Software Requirements Specification

for

Pennywise

Version 1.0

Prepared by

Group 11: **Group Name: InfiniteLoop**

Anushika	240315	anushikad24@iitk.ac.in
Anushka Rajora	240163	anushkar24@iitk.ac.in
Gowra Akshitha	240409	gowrak24@iitk.ac.in
Chinthala Vinayasri	240310	chinthalav24@iitk.ac.in
Lingamsetti Rajasa	240596	lrajasa24@iitk.ac.in
Medini	240645	medinim24@iitk.ac.in
Raparthi Bhavishitha	240852	bhavisht24@iitk.ac.in
Anshika Singh	240145	anshikas24@iitk.ac.in
Nagella Venkata Snehitha	240673	snehithanv24@iitk.ac.in
Rachana Reddy V	240822	rachanav24@iitk.ac.in

Course: CS253

Mentor TA: Ujjwal Kajal

Date: 23rd January 2026

CONTENTS II

REVISIONS II

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
v1.0	Chinthala Vinayasri, Daida Anushika, Raparthi Bhavishitha, Anushka Rajora, Gowra Akshitha, Lingamsetti Rajasa, Medini, Snehitha, Anshika, Rachana	First version of the SRS of Pennywise	24/01/2026

1 Introduction

1.1 Product Scope

The Pennywise system is a Personal Finance Planning and Expense Management software application designed to assist users in managing their personal finances in a structured and efficient manner. The system enables users to record income and expenses, categorize financial transactions, and define budget limits based on configurable, rule-based policies. By providing a centralized platform for financial data, Pennywise helps users gain better control over their spending and saving habits.

The primary goal of Pennywise is to promote financial awareness and discipline through analytical summaries, visual representations, and timely alerts. The system generates reports and dashboards that highlight spending patterns, budget utilization, and savings trends over time. Alerts are provided to warn users about budget overruns, excessive spending. The product aims to be reliable, user-friendly, and maintainable while demonstrating the application of standard software engineering practices.

1.2 Intended Audience and Document Overview

This document is for the following audience:

Users (Individuals managing personal finances):
The primary users of the Pennywise system are individuals who wish to manage their personal finances more effectively. These users need a system that is intuitive, reliable, and easy to navigate in order to record income, track expenses, set budgets, and monitor savings. Clear interfaces, meaningful visualizations, and timely alerts will help users understand their spending behaviour and make informed financial decisions. Ultimately, the goal is for users to seamlessly integrate Pennywise into their daily lives and develop better financial discipline and awareness.

Project Instructors and Evaluators:
Instructors and evaluators will use this document to understand the scope, objectives, and completeness of the Pennywise system. The SRS provides a clear description of system requirements, design constraints, and expected functionality, enabling evaluators to assess whether the project meets academic and technical expectations. This document also serves as a reference for grading, feedback, and verification of adherence to software engineering principles.

Developers:
Developers use this document as the primary reference for designing and implementing the Pennywise system. It provides detailed descriptions of system functionality, interfaces, and constraints, ensuring that all components such as expense tracking, budgeting rules, analytics, and alerts work together smoothly. Developers are responsible for building a secure, maintainable, and scalable system that fulfills the specified requirements while delivering a positive user experience.

Document Overview:

- ***Introduction:*** *Describes the scope of the product, the intended audience, and relevant definitions.*
- ***Overall Description:*** *Provides a high-level overview of the system functionality, assumptions, and design constraints.*
- ***Specific Requirements:*** *Lists external interface requirements, functional requirements, and use case models.*
- ***Non-Functional Requirements:*** *Describes performance, security, and software quality attributes.*
- ***Other Requirements:*** *Includes additional constraints and compliance considerations.*
- ***Appendices:*** *Contains the data dictionary and group activity log.*

1.3 Definitions, Acronyms and Abbreviations

The following terms, acronyms, and abbreviations are used throughout this document.

- ***API*** *– Application Programming Interface*
- ***Budget*** *– A predefined financial limit allocated for a specific category or time period*
- ***Budget Overrun*** *– A condition where recorded expenses exceed the allocated budget*
- ***Dashboard*** *– A graphical interface displaying financial summaries and analytics*
- ***Expense*** *– Any monetary outflow recorded by the user*
- ***Income*** *– Any monetary inflow recorded by the user*
- ***SRS*** *– Software Requirements Specification*
- **SQL -** Structured Query Language
- ***UI*** *– User Interface*
- ***User*** *– An individual who uses the Pennywise system to manage personal finances*

1.4 Document Conventions

- *The document follows **IEEE Software Requirements Specification (SRS)** standards.*
- ***NORMAL font** of size **11 or 12** is used throughout the document.*
- *The document is **single-spaced** with **1-inch margins** on all sides.*
- *Section and subsection headings follow a **hierarchical numbering format** as per the template.*
- ***Bold text** is used for section and subsection titles.*
- *Lists and tables are used where necessary to improve **clarity and readability**.*
- *All requirements are written in a **clear, precise, and unambiguous** manner to avoid misinterpretation.*

1.5 References and Acknowledgments

The following documents and resources were referenced during the preparation of this Software Requirements Specification:
IEEE Std 830-1998: IEEE Recommended Practice for Software Requirements Specifications
Course material and guidelines provided for CS253 – Software Engineering

We utilised Figma, Canva to craft visually compelling graphs and flowcharts, effectively translating our ideas into a concise and impactful pictorial representation.

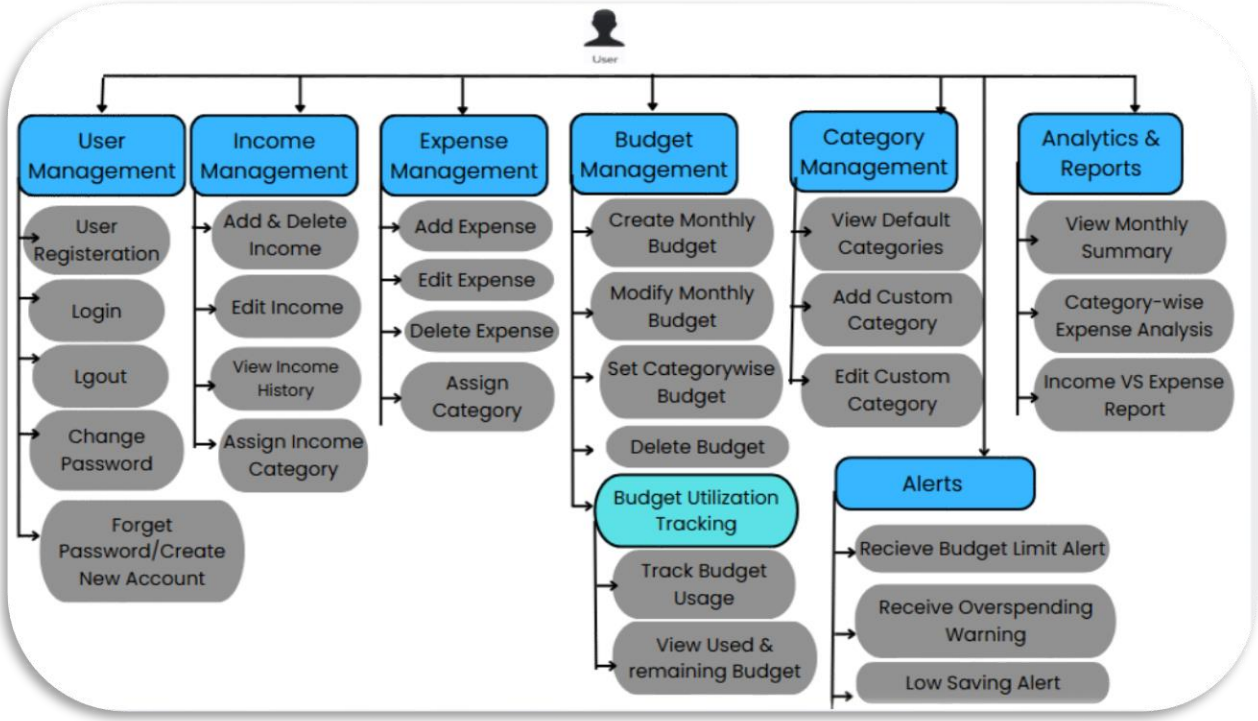
The team Infinite Loop acknowledges the guidance and support provided by the course instructors and teaching assistants throughout the development of the Pennywise project.

