Software Requirements Specification

for

Smart Savings Manager

Version 1.0 approved

Prepared by Jagadheep S

Coimbatore Institute of Technology

28 June 2025

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

The purpose of this application is to assist individuals in managing their monthly budgets efficiently based on their lifestyle type (Family Man, Bachelor, or Child under 18). It provides a personalized budget breakdown, including fixed and variable expenses, savings recommendations, and visual insights via pie charts. This SRS covers the entire system, detailing its features, functionalities, and constraints to guide developers, testers, and stakeholders throughout the software development lifecycle.

1.2 Document Conventions

This SRS follows the IEEE 830-1998 standard for Software Requirements Specification. Throughout the document, bold text is used for section headings to highlight structural divisions clearly, while italic text emphasizes special terms or important concepts. Monospace font is applied to code snippets and file names to distinguish them from regular text. Requirements are labeled with High, Medium, or Low priorities to indicate their relative importance, and these priorities are not assumed to be inherited by detailed requirements unless explicitly stated. All diagrams and figures included in this document follow a consistent naming convention, such as "Fig 1.1: Budget Flow Diagram," to maintain clarity and ease of reference.

1.3 Intended Audience and Reading Suggestions

This document is intended for a range of stakeholders, including developers, project managers, testers, end users, marketing staff, and documentation writers. Developers can use this SRS to understand both functional and non-functional requirements necessary to build the Smart Savings Manager application. Project managers will find it useful for tracking the scope, milestones, and deliverables of the project. Testers can refer to the specified requirements for designing test cases and ensuring the application meets its intended goals. End users and marketing staff may use this document to gain a deeper understanding of the application's features and its value proposition. The document is organized to start with overview sections for general comprehension (Section 1), followed by detailed functional requirements (Section 3), and then progressing to specific system features and implementation details. Readers are recommended to begin with the introductory sections and proceed to areas most relevant to their role.

1.4 Product Scope

The Smart Savings Manager is a web-based application developed to assist users in managing their personal finances effectively. It allows users to select their lifestyle type—Family Man, Bachelor, or Child under 18—enter their monthly income or pocket money, and indicate whether they pay rent. Based on these inputs, the application generates a personalized budget breakdown that includes categories such as rent, food, utilities, school fees, transport, and savings. The results are displayed both as a text summary and visually using interactive pie charts through Chart.js. The primary objective of this application is to empower users to make informed financial decisions while providing an intuitive and responsive user interface designed with Bootstrap.

Additionally, it supports basic validation to prevent incorrect inputs and is designed for potential future enhancements like expense tracking, saving goals, and the ability to download reports in PDF format. This project aligns with modern personal finance management trends and aims to support users in achieving their financial goals efficiently.

1.5 References

This Software Requirements Specification refers to several important standards and resources that support the development of the “Smart Savings Manager” application. The **IEEE 830-1998 SRS Standard** by IEEE provides comprehensive guidelines for structuring software requirements specifications and can be accessed at https://standards.ieee.org/ieee/830/1222/. The **Bootstrap Documentation** (version 5.x) offers detailed guidance for building a responsive and user-friendly interface and is available at https://getbootstrap.com/docs/5.0/gettingstarted/introduction/. For implementing interactive data visualizations, the project refers to the **Chart.js Documentation** (version 4.x), accessible at https://www.chartjs.org/docs/. Additionally, the application leverages web technologies defined in the **HTML5 and CSS3 specifications**, published by the W3C and available at https://www.w3.org/TR/html52/ and https://www.w3.org/Style/CSS/. These references provide foundational knowledge and best practices essential for ensuring the quality and effectiveness of the Smart Savings Manager system.

2. Overall Description

2.1 Product Perspective

The Smart Savings Manager App is a new, self-contained product designed to assist users in managing their finances effectively. It is not a follow-on member of an existing product family nor a direct replacement for any specific system, but it addresses the growing need for personal financial management tools in a userfriendly digital format. The app integrates features for budgeting, expense tracking, and financial goal setting, catering to both individual users and financial advisors. It interacts with a larger ecosystem that includes system utilities such as notifications and authentication services, which are essential for its operation. The app interfaces with external systems through secure authentication protocols and notification services, ensuring seamless user experience and data integrity. A use case diagram provided earlier illustrates the major components (User Features, Admin Features, System Utilities) and their interconnections, highlighting the external interfaces with authentication and notification systems.

2.2 Product Functions

- View Budget Suggestion
- Track Expenses
- Get Savings Tips
- Set Financial Goals
- Set Reminders
- Export Data
- Register/Login
- Select User Type
- Enter Income/Pocket Money
- View Reports
- Generate Analytics
- Manage Content
- Manage Users
- Monitor Budget Reports Send Notifications

These functions are organized into User Features and Admin Features, supported by System Utilities, providing a comprehensive financial management solution. The use case diagram effectively groups related requirements and depicts their relationships.

2.3 User Classes and Characteristics

- **General Users**: Frequent users with basic technical skills, utilizing core functions like tracking expenses, viewing budget suggestions, and setting reminders. They require an intuitive interface and are the most important user class to satisfy.
- **Financial Advisors**: Infrequent but expert users who leverage advanced features like generating analytics and managing content. They need detailed reports and higher privilege levels, making them a secondary but critical user class.
- **Admins**: Rare users with high technical expertise, responsible for managing users and monitoring budget reports. They require administrative privileges and are less numerous but essential for system maintenance.
- Characteristics include varying educational levels (general users may have minimal financial knowledge, while advisors and admins are highly skilled) and security needs (admins require elevated access controls).

2.4 Operating Environment

The software will operate on mobile devices running iOS (latest stable version) and Android (latest stable version), supported by hardware with at least 2GB RAM and 1GHz processor. It must coexist with standard mobile operating system features (e.g., push notifications) and third-party apps for data export (e.g., PDF viewers). Internet connectivity is required for real-time features like notifications and analytics.

2.5 Design and Implementation Constraints

The development must adhere to corporate security policies ensuring data encryption and user privacy compliance (e.g., GDPR). Hardware limitations include mobile device memory constraints, necessitating optimized performance. The app must interface with existing authentication systems using standard protocols (e.g., OAuth). Development will use specific technologies like React Native for cross-platform compatibility and a MySQL database. Language requirements include English as the primary interface, with potential multilingual support. Security considerations mandate secure API calls and data storage

2.6 User Documentation

- **User Manual**: A detailed guide covering all user features, delivered in PDF format.
- **On-line Help**: Contextual help within the app, accessible via a help menu.
- **Tutorials**: Video tutorials for key functions like expense tracking and goal setting, hosted online and linked in the app.
- Documentation will follow standard mobile app documentation guidelines.

2.7 Assumptions and Dependencies

It is assumed that users have basic mobile device proficiency and internet access. The app depends on third-party authentication services for login functionality and a reliable notification system for reminders. Development assumes the availability of a stable API for data export and analytics generation. Any changes in these external components or lack of internet connectivity could impact the project. Reused software components, such as authentication libraries, are assumed to be compatible with the chosen development stack.

