## PROJECT REPORT of APC (Spring-boot)

**On**

FINANCE MANAGEMENT SYSTEM

Project ID : APC-2015

**COMPUTER SCIENCE AND ENGINEERING**

## B.E. Batch 2023



**Submitted By:**

**Under the Guidance of**

**Mr M. Imran Ravneet Kaur- 2310991188 Priyanshu- 2310991183 Rishu-2310991350**

**Kunal Bhatia- 2310990638**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

CHITKARA UNIVERSITY

**INDEX**

**Page No.**

1. **Abstract/Keywords 3**
2. **Introduction to the project 4**
   1. **Background**
   2. **Problem Statement**
3. **Software and Hardware Requirement Specification 6**
   1. **Methods**
   2. **Programming/Working Environment**
   3. **Requirements to run the application**
4. [Database Analyzing, design and implementation](#dy6vkm) 10
5. [Key Features and GUI](#bookmark) 13
6. [Code-Implementation and web overview](#bookmark1) 15
7. Limitations 21
8. [Conclusion](#s8eyo1) 23
9. [Future Scope](#dp8vu) 24

# 1. Abstract

The Finance Management System (FMS) is a **web-based application** designed to simplify and optimize the process of financial tracking, management, and analysis for both individuals and organizations. In today’s fast-paced and data-driven world, financial management has become increasingly complex, with people relying heavily on multiple income streams, diverse expense categories, and long-term investment planning. Managing these finances manually through registers or traditional spreadsheets is not only inefficient but also highly error-prone and lacks advanced insights. To address these limitations, the Finance Management System provides a **centralized, secure, and automated platform** where users can efficiently record their **income, expenses, savings, and investments** while also gaining meaningful insights through real-time analytics and visualization tools.

The system is developed using **modern web technologies and frameworks** to ensure scalability, usability, and security. The backend is powered by **Spring Boot**, which manages business logic and RESTful APIs, while the frontend uses **Thymeleaf templates** integrated with **HTML5, CSS3, and Bootstrap** to create a responsive and user-friendly interface. Data is persistently stored in a **MySQL database**, with the integration of **Hibernate JPA** for seamless Object-Relational Mapping (ORM). The security of user data is ensured using **Spring Security**, providing features such as authentication, authorization, and role-based access control for different types of users, such as administrators and standard users. The application is deployed on **Apache Tomcat** or can be containerized using **Docker** for scalable deployment in enterprise environments.

One of the most significant advantages of FMS is its ability to generate **dynamic reports and visualizations**. Unlike manual methods, where data analysis is limited to basic calculations, this system enables users to generate **daily, weekly, or monthly financial reports**, which can be exported in **PDF or Excel** formats. It also incorporates **charts and graphs** for visual representation of financial health, allowing users to instantly recognize spending patterns, track savings progress, and compare budgets against actual expenses. Such functionality not only enhances decision-making but also motivates users toward better financial discipline and planning.

The Finance Management System also addresses the **major drawbacks of traditional financial management practices**. These include:

* **Time consumption**, since manual entry and categorization of data require repetitive effort.
* **Error-prone calculations**, as spreadsheets or registers often lead to mistakes in summations, averages, or categorizations.
* **Unorganized reporting**, where the lack of structured data leads to confusion and inefficiency.
* **Data loss risks**, since files stored locally on systems are vulnerable to corruption, accidental deletion, or system crashes.
* **Lack of insights**, as manual methods do not provide predictive or visual analytics.

By overcoming these challenges, the FMS emerges as a **reliable and future-ready solution**. The system is **modular and extensible**, meaning it can be enhanced in the future with advanced features such as **AI-powered financial forecasting**, **multi-currency support**, **payment gateway integration**, and **cloud deployment** on platforms like AWS or Azure. Such scalability ensures that the system is not only suitable for personal use but can also be extended to serve the needs of startups, enterprises, or financial institutions.

# Introduction to the Project

### Background

In the modern era, financial management has become an integral part of both individual and organizational life. Every individual deals with multiple financial transactions on a daily basis, ranging from basic expenses such as groceries, bills, and rent to complex financial activities like investments, savings, and loan repayments. Similarly, organizations manage vast amounts of financial data involving employee salaries, vendor payments, operational costs, and revenue tracking. Managing this wide variety of financial activities using traditional methods such as manual bookkeeping or spreadsheets is often highly inefficient and unreliable

Spreadsheets, while somewhat useful for storing numbers, lack features like **real-time analytics, categorization, error checking, and secure storage**. As financial data grows in volume and complexity, spreadsheets become increasingly difficult to maintain, prone to duplication, and vulnerable to accidental errors or data loss. Moreover, manual financial records lack the ability to provide insights into spending patterns, long-term savings potential, or comparative analysis between budgets and actual expenses. This leads to poor financial planning, unorganized reporting, and missed opportunities for optimization.

The Finance Management System (FMS) is designed to overcome these limitations by offering a **centralized, automated, and intelligent platform**. With FMS, users can record their income and expenses, categorize transactions, set budgets, and generate detailed reports all within a single application. Unlike traditional systems, it not only ensures **accuracy and efficiency** but also provides **data-driven insights** to support better decision-making.

To achieve these objectives, the project integrates modern and reliable technologies:

* **Spring Boot** serves as the backend framework, offering a powerful environment for building scalable and secure REST APIs and business logic.
* **Thymeleaf**, combined with HTML5, CSS3, and Bootstrap, ensures a **dynamic and responsive user interface** that is easy to navigate and user-friendly.
* **MySQL** acts as the persistent storage solution, allowing structured and secure storage of financial data.
* **Hibernate JPA** simplifies database communication by enabling object-relational mapping.
* **Spring Security** ensures that user data remains private, supporting authentication and authorization mechanisms.

By integrating these technologies, the Finance Management System is built to provide a **secure, accurate, user-friendly, and scalable platform** for managing finances. It reduces human error, saves time, and makes complex financial management tasks accessible to every user, from individuals to organizations.

### Problem Statement

Despite the availability of tools such as spreadsheets and manual registers, users continue to face significant challenges in managing their financial activities efficiently. These challenges can be summarized as follows:

* **Manual Data Entry Errors:**  
  Entering financial transactions manually increases the risk of mistakes such as incorrect figures, missing entries, or duplication of records, ultimately leading to inaccurate financial reports.
* **Lack of Reporting and Visualization:**  
  Traditional methods fail to provide comprehensive reports and visual insights into financial data. Users are unable to identify trends in spending or compare current financial health with past data.
* **Data Fragmentation:**  
  Without a centralized platform, financial data is often scattered across multiple files, registers, or applications, making it difficult to maintain consistency and accuracy.
* **Limited Insights for Decision-Making:**  
  Users lack intelligent tools to analyze their income, expenses, and savings patterns. As a result, financial decisions are based on guesswork rather than data-driven insights.
* **Risk of Data Loss:**  
  Local files and spreadsheets stored on personal computers are vulnerable to hardware failures, system crashes, or accidental deletion, which can result in permanent data loss.

### ****Problem to Solve****

The core problem is the **absence of a centralized, secure, and automated system** that can handle financial records efficiently while also providing meaningful insights. Users require a platform that can:

* Record and categorize financial transactions with accuracy.
* Provide real-time analytics and dynamic reports.
* Offer secure data storage with user authentication.
* Support budgeting and alert users when expenses exceed limits.
* Enhance decision-making through visualization of financial health.

The Finance Management System is specifically developed to address these issues. By offering a centralized web-based platform with features like income/expense tracking, budgeting, and real-time reporting, FMS ensures **efficient financial management, improved organization, and smarter decision-making** for individuals and organizations alike.