2 Overall Description

2.1 Product Overview

The Personal Income and Expense Tracker System is a finance management application that helps users track their income, expenses, budgets, and savings in one place. This product is a standalone system, which means it works independently and does not depend on any other software to perform its main functions. It securely stores user data and provides useful financial information through reports and analysis.

The system allows users to register and log in securely. Users can add, edit, and delete income and expense records, assign categories, and view their complete transaction history. It also supports monthly budgeting where users can set budgets for different categories and monitor how much they have spent. The system gives alerts when spending reaches the warning level or goes above the maximum limit. It also provides reports such as monthly summaries, category-wise spending, income vs expense comparison, and savings calculation.



2.2 Product Functionality

The platform includes the following features to enhance the user experience:

- User Management (register, login/logout, secure authentication)
- Income Management (add/edit/delete income, categorize sources, view income history)
- Expense Management (add/edit/delete expenses, assign categories, view expense history)
- Transaction Filtering (filter by date and category)
- Category Management (view default categories, add/edit/delete custom categories with constraints)
- Budget Management (create/modify monthly and category-wise budgets)
- Budget Utilization Tracking (used vs remaining budget monitoring)
- Alerts (warning/max budget alerts, low savings)
- Reports & Analytics (monthly summaries, category-wise analysis, income vs expense, savings)

2.3 Design and Implementation Constraints

2.3.1 Hardware limitations:

Memory Requirements:
The application must have a dedicated database to store all the user information and item details

Timing Requirements:
Users expect quick response times. Any delays in loading pages or processing user requests can lead to a poor user experience.

2.3.2 Security Considerations:

- The system shall enforce **secure authentication mechanisms**, including password hashing and session management.
- Sensitive user data shall not be transmitted to external systems.

2.3.3 Tools, Languages and Databases:

MySQL

- Used for persistent storage of: User accounts, Income and expense records, Categories, Budgets and alerts

Frontend:

- React (JavaScript library)
- HTML
- CSS
- JavaScript

Backend:

- Node.js
- Express.js

2.3.4 Communication protocols:

The system uses HTTP/HTTPS protocols for client–server communication, with HTTPS preferred to ensure secure transmission of sensitive user and financial data through encrypted request–response mechanisms.

2.3.5 Design Conventions and Programming Standards:

- The codebase shall be structured so that new features (e.g., additional reports or alert rules) can be added with minimal changes to existing code.
- Industry-accepted coding standards shall be followed for all programming languages used in the project.
- The system shall follow a modular and layered architecture, separating user interface, business logic, and data access layers.

2.4 Assumptions and Dependencies

2.4.1 Assumptions:

- The system will be used primarily as a **web-based application** accessed through modern web browsers.
- The application is intended for **personal finance tracking only** and does not require integration with real banking systems.
- Financial data entered by users is assumed to be **manually provided and accurate**.
- The system does not require real-time financial predictions or live market data.
- Security requirements are assumed to be sufficient with **standard authentication and encryption mechanisms**.

2.4.1 Dependencies:

- The backend relies on **Node.js runtime and the Express framework** for handling server-side logic and API development.
- The system depends on **frontend libraries** such as React for building the user interface.

- The project depends on a **MySQL database management system** for persistent data storage and retrieval.
- Several **open-source Node.js packages** are reused for essential functionality, including authentication, password encryption, input validation, and database connectivity.
- Data visualization features depend on **charting libraries** for generating graphical reports and analytics.
- The system relies on **web browsers** to support modern JavaScript execution and HTTP/HTTPS communication.
- Development and collaboration depend on **version control tools** such as Git.
- **RESTful APIs**: The project utilizes standard API protocols to facilitate seamless data exchange between the frontend and the backend server.