3. External Interface Requirement

3.1 User Interfaces

The Smart Savings Manager App will feature an intuitive graphical user interface (GUI) tailored for mobile devices, adhering to material design principles and iOS Human Interface Guidelines. Each screen will include a consistent layout with a top navigation bar (containing a menu button and help icon), a main content area, and a bottom action bar with standard buttons (e.g., save, cancel). Key screens include a dashboard, budget suggestion view, expense tracking form, and analytics report page. All screens will support a help function accessible via the help icon, displaying contextual guidance. Error messages will appear as modal pop-ups with clear descriptions and an OK button. Keyboard shortcuts are not applicable, but touch gestures (e.g., swipe to refresh) will be supported. The user interface design will be detailed in a separate User Interface Specification document.

3.2 Hardware Interfaces

The app will interface with mobile device hardware including touchscreens, cameras (for scanning receipts), and speakers (for audio notifications). Supported devices include smartphones and tablets with at least 2GB RAM and 1GHz processors running iOS (latest stable version) or Android (latest stable version). Data interactions include touch input for navigation and camera data for receipt parsing, processed via the app's image recognition module. Control interactions involve triggering notifications through the device's speaker or vibration motor. Communication will use standard mobile hardware APIs (e.g., Android Camera API, iOS AVFoundation) with no custom protocols required.

3.3 Software Interfaces

The app will connect with the following software components:

- Operating Systems: iOS (latest stable version) and Android (latest stable version) for core functionality.
- Database: MySQL (version 8.0) for storing user data, budgets, and expenses, with SQL queries for data retrieval and updates.
- Libraries: React Native (latest stable version) for cross-platform development, and a third-party authentication library (e.g., Firebase Authentication v9.0) for user login.
- Integrated Components: A notification service API (e.g., Firebase Cloud Messaging) for sending reminders and alerts.
- Data Items: Incoming data includes user login credentials and income entries; outgoing data includes exported reports (PDF) and analytics data. The purpose is to facilitate user authentication, data storage, and communication.
- Services: RESTful APIs will handle data exchange with the database and notification services, using JSON for message formatting. Shared data includes user profiles and financial records, stored in a centralized MySQL database. Implementation requires secure API endpoints with HTTPS

3.4 Software Interfaces

The app requires internet connectivity for features like notifications, data synchronization, and report generation. It will use HTTPS for secure web communication, supporting RESTful API calls to the backend server. Email functionality will integrate with a third-party service (e.g., SendGrid) for account verification, using SMTP with TLS encryption. Data transfer rates should support real-time updates (minimum 1 Mbps). Synchronization will occur automatically upon data changes, with a fallback to manual sync. Security measures include end-to-end encryption for data in transit and compliance with GDPR standards.

4. System Features

4.1 Budget Management

4.1.1 Description and Priority

This feature allows users to view budget suggestions, track expenses, and set financial goals. It is of High priority due to its core role in the app's purpose of aiding financial planning.

4.1.2 Stimulus/Response Sequences

- **User Action**: Selects "View Budget Suggestion" from dashboard.

  System Response: Displays a personalized budget plan based on entered income.
- **User Action**: Enters expense details in "Track Expenses" form.

  System Response: Updates expense log and notifies user if over budget.
- **User Action**: Sets a financial goal via "Set Financial Goals" option.

  System Response: Saves goal and tracks progress with periodic reminders.

4.1.3 Functional Requirements

- **REQ-1**: The system shall generate and display a budget suggestion based on user-entered income and expense data within 5 seconds.
- **REQ-2**: The system shall allow users to input and categorize expenses with a maximum of 3 custom categories, saving data to the database upon submission.
- **REQ-3**: The system shall alert users with a notification if expenses exceed 90% of the suggested budget, displaying an error message if invalid data (e.g., negative amounts) is entered.
- **REQ-4**: The system shall enable users to set and edit financial goals, storing them securely and providing progress updates monthly.

4.2 User Authentication

4.2.1 Description and Priority

This feature handles user registration, login, and user type selection. It is of High priority to ensure secure access and personalized experiences.

4.2.2 Stimulus/Response Sequences

- **User Action**: Clicks "Register/Login" and enters credentials.
  System Response: Validates credentials and logs user in, redirecting to the dashboard.

- **User Action:** Selects user type (e.g., General User, Financial Advisor) during registration.
  System Response: Assigns appropriate feature access based on user type.
- **User Action**: Enters incorrect login details.
  System Response: Displays an error message and allows three retry attempts before locking the account for 15 minutes.

4.2.3 Functional Requirements

- **REQ-1**: The system shall support email-based registration and login using OAuth 2.0, completing the process within 10 seconds.
- **REQ-2**: The system shall allow users to select their user type during registration, restricting access to admin features for non-admin users.
- **REQ-3**: The system shall validate login credentials against the database, displaying an "Invalid Credentials" error if mismatched, and lock the account after three failed attempts.
- **REQ-4**: The system shall ensure all authentication data is encrypted using AES-256 during transmission and storage.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The Smart Savings Manager App must process budget suggestions and expense tracking within 5 seconds under normal conditions (e.g., 100 concurrent users). Analytics generation should complete within 10 seconds for datasets up to 1,000 entries. Real-time notifications must be delivered within 2 seconds of a trigger event (e.g., exceeding budget). These requirements ensure a responsive user experience, with the rationale being to maintain user engagement and satisfaction on mobile devices with typical network conditions (1-10 Mbps). Performance may degrade gracefully under high load (e.g., 1,000 users), with a maximum delay of 15 seconds.

5.2 Safety Requirements

The app must prevent financial data loss by implementing automatic backups to a secure cloud server every 24 hours. Users should be unable to delete critical data (e.g., income entries) without confirmation via a two-step process. No harm should result from app use, adhering to GDPR and local financial regulations. Safeguards include encrypted data storage and a panic button to log out remotely if the device is lost. No specific safety certifications are required, but compliance with ISO 27001 standards is targeted.

5.3 Security Requirements

All user data, including financial details and login credentials, must be encrypted using AES-256 during transmission (HTTPS) and storage. User identity authentication requires email verification and OAuth 2.0, with multi-factor authentication (MFA) as an optional setting. The app must comply with GDPR and CCPA for data privacy, ensuring users can request data deletion. Access to admin features is restricted to verified admin accounts with role-based access control. Security audits will align with ISO 27001 certification goals.

5.4 Software Quality Attributes

- **Usability**: The app must achieve a System Usability Scale (SUS) score of at least 80, prioritizing ease of use over ease of learning for general users.
- **Reliability**: The system must operate without failure for 99.9% of the time under normal usage, with automatic error recovery within 30 seconds.
- **Portability**: The app must function seamlessly across iOS and Android platforms with no more than 5% feature variance.
- **Maintainability**: Code changes must be implementable with less than 10% effort increase due to modular design.

- **Interoperability**: The app must integrate with third-party tools (e.g., PDF exporters) via standard APIs, ensuring 100% compatibility with supported versions.

5.5 Business Rules

- Only admins can manage users and monitor budget reports.
- Financial advisors can generate analytics and manage content but cannot alter user data.
- General users can access all personal financial features but are restricted from admin functions. Expense entries exceeding the budget require advisor approval before processing.
- Notifications can only be sent by admins or triggered automatically for user-defined reminders.
- Data export is limited to registered users with a maximum of one export per week to ensure server load management.

