PERSONAL FINANCE TRACKER

PROJECT REPORT

Submitted by

Vaibhav Lohani (23BCS11415)

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

Chandigarh University

November - 2025

BONAFIDE CERTIFICATE

Certified that this project report **" Personal Finance Tracker "** is the bonafide work of " Vaibhav Lohani (23BCS11415)" who carried out the project work under my/our supervision.

Er. Pravindra Kumar Gole

SIGNATURE

INTERNAL EXAMINER **EXTERNAL EXAMINER**

Table of Contents

ABSTRACT

This project presents a **Personal Finance Tracker**, a web-based application designed to help users efficiently manage their income, expenses, and savings. The system enables users to record, categorize, and analyze their financial transactions through an intuitive interface. Built using **Spring Boot**, **PostgreSQL**, **HTML**, **CSS**, and **Java** the tracker ensures data persistence, security, and scalability.

The key motivation behind the project is the increasing need for personal financial management tools that empower users to make informed decisions about their spending habits. Many individuals struggle to monitor their daily expenses or understand their savings progress without structured digital tools. This project bridges that gap by providing a customizable and secure financial monitoring platform.

The system offers essential modules for user registration, authentication, transaction recording, budget planning, and visual analytics. It employs a **Model–View–Controller (MVC)** architecture, ensuring separation of concerns and smooth data flow. Users can add, edit, and delete transactions, and the dashboard presents expense summaries and charts for better financial insights.

Overall, the Personal Finance Tracker enhances individual financial awareness, promotes responsible spending, and demonstrates practical applications of modern backend frameworks like Spring Boot integrated with relational databases.

CHAPTER 1 INTRODUCTION

1.1 Client Identifiction

The primary users of the **Personal Finance Tracker** are individuals, families, and small-scale business owners who wish to manage and analyze their personal or organizational financial data efficiently. In the modern digital era, financial transactions have become more frequent and complex due to the rise of online banking, e-commerce, and multiple modes of digital payments such as UPI, debit/credit cards, and wallets. As a result, individuals often struggle to keep track of where their money goes and how much they save or invest each month.

This project is also developed with the **academic community** in mind — particularly students, educators, and software developers who aim to understand how real-world web applications are structured using **Spring Boot** as the backend framework and **PostgreSQL** as a robust relational database. The application is designed to demonstrate the integration of user authentication, database connectivity, and data visualization within a secure and maintainable full-stack environment.

The client needs addressed by this project include:

- Easy-to-use interfaces for recording income and expenses.
- Secure user authentication and data storage.
- Real-time graphical visualization of spending patterns.
- Ability to analyze financial health using categorized data.

By targeting both end-users and developers, the **Personal Finance Tracker** bridges the gap between usability and educational utility. It serves as both a **personal financial management solution** and a **practical example of enterprise-level software design** in a simplified form.

1.2 Identification of Problem

Financial mismanagement is a common issue among individuals, especially those who lack proper tools to monitor their income, expenditure, and savings. Many rely on manual methods such as spreadsheets, diaries, or mobile notes, which are:

- Time-consuming and prone to human errors.
- Inconsistent and lack analytical insights.
- Not easily accessible across devices.

While commercial finance management software exists, such as **Mint**, **YNAB**, and **PocketGuard**, these solutions often come with limitations:

- Paid subscriptions or hidden costs.
- Limited customization options.
- Privacy concerns due to cloud-based data storage.
- Complex interfaces unsuitable for beginners.

Therefore, the identified problem is the **lack of a free, customizable, and self-hosted personal finance management tool** that gives users full control over their financial data without compromising security or usability.

The **Personal Finance Tracker** project aims to address this problem by developing a lightweight yet powerful web application that:

- Records, categorizes, and analyzes financial transactions.
- Provides meaningful visual insights through graphical dashboards.
- Ensures complete data privacy with local database storage.
- Operates seamlessly in a browser-based environment.

1.3 Purpose

The specific purposes of this project include:

1. **Automate Financial Tracking:**
   Eliminate the need for manual entry and calculations by automating expense and income management using a centralized system.