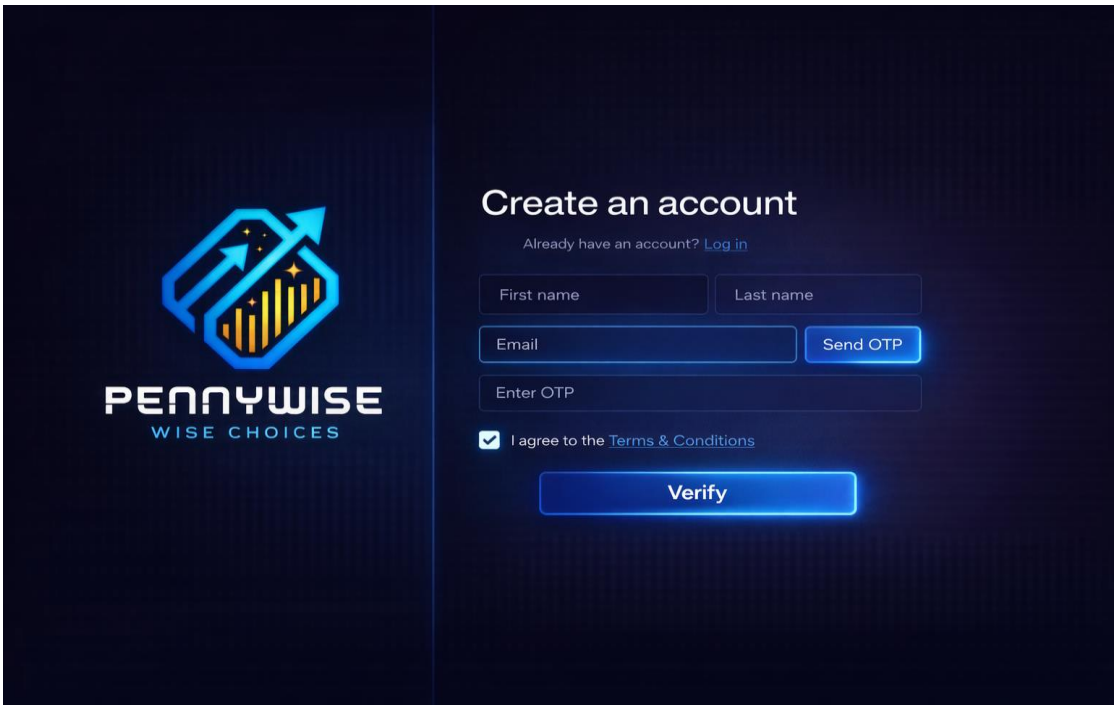
3 Specific Requirements

3.1 External Interface Requirements

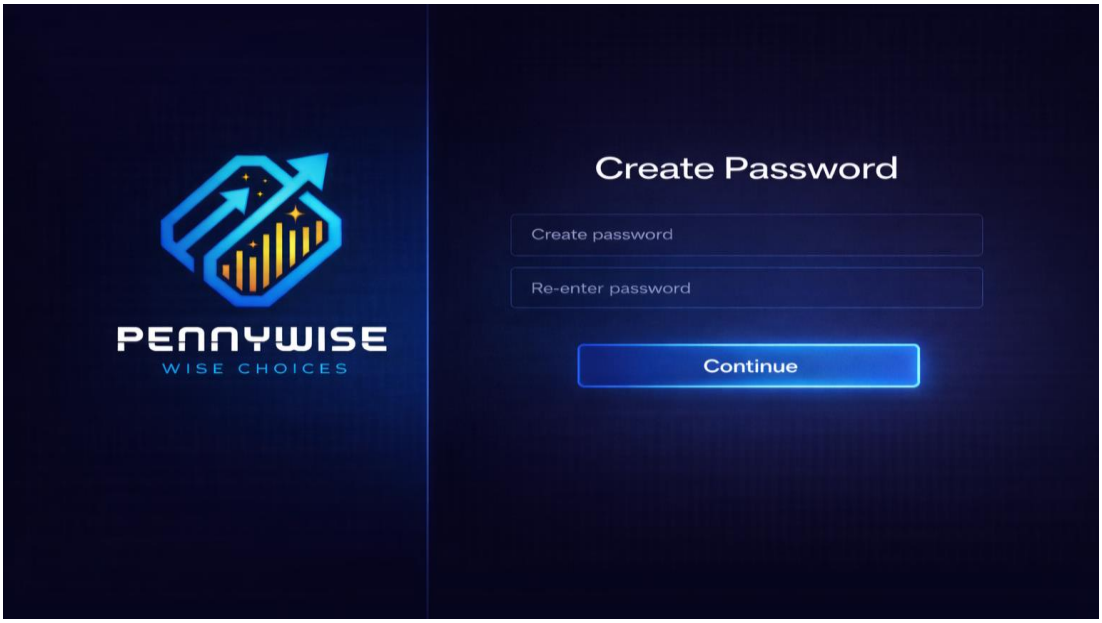
3.1.1 User Interfaces

Users shall be able to access the system through a web-based interface. To use the platform, users shall create an account using a Gmail address with an OTP-based verification mechanism. After successful registration and login, users shall be able to access various web pages that allow them to select predefined expenditure categories, customize categories as needed, and set budget limits for their spending. Users shall be able to record and maintain their transaction history by adding expense entries under the selected categories. The system shall enable users to view expenditure analysis through visual representations such as pie charts and graphs, showing total spending across selected categories on a monthly and yearly basis. Additionally, users shall be able to view their spending patterns on a daily and monthly basis to better understand and track their financial behavior.