6. Other Requirements

6.1 Database Requirements

The Smart Savings Manager App requires a MySQL database (version 8.0) to store user profiles, financial data, and analytics. The database must support a minimum of 10,000 user records with a query response time of under 2 seconds. Data integrity must be ensured through primary key constraints and regular backups every 24 hours. The schema should include tables for users, expenses, budgets, and goals, with indexing on frequently queried fields (e.g., user ID, date).

6.2 Internationalization Requirements

The app must support English as the default language, with plans to add multilingual support for Spanish, French, and Hindi. Text strings must be stored in a separate resource file for easy translation.
Date and currency formats should adapt to the user's locale (e.g., MM/DD/YYYY for US, DD/MM/YYYY for India), with currency conversion available for major currencies (USD, EUR, INR) based on real-time exchange rates.

6.3 Legal Requirements

The app must comply with GDPR and CCPA, providing users the right to access, rectify, or delete their data upon request within 30 days. User consent must be obtained for data collection and processing, documented via an in-app terms of service agreement. All financial data handling must adhere to local financial regulations (e.g., RBI guidelines in India).

6.4 Reuse Objectives

The project aims to reuse the authentication module from a previous xAI project, ensuring 80% code reusability. The notification system design will be modular for potential reuse in future financial apps, with documentation provided for integration.

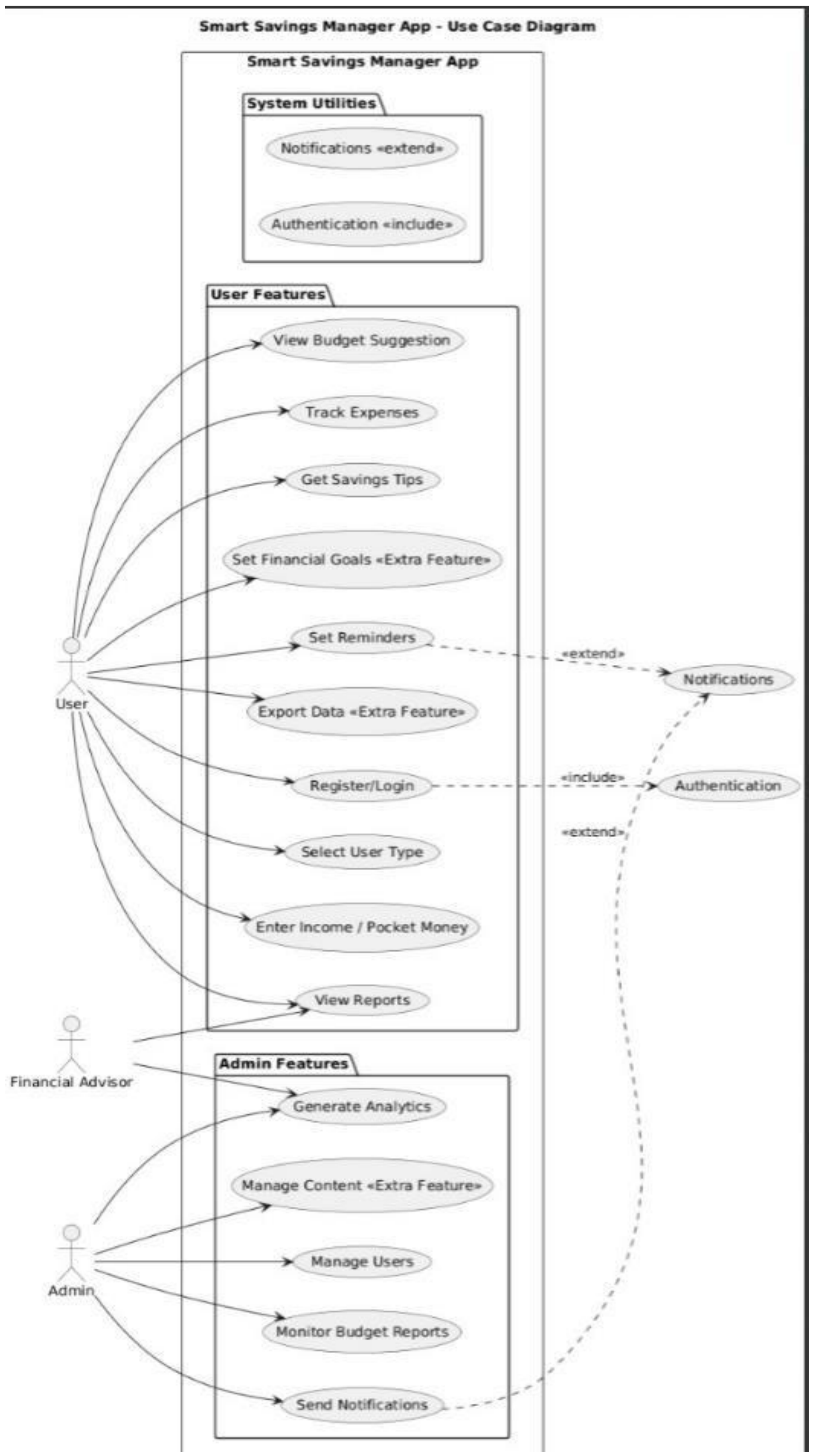
Appendix A: Glossary

- Admin: A user with elevated privileges to manage users and monitor reports.
- Financial Advisor: A user type with access to analytics and content management features.
- General User: A standard user with access to personal financial management tools.
- GDPR: General Data Protection Regulation, a legal framework for data protection in the EU.
- CCPA: California Consumer Privacy Act, a data privacy law in California.
- OAuth 2.0: An authorization framework used for secure API authentication.
- AES-256: Advanced Encryption Standard with 256-bit key length for data security.

- RESTful API: A web service architecture using HTTP methods for communication.
- SRS: Software Requirements Specification, a document outlining system requirements.

Appendix B: Analysis Models

Use case diagram:



Use case Description

UC 1: Register/Login Primary Actor: User/Admin
Secondary Actor: Authentication Service
Precondition: User/Admin is not logged in
Trigger: User/Admin opens the system
Main Success Scenario:
1. User/Admin provides login credentials.

2. System validates credentials.
3. User/Admin is redirected to their respective dashboard.

Exception Scenarios:
4. Invalid credentials → System shows error message.
5. Account locked/inactive → System shows "Contact Admin".

UC 2: Authenticate User

Primary Actor: System
Precondition: User/Admin credentials submitted
Trigger: Login attempt made Main Success
Scenario:
1. System checks credentials.
2. If valid, session is created.

Exception Scenario:
3. Invalid credentials → Deny access.

UC 3: View Budget Suggestions

Primary Actor: User
Precondition: User is logged in
Trigger: User selects “View Budget Suggestions” Main
Success Scenario:
1. System analyzes user income and expenses.
2. Displays budget suggestions tailored to user data.