2. **Enable Categorization and Insights:**
   Allow users to classify transactions into meaningful categories (e.g., Food, Rent, Travel, Salary) and visualize spending habits through dynamic charts.

3. **Ensure Data Security and Privacy:**
   Protect user data using **Spring Security** authentication mechanisms, encrypt sensitive credentials, and maintain database integrity using PostgreSQL.

4. **Provide Analytical Reports:**
   Generate summaries that highlight total income, total expenses, and savings over customizable time periods.

5. **Educate and Demonstrate Full-Stack Integration:**
   Serve as an academic example of a modern **Spring Boot MVC project** demonstrating CRUD operations, ORM with Hibernate, REST APIs, and responsive UI design.

1.4 Timeline

Week	Dates	Task Description	Status

1	June 9 – June 15	Requirements analysis, initial UI setup, BFS & DFS implementation	Completed
2	June 16 – June 23	Dijkstra's algorithm, Floyd-Warshall algorithm,animations, reset, metrics, bug fixes, report documentation	Completed

1.5 Organization of the Report

This This report has been organized into a structured and logical flow to ensure clarity, comprehensiveness, and academic rigor. The layout mirrors the standard university project report structure used for undergraduate engineering submissions.

- **Chapter 1: Introduction**
  Provides an overview of the project, identifies the client base, defines the problem statement, outlines objectives, and presents the project timeline.
- **Chapter 2: Literature Review / Background Study**
  Reviews existing financial management tools, highlights their advantages and drawbacks, and establishes the research gap that this project addresses.
- **Chapter 3: Design Flow and Methodology**
  Details the architectural design, system components, and implementation methodology used in the development process, including MVC architecture and technology stack.
- **Chapter 4: Results, Analysis, and Validation**
  Describes system outcomes, experimental validations, testing results, and performance evaluation. It also includes screenshots and sample outputs.
- **Chapter 5: Conclusion and Future Work**
  Summarizes the outcomes of the project, discusses challenges faced, and provides recommendations and potential enhancements for future versions.
- **References, Appendix, and User Manual**
  Includes bibliographic citations, additional materials like code structure, and a user manual for operating the application.

CHAPTER 2 LITERATURE REVIEW

2.1 Timeline of the reported problem

Pathfinding The need for **personal finance management systems** has evolved significantly over the past few decades. In the early 1990s, most individuals relied on **manual bookkeeping** or spreadsheet-based methods such as Microsoft Excel to manage their income and expenses. While functional, these methods were limited by the user's knowledge of formulas, manual entry errors, and lack of visual representation.

In the early 2000s, the introduction of software like **Microsoft Money** and **Quicken** marked the beginning of digital personal finance tracking. These applications automated expense recording and budgeting, but they were primarily desktop-based and lacked real-time synchronization or portability. The late 2000s brought about **cloud-based tools** such as **Mint**, **YNAB (You Need A Budget)**, and **PocketGuard**, which introduced data analytics and financial goal tracking but often required online access and user data sharing with third-party servers.

By the 2010s, as cybersecurity and data privacy concerns grew, users increasingly sought **self-hosted or open-source solutions** to maintain control over their financial information. With the evolution of backend technologies like **Spring Boot**, **Django**, and **Node.js**, it became feasible to create lightweight yet powerful applications that could securely handle financial data with relational databases like **PostgreSQL** and **MySQL**.

In 2020 and beyond, the global push for digitalization, especially after the pandemic, emphasized the importance of **personal financial literacy**. People began prioritizing tracking their daily expenses, understanding savings, and managing budgets digitally. This trend set the stage for developing modern, **secure, open-source, and customizable personal finance applications** that combine simplicity with data analytics capabilities — exactly what the **Personal Finance Tracker** aims to deliver.

Thus, the timeline of financial management evolution highlights the transition from **manual record-keeping → desktop applications → cloud-based tools → open-source, privacy-focused, full-stack solutions**.

2.2 Proposed solutions

The Various solutions have been proposed and developed to address financial management problems, each contributing unique functionalities. However, they differ in terms of platform, data ownership, accessibility, and security.