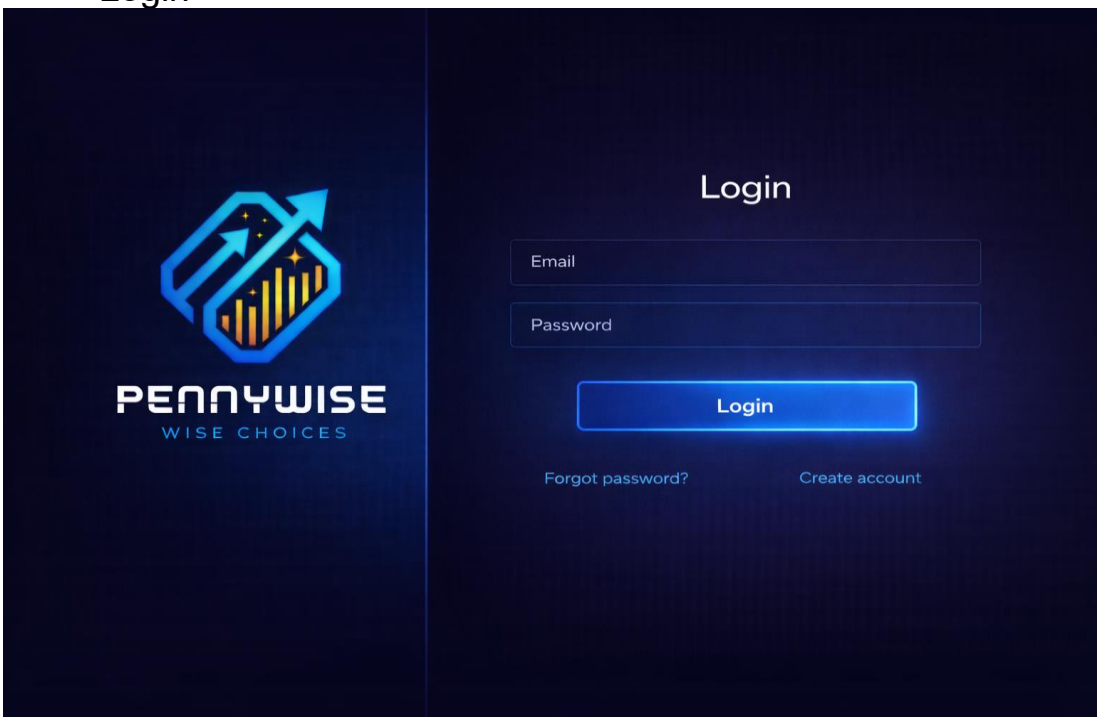
- Create an account



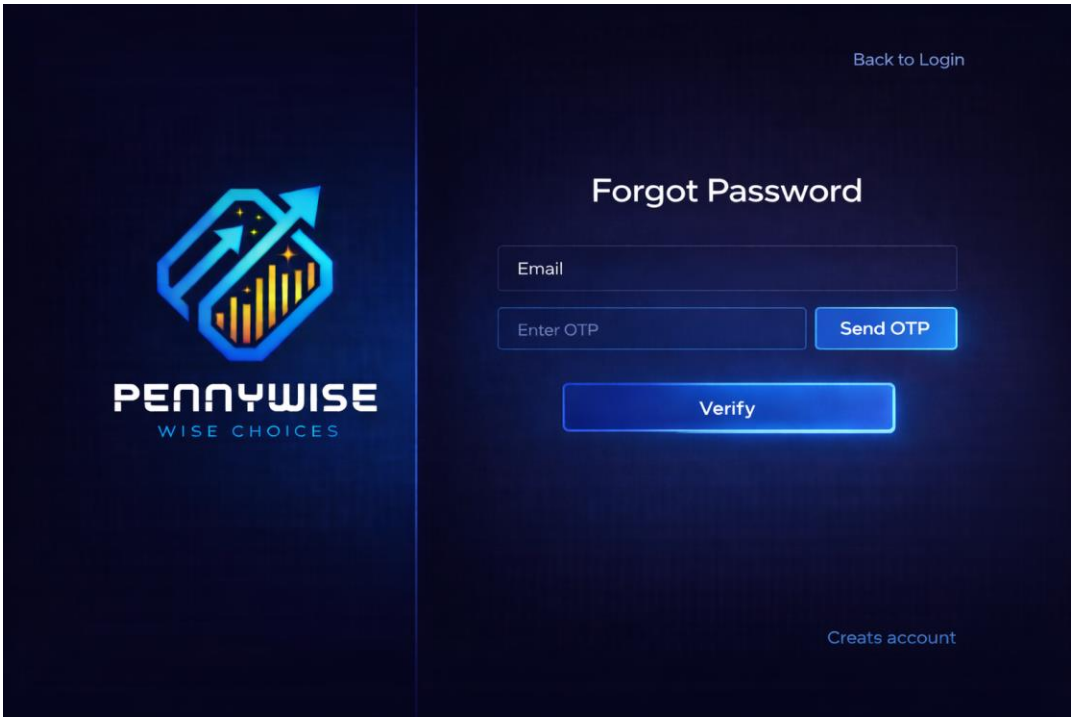
- Create Password



- Login



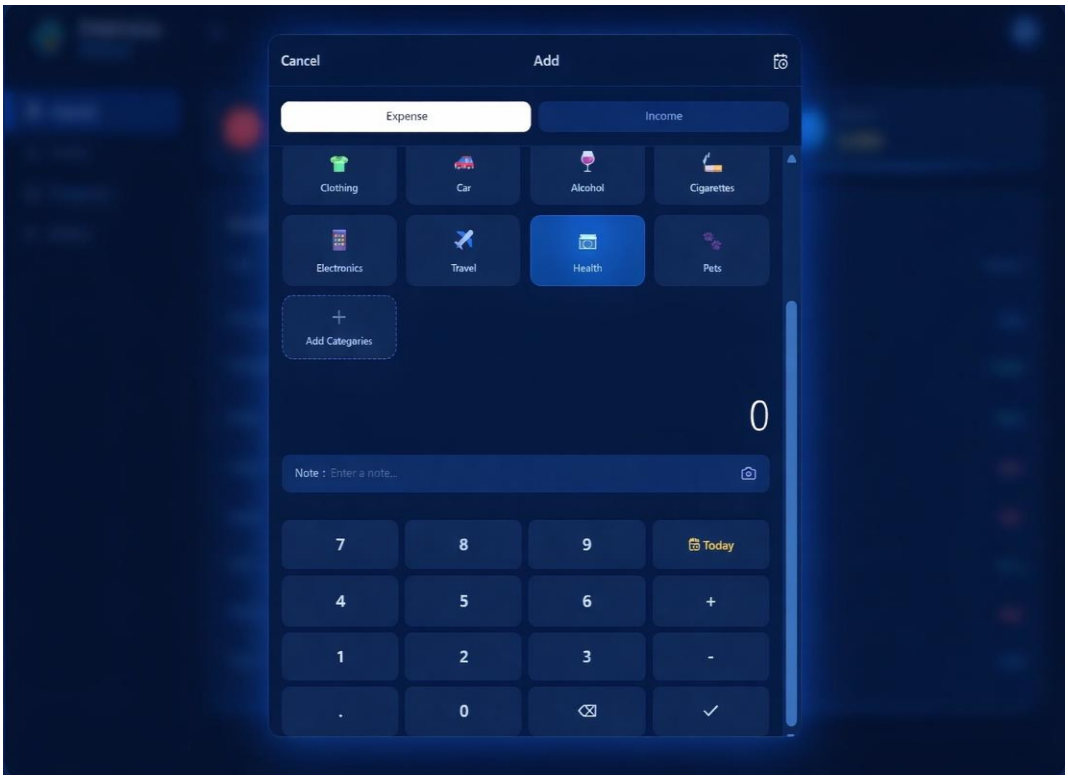
- Forgot Password



- Home Page

After logging in, the user is directed to the **Home** page, which provides a daily overview of financial activity. This page displays total income, total expenses, and the current balance, along with a chronological list of transactions showing the amount spent or earned each day and the corresponding expense categories. By selecting the **Add Transaction** option, the user is navigated to the next page to record a new income or expense entry.

- Add Transaction



Upon selecting the **Add Transaction** option, the user is navigated to a transaction entry page where they can record a new expense or income. On this page, the user selects the transaction type and category, enters the amount, and optionally adds a note for better understanding of the purpose of the expense or income. The interface also allows the user to add new categories if required, ensuring flexibility in managing and organizing financial transactions.

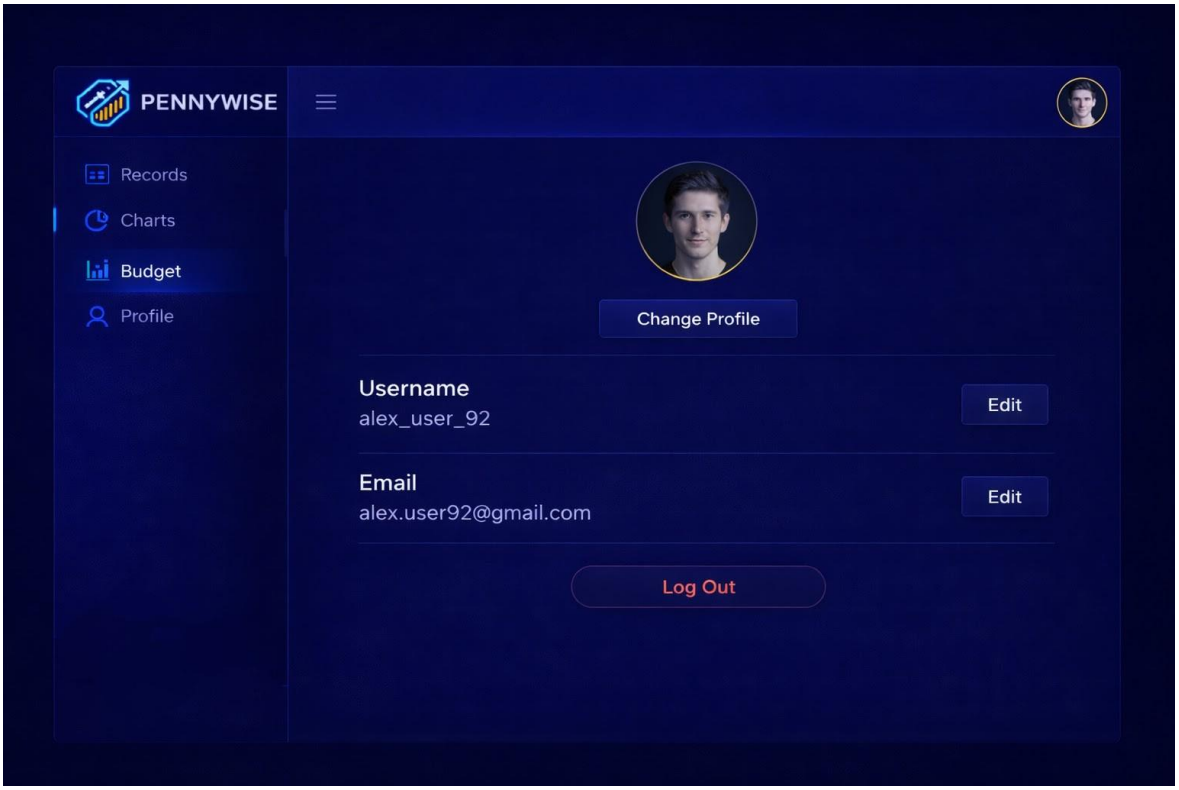
- Budget

Upon selecting the **Budget** option, the user is directed to a budget management page where all existing budget categories are displayed along with their allocated budget, remaining balance, and expenses incurred. The user can select an existing category to view or modify its details, edit the allocated budget amount, or set a budget for a category if it has not been previously defined. Additionally, the page provides an **Add Category** option, enabling the user to create new categories and assign budgets as required, thereby allowing flexible and effective control over category-wise budgeting.

- Charts

Upon selecting the **Charts** option, the user is navigated to a dashboard page that provides a comprehensive visual representation of expenses. This page allows the user to toggle between **monthly** and **yearly** views, displaying the corresponding expense summaries. It includes a **line graph** illustrating expense trends, showing expense versus day in the selected month and expense versus month in the selected year. A **pie chart** presents the distribution of total expenses across different budgeted categories, clearly indicating how much money has been spent in each category during the selected period. Additionally, the page displays a detailed breakdown of category-wise expenses, along with a monthly or yearly summary that highlights total income, total expenses, and savings, enabling the user to effectively analyze and manage their spending patterns.

- Profile



3.1.2 Hardware Interfaces

The software interacts with server-side hardware to store user transaction history, perform financial data analysis, generate expenditure visualizations, and trigger rule-based alerts. The server hardware handles data processing and retrieval requests efficiently, ensuring accurate analytics, timely alerts, and smooth system operation. Users can access the platform using a computer equipped with a modern web browser and an active internet connection. The system's responsive web interface ensures that transaction entry, data visualization through pie charts and graphs, and alert notifications are accessible and function smoothly on both desktop and mobile devices.

3.1.3 Software Interfaces

The system interfaces with web server software to handle HTTP/HTTPS requests and responses generated by user interactions. Backend services process business logic related to transaction management, budgeting, analysis, and alert generation. The application interfaces with a database management system to store and retrieve user transaction history, category data, budget information, and alert rules. Additionally, the system uses data visualization libraries to generate pie charts and graphical reports, and internal notification components to display alerts and warnings within the user interface

3.2 Functional Requirements

3.2.1 User Management

The system shall provide user management functionalities to ensure secure and personalized access to financial data.

User Registration:
The system shall allow new users to create an account by providing required details such as username, email address, and password.

User Login/Logout:
The system shall authenticate registered users and allow them to log in securely. Logged-in users shall be able to log out safely at any time.

Email Verification:
The application will verify the user's email by sending a one-time password (OTP) to their registered email address.

Password Validation:
The system shall enforce password policies such as minimum length, use of alphanumeric characters, and secure storage using encryption or hashing techniques.

Secure Authentication:
The system shall implement secure authentication mechanisms to prevent unauthorized access, including session management and protection against invalid login attempts.

3.2.2 Income Management

The system shall allow users to manage and track their sources of income effectively.

Add Income:
Users shall be able to add income details.

Edit Income:
The system shall allow users to modify previously recorded income entries.

Delete Income:
Users shall be able to remove incorrect or outdated income records.

Categorize Income Sources:
Users shall be able to classify income under categories such as food, shopping, travel, or other custom sources.

3.2.3 Expense Management

The system shall provide comprehensive expense tracking features to monitor user spending.

Add Expense:
Users shall be able to record expenses by entering amount,category.

Edit Expense:
The system shall allow users to update expense details if changes are required.