# Software and Hardware Requirement Specification

A robust software system requires well-defined methods, a proper working environment, and clear specifications for both hardware and software. The Finance Management System (FMS) is designed with modern technologies and proven software design principles to ensure **scalability, maintainability, and security**. The following subsections describe the methodology, programming environment, and requirements to successfully run the application.

### 3.1 Methods

The Finance Management System is developed using the **MVC (Model–View–Controller)** architectural pattern. This design pattern is widely used in enterprise applications because it separates concerns, enhances modularity, and ensures scalability.

1. **Model Layer (Hibernate JPA):**

* Responsible for data handling, storage, and retrieval operations.
* Maps Java classes to relational database tables using Hibernate ORM.
* Ensures that all income, expense, budget, and report data is persisted in the MySQL database with referential integrity.
* Supports CRUD (Create, Read, Update, Delete) operations seamlessly.

1. **View Layer (Thymeleaf + Bootstrap):**

* Provides the user interface of the system.
* Generates dynamic HTML pages by integrating backend data with **Thymeleaf templates**.
* Ensures responsive and user-friendly design through **Bootstrap, CSS3, and JavaScript**.
* Displays dashboards, reports, and visualizations in an intuitive manner.

1. **Controller Layer (Spring MVC):**

* Acts as the mediator between the Model and the View.
* Handles incoming **HTTP requests** and routes them to appropriate services.
* Manages user sessions, navigation, and response rendering.
* Implements APIs for adding expenses, viewing budgets, and generating reports.

1. **Service Layer (Business Logic):**

* Implements the core functionality of the system such as:
* Budget limit checks and alert generation.
* Expense and income categorization.
* Real-time calculation of savings and balances.
* Dynamic report generation (daily, weekly, monthly).
* Ensures that all operations follow business rules and validation checks before interacting with the database.

This layered methodology ensures that the application is **scalable, maintainable, and secure** while providing a clear separation of responsibilities.

### 3.2 Programming/Working Environment

The Finance Management System is developed using modern frameworks and tools to ensure high performance and flexibility.

1. **Frontend (User Interface):**

* **Thymeleaf** for rendering server-side dynamic content.
* **HTML5, CSS3, Bootstrap** for responsive and interactive design.
* JavaScript-based libraries like **Chart.js/D3.js** for data visualization (graphs, charts, and analytics).

1. **Backend (Business Logic):**

* **Spring Boot** to create REST APIs and manage services.
* **Spring MVC** to structure controllers and manage request-response cycles.
* **Service Layer** for implementing logic like expense calculations, budget alerts, and report generation.

1. **Database:**

* **MySQL** serves as the relational database for persistent storage.
* Ensures structured storage of user accounts, income, expenses, and reports.
* Supports queries for generating insights and exporting reports.

1. **ORM (Object-Relational Mapping):**

* **Hibernate (JPA)** automates mapping between Java objects and database tables.
* Reduces boilerplate SQL code and improves maintainability.

**5, Security:**

* **Spring Security** is used for authentication and authorization.
* Supports **JWT (JSON Web Token)** or session-based authentication.
* Provides role-based access (Admin/User).

1. **Build & Dependency Management:**

* **Maven** is used to manage dependencies, plugins, and build processes.
* Ensures modular and consistent project structure.

1. **Deployment:**

* Can be deployed on **Apache Tomcat** server.
* Containerized deployment supported via **Docker**, enabling easy scalability and cloud hosting.

1. **Visualization:**

* Advanced libraries like **Chart.js** and **D3.js** are integrated for interactive graphs.
* Financial KPIs (Key Performance Indicators) such as monthly savings rate, budget adherence, and category-wise expenditure are visualized.

This working environment ensures that the Finance Management System is **user-friendly, scalable, secure, and easy to deploy** in different environments (local, enterprise, or cloud).

### 3.3 Requirements to Run the Application

To run the Finance Management System efficiently, specific hardware and software requirements must be met.

### ****Hardware Requirements****

* **Processor:** Minimum **Dual-Core (Intel i3/AMD equivalent)**, recommended **Quad-Core** for smooth server operations.
* **RAM:** Minimum **4 GB**, recommended **8 GB** for handling concurrent users.
* **Storage:** At least **500 MB** for project files, excluding database storage. With larger datasets, storage requirements may increase.
* **Display:** A standard **1024x768 resolution** monitor or higher for proper UI rendering.
* **Network:** Internet connection for deployment, updates, and access via browsers.

### ****Software Requirements****

* **Operating System:** Windows 10/11, Linux (Ubuntu, Fedora), or macOS.
* **Java Development Kit (JDK):** Version **11 or higher** for compatibility with Spring Boot.
* **Database Server:** **MySQL Server 8.0+** for reliable database storage.
* **Build Tool:** **Maven 3.6+** for project dependency management.
* **Frameworks & Libraries:** Spring Boot, Hibernate, Thymeleaf, Chart.js.
* **Application Server:** **Apache Tomcat 9+** or Docker for containerized deployment.
* **Web Browser:** Google Chrome, Mozilla Firefox, or Microsoft Edge for accessing the web application.

# Database Analyzing, Design, and Implementation

The Finance Management System (FMS) relies on **MySQL** as the relational database management system (RDBMS) to ensure secure, structured, and efficient data storage. The database has been designed using **Entity-Relationship (ER) modeling** to clearly define entities, attributes, and relationships. This ensures data consistency, integrity, and scalability while supporting the system’s diverse functionalities such as income/expense tracking, budgeting, and reporting.

#### 4.1 Database Analysis

During the analysis phase, the primary objective was to identify the different types of data to be stored and the relationships among them. The following considerations were made:

* **Users**: Each user should have a unique profile for storing personal and financial data securely.
* **Transactions (Income & Expense)**: Users need to store multiple financial transactions, categorized for better tracking.
* **Budgets**: Users should be able to set spending limits for specific categories and time periods.
* **Categories**: Categorization (e.g., Salary, Food, Rent, Investments, Travel) helps organize transactions and generate meaningful reports.
* **Reports**: Automatically generated reports allow users to analyze financial health over time.

The analysis concluded that a **normalized database design (up to 3NF)** would be suitable to avoid redundancy and maintain accuracy.

#### 4.2 Entities and Relationships

The major entities and their relationships are as follows:

* **User Entity**: Stores login and profile details.
* **Income Entity**: Records all income transactions made by a user.
* **Expense Entity**: Records all expenses made by a user.
* **Budget Entity**: Defines budget constraints and links them with users.
* **Category Entity**: Provides classification of income and expenses.
* **Reports Entity**: Stores metadata of reports generated for a user.

**Relationships:**

1. One **User** → Many **Income** records
2. One **User** → Many **Expense** records
3. One **User** → Many **Budgets**
4. One **Category** → Many **Income/Expense** transactions
5. One **User** → Many **Reports**

This relational structure ensures that each transaction and report is traceable to a specific user and category.

#### 4.3 Database Schema Example

**User Table**

* user\_id (PK)
* username
* password
* role
* email

**Income Table**

* income\_id (PK)
* user\_id (FK → User)
* category\_id (FK → Category)
* amount
* date

**Expense Table**

* expense\_id (PK)
* user\_id (FK → User)
* category\_id (FK → Category)
* amount
* date

**Budget Table**

* budget\_id (PK)
* user\_id (FK → User)
* limit\_amount
* month
* status

**Category Table**

* category\_id (PK)
* name
* type (Income/Expense)

**Reports Table**

* report\_id (PK)
* user\_id (FK → User)
* report\_type (Monthly/Annual/Custom)
* generated\_date

#### 4.4 Implementation Details

* **Primary Keys (PK):** Ensure unique identification of records.
* **Foreign Keys (FK):** Maintain referential integrity between tables.
* **Indexes:** Added on frequently searched fields such as user\_id, category\_id, and date to speed up query execution.
* **Normalization:** Applied up to **Third Normal Form (3NF)** to remove redundancy and ensure consistency.
* **Transactions & Constraints:** MySQL transactions are used to maintain atomicity, while constraints such as NOT NULL, UNIQUE, and CHECK improve data accuracy.
* **Scalability:** The design supports future expansion, such as adding investment tracking, recurring transactions, or integration with external APIs.