UC 4: Track Expenses

Primary Actor: User
Precondition: User is logged in
Trigger: User navigates to “Track Expenses” Main
Success Scenario:
1. User adds new expenses or views history.
2. System updates expense records and graphs.

UC 5: Get Savings Tips

Primary Actor: User
Precondition: User is logged in
Trigger: User selects “Get Savings Tips”
Main Success Scenario:
1. System fetches personalized savings tips.
2. Displays them in the user dashboard.

UC 6: Set Financial Goals

Primary Actor: User
Precondition: User is logged in Trigger:
User selects “Set Financial Goals” Main
Success Scenario:
1. User inputs target savings amount and timeframe.
2. System saves goals and monitors progress.

UC 7: Set Reminders

Primary Actor: User

Precondition: User is logged in
Trigger: User sets reminder for payments/savings Main
Success Scenario:

1. User sets reminders with date and purpose.
2. System stores reminders and triggers notifications.

UC 8: Export Data

Primary Actor: User

Precondition: User has data to export
Trigger: User selects "Export Data" Main
Success Scenario:

1. System generates exportable file (PDF/Excel).
2. User downloads data successfully.

UC 9: Select User Type

Primary Actor: User

Precondition: User is registering for the first time Trigger:
User selects user type (User/Admin/Advisor) Main
Success Scenario:

1. System saves selected user type.
2. Tailors interface and permissions accordingly.

UC 10: Enter Income/Pocket Money

Primary Actor: User

Precondition: User is logged in Trigger:
User navigates to "Enter Income" Main
Success Scenario:

1. User inputs income or pocket money.
2. System updates financial records.

UC 11: View Reports

Primary Actor: User

Precondition: Financial data exists
Trigger: User selects "View Reports"

Main Success Scenario:

1. System compiles reports on income, expenses, and savings.
2. Displays graphs and summaries to the user.

UC 12: Generate Analytics

Primary Actor: Admin

Precondition: User financial data available

Trigger: Admin selects "Generate Analytics"
Main Success Scenario:

1. System analyzes usage patterns and savings trends.
2. Generates analytics reports for admin.

UC 13: Manage Content

Primary Actor: Admin

Precondition: Admin is logged in Trigger:
Admin selects “Manage Content” Main
Success Scenario:

1. Admin adds/edits tips and savings suggestions.
2. Updates reflected for all users.

UC 14: Manage Users

Primary Actor: Admin

Precondition: Admin is authenticated
Trigger: Admin selects “Manage Users” Main
Success Scenario:

1. Admin views user list.
2. Can activate, deactivate, or delete users.

UC 15: Monitor Budget Reports

Primary Actor: Admin

Precondition: Users have submitted data
Trigger: Admin navigates to “Monitor Budget Reports” Main
Success Scenario:

1. System displays budget reports from users.
2. Admin reviews and downloads reports.

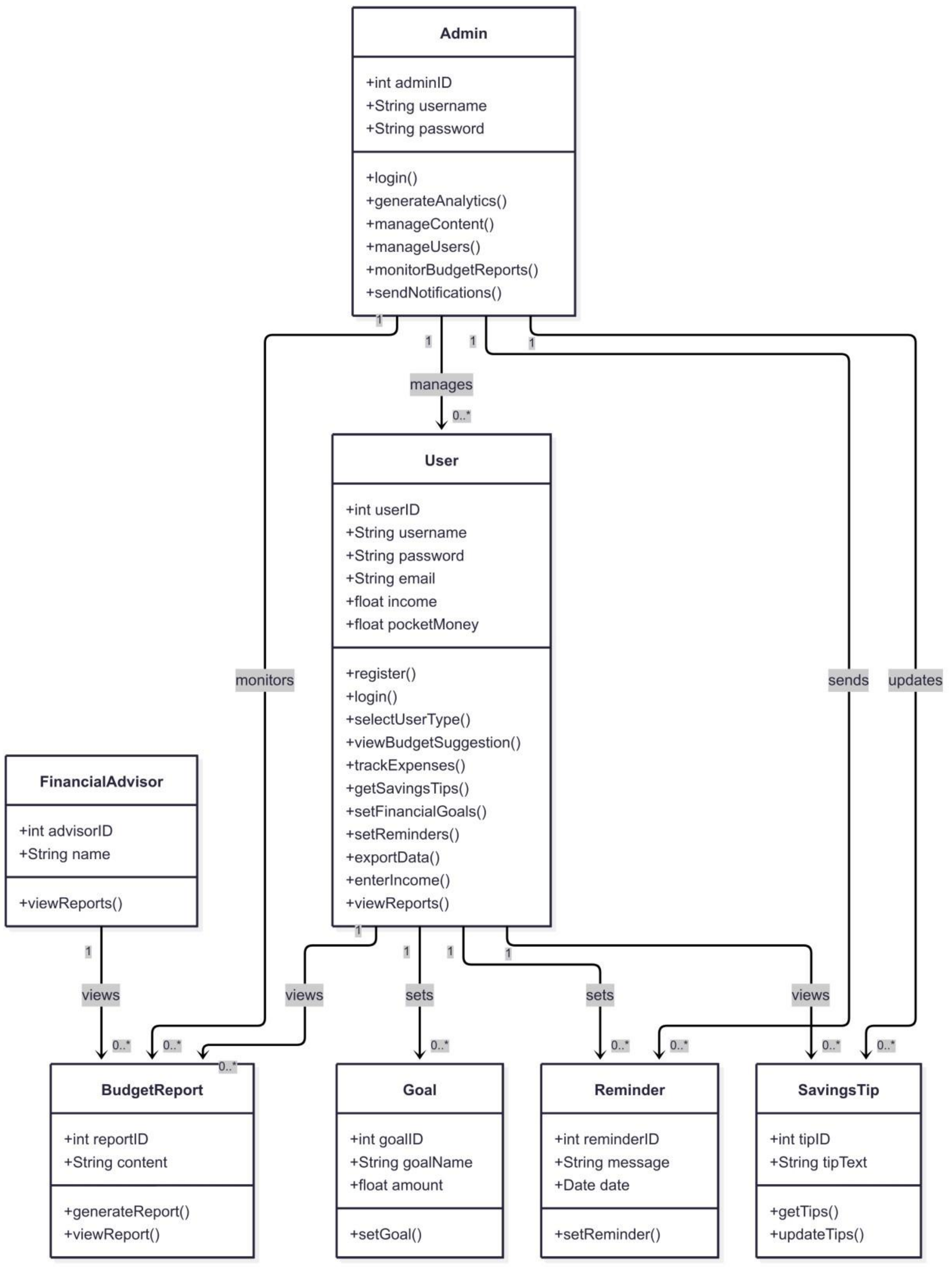
UC 16: Send Notifications

Primary Actor: Admin/System

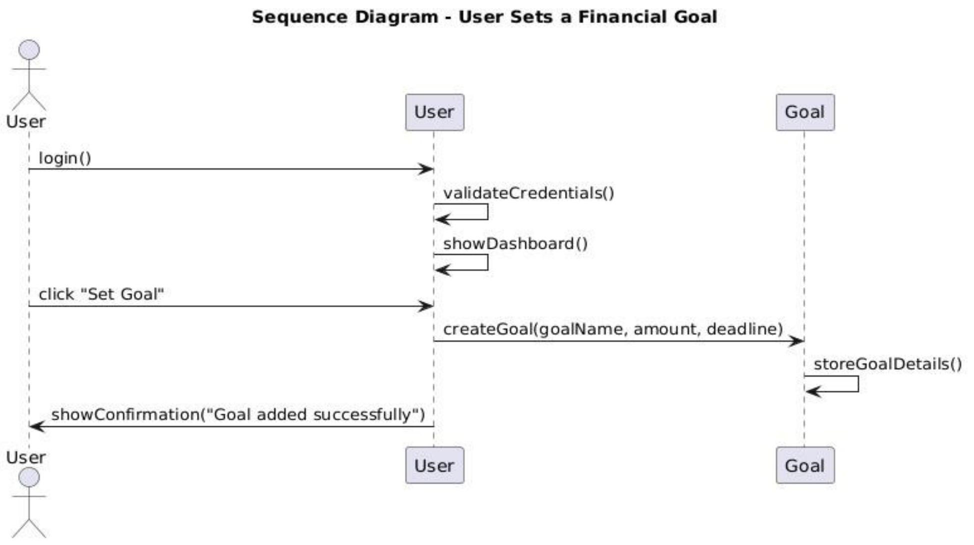
Precondition: Notification rules defined
Trigger: Admin selects “Send Notification” or system triggers reminder Main
Success Scenario:

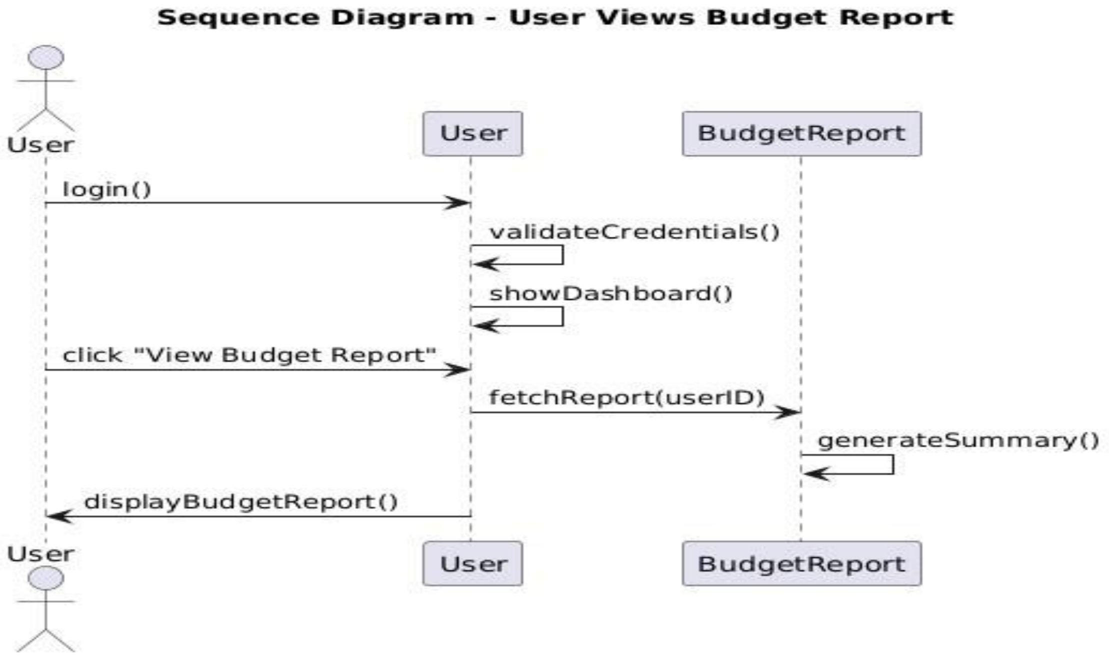
1. Notification sent to users about updates, reminders, or tips.
2. System logs notification delivery status.

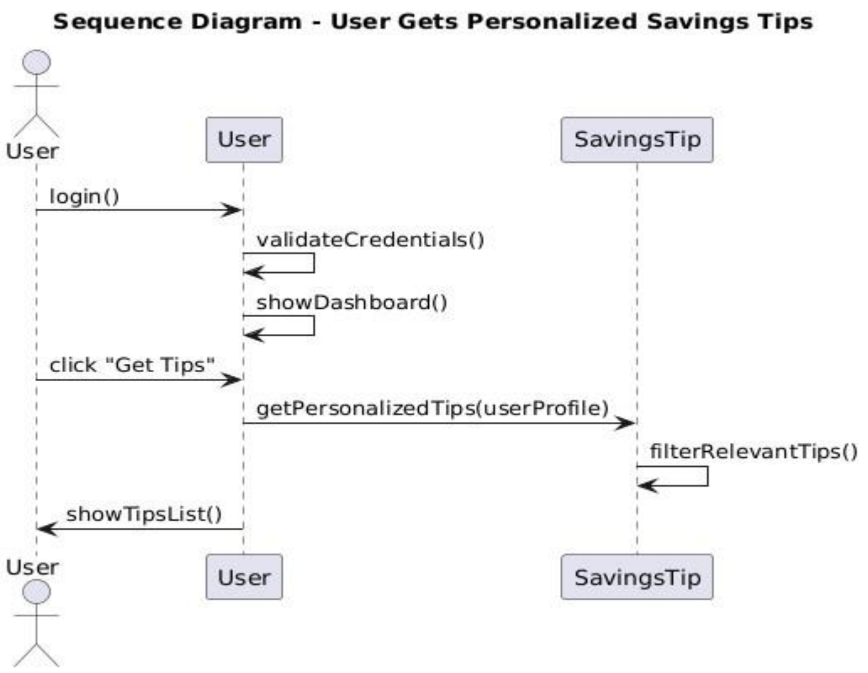
Class diagram:

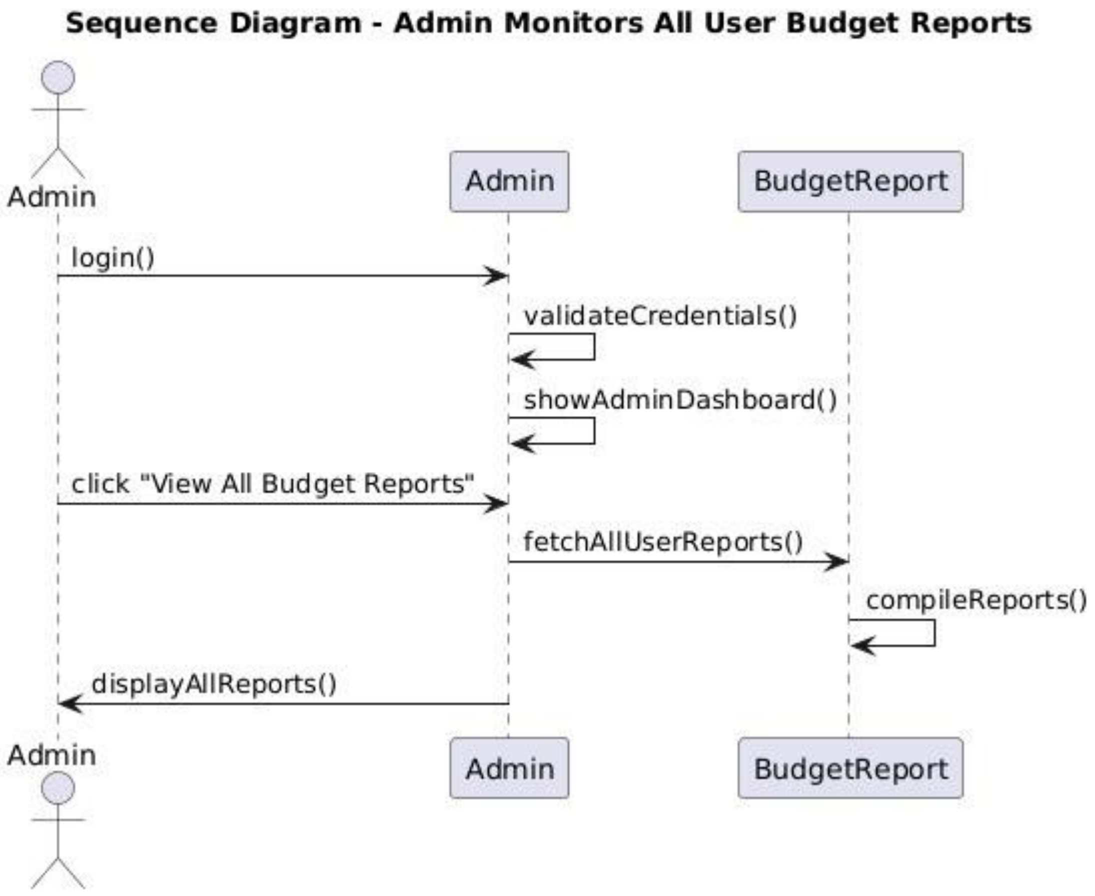


Sequence Diagram:

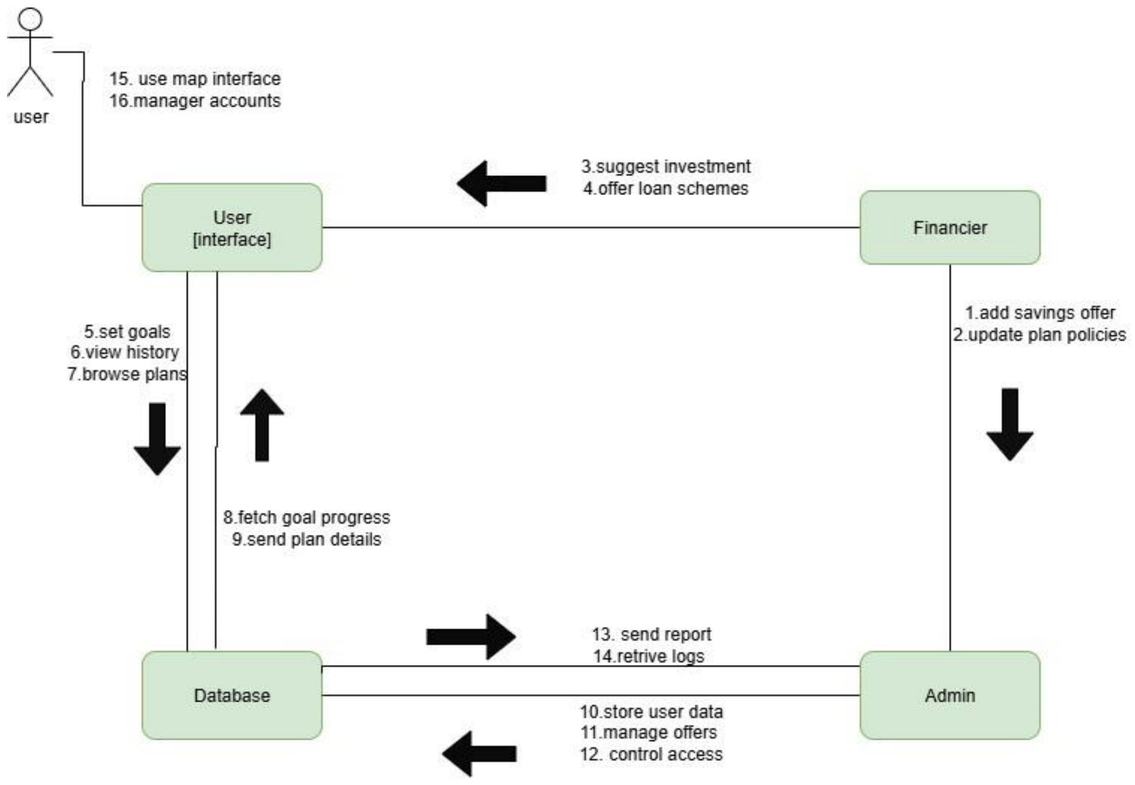

Sequence Diagram - User Sets a Financial Goal
User
User
Goal
login()
validateCredentials()
showDashboard()
click "Set Goal"
createGoal(goalName, amount, deadline)
storeGoalDetails()
showConfirmation("Goal added successfully")
User
User
Goal