Limitations of Existing Systems

Despite their usability, most of the above systems share critical drawbacks:

- **Dependence on Internet Connectivity:** Users cannot access data offline.
- **Cloud Storage Risks:** User data may be shared or breached.
- **Limited Free Access:** Most systems are subscription-based.
- **Lack of Customization:** Users cannot modify or extend functionality.
- **Complexity:** Steep learning curve for non-technical users.

Proposed Solution — Personal Finance Tracker

The proposed system eliminates these limitations through a **self-hosted, secure, and customizable full-stack web application**.

The **Personal Finance Tracker** is developed using:

- **Spring Boot** (Backend Framework): To provide RESTful APIs, modular design, and security through Spring Security.
- **PostgreSQL** (Database): For structured storage and relational data management.
- **HTML/CSS/Bootstrap** (Frontend): For responsive and interactive user interface.

2.3 Bibliometric analysis

Tool / Platform	Technology Stack	Target Users	Features Offered	Deployment
Mint	Web & Mobile (Cloud)	General Users	Bank Integration, Budget Tracking	Cloud
YNAB	Web & Mobile	Budget-Oriented Users	Predictive Budgeting, Cloud Sync	Cloud
PocketGuard	Android/iOS	General Users	Expense Overview, Goal Setting	Cloud
Google Sheets	Web Spreadsheet	All Users	Manual Data Entry, Graphs	Web
Firefly III	PHP, MySQL	Developers, Finance Enthusiasts	Open-Source, API Support	Self-hosted
Personal Finance Tracker (This Project)	**Spring Boot + PostgreSQL + Thymeleaf**	**Individuals, Students, SMEs**	**Expense Categorization, Secure Login, Visual Reports**	**Local / Server**

2.4 Review Summary

After studying the available literature and analyzing the existing solutions, it becomes clear that there is a significant **gap in privacy-centric, self-hosted finance management tools** that balance simplicity, flexibility, and analytical power.
While commercial products dominate the market with attractive interfaces, their **data dependency on external servers** and **subscription-based models** restrict accessibility. Moreover, open-source tools like Firefly III, though powerful, are not beginner-friendly and require complex setups.
The **Personal Finance Tracker** overcomes these issues by offering:

- A **secure local deployment** using Spring Boot and PostgreSQL.
- A **modular architecture** for easy feature expansion.
- An **intuitive interface** for students and general users.
- Free usage with full data privacy.

This project bridges the educational and practical aspects of software engineering by creating an application that demonstrates real-world backend–frontend integration while solving an everyday problem — **personal finance tracking**.

2.5 Problem Defination

Despite the variety of existing solutions, many users still lack control over their financial data or cannot access tools that meet their exact needs.
The **core problem** identified is:

"To design and develop a secure, user-friendly, and open-source web application that enables users to manage and analyze their financial transactions locally without relying on external servers."

What is to be done:

- Develop a web-based personal finance management application.
- Implement authentication for multiple users using Spring Security.
- Enable CRUD (Create, Read, Update, Delete) operations for income and expenses.
- Categorize transactions (e.g., Food, Rent, Salary, Utilities).
- Generate real-time analytics using graphical representations.
- Ensure full data persistence through PostgreSQL.

How it is to be done:

- Use **Spring Boot** to build backend REST APIs and handle business logic.
- Implement **Thymeleaf** templates for dynamic UI rendering.

- Store user data securely using **PostgreSQL** with JPA/Hibernate ORM.
- Apply **MVC architecture** for separation of concerns.
- Validate and test all modules thoroughly before deployment.

2.6 Goals / Objectives

To ensure the project stays on track and measurable, the following **SMART objectives** were set:

Goal ID	Objective Statement
G1	Design and develop a web application for personal finance management using Spring Boot.
G2	Implement CRUD functionality for income and expense records with PostgreSQL persistence.
G3	Enable secure authentication and session management using Spring Security.
G4	Visualize categorized data using interactive charts for monthly and yearly analysis.
G5	Test, validate, and document the entire application with user and technical manuals.