# Key Features and GUI

The Finance Management System (FMS) is designed to be **feature-rich, secure, and user-friendly**. Its interface prioritizes simplicity and responsiveness, ensuring that both technical and non-technical users can seamlessly interact with the system. The integration of **Spring Boot** for backend logic, **Spring Security** for authentication, and **Thymeleaf + Bootstrap** for the frontend provides a cohesive, modern web experience.

#### 5.1 Key Features

1. **User Authentication & Authorization**

* Secure login, registration, and logout functionalities powered by **Spring Security**.
* Passwords are encrypted using hashing algorithms to ensure confidentiality.
* Role-based access control (e.g., Admin vs. Regular User) ensures data privacy and restricted access.

1. **Expense & Income Tracking**

* Users can add, update, delete, and view detailed financial transactions.
* Each transaction stores essential details such as **amount, date, category, and type (income/expense)**.
* Smart filters allow viewing transactions by **date range, category, or amount**.

1. **Categorization of Transactions**

* Every transaction is linked to a predefined or custom category (e.g., **Food, Rent, Travel, Salary, Investments**).
* Categories provide a structured approach for monitoring spending habits and income sources.

1. **Budget Management**

* Users can set monthly/weekly budgets per category or overall spending limit.
* The system provides **real-time alerts and notifications** when expenses exceed the allocated budget.
* Color-coded progress indicators visually show how close a user is to exceeding their budget.

1. **Dashboard (Real-time Financial Overview)**

* A personalized dashboard displays a **summary of income, expenses, and savings** at a glance.
* Graphs and charts provide visual representation of financial health.
* Quick stats highlight top spending categories and remaining budget.

1. **Report Generation**

* Automated reports generated dynamically using transaction history.
* Users can **export reports in PDF/Excel** for offline storage or sharing.
* Reports include both **tabular summaries** and **graphical insights** using Chart.js / D3.js.

1. **Admin Panel**

* An **admin-exclusive interface** for managing users, categories, and transactions.Admins can view system-wide analytics, monitor suspicious activity, and maintain overall database integrity.

Provides CRUD operations for categories to ensure users have the flexibility to classify financial data accurately.

#### 5.2 Graphical User Interface (GUI)

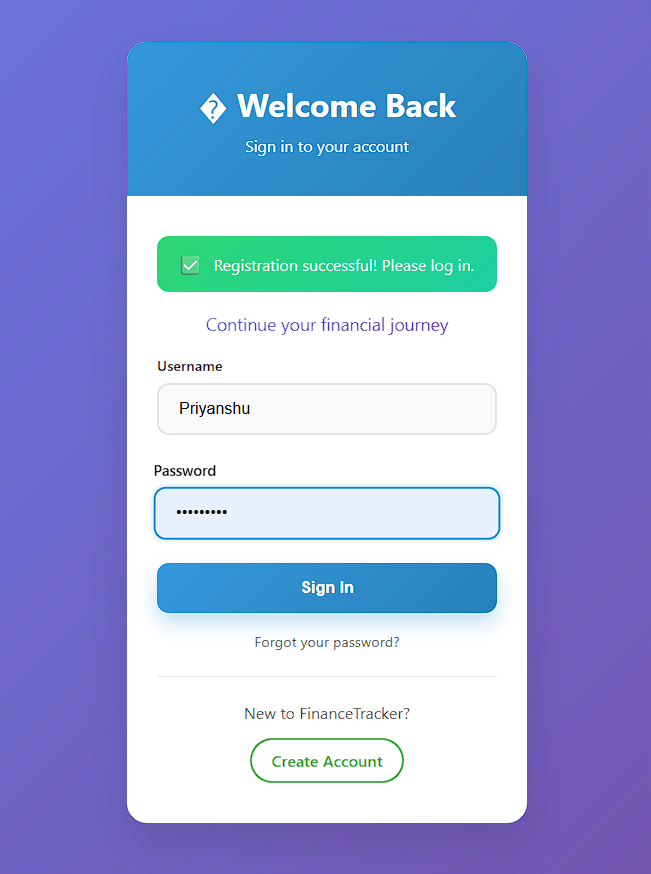
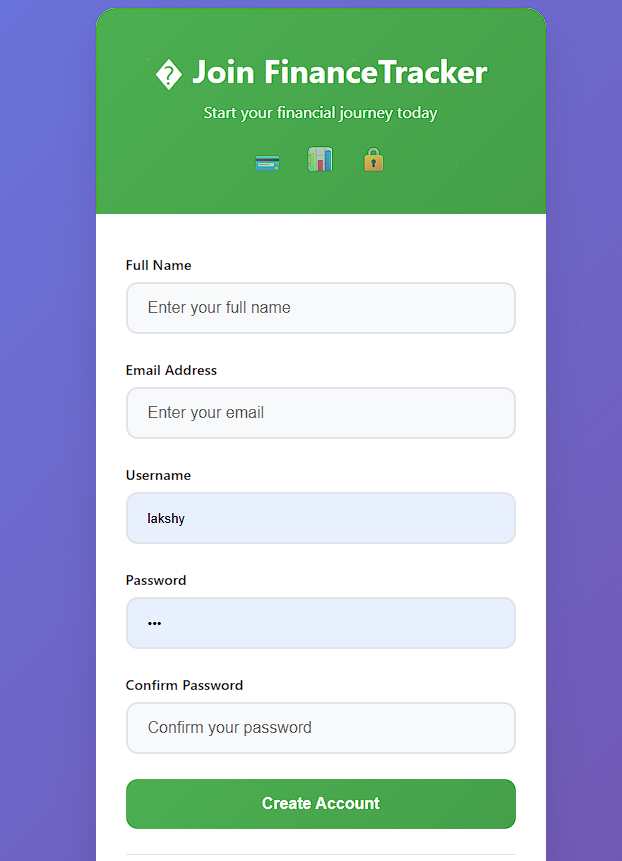
The **GUI** is implemented using **Thymeleaf templates** integrated with **Spring Boot**, styled with **Bootstrap** to ensure a **responsive, mobile-friendly layout**.

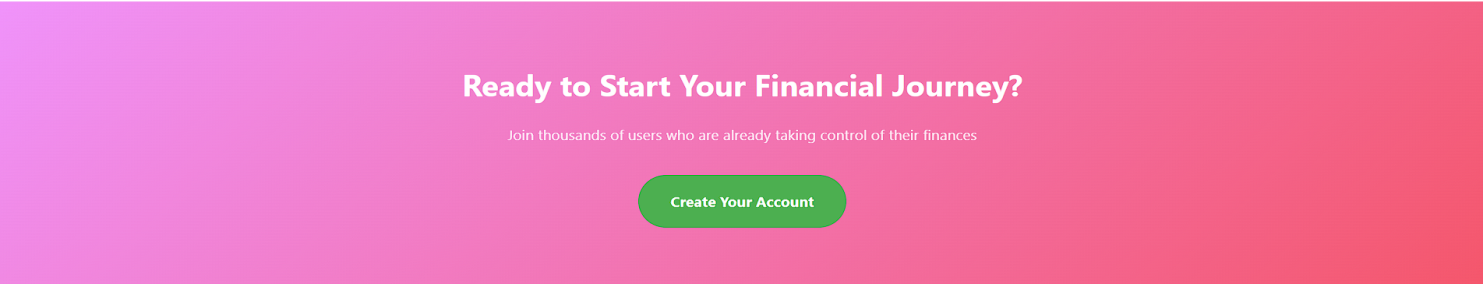
* **Login/Registration Screens**: Simple and secure forms with validations for input fields like email, password strength, and uniqueness of username.
* **Dashboard**: Displays financial summaries in the form of cards, pie charts, and bar graphs.
* **Transaction Management Pages**: User-friendly forms for adding/editing income and expenses with category dropdowns and date pickers.
* **Budget Management View**: Interactive progress bars showing budget usage with color indicators (green = safe, orange = nearing limit, red = exceeded).
* **Reports Page**: Downloadable reports with embedded visualizations (line charts, pie charts, bar charts).
* **Admin Panel**: Structured tables and management options for user accounts, transaction logs, and system categories.