Delete Expense:
Users shall be able to delete incorrect or unnecessary expense entries.

Assign Category:
Each expense shall be assigned to a predefined or custom category.

View Expense History:
The system shall provide a History view that displays only expense transactions recorded by the user.
The expense history shall present a chronological list of all expense transactions.
Each record shall include details such as:
Date of transaction
Expense category
Amount spent

The History view shall allow users to review past spending for better financial analysis and tracking.

3.2.4 Category Management

The system shall enable users to organize income and expenses using flexible categories.

View Default Categories:
The system shall provide a set of predefined categories such as Food, Shopping, Travel, Utilities, Entertainment and Others.

Add Custom Categories:
Users shall be able to create personalized categories according to their financial needs.

Edit Categories:
The system shall allow modification of category names and descriptions.

Delete Categories (with Constraints):
Categories associated with existing income or expense records shall not be **deleted** *unless those records are reassigned to others category.*

3.2.5 Budget Management

The system shall assist users in planning and controlling their spending through budgeting features.

Create Monthly Budgets:
Users shall be able to define budgets for a specific month.

Assign Budgets per Category:
The system shall allow allocation of budget limits for individual expense categories.

Modify Budgets:
Users shall be able to update budget values as financial priorities change.

View Budget Utilization:
The system shall display current spending against allocated budgets using numerical indicators.

3.2.6 Analytics & Reports

The system shall provide analytical insights through graphical representations to help users understand income, expenses, and savings behavior over time.

Expense Distribution Charts (Pie Charts)
The system shall display pie charts representing expense distribution across different categories.
Separate pie charts shall be available for:

Monthly expense distribution
Yearly expense distribution

These charts shall show the contribution of each expense category for the selected time- period.

Daily Expense Trend (Dotted Graph)

The system shall provide a dotted graph representing daily expenditure for a selected month.
Each dot shall indicate the total expense incurred on a particular day.
This graph shall help users analyze daily spending patterns and irregular expenses.

Monthly Expense Trend (Dotted Graph)
The system shall display a dotted graph showing total expenditure for each month within a selected year.

Each dot shall represent the total expenses for a month.

This graph shall assist users in identifying long-term spending trends and seasonal variations.

3.2.7 Alerts :

The system shall provide notifications to promote financial discipline.

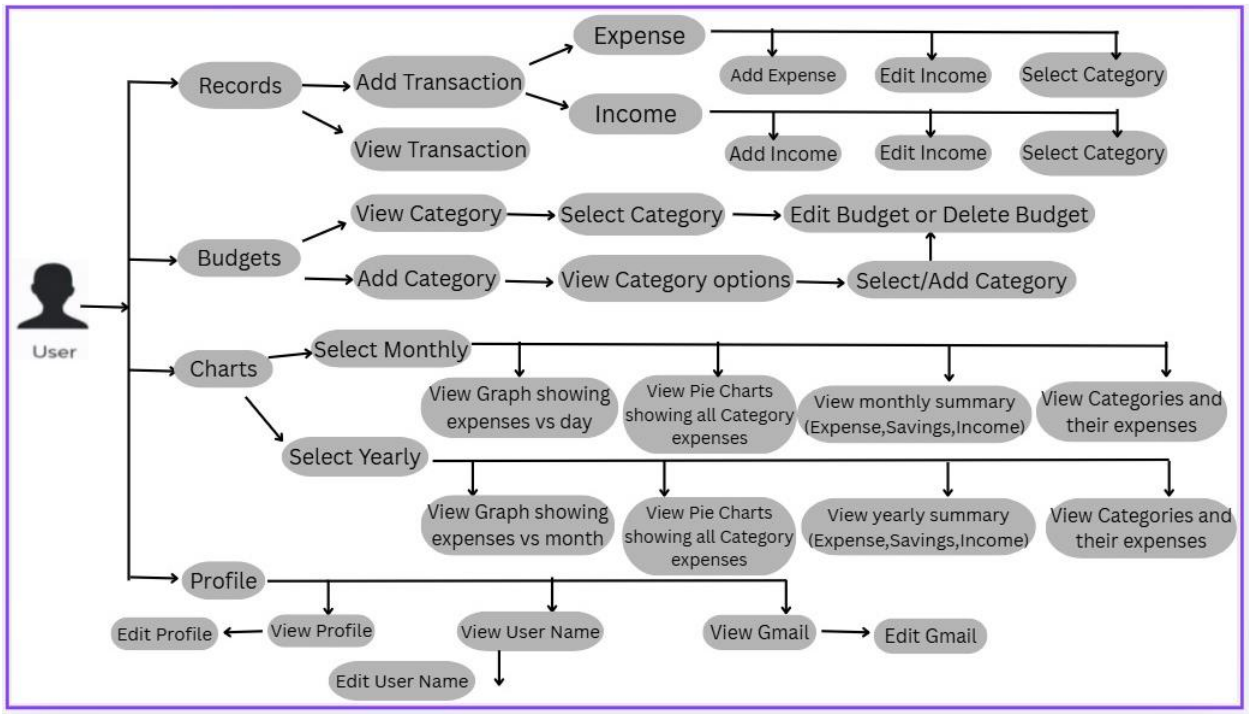
Budget Overrun Alerts:

Alerts shall be triggered when spending exceeds defined budget limits.

Low Savings Alerts:

The system shall notify users when monthly savings fall below a recommended threshold.

3.3 Use Case Model



Author *– Vinayasri,Anushika*

Purpose-The purpose of this use case is to describe how a user interacts with the Pennywise system to manage income and expenses, organize categories, set and modify budgets, view analytical charts and summaries, and update profile information to effectively track and control personal finances

Requirement Traceability-This use case traces to requirements related to user authentication, income and expense management, category and budget management, financial analysis through charts and summaries, and profile viewing and editing functionalities.

Priority – The priority of this use case is high, as it represents the core functionality of the Pennywise system and is essential for user interaction with the application.

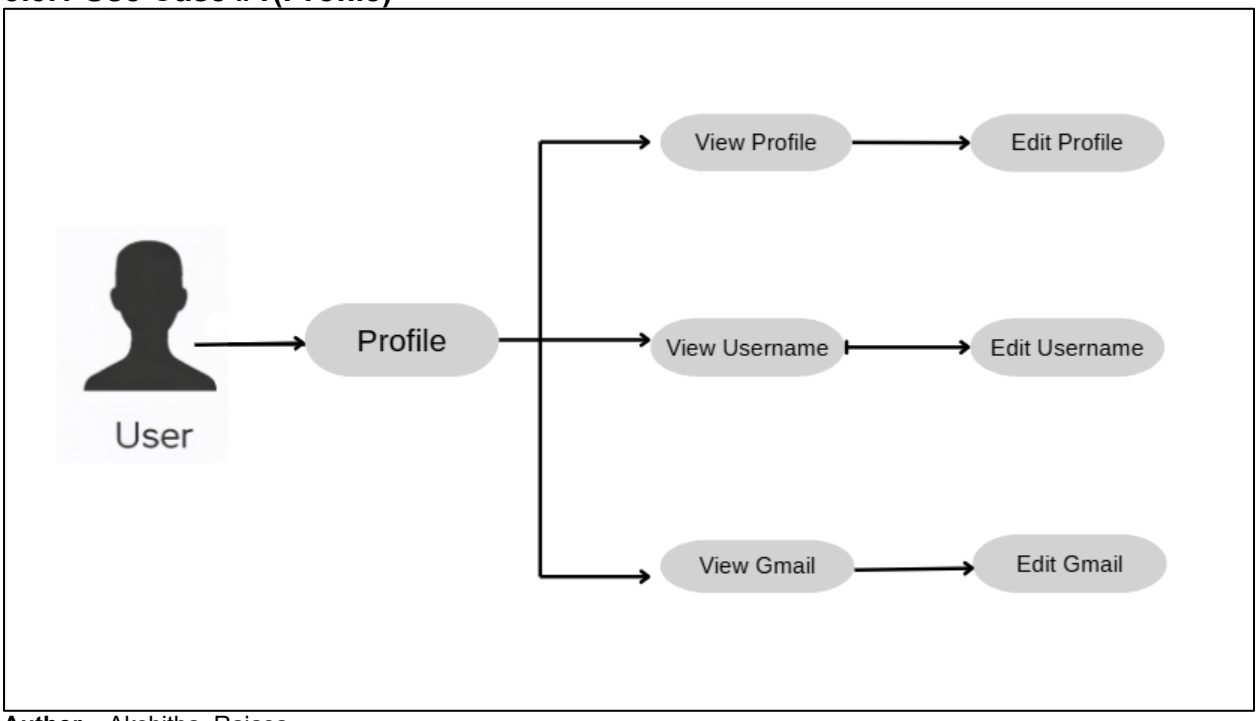
Preconditions – The user must be registered, successfully logged into the system, and have access to the required system services and an active internet connection.