Each of these goals is **specific**, **measurable**, **achievable**, **realistic**, and **time-bound**, ensuring structured project execution and evaluation.

CHAPTER 3 DESIGN FLOW / PROCESS

3.1 Evaluation & Selection of Specifications / Features

Designing the **Personal Finance Tracker** required a careful evaluation of user requirements, technological feasibility, and system scalability. The project aimed to create a platform that balances functionality, performance, and simplicity while ensuring complete data security.

The evaluation began with an analysis of the **core features** that a finance tracking application must include, followed by the selection of the most appropriate technologies and design principles.

Targeted Core Features

Based on requirement analysis, the following features were identified as crucial for the project:

1. **User Authentication System:**
Secure login and registration using Spring Security to protect user data and provide personalized access.

2. **Transaction Management:**
CRUD (Create, Read, Update, Delete) operations for managing income and expenses with attributes like date, category, amount, and description.

3. **Categorization of Transactions:**
Group transactions into categories such as Food, Rent, Utilities, Salary, Entertainment, and Others for easier data organization.

4. **Visualization Dashboard:**
Display graphical representations of user data using charts and summaries to provide quick insights into spending and savings trends.

5. **Search and Filter Options:**
Allow users to search and sort their financial records by date, category, or amount for efficient tracking.

6. **Database Connectivity:**
Use PostgreSQL to store and manage all user and transaction data reliably with proper indexing and relationships.

7. **Responsive UI:**
Develop an intuitive interface using Thymeleaf and Bootstrap that works seamlessly across desktop and mobile browsers.

8. **Error Handling and Validation:**
Include form validations, error alerts, and exception handling to ensure data integrity and prevent runtime issues.

9. **Security and Session Management:**
Implement Spring Security to authenticate users, manage sessions, and prevent unauthorized access.

10. **Scalable Architecture:**
Design a structure that allows easy integration of future features such as multi-user support, data export, and AI-based analytics.

3.2 Design Constraints

During the design and implementation phases, several constraints were considered:

Constraint Type	Description
Technical	Limited to technologies such as Spring Boot, Thymeleaf, PostgreSQL, and standard Java libraries. No external paid APIs or frameworks were used.
Economic	The entire project had to be developed using open-source and cost-free tools like PostgreSQL, Maven, and IntelliJ IDEA Community Edition.
Time-based	The project had to be completed within 4 weeks, restricting the inclusion of advanced modules like API integration or predictive analytics.
Security	Data encryption and authentication were mandatory to protect financial records from unauthorized access.
Scalability	The system must be extendable to support multi-user environments in future releases without major architectural changes.
Usability	The UI had to remain simple and intuitive, suitable for both technical and non-technical users.
Maintainability	The codebase should follow modular and object-oriented principles, ensuring easy debugging and feature enhancement.

3.3 Analysis and Feature Finalization Subject to Constraints

Following design evaluations and constraint analysis, certain features were **modified, postponed, or finalized** based on feasibility and educational value.

Features Removed or Deferred

1. **Multi-Currency Support:**
   Although valuable, it required additional conversion APIs and currency tables, which were beyond the scope of this version.
2. **Cloud Synchronization:**
   For privacy and simplicity, all data remains stored locally on PostgreSQL rather than in cloud environments.
3. **PDF Export of Reports:**
   Deferred to future versions due to time constraints.
4. **AI-Powered Spending Suggestions:**
   Considered as part of future scope but excluded from the initial implementation.

Modified Features

1. **Graph Visualization:**
   Instead of complex JavaScript libraries, lightweight charting through Thymeleaf-integrated chart libraries (like Chart.js) was implemented.
2. **Login System:**
   Simplified to include username, email, and password without multi-factor authentication, which can be added in future updates.
3. **Responsive Design:**
   Optimized for major screen sizes using Bootstrap grid layout rather than developing a separate mobile version.