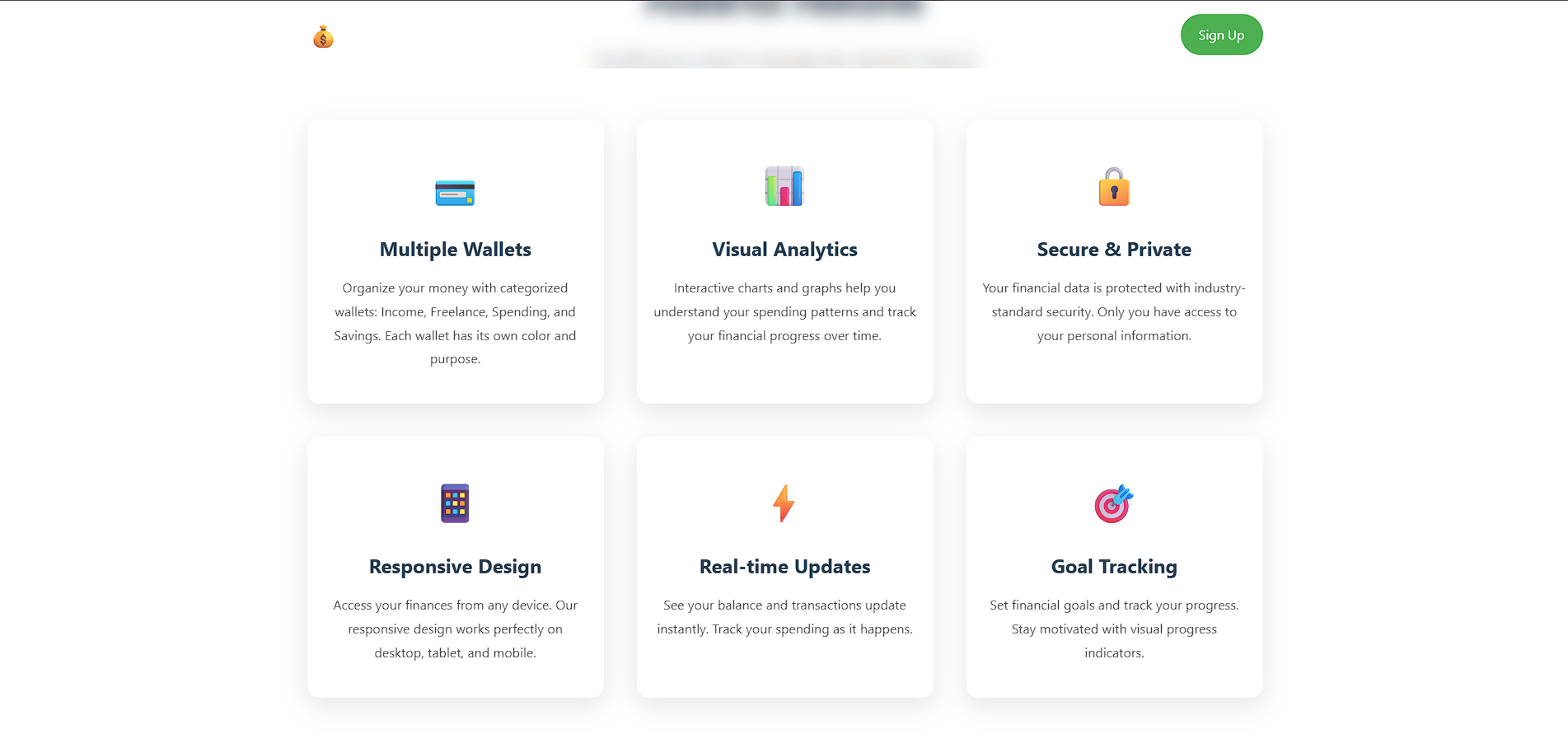
**The GUI prioritizes:**

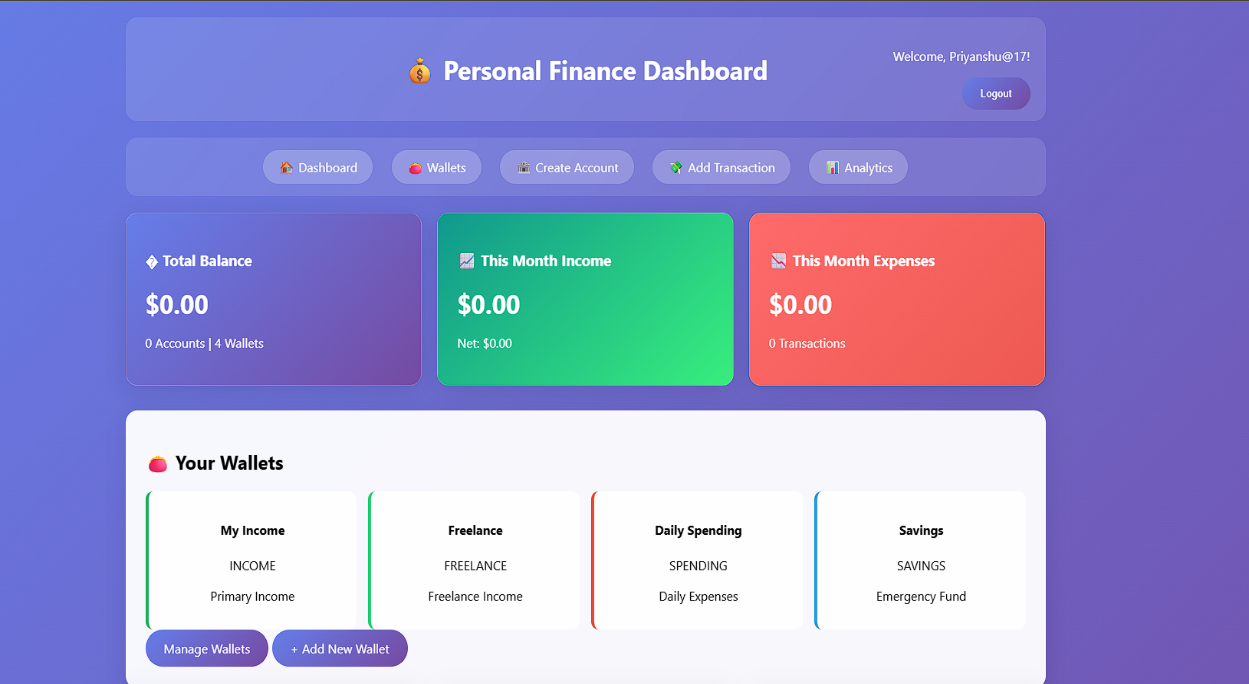
* **Responsiveness**: Automatically adjusts to various screen sizes (desktop, tablet, mobile).
* **Consistency**: Uniform color schemes, typography, and UI components.
* **Usability**: Clear navigation bar, tooltips, and error messages for enhanced user experience.
* **Interactivity**: Charts, dynamic tables, and alerts improve engagement and usability