Postconditions – After successful execution of this use case, transactions and budget updates are stored in the database, analytical views are updated with the latest data, and any profile changes made by the user are saved and reflected in the system.

Actors – The primary actor is the user, while the system and database act as supporting actors to process requests and store data.

Exceptions: None

3.3.1 Use Case #1(Profile)



Author – Akshitha, Rajasa

Purpose – The user can view and edit Profile, Username, Gmail.

Requirements Traceability – Profile page, View profile, edit profile, view username, edit username, view Gmail, Edit Gmail.

Priority – Medium

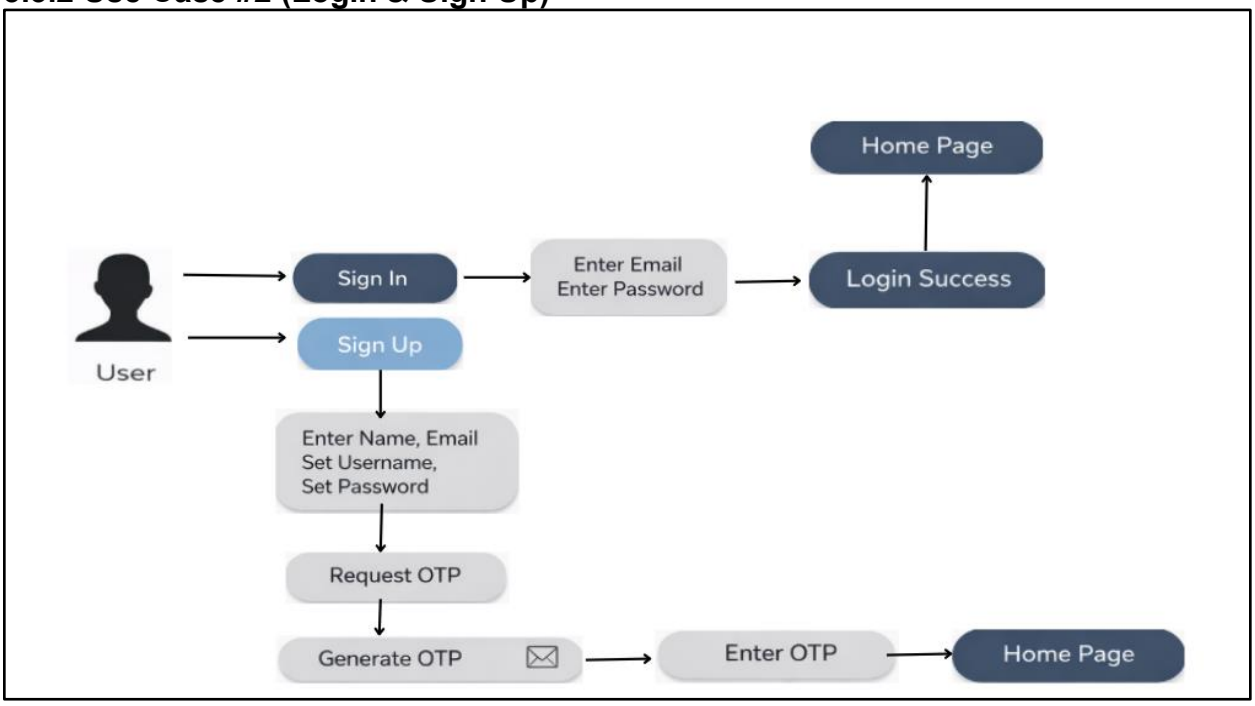
Preconditions –The user should have an account

Postconditions - The user can view the edited profile.

Actors – User of the website, and server.

Exceptions – None

3.3.2 Use Case #2 (Login & Sign Up)



Author *– Vinayasri,Anushika.*

Purpose - Allow users to securely log in or create a new account and access the Home Page.

Requirements Traceability – User login, User registration, OTP verification, Profile photo upload, Redirect to Home Page

Priority - High – Required for system access.

Preconditions - User has internet access, Valid email ID, for login: user already registered.

Post conditions - User is authenticated, User redirected to Home Page, New user account created (if sign up).

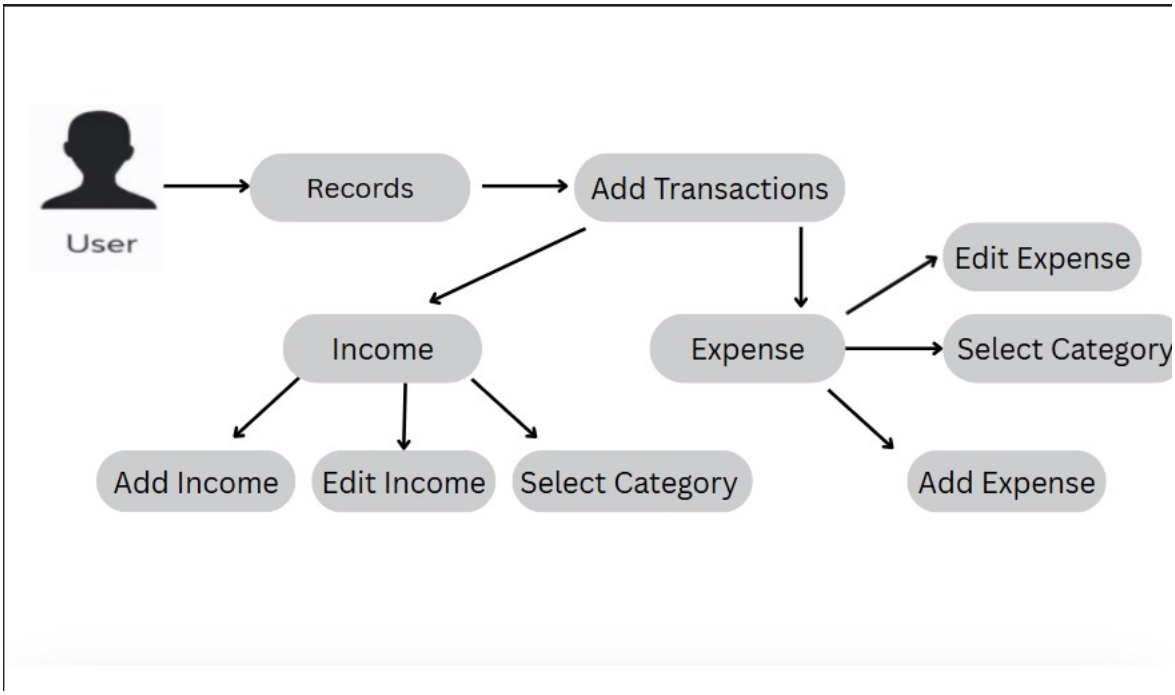
Actors – User (Primary), Email Service (Supporting).

Exceptions - Invalid login credentials.

Includes-Verify OTP, upload User Photo

Notes/Issues - Define OTP validity time, Password rules required, Limit photo size and format

3.3.3 Use Case #3(income and expense Management)



Author: Bhavishitha, Vinayasri, Anushka

Purpose: This use case allows the user to manage income records by adding, editing, deleting income and viewing the income history. allows the user to manage expense records by adding, editing, deleting expenses and viewing the expense history.

Requirements Traceability: Add income, edit income, add income category, Add Expense, Edit Expense, Delete Expense, Assign Expense Category, View Expense History.

Priority: High

Preconditions: User must be logged in; expense categories must be available in the system.