Finalized Features

- Secure authentication using **Spring Security**
- PostgreSQL database with **JPA/Hibernate ORM**
- CRUD functionality for financial records
- Category-based expense tracking
- Interactive dashboard with visual reports
- Error validation and simple session management

The final system thus ensures both **educational value** and **practical application** within the academic timeline.

3.4 Design Flow

The **Personal Finance Tracker** follows a **Model–View–Controller (MVC)** architecture pattern to achieve separation of concerns, maintainability, and modularity.

System Design Approach

1. **Model (M):**
   Represents the core data of the application, including entities such as *User*, *Transaction*, and *Category*.
   - Each model class is mapped to a database table using JPA annotations.
   - Relationships (e.g., One-to-Many between User and Transactions) are handled through Hibernate ORM.
2. **View (V):**
   Managed by **Thymeleaf templates**, it presents data to users in a visually appealing manner.
   - HTML/CSS with Bootstrap ensures responsiveness.
   - Charts and graphs provide financial insights.
3. **Controller (C):**
   Handles all client requests and responses.
   - Spring Boot controllers manage routing, CRUD operations, and service-layer communication.
   - REST endpoints ensure modular and scalable communication between components.

CHAPTER 4 RESULTS ANALYSIS AND VALIDATION

4.1 Implementation of Solution

The The implementation of the **Personal Finance Tracker** was carried out in a systematic and modular manner, following the **Model–View–Controller (MVC)** architecture. This structure ensured that each part of the system—data management, business logic, and user interface—was developed and tested independently before final integration.

The project successfully delivers all the key functionalities envisioned during the planning phase, including secure user authentication, transaction management, categorization, and financial analytics visualization.

Core Functional Modules Implemented

1. **User Authentication Module**
   - Developed using **Spring Security**.
   - Enables registration and login functionalities.
   - Passwords are stored in encrypted form using hashing algorithms.
   - Ensures session control and logout support.
   - Protects unauthorized access to user-specific financial data.
2. **Transaction Management Module**
   - Supports CRUD operations (Create, Read, Update, Delete) for income and expense entries.
   - Each transaction is linked to a user via a foreign key relationship.
   - Validation checks are applied for mandatory fields such as amount, date, and category.
3. **Categorization and Analytics Module**
   - Allows users to classify transactions into pre-defined or custom categories.
   - Uses **Chart.js** for generating interactive visualizations (pie charts and bar graphs).
   - Displays monthly income, expenditure, and savings summaries dynamically.
4. **Database Integration Module**
   - **PostgreSQL** database configured with relational mapping using Hibernate ORM.
   - Tables created:
     - *users*: stores login credentials and user profile details.
     - *transactions*: stores all income and expense records.
     - *categories*: stores category names and identifiers.
   - Referential integrity maintained through primary and foreign key constraints.
5. **User Interface Module**
   - Implemented using **Thymeleaf** and **Bootstrap** for an appealing and responsive design.
   - Includes templates for login, registration, dashboard, and reports.
   - Displays real-time data updates through dynamic content rendering.
6. **Error Handling and Validation**
   - Proper feedback messages are displayed for invalid data entry.
   - Exceptions such as database disconnections or invalid sessions are caught and handled gracefully.

4.2 Implementation of Solution

To develop, test, and evaluate the system effectively, various software tools and libraries were employed. Each tool played a specific role in different stages of implementation.

Tool / Technology	Purpose
Spring Boot 3.0	Backend application framework; manages controllers, services, and dependency injection.
PostgreSQL 14	Database management system for persistent data storage.
Hibernate / JPA	Object Relational Mapping (ORM) tool to manage database operations.
Spring Security	Handles authentication and authorization of users.
Chart.js	Used to render financial data graphs (pie and bar charts).
Thymeleaf	Server-side templating engine for rendering dynamic web pages.
Bootstrap 5	Provides CSS framework for responsive user interface design.
JUnit 5	Framework for unit testing backend modules.
Maven	Build automation and dependency management.
IntelliJ IDEA / VS Code	Integrated Development Environments used for coding, debugging, and testing.