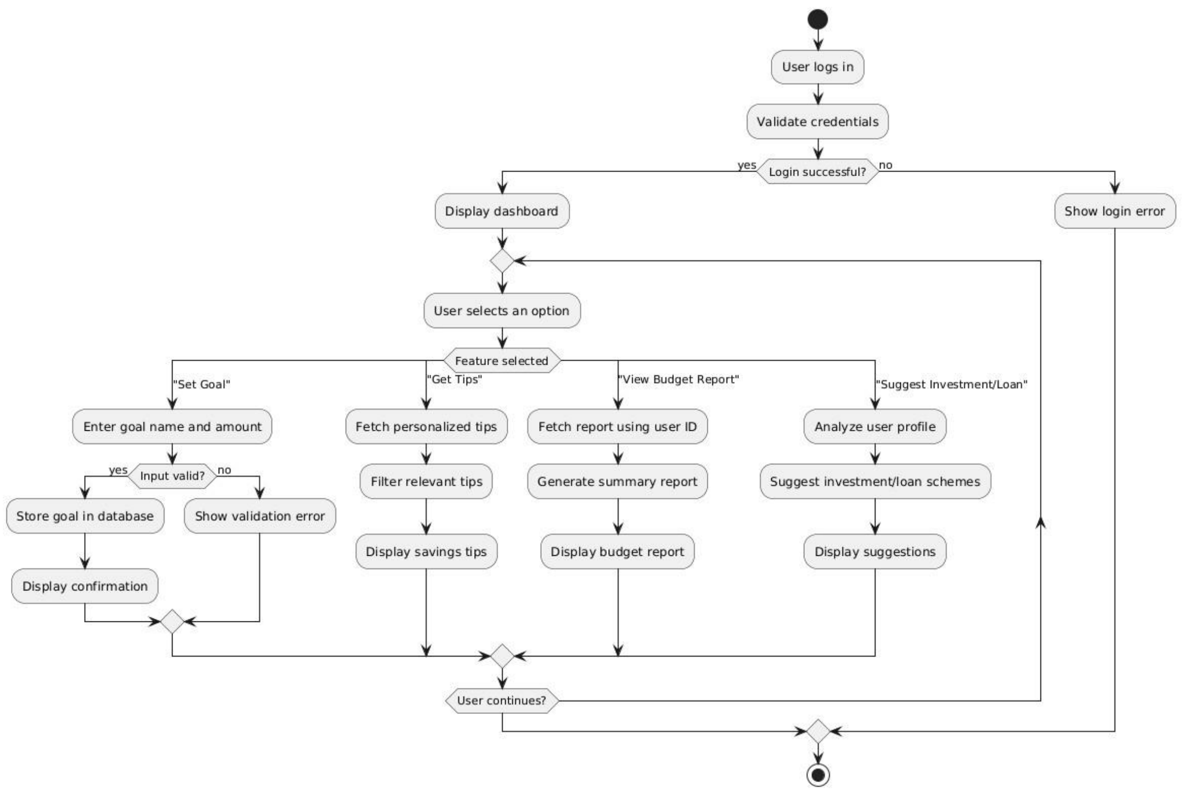

Sequence Diagram - User Gets Personalized Savings Tips
User
User
SavingsTip
login()
validateCredentials()
showDashboard()
click "Get Tips"
getPersonalizedTips(userProfile)
filterRelevantTips()
showTipsList()
User
User
SavingsTip


Sequence Diagram - Admin Monitors All User Budget Reports
Admin
Admin
BudgetReport
login()
validateCredentials()
showAdminDashboard()
click "View All Budget Reports"
fetchAllUserReports()
compileReports()
displayAllReports()
Admin
Admin
BudgetReport