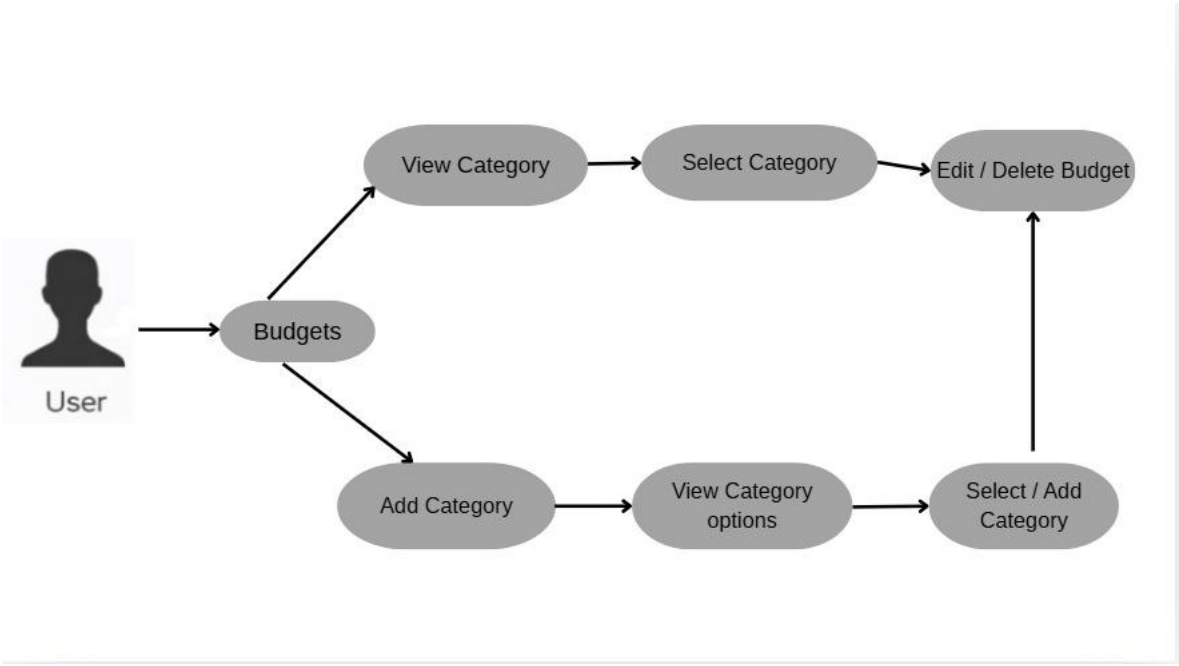
Post Conditions: Income history is displayed to the user; Expense records are updated in the database and changes are reflected in expense history and reports.

Actors: User, System, Database

Exceptions: Invalid inputs, Income record not found, Database or server failure.

Notes: Filtering and sorting of income history can be added as enhance

3.3.4 Use Case #4 (Budget Management)



Author: Akshitha

Purpose: This use case allows the user to manage budget by editing, deleting budget and adding budget to new category.

Requirements Traceability: View Category, Add Category, edit/delete budget.

Priority: High

Preconditions: User must be logged in, and Budget Category must be available in the system.

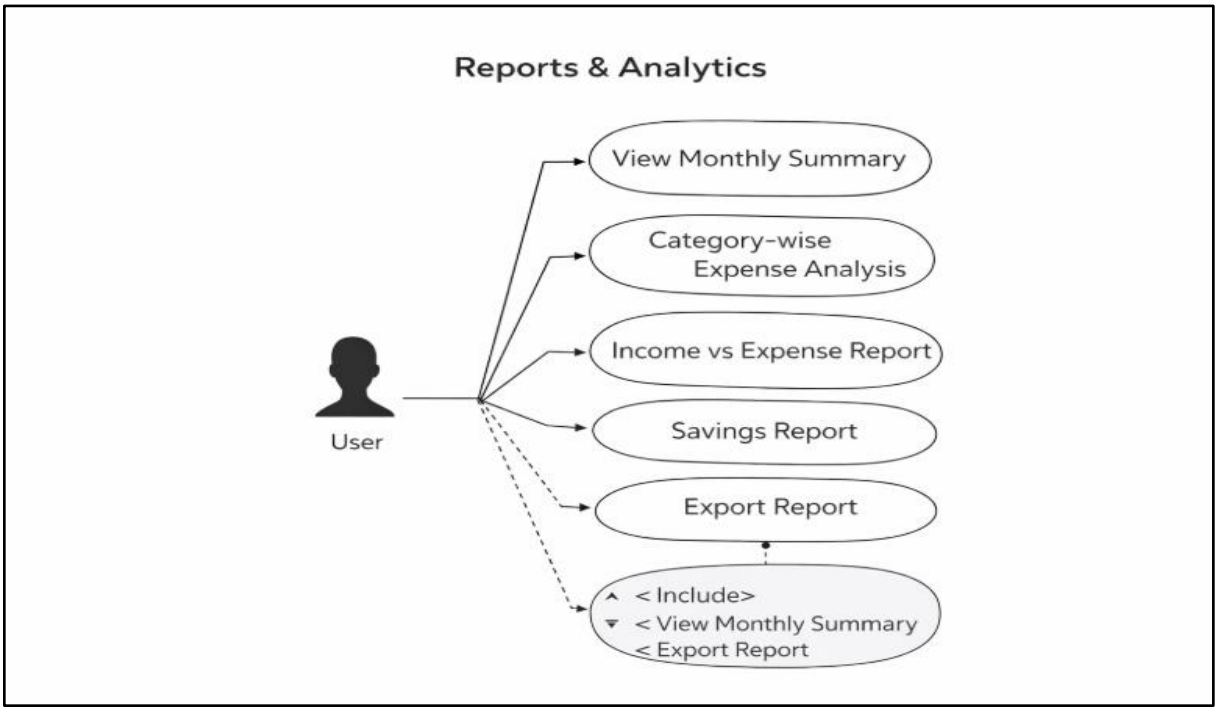
Post Conditions: Budget records are updated in the database and changes are displayed to the users.

Actors: User, System, Database

Exceptions: Invalid inputs, Database or server failure

Includes: Viewing Budget, Edit/deleting budget.

3.3.5 Use Case #5 (Reports & Analytics)



Author: Medini

Purpose: This use case allows the user to analyze financial data by viewing monthly summaries, generating category-wise expense reports, comparing income versus expenses, tracking savings, and exporting reports for external use.

Requirements Traceability: View Monthly Summary, Category-wise Expense Analysis, Income vs Expense Report, Savings Report, Export Report.

Priority: Medium

Preconditions: User must be logged in, and financial records (income and expenses) must be available in the system.

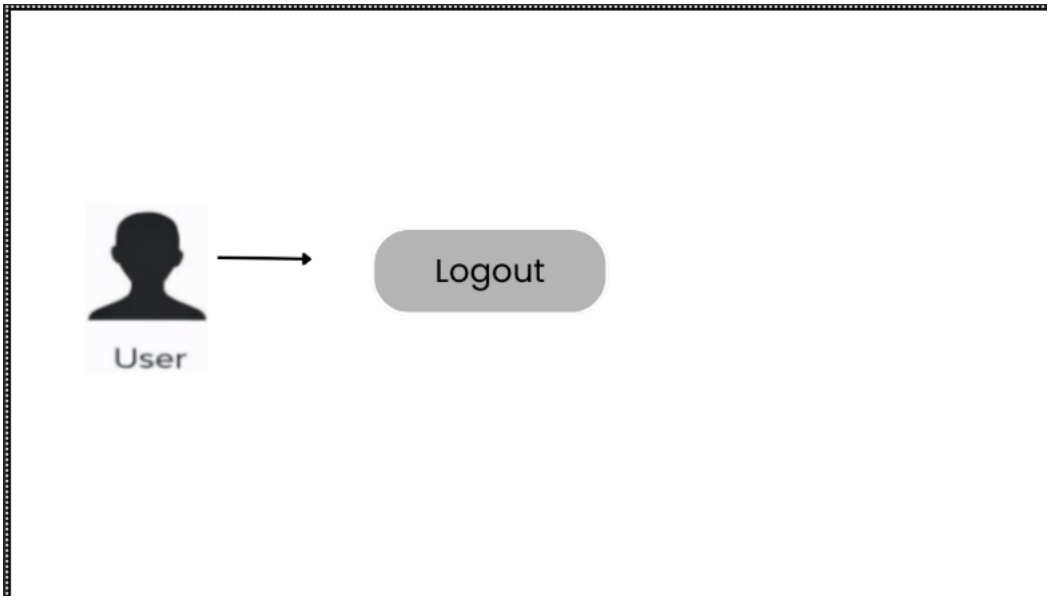
Post Conditions: Analytical reports are generated and displayed to the user, and exported reports are stored or downloaded successfully.