4.3 Project Management and Communication

The project was managed following **Agile development principles**, emphasizing iterative development, frequent testing, and adaptability to feedback.

- **Task Division:**
  The project was divided into smaller modules (Authentication, Transaction Management, Analytics, UI) and assigned week-by-week milestones.
- **Documentation:**
  Continuous documentation of code, APIs, and database schema ensured traceability and ease of debugging.
- **Version Control:**
  Git was used for version control and maintaining multiple versions of code throughout the development process.
- **Testing Coordination:**
  Daily testing cycles were conducted to identify bugs early, reducing debugging time at later stages.
- **Communication Tools:**
  Progress tracking was done via checklists and status reports, aligning with weekly project goals and mentor feedback.

Through systematic planning and regular validation, the project was executed efficiently within the four-week timeline.

4.4 Testing and Validation

To ensure the correctness, accuracy, and reliability of the system, rigorous **testing and validation** were conducted at various stages of development. Both **manual** and **automated tests** were performed.

1. Unit Testing

Each function, class, and method was independently tested using **JUnit 5** to ensure correctness in isolation.

- Example: Functions responsible for saving or retrieving transactions were tested using mock
- Result: All unit tests passed successfully with an average execution time below 1 second.

2. Integration Testing

After unit testing, integration testing was carried out to verify proper communication between the backend (Spring Boot) and database (PostgreSQL).

- Test Scenarios:
  - User registration and login validation.
  - Data persistence after CRUD operations.
  - Chart generation from database entries.
- Result: Successful synchronization between all components without data loss or timeout errors.

3. Functional Testing

Functional testing was performed manually to ensure the application behaved according to the requirements.

- Key Features Tested:
  - Adding, editing, deleting transactions.
  - Category filtering and visualization.
  - Authentication flow and session management.
- Result: All functions performed correctly with appropriate feedback and navigation.

4. Validation Testing

Validation testing ensured that the application adhered to data integrity and UI constraints. Input fields like *Amount* and *Date* were tested for proper validation.

- Invalid entries such as alphabets in numeric fields or blank submissions were rejected.
- Database constraints ensured that foreign key references were correctly maintained.

5. Performance Testing

- The system was stress-tested with 1000 sample transaction entries.
- Response time for CRUD operations remained under 1.2 seconds.
  - The dashboard dynamically loaded chart data within 2 seconds, ensuring smooth user experience.

6. Security Testing

- Verified that unauthorized access to URLs (like /dashboard without login) was blocked.
- Passwords were stored as encrypted hashes, ensuring no plaintext exposure.
- Session hijacking and CSRF (Cross-Site Request Forgery) were mitigated using built-in Spring Security measures.

CHAPTER 5 CONCLUSION AND FUTURE WORK

5.1 Conclusion

The **Personal Finance Tracker** project was designed and implemented to create a comprehensive, secure, and user-friendly web application that allows individuals to manage their income and expenses efficiently. The project successfully meets its primary objectives — offering real-time tracking, analytical visualization, and secure data management — all within a self-hosted environment built on modern open-source technologies.

This system provides users with an intuitive dashboard that helps them visualize spending patterns, understand their savings habits, and make informed financial decisions. Unlike commercial financial tracking tools that depend on external servers, this application ensures **complete data ownership and privacy**, as all information is stored securely in a **PostgreSQL database** under the user's control.

The project was developed using the **Spring Boot** framework, which facilitated modularity and reduced development time through dependency injection and embedded configurations. The combination of **Spring Security** for authentication, **Thymeleaf** for user interface rendering, and **Chart.js** for graphical visualization resulted in a highly interactive and secure web solution.

From an educational standpoint, the **Personal Finance Tracker** serves as a valuable demonstration of full-stack development principles, integrating front-end, back-end, and database layers seamlessly. The project also illustrates the implementation of software engineering methodologies such as modular design, MVC architecture, validation testing, and iterative development.