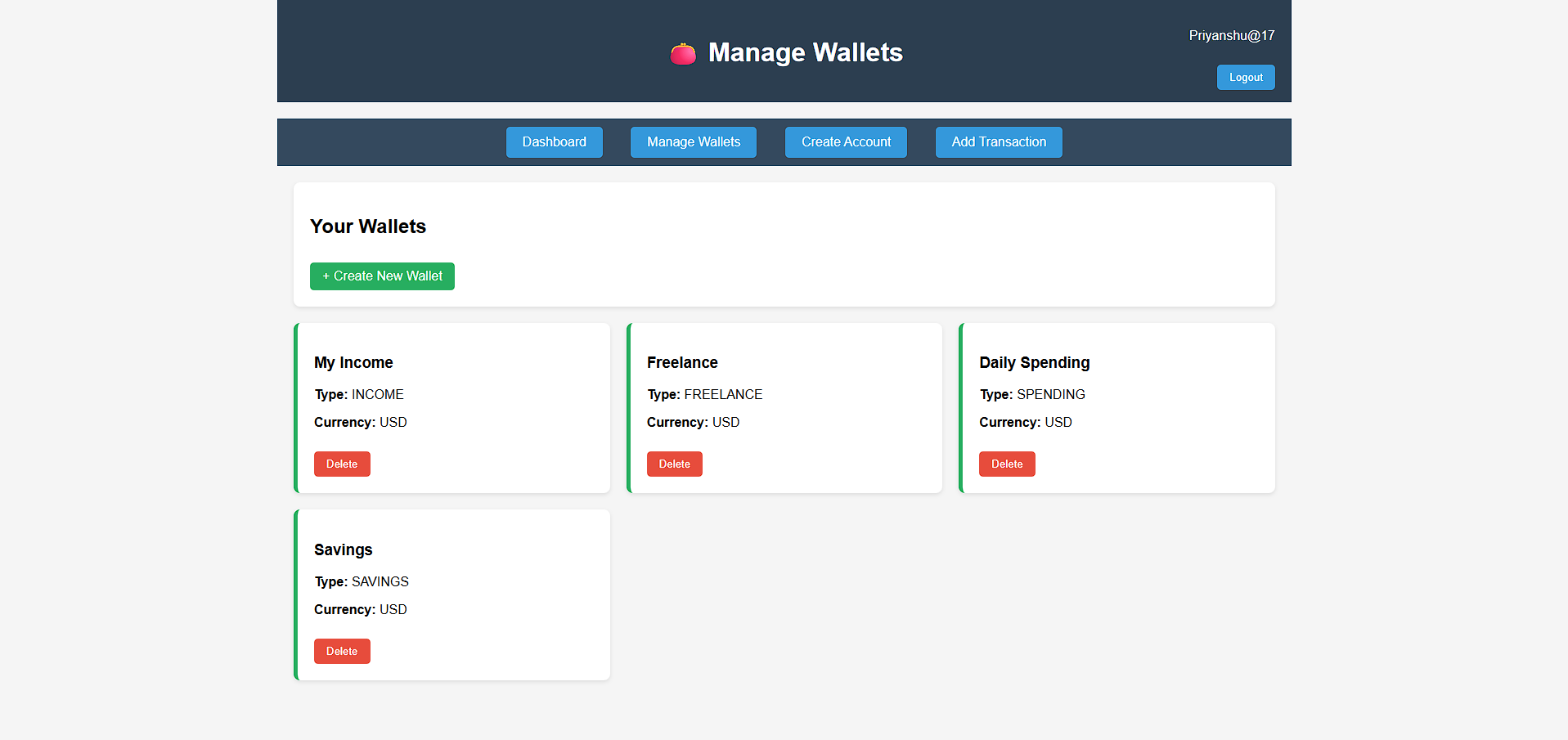
1. **Code Implementations and web overview (screenshots)**

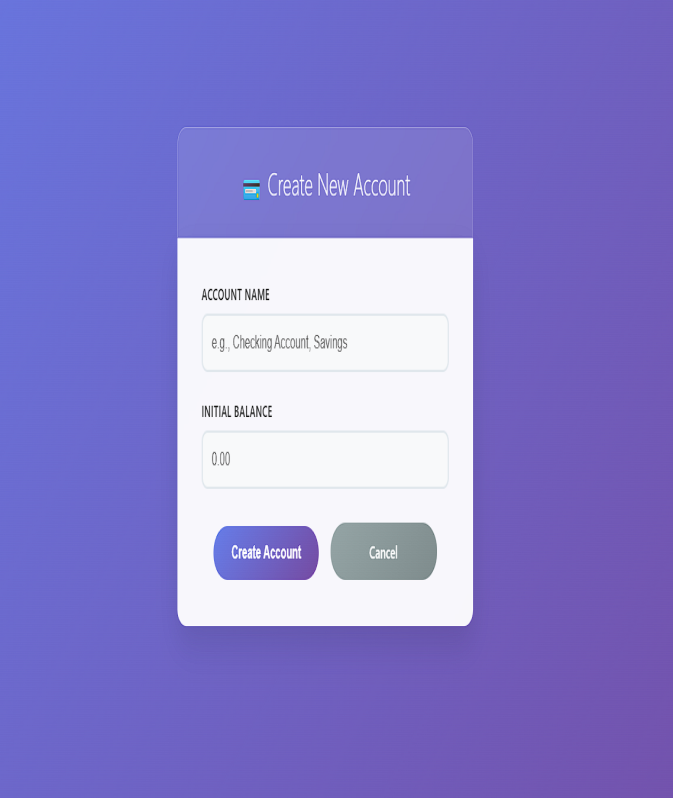
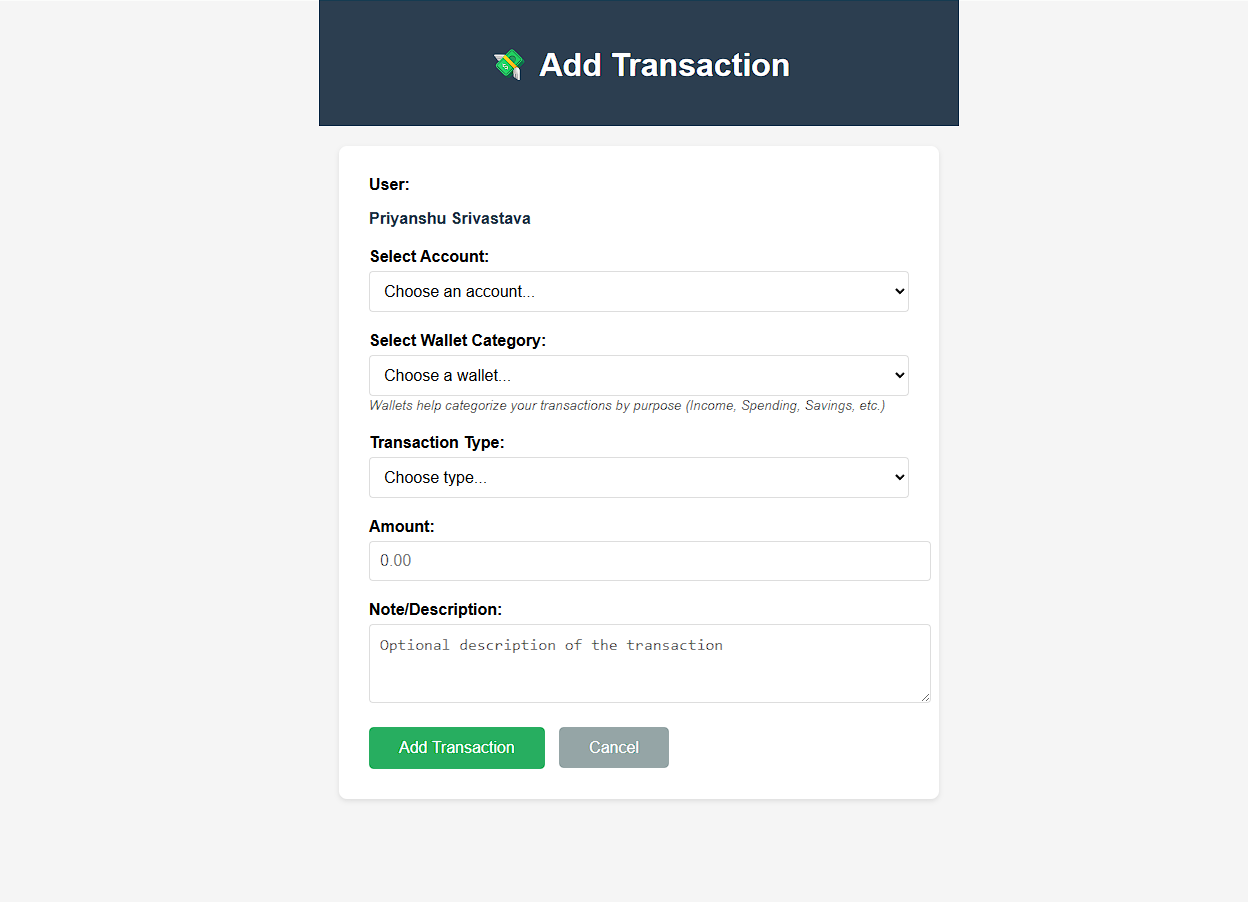


