monetary value for the goal |
| TargetDate | Deadline to achieve the goal |
| CurrentAmount | Current progress towards achieving the goal |
| CategoryID | Associated category for tracking progress |
| **Responsibilities:** | |
| **Name:** | **Collaborator** |
| setFinancialGoal() | Profile, Category |
| trackProgress() | Transaction, Budget |

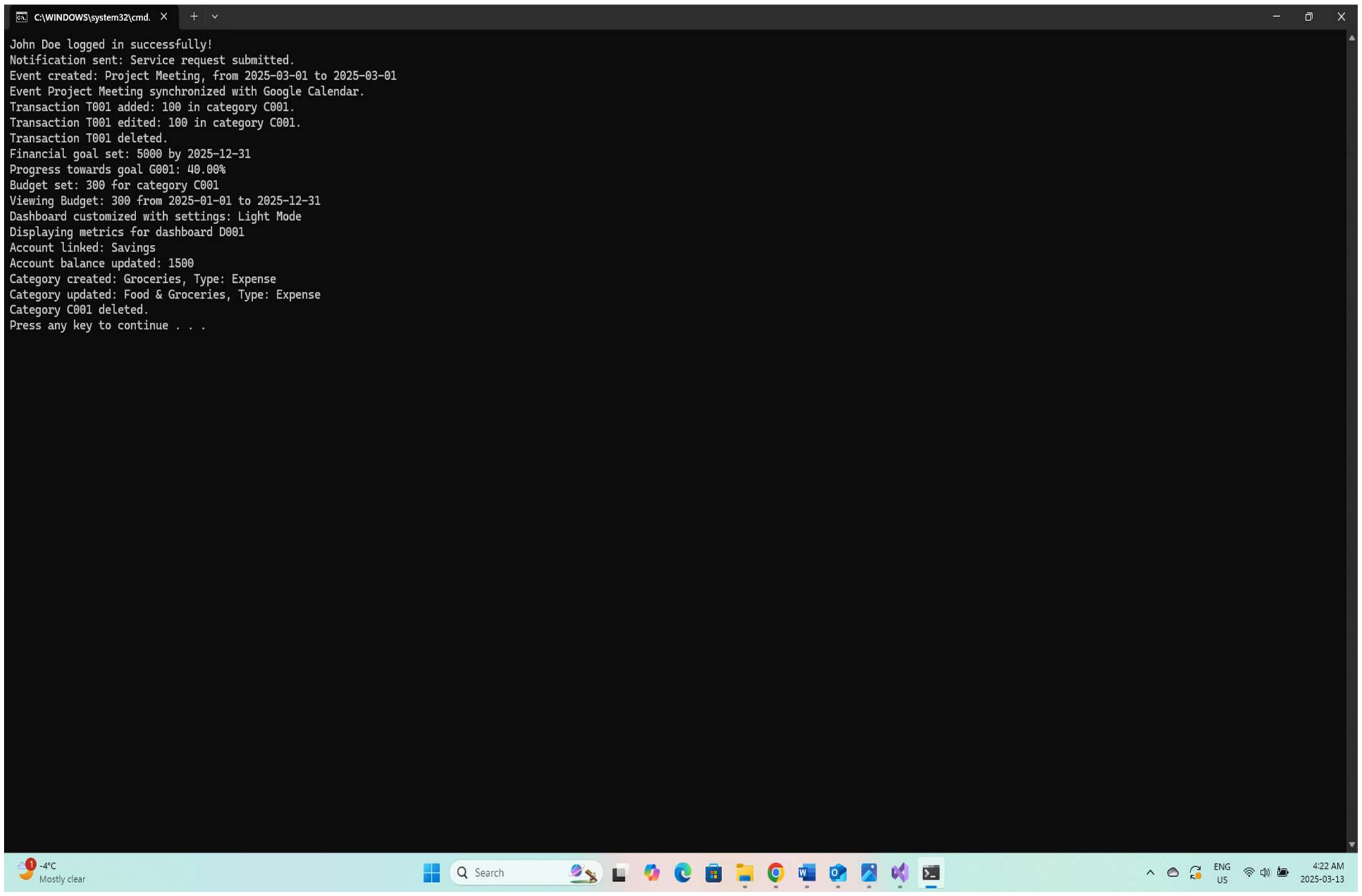
| **Class: Calendar Event** | |
|---|---|
| Represents scheduled financial events within the Loonie app, such as bill due dates and budget reminders. The Calendar Event class enables users to create, store, and synchronize these events with their Google Calendar, ensuring that they remain informed of important financial milestones. Its goal is to enhance user engagement by helping users stay organized and manage their financial obligations effectively. | |
| **Attributes:** | |
| **Name** | **Description** |
| EventID | ID for Calendar events |
| Title | Title of the event |
| StartDate/EndDate | Start and end dates for events |
| Description | Additional details about Events |
| **Responsibilities:** | |
| **Name:** | **Collaborator** |
| createEvents() | Profile |
| syncWithGoogleCal() | |
| addNotification() | Notification |