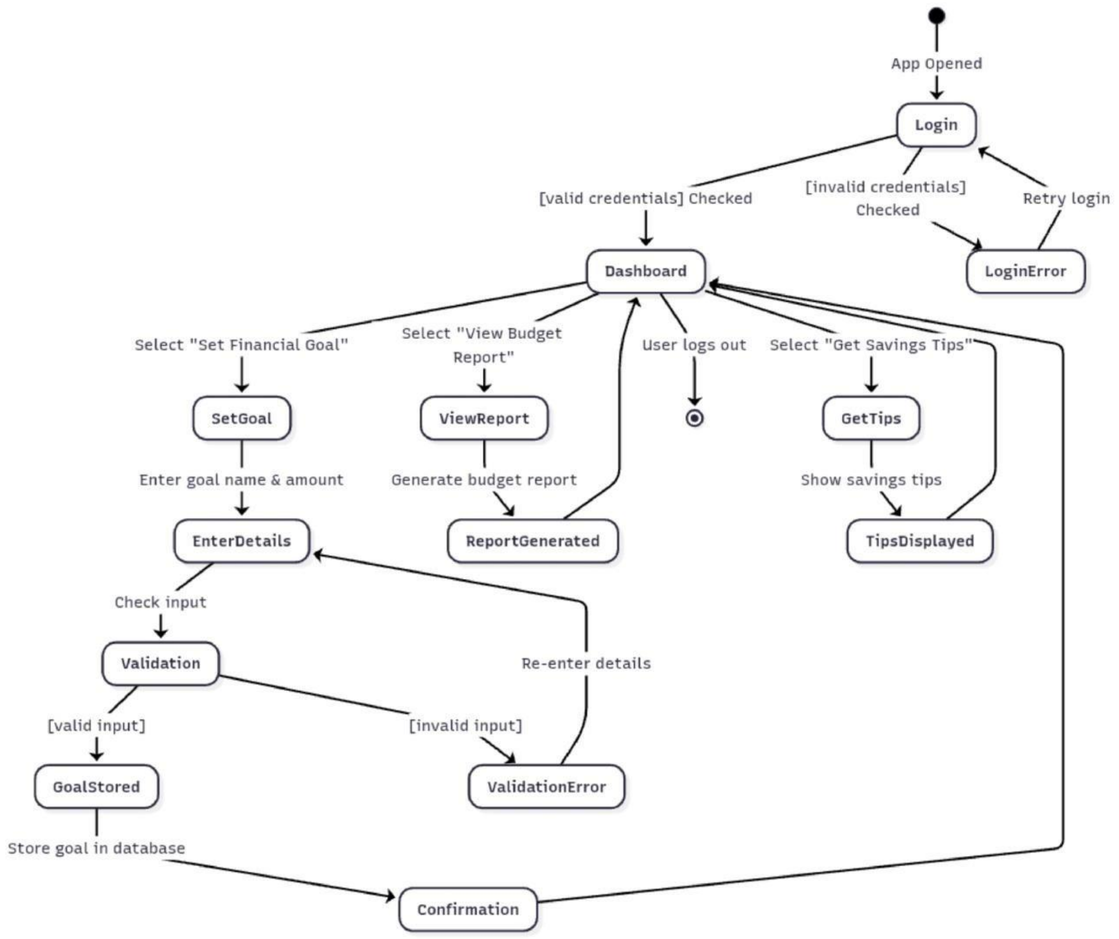
Collaboration Diagram



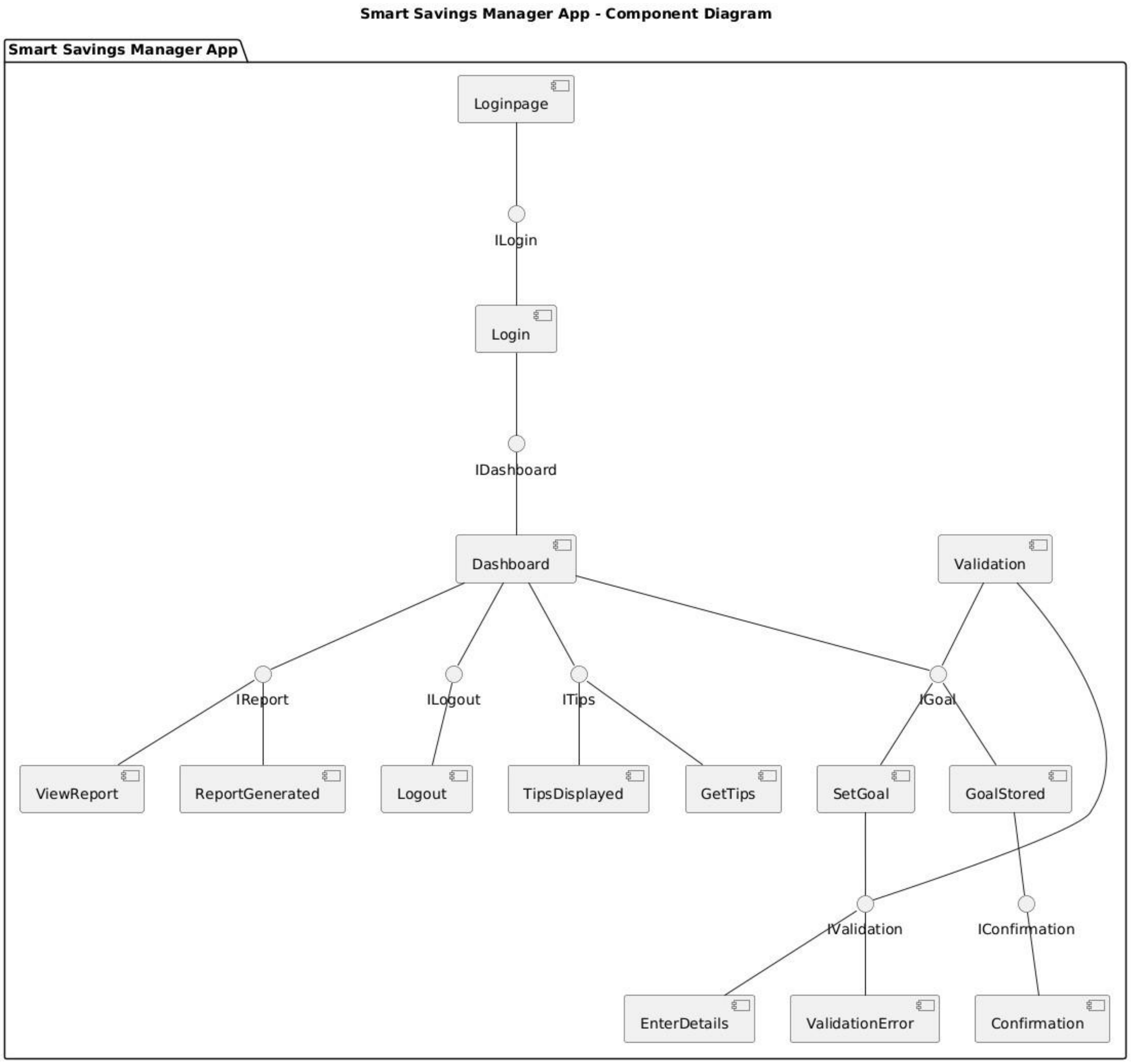
Activity Diagram



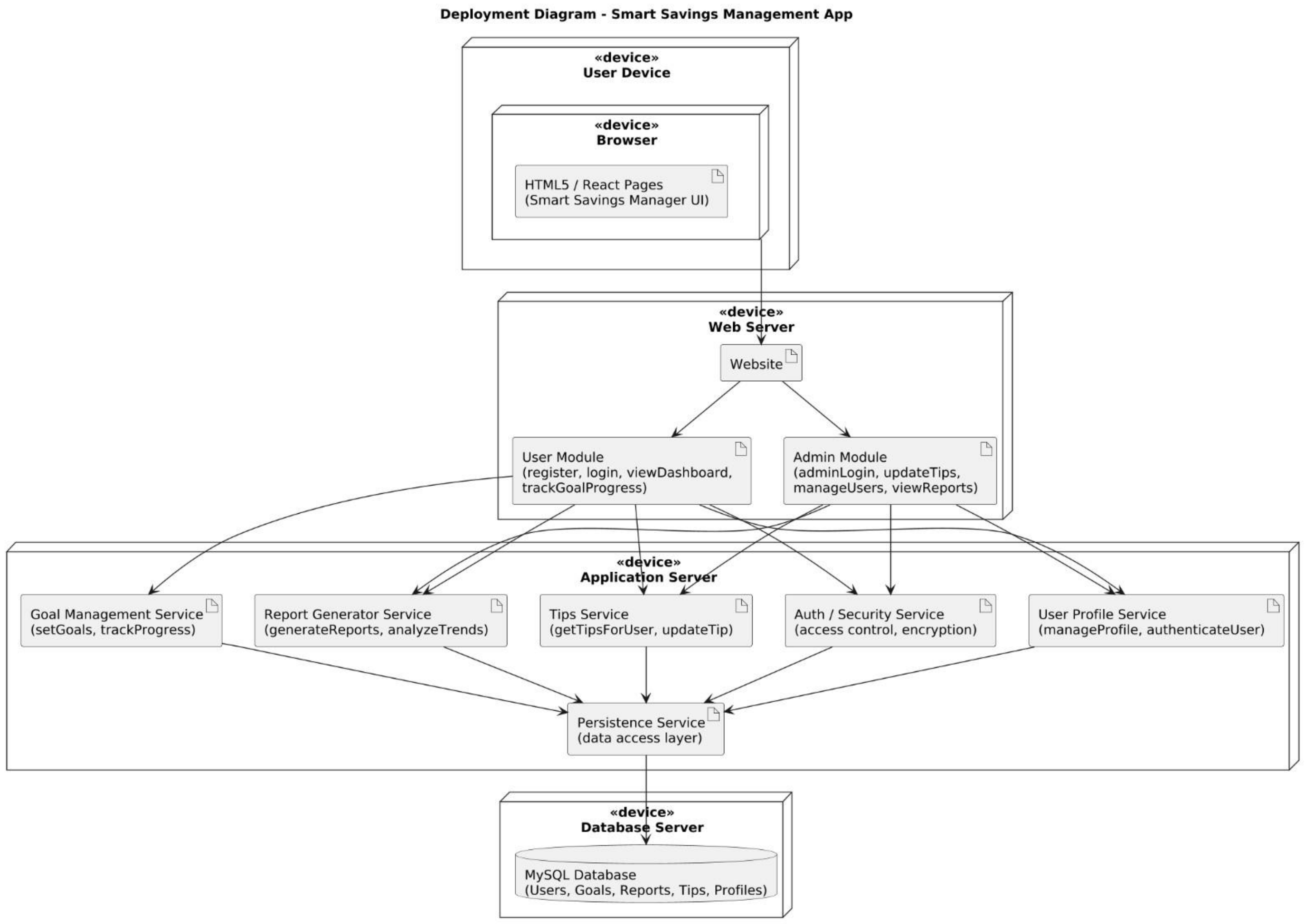
State Diagram



Component Diagram



Deployement Diagram


Deployment Diagram - Smart Savings Management App
«device» User Device
«device» Browser
HTML5 / React Pages (Smart Savings Manager UI)
«device» Web Server
Website
User Module (register, login, viewDashboard, trackGoalProgress)
Admin Module (adminLogin, updateTips, manageUsers, viewReports)
«device» Application Server
Goal Management Service (setGoals, trackProgress)
Report Generator Service (generateReports, analyzeTrends)
Tips Service (getTipsForUser, updateTip)
Auth / Security Service (access control, encryption)
User Profile Service (manageProfile, authenticateUser)
Persistence Service (data access layer)
«device» Database Server
MySQL Database (Users, Goals, Reports, Tips, Profiles)

Appendix C: To Be Determined List

- TBD-1: Specific third-party authentication library version (e.g., Firebase Authentication v9.0) to be confirmed during development.
- TBD-2: Exact multilingual support timeline and resource file structure for Spanish, French, and Hindi.
- TBD-3: Detailed performance benchmarks for 1,000 concurrent users under high load conditions.
- TBD-4: Specific currency conversion API and real-time exchange rate update frequency.
- TBD-5: Final backup server location and encryption method for 24-hour data backups.