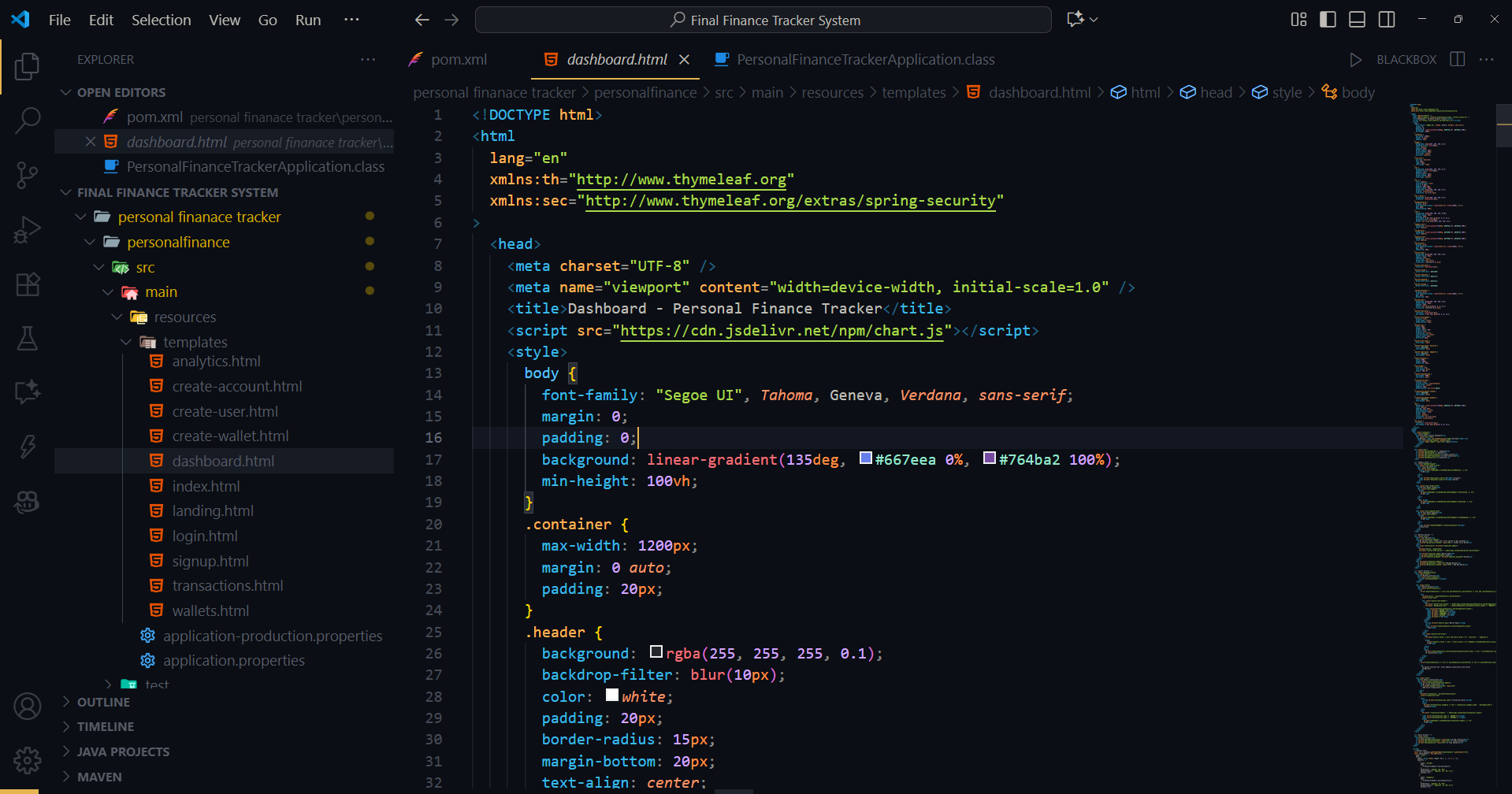


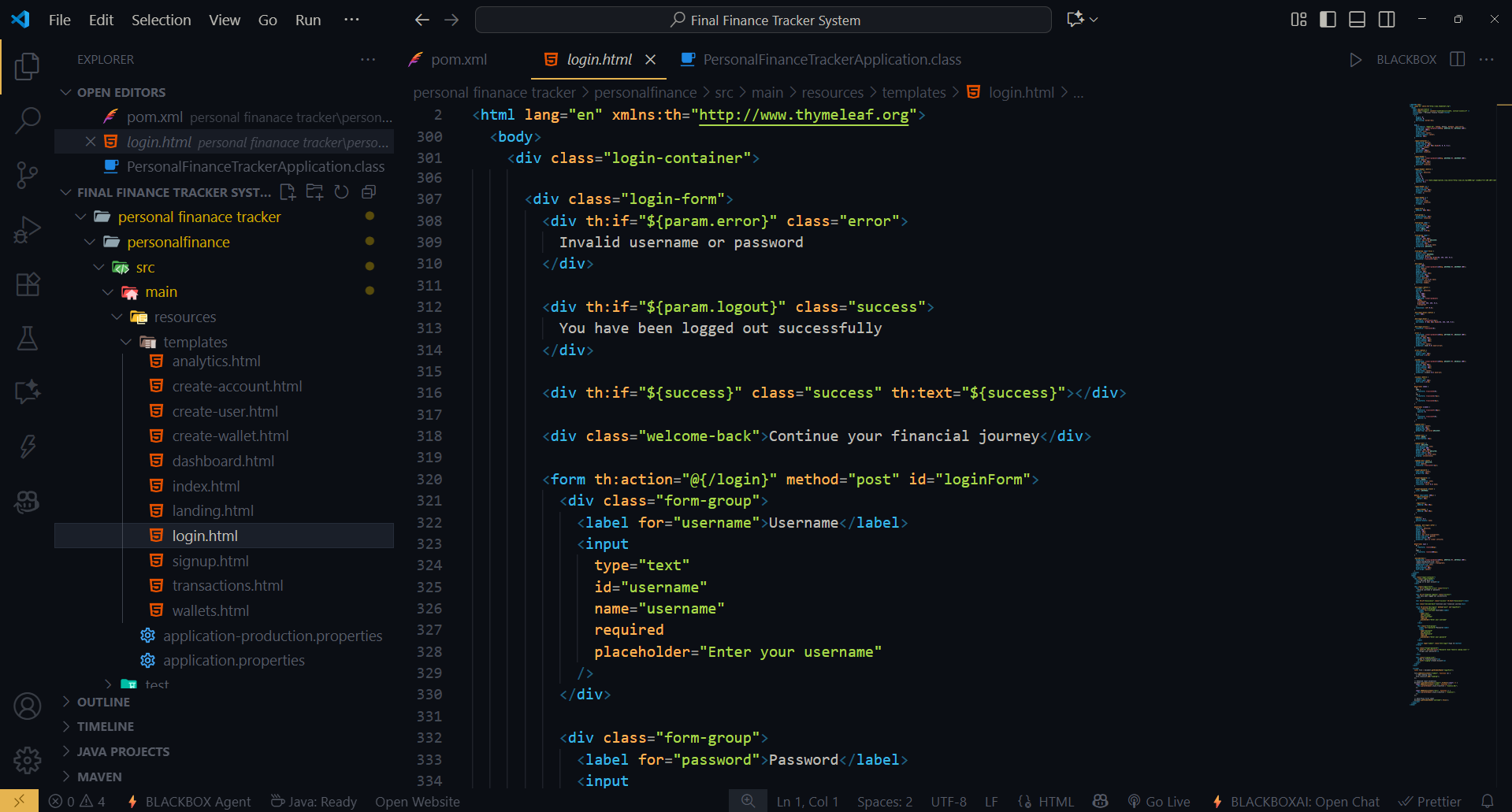


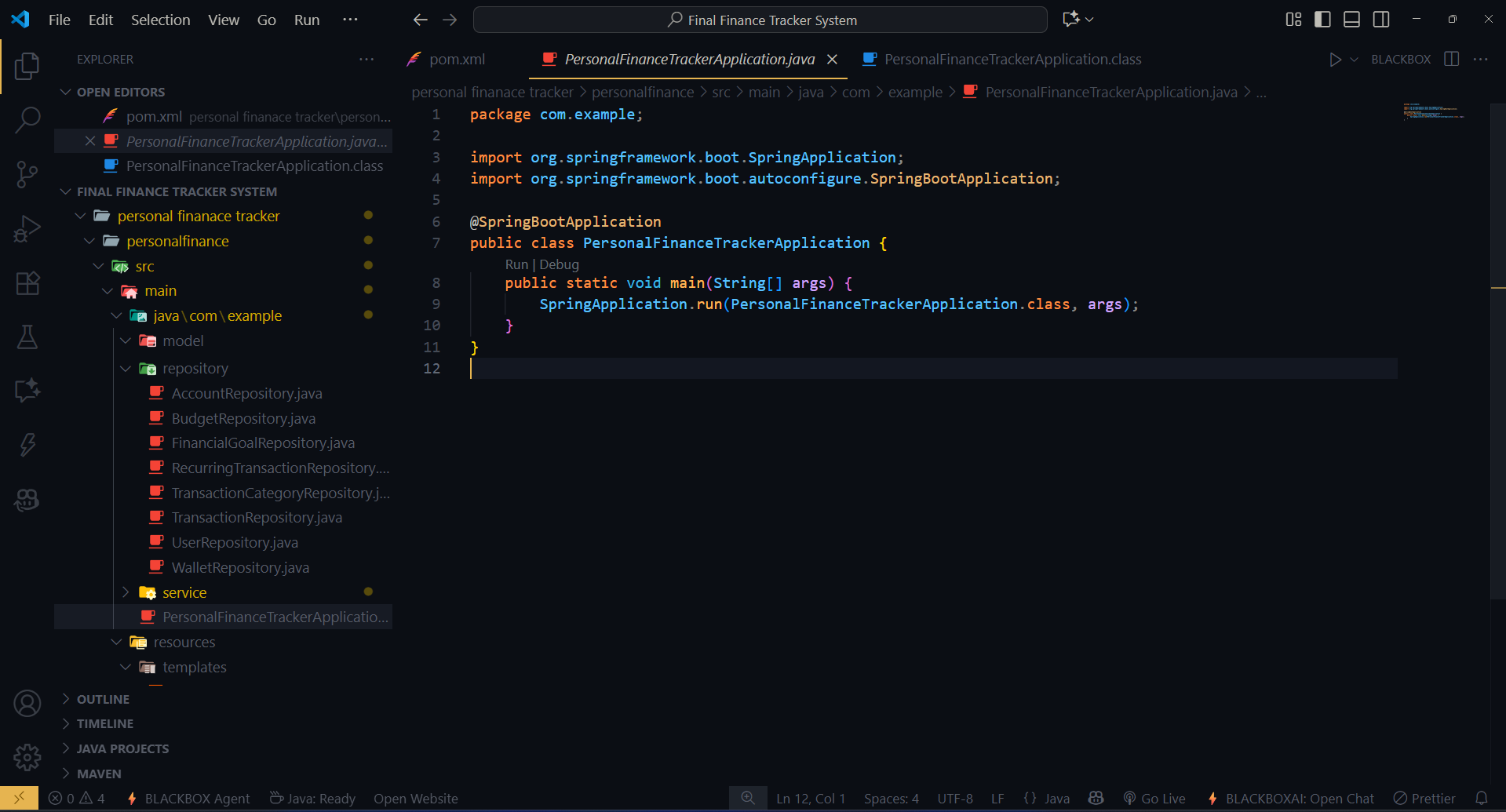


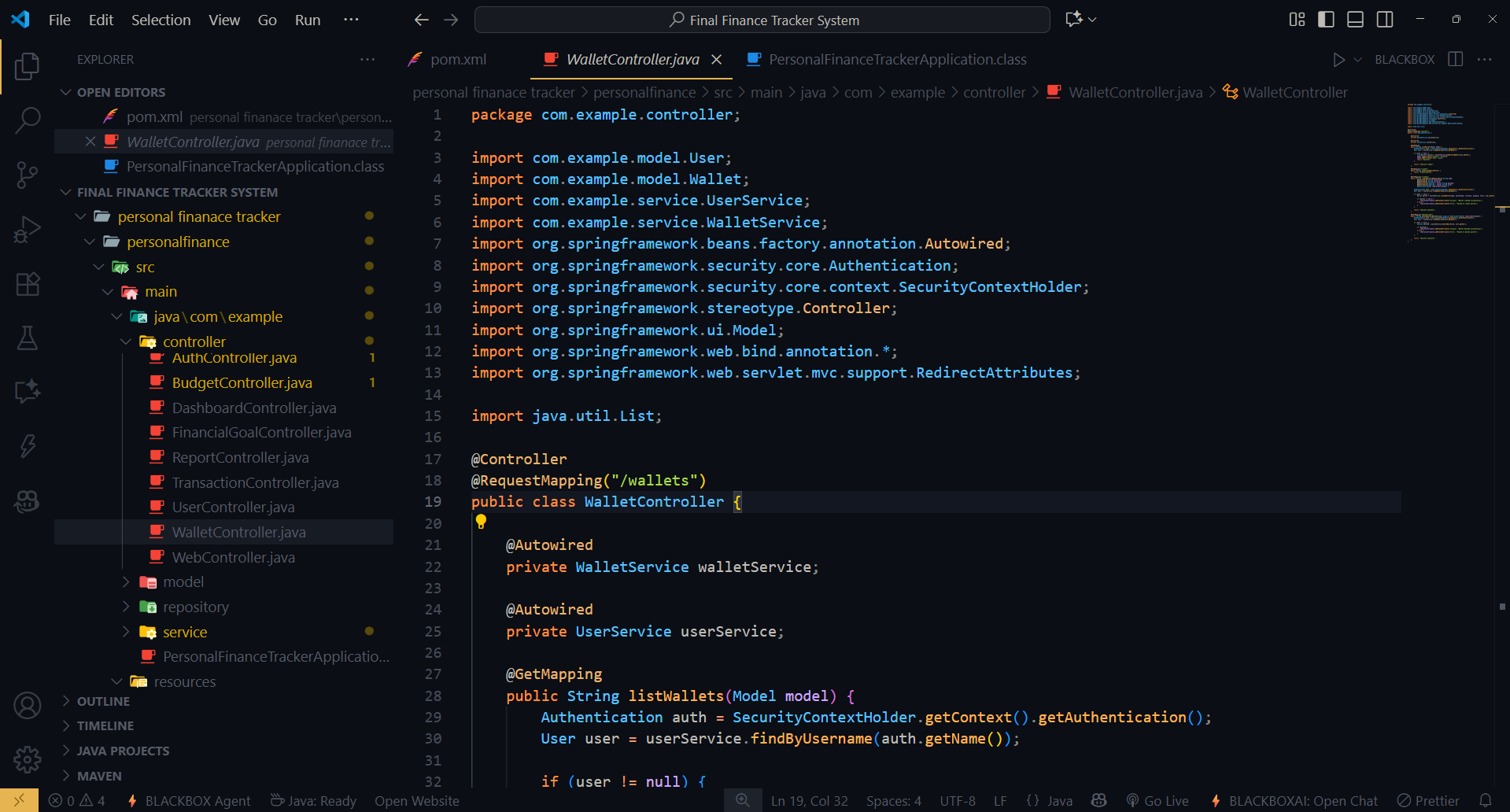












# L**imitations**

While the Finance Management System (FMS) provides a reliable and efficient solution for income-expense management, like any software system, it has certain limitations in its current version. These limitations highlight opportunities for future enhancements:

1. **Web-Based Interface Only**

* The system is currently limited to a browser-based interface, accessible only via desktops or laptops.
* A dedicated **mobile application (Android/iOS)** has not yet been developed, which limits accessibility for users who prefer on-the-go tracking.

1. **Single Currency Support**

* At present, the system is designed to work with a **single currency**.
* This restricts its usability for international users or organizations handling multiple currencies.
* Future versions could integrate APIs for **real-time currency conversion** and **multi-currency support**.

1. **Dependence on Internet and Server Uptime**

* Since it is a web-based system, continuous **internet connectivity and server availability** are mandatory.
* In case of network outages or server downtime, the system becomes inaccessible, affecting users who need uninterrupted access to financial data.

1. **Limited AI and Predictive Insights**

* While the system offers reporting and visualization features, **AI-powered financial predictions, forecasting, and smart recommendations** are not yet fully implemented.
* Future integration with **machine learning models** could help users with investment suggestions, spending trend predictions, and automated savings strategies.

1. **Scalability Constraints**

* The current deployment is suitable for individuals, small businesses, and medium-scale organizations.
* However, for **large-scale enterprise data handling**, performance may degrade without advanced cloud-based scaling solutions.
* Future improvements may include migration to **cloud-native architectures** like AWS, Azure, or Kubernetes for distributed scalability.

1. **No Blockchain or Advanced Security Modules**

* Although Spring Security ensures authentication and data privacy, advanced features like **blockchain for transaction immutability** or **multi-factor authentication** are not yet integrated.

By addressing these limitations, the Finance Management System can evolve into a comprehensive, globally scalable, and AI-driven financial solution.