| **Class: Service Request** | |
|---|---|
| Represents user requests for assistance or support within the Loonie app. The Service Request class facilitates the submission, tracking, and management of these requests, allowing users to seek help for issues or inquiries related to their financial management. Its goal is to enhance user experience by providing a structured way to address user concerns and improve app functionality. | |
| **Attributes:** | |
| **Name** | **Description** |
| RequestID | ID for service request |
| ServiceType | Type of service requested |
| Description | Details about the request |
| Status | Current status of request processing |
| **Responsibilities:** | |
| **Name:** | **Collaborator** |
| submitRequest() | Profile |
| trackRequest() | Profile |

| **Class: Notification** | |
|---|---|
| Represents alerts and messages within the Loonie app that inform users about important updates, reminders, or actions required. The Notification class ensures users are kept informed about their financial activities, important deadlines, and other relevant information to enhance their user experience and engagement with the app. | |
| **Attributes:** | |
| **Name** | **Description** |
| NotificationID | ID for notification |
| Message | Content of notification messages |
| SentDate | Date when notification was sent |
| NotificationType | Type of notification |
| **Responsibilities:** | |
| **Name:** | **Collaborator** |
| sendNotification() | Calendar Event, Profile |
| markAsRead() | User |
| saveNotification() | Profile |