Actors: User, System, Database

Exceptions: No data available for the selected period, report generation failure, export failure, invalid date range, or database/storage error.

Includes: View Monthly Summary, Category-wise Expense Analysis, Income vs Expense Report, Savings Report.

3.3.6 Use Case #6(Logout)



Author: Anushka
Purpose: This use case displays the logout

Requirements Traceability: None

Priority: High

Preconditions: The user should have an account

Post Conditions: The user is logged out of their account, and the login page is displayed.

The user needs to log back in to come back.

Actors: Users of the website

Exceptions: None

Includes: None

4 Other Non-functional Requirements

4.1 Performance Requirements

- ***Response Time****: The system shall load the user dashboard and display current budget status within 2 seconds of a successful login under normal load conditions.*
- ***Transaction Processing****: Adding a new income or expense record shall take no longer than 1 second to process and reflect in the database.*
- ***Report Generation****: Generating monthly financial summaries or graphical visualizations for up to one year of data shall take no longer than 3 seconds.*
- ***Concurrency****: The system shall support at least 50 concurrent users without performance degradation, ensuring stability during demonstration or testing phases.*
- ***Database Latency****: All databases read/write operations must complete within 500 milliseconds to ensure a smooth user interface experience.*

4.2 Safety and Security Requirements

- ***Data Privacy****: User financial data is strictly private. Users shall not be able to access or view the data of other users.*
- ***Authentication Security****: Passwords shall not be stored in plain text. The system must use a strong hashing algorithm to salt and hash passwords before storage in the database.*
- ***Input Validation****: To prevent SQL Injection and Cross-Site Scripting (XSS) attacks, all user inputs (category names, amounts, descriptions) must be sanitized and validated on the server side before processing.*
- ***Error Handling****: System error messages displayed to the user shall not reveal sensitive system details (e.g., database stack traces) that could be exploited by attackers.*

4.3 Software Quality Attributes

4.3.1 Usability

Goal: The system should be intuitive enough for a new user to record a transaction without consulting a manual.
Implementation: We will adhere to the "Three-Click Rule," ensuring that users can access any major feature (Add Expense, View Report, Check Budget) within three clicks from the dashboard. Visual cues (Red for Expenses, Green for Income) will be used consistently to aid rapid comprehension.

4.3.2 Maintainability

Goal: The codebase should be easy for future developers (or graders) to understand and extend.
Implementation: The system will be developed using a combination of **Layered Architecture + MVC (Model–View–Controller)** architectural pattern to separate business logic from the user interface. Code will adhere to standard naming

conventions, and all complex functions will include documentation comments explaining their logic.

4.3.3 Reliability

Goal: The system must ensure data accuracy, which is critical for financial applications.
Implementation: The database will enforce ACID (Atomicity, Consistency, Isolation, Durability) properties to ensure that a transaction is either fully completed or fully rolled back in case of a system crash, preventing partial data corruption

5 Other Requirements

5.1 Database Requirements

The system shall use a Relational Database Management System (RDBMS) (such as MySQL). This is required to maintain the structured relationships between Users, Transactions, Categories, and Budgets, and to support complex SQL queries needed for generating analytical reports.

5.2 Legal and Compliance Disclaimer

Data Ownership: Users shall retain full ownership of their data and must have the ability to "Export" their data or "Delete Account," which effectively wipes their information from the server.

5.3 Technical Constraints

***Browser Compatibility**: The web application must be fully functional on the latest versions of major web browsers (Chrome, Firefox, Edge) without requiring proprietary plugins.*

***Currency Format**: The system shall store all monetary values using a suitable decimal data type rather than floating-point types to avoid rounding errors common in financial software.*

***Input Validation**: To prevent SQL Injection and Cross-Site Scripting (XSS) attacks, all user inputs (category names, amounts, descriptions) must be sanitized and validated on the server side before processing.*

Appendix A – Data Dictionary

A.1 User Data

Data Item	Description	Type	Operations / Usage
User ID	Unique identifier for each user	Integer	Generated during registration, used for authentication and data linkage
Username	Name chosen by the user	String	Create, Read, Update
Email	Registered Gmail address of the user	String	Used for login, OTP verification, notifications
Password Hash	Encrypted user password	String	Stored securely, used for authentication
Profile Photo	User profile image	File/Image	Upload, Update
Account Status	Status of user account (Active/Inactive)	Enum	Access control

A.2 Transaction Data

Data Item	Description	Type	Operations / Usage

Transaction ID	Unique identifier for each transaction	Integer	Auto-generated
Transaction Type	Type of transaction (Income / Expense)	Enum	Selected during transaction entry
Amount	Monetary value of the transaction	Decimal	Add, Update
Date	Date of the transaction	Date	Used for reports and filtering
Category ID	Linked category of the transaction	Integer	Foreign key reference
Note	Optional description of the transaction	String	User-provided clarification
User ID	Owner of the transaction	Integer	Data ownership and isolation

A.3 Category Data

Data Item	Description	Type	Operations / Usage
Category ID	Unique category identifier	Integer	Auto-generated
Category Name	Name of the category (Food, Travel, etc.)	String	Create, Read, Update, Delete
Category Type	Expense or Income category	Enum	Classification
Is Default	Indicates system-defined or user-defined category	Boolean	Constraint enforcement
User ID	Owner of custom category	Integer	Access control

A.4 Budget Data

Data Item	Description	Type	Operations / Usage
Budget ID	Unique identifier for each budget	Integer	Auto-generated
Month	Budget month	Date (MM-YYYY)	Budget allocation
Category ID	Category assigned to budget	Integer	Foreign key
Budget Amount	Allocated budget amount	Decimal	Create, Update
Spent Amount	Total expense incurred	Decimal	Auto-calculated
Remaining Amount	Budget Amount − Spent Amount	Decimal	Display, Alerts
User ID	Budget owner	Integer	Access control

A.5 Analytics & Report Data

Data Item	Description	Type	Operations / Usage
Total Income	Total income for selected period	Decimal	Dashboard summary
Total Expense	Total expense for selected period	Decimal	Dashboard summary
Savings	Income − Expense	Decimal	Financial analysis
Daily Expense	Total expense per day	Decimal	Line graph (monthly view)
Monthly Expense	Total expense per month	Decimal	Line graph (yearly view)
Category Expense	Expense per category	Decimal	Pie chart visualization

A.6 Alert Data

Data Item	Description	Type	Operations / Usage
Alert ID	Unique identifier for alert	Integer	Auto-generated
Alert Type	Budget overrun / Low savings	Enum	Trigger condition
Alert Message	Notification message	String	Display to user
Trigger Value	Threshold value for alert	Decimal	Rule-based
User ID	Alert recipient	Integer	Access control

A.7 System States

State	Description
Logged Out	User is not authenticated
Logged In	User is authenticated and active
Budget Exceeded	Expense exceeds allocated budget
Warning Level	Expense nearing budget limit
Data Updated	Transaction or budget successfully modified

Appendix B - Group Log

Date	Summary of the meeting
January 6	Decided Team Name and Potential Ideas for Project
January 9	Finalized the Project Idea
January 12	Started working on the SRS Document, and started discussing user and system requirements and divided work amongst team members
January 16	Discussed the use case diagrams
January 18	Met with our project TA Ujjwal Kajal and discussed our ideas
January 20	Has discussed doubts regarding SRS, worked together.
January 23	Finalized SRS and completed work.