Key Achievements

1. **End-to-End Functionality:**
   Full implementation of CRUD operations for income and expenses, including category management and summary reports.
2. **Data Security and Integrity:**
   Password encryption and restricted access control through Spring Security ensure confidentiality.
3. **Usability and Accessibility:**
   A simple and clean interface with responsive design ensures usability across all devices.
4. **Data Visualization:**
   Integration of charts provides instant insights into financial behavior, encouraging informed budgeting decisions.
5. **Educational and Practical Relevance:**
   The project not only serves as a financial tracking tool but also as a case study for students and developers learning enterprise-level web application design.

Deviations and Challenges

Although the project achieved its primary objectives, a few deviations occurred due to time and scope constraints:

- Advanced features like **multi-currency support** and **AI-based spending predictions** were not implemented.
- The current version supports **single-user operation**, while multi-user collaborative tracking remains part of future work.
- The **export-to-PDF and CSV** reporting feature was postponed to maintain focus on stability and usability.

Despite these limitations, the developed prototype functions effectively and can be easily extended to include additional modules.

5.2 Future Work

While The project has demonstrated significant potential for real-world application and can be enhanced further in several technical and functional aspects.
The following suggestions outline possible directions for **future development**:

1. Multi-User and Role-Based System

Future iterations of the application can include **multi-user authentication** with **role-based access control** (e.g., admin, standard user). This would allow families, small businesses, or organizations to track collective expenses and budgets under a shared platform.

2. AI and Predictive Analytics

Machine learning algorithms can be integrated to analyze user spending patterns and provide **budget optimization recommendations**, such as identifying recurring expenses, predicting monthly savings, and suggesting cost-cutting measures.

3. Cloud Integration

Deploying the system on cloud platforms such as **AWS**, **Azure**, or **Google Cloud** would enable real-time synchronization across devices, offering global accessibility while maintaining database backup and recovery options.

4. Advanced Data Visualization

Enhanced graphical interfaces using **React.js** or **Angular** in combination with APIs can produce dynamic dashboards with advanced filters, multiple chart types, and real-time updates.

5. Expense Forecasting and Budget Alerts

Introduce intelligent alerts to notify users when their monthly spending exceeds budget limits or when recurring payments (e.g., rent, subscriptions) are due. Integration with **email or SMS APIs** could automate notifications.

6. Data Export and Reporting

Allow users to generate detailed reports and export financial summaries into **PDF, Excel, or CSV formats**, enabling external sharing and offline analysis.

7. Mobile Application Version

Developing a **mobile application** using **Flutter** or **React Native** can make the tool more accessible, allowing users to record expenses and track budgets on the go.

8. Integration with Financial APIs

Connecting to third-party APIs such as bank feeds, currency converters, or tax calculators could automate the retrieval of real-time transaction data and provide a more holistic financial overview.

9. Enhanced Security Measures

Future upgrades can include:

- Two-factor authentication (2FA)
- Password recovery options
- Database encryption for sensitive financial details
- Session timeout and audit logging

10. Gamified Savings Interface

To encourage consistent usage, gamification techniques such as achievement badges, savings goals, and progress tracking could be implemented, making the financial management process more engaging.

Suggested Improvements:

Implementing these future improvements would transform the Personal Finance Tracker from a simple finance management tool into a comprehensive, intelligent, and interactive financial assistant. With scalability, cross-platform accessibility, and predictive analytics, the system could serve individuals, small enterprises, and educational institutions alike.

Moreover, the inclusion of AI and data science features would align the project with modern trends in FinTech innovation.

REFERENCES

- **Spring Boot Documentation**, *Spring Framework*, [Online]. Available: https://spring.io/projects/spring-boot
(Accessed: October 2025)

- **PostgreSQL Official Documentation**, *PostgreSQL Global Development Group*, [Online]. Available: https://www.postgresql.org/docs/

- **Thymeleaf Documentation**, *Thymeleaf.org*, [Online]. Available: https://www.thymeleaf.org/documentation.html

- **Chart.js Library**, *Chart.js Team*, [Online]. Available: https://www.chartjs.org/docs/latest/
- **Bootstrap 5 Framework**, *GetBootstrap*, [Online]. Available: https://getbootstrap.com/docs/5.0/
- **Spring Security Guide**, *Spring.io*, [Online]. Available: https://spring.io/guides/topicals/spring-security-architecture