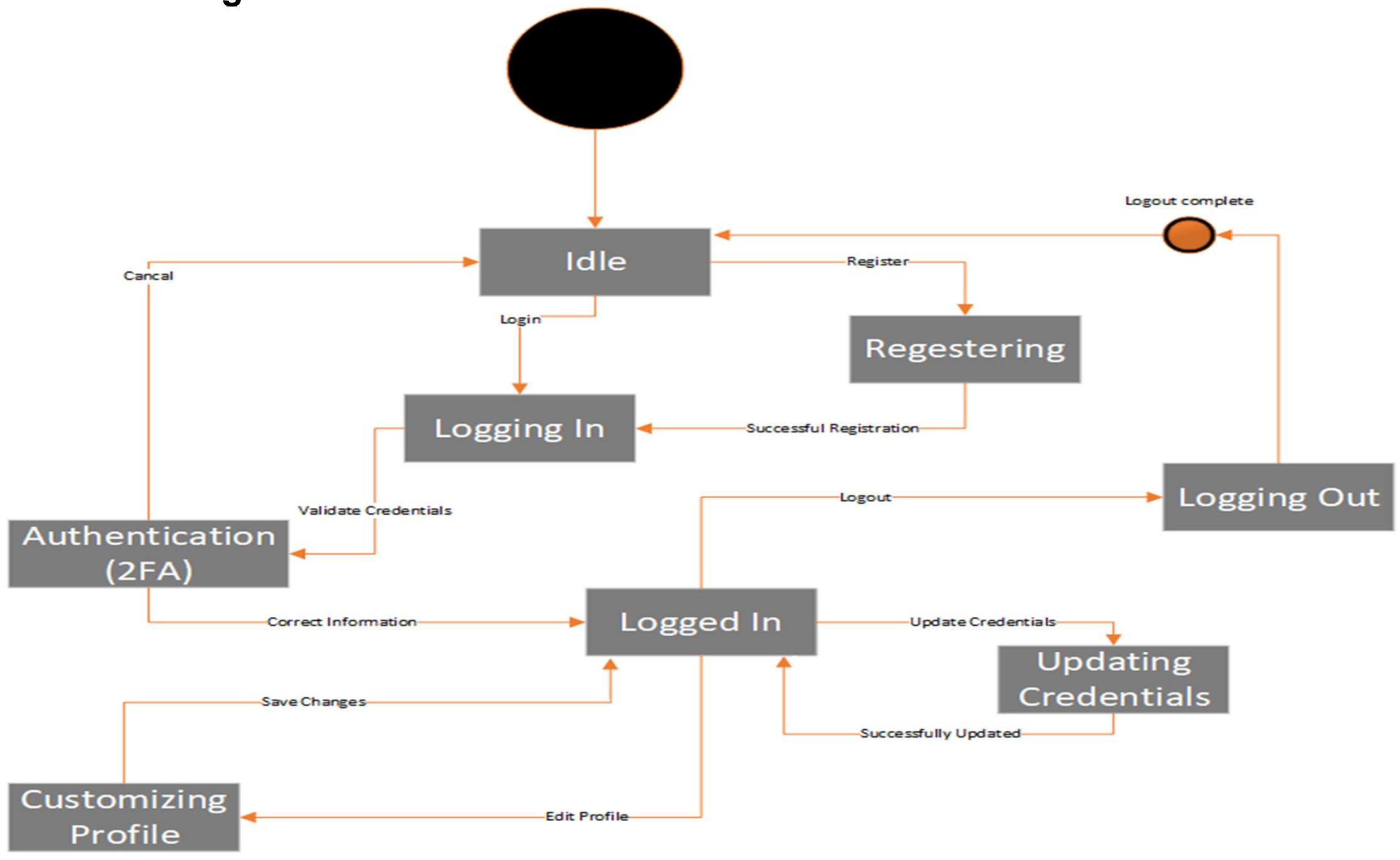
| **Class: Profile** | |
|---|---|
| Represents the central interface within the Loonie app where users can view key financial metrics and insights. The profile class allows users to customize their view according to their preferences and displays notifications. | |
| **Attributes:** | |
| **Name** | **Description** |
| ProfileID | ID for profiles |
| ConfigurationSettings | Customizable settings defined by users |
| FinancialGoals | List of financial goals associated with the profile |
| Budget | Budgets created by the user |
| AccountBalances | Summary balances across linked accounts |
| Notifications | Notifications received by the user |
| CalendarEvents | Events tied to the profile |
| **Responsibilities:** | |
| **Name:** | **Collaborator** |
| customizeDashboard() | Notification, Financial Goals, Budget |
| diplayMetrics() | Budget, Account, Transaction |
| displayServiceRequest() | Service request |
| createServiceRequest() | Service request |