# 8. Conclusion

The **Finance Management System (FMS)** provides a **robust, secure, and scalable platform** for managing personal and organizational finances. By integrating modern technologies such as **Spring Boot, Hibernate, Thymeleaf, MySQL, and Spring Security**, it successfully overcomes the challenges of traditional manual finance management.

Through its **real-time dashboards, income-expense tracking, budget management, and automated reporting capabilities**, the system helps users make informed decisions and gain better insights into their financial health. The inclusion of features such as **categorization, budget alerts, and exportable reports** ensures that users can maintain transparency, efficiency, and accuracy in financial record-keeping.

The project demonstrates the importance of adopting **automation and digitization** in financial workflows. By reducing manual errors, minimizing the risk of data loss, and enhancing decision-making with analytics, the Finance Management System contributes to more efficient financial planning.

Moreover, the system is designed with a **modular architecture**, making it adaptable for future technological advancements such as **mobile app development, AI-based predictions, cloud scaling, and blockchain-based transaction security**. This ensures the solution remains **future-ready and sustainable** in the long run.

In conclusion, the Finance Management System not only solves existing challenges in finance tracking but also lays the foundation for a smarter, more connected financial ecosystem. With continued improvements and integration of emerging technologies, it has the potential to become a **comprehensive digital finance assistant** for both individuals and organizations.

# 

# Future Scope

The Finance Management System (FMS) has been designed with a modular and scalable architecture, which makes it adaptable to future advancements in technology and user requirements. While the current version offers a strong foundation for income-expense management, several enhancements can be introduced in upcoming iterations to make the system more comprehensive and globally scalable.

1. **Mobile App Integration**

* A dedicated **Android and iOS mobile application** will significantly enhance accessibility and user convenience.
* Users will be able to record expenses, track income, and view dashboards on-the-go.
* Push notifications can be introduced to remind users about budget limits, bill payments, or financial insights.