- **Hibernate ORM Documentation**, *Hibernate.org*, [Online]. Available: https://hibernate.org/orm/documentation/

- **JUnit 5 User Guide**, *JUnit.org*, [Online]. Available: https://junit.org/junit5/docs/current/user-guide/
- Chirag Jagani, "**Personal Finance Tracker Built with Spring Boot and PostgreSQL**," *GitHub Repository*, 2024. [Online]. Available: https://github.com/chiragjagani/Personal-Finance-Tracker-Built-with-Spring-Boot-and-PostgreSQL

- **W3Schools**, "*HTML, CSS, and JavaScript Tutorials*," [Online]. Available: https://www.w3schools.com/

- **Oracle**, "*Java SE Documentation*," [Online]. Available: https://docs.oracle.com/en/java/javase/

APPENDIX

A. Database Schema

The database schema for the **Personal Finance Tracker** project is designed using **PostgreSQL** and connected through **Spring Data JPA**.

This schema ensures:

- Each user can have multiple transactions.
- Each transaction is associated with a category.
- Data integrity maintained using primary and foreign keys.

B. Project Dependencies (Maven pom.xml Summary)

Dependency	Purpose
spring-boot-starter-web	Provides web server and REST API functionality
spring-boot-starter-data-jpa	Enables Hibernate ORM and JPA repository support
spring-boot-starter-security	Adds authentication and access control features
postgresql	JDBC driver for PostgreSQL database
thymeleaf	Template engine for rendering frontend views
spring-boot-starter-test	Includes JUnit and Mockito for testing
chart.js	Used for generating visual charts on dashboard

C. Project Hardware and Software Requirements

Category	Specification
Processor	Intel Core i3 or higher
RAM	Minimum 4 GB (Recommended: 8 GB)
Storage	200 MB (application) + 100 MB (database)
Operating System	Windows 10/11, Linux, or macOS
Database	PostgreSQL 14 or above
IDE	IntelliJ IDEA / VS Code / Eclipse
Java Version	JDK 17 or later
Web Browser	Chrome / Edge / Firefox

USER MANUAL

1. Introduction

The **Personal Finance Tracker** is a web application designed to help users manage income and expenses effectively. It provides insights into financial health through graphical summaries and detailed transaction management.

2. Installation Steps

Step 1: Install Prerequisites

Ensure the following are installed on your system:

- **Java JDK 17+**
- **PostgreSQL Database**
- **Maven Build Tool**
- **IntelliJ IDEA / VS Code**

Step 2 : Configure the Database

1. Open **pgAdmin** or your PostgreSQL client.
2. Create a new database:
3. CREATE DATABASE personal_finance_tracker;
4. Update the credentials in application.properties:
5. spring.datasource.url=jdbc:postgresql://localhost:5432/personal_finance_tracker
6. spring.datasource.username=postgres
7. spring.datasource.password=your_password

Step 3: Build and Run the Project

Use Maven to run the application:

mvn spring-boot:run

Once the build is complete, open your browser and visit:

👉 **http://localhost:8080/**

3. Using the Application

3.1 Registration

- Navigate to /register
- Fill in your **username**, **email**, and **password**
- Click on “Register” to create an account

3.2 Login

- Go to /login
- Enter credentials to access the personalized dashboard
- Incorrect credentials display a validation alert

3.3 Adding a Transaction

- Click **Add Transaction**
- Select **Category**, enter **Amount**, **Date**, and optional **Description**
- Click **Save** to store data in the database

3.4 Viewing and Managing Transactions

- All entries are listed in a table format
- You can **Edit** or **Delete** transactions using respective buttons
- Search and filter options are available for easy navigation

3.5 Dashboard and Analytics

- The Dashboard displays:
    - Total Income

- Total Expenses
- Net Balance
- Pie chart showing expense categories
- Bar chart for month-wise expense trends

3.6 Logout

- Use the logout option in the menu to end your session securely