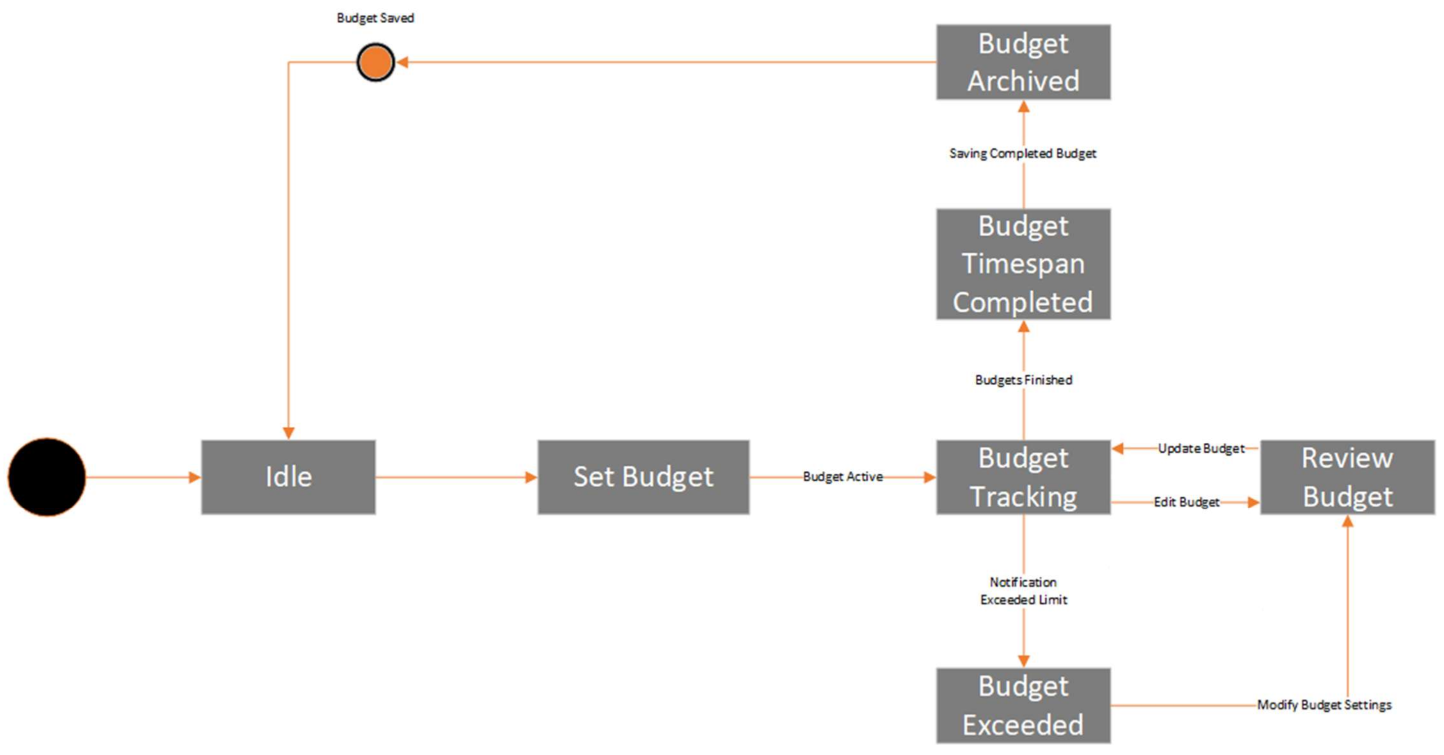
## 7.4 Gen AI Class Diagram



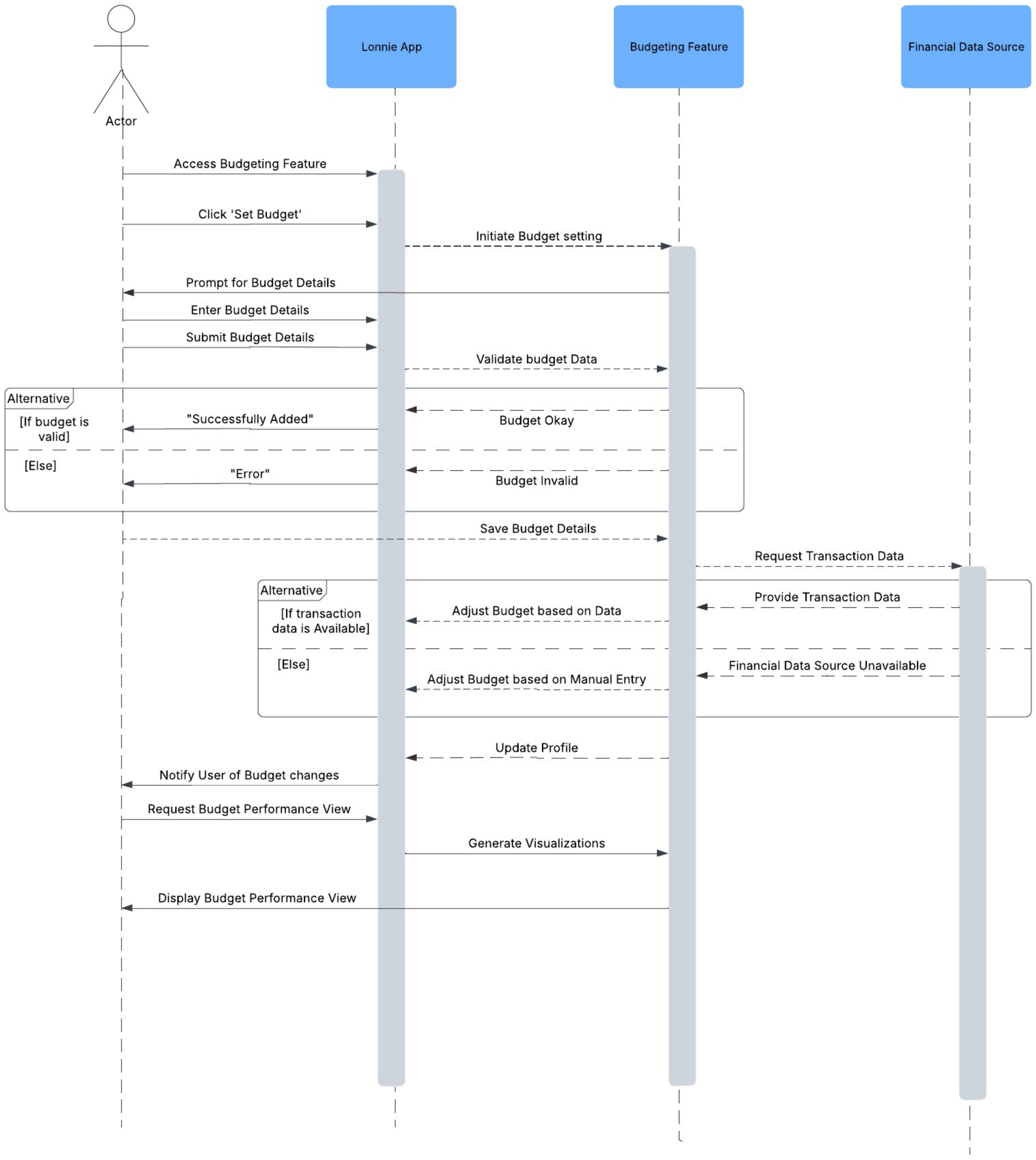
```
John Doe logged in successfully!
Notification sent: Service request submitted.
Event created: Project Meeting, from 2025-03-01 to 2025-03-01
Event Project Meeting synchronized with Google Calendar.
Transaction T001 added: 100 in category C001.
Transaction T001 edited: 100 in category C001.
Transaction T001 deleted.
Financial goal set: 5000 by 2025-12-31
Progress towards goal G001: 40.00%
Budget set: 300 for category C001
Viewing Budget: 300 from 2025-01-01 to 2025-12-31
Dashboard customized with settings: Light Mode
Displaying metrics for dashboard D001
Account linked: Savings
Account balance updated: 1500
Category created: Groceries, Type: Expense
Category updated: Food & Groceries, Type: Expense
Category C001 deleted.
Press any key to continue . . .
```

## 7.5 State Diagram





## 7.6 Sequence Diagram


Actor
Lonnie App
Budgeting Feature
Financial Data Source
Access Budgeting Feature
Click 'Set Budget'
Initiate Budget setting
Prompt for Budget Details
Enter Budget Details
Submit Budget Details
Validate budget Data
Alternative
[If budget is valid]
"Successfully Added"
Budget Okay
[Else]
"Error"
Budget Invalid
Save Budget Details
Request Transaction Data
Alternative
[If transaction data is Available]
Provide Transaction Data
Adjust Budget based on Data
[Else]
Financial Data Source Unavailable
Adjust Budget based on Manual Entry
Update Profile
Notify User of Budget changes
Request Budget Performance View
Generate Visualizations
Display Budget Performance View