1. **AI-Powered Insights**

* By integrating **machine learning and artificial intelligence algorithms**, the system can move beyond static reporting and offer **predictive analytics**.
* AI can analyze historical data to **forecast expenses, predict savings trends, and recommend investment opportunities**.
* This will transform the system from being just a tracking tool into a **smart financial advisor**.

1. **Payment Gateway Integration**

* Incorporating **payment gateways and bank synchronization APIs** will allow users to directly manage bill payments, track transactions in real-time, and automate recurring payments.
* This feature will simplify financial management by reducing the gap between tracking and actual financial operations.

1. **Multi-Currency Support**

* With globalization and international usage in mind, the system can integrate **currency conversion APIs** to provide **multi-currency support**.
* This feature will be beneficial for multinational organizations and individuals handling cross-border transactions.

1. **Cloud Deployment (AWS/Azure)**

* Hosting the system on **cloud platforms like AWS or Microsoft Azure** will improve **scalability, availability, and global reach**.
* Cloud deployment will enable **automatic backups, disaster recovery, and elastic scaling** for high user loads.
* This ensures reliability for enterprise-scale usage.

1. **Block chain Integration**

* Incorporating **blockchain technology** will enhance **security, transparency, and immutability** of financial transactions.
* Users will gain confidence in the authenticity of records, making the system suitable for high-value transactions and organizational accounting.

1. **Advanced Data Visualization**

* The current system already supports charts and graphs; however, future versions can integrate **interactive dashboards** using libraries like **Chart.js, D3.js, or Power BI connectors**.
* Features such as **real-time KPIs, drill-down analytics, and predictive charts** will allow users to visualize spending habits and financial health in greater detail.

****In summary**, the future development of the Finance Management System lies in transforming it into a **fully intelligent, global, and secure financial assistant**. With the integration of AI, cloud, blockchain, and advanced analytics, the system can evolve from a financial tracker into a **decision-support system**, empowering both individuals and organizations to achieve financial stability and growth.**