"FINFLOW: THE EXPENSE MANAGER FOR YOUR BETTER FUTURE"

A

Project Report

submitted

in partial fulfillment

for the award of the Degree of

Bachelor of Technology

in Department of Information Technology

Project Mentor:
Dr. Sunita Gupta
Associate Professor-2

Submitted By :
Kashish Arora
21ESKIT061

Department of Information Technology
Swami Keshvanand Institute of Technology, M & G, Jaipur
Rajasthan Technical University, Kota
Session 2024-2025

Swami Keshvanand Institute of Technology, Management & Gramothan, Jaipur

Department of Information Technology

CERTIFICATE

This is to certify that Miss Kashish Arora, a student of B.Tech(Information Technology) 8th semester has submitted his/her Project Report entitled ”FINFLOW: The Expense Tracker ” under my guidance.

Coordinator	**Coordinator**
Dr. Sunita Gupta	Ms. Priyanka Yadav
Associate Professor-2	Assistant Professor
Signature............	Signature............

DECLARATION

I hereby declare that the report of the project entitled **”FINFLOW: The Expense Tracker”** is a record of original work carried out by us at Swami Keshvanand Institute of Technology, Management and Gramothan, Jaipur under the esteemed mentorship of **Dr. Sunita Gupta** (Dept. of Information Technology) and with the valuable coordination of **Ms. Priyanka Yadav** (Dept. of Information Technology).

This project report has been submitted as a part of the partial fulfillment of the requirements for the award of the degree of Bachelor of Technology (B.Tech) in the Department of Information Technology. To the best of our knowledge and belief, it has not been submitted anywhere else under any other program.

Team Members	**Signature**
Kashish Arora, 21ESKIT061	______________

Acknowledgement

A project of such thoughtful planning and technical execution cannot be completed without the guidance and support of several individuals. We take this opportunity to express our heartfelt gratitude to all those who have supported us throughout the development of this project.

We are deeply grateful to our mentor **Dr. Sunita Gupta** for her constant guidance, encouragement, and insightful feedback. Her mentorship was instrumental in shaping our ideas and helping us navigate challenges with confidence.

We would also like to extend our sincere thanks to our project coordinator **Ms. Priyanka Yadav** for her valuable support, cooperation, and motivation, which helped us stay focused and on the right track throughout the project journey.

We further express our sincere appreciation to **Dr. Anil Chaudhary**, HOD, Department of Information Technology, for providing the necessary resources, support, and a collaborative environment that made this project possible.

We are thankful to all faculty members of Swami Keshvanand Institute of Technology, Management and Gramothan, Jaipur, and our fellow colleagues for their cooperation, encouragement, and support during every phase of our project.

Last but not the least, we wish to thank everyone who directly or indirectly contributed to the successful completion of this project.

Team Member:

Kashish Arora, 21ESKIT061

Table of Content

1 Introduction **2**

1.1 Problem Statement and Objective . . . 2

1.2 Literature Survey / Market Survey / Investigation and Analysis . . . 2

1.3 Introduction to Project . . . 3

1.4 Proposed Logic / Algorithm / Business Plan / Solution / Device . . . 3

1.5 Scope of the Project . . . 3

2 Software Requirement Specification **5**

2.1 Overall Description . . . 5

2.1.1 Product Perspective . . . 5

2.1.1.1 System Interfaces . . . 5

2.1.1.2 User Interfaces . . . 6

2.1.1.3 Hardware Interfaces . . . 6

2.1.1.4 Software Interfaces . . . 6

2.1.1.5 Communication Interfaces . . . 6

2.1.1.6 Memory Constraints . . . 7

2.1.1.7 Operations . . . 7

2.1.1.8 Project Functions . . . 7

2.1.1.9 User Characteristics . . . 7

2.1.1.10 Constraints . . . 8

2.1.1.11 Assumptions and Dependencies . . . 8

3 System Design Specification **9**

3.1 System Architecture . . . 9

3.2 Module Decomposition Description . . . 9

3.3 High Level Design Diagrams . . . 11

3.3.1 Use Case Diagram . . . 11

3.3.2 Sequence Diagram 12
3.3.3 Data Flow Diagram (DFD) 12
3.3.4 Class Diagram 13
3.3.5 ER Diagram 14
3.3.6 Communication Diagram 14
3.3.7 Deployment Diagram 15

4 Methodology and Team **16**
4.1 Introduction to Waterfall Framework 16
4.2 Team Member, Roles & Responsibilities 18

5 System Testing **20**
5.1 Functionality Testing 20
5.2 Performance Testing 21
5.3 Usability Testing 22

6 Test Execution Summary **23**

7 Project Screen Shots **25**

8 Project Summary and Conclusions **30**
8.1 Conclusion 30

9 Future Scope **31**

References **32**

List of Figures

3.1 System Architecture 10
3.2 Use Case Diagram for Expense Tracker 11
3.3 Sequence Diagram: User Adds Expense 12
3.4 DFD Level 0: Overview of Expense Tracker 12
3.5 DFD Level 1: User Expense Management 13
3.6 Class Diagram: Entity Relationships 13
3.7 ER Diagram for Expense Tracker 14
3.8 Communication Diagram: User and System Interaction 14
3.9 Deployment Diagram for Expense Tracker Web App 15

4.1 **Waterfall Model** 16

7.1 HOMEPAGE 25
7.2 Registration Page 26
7.3 Login Page 26
7.4 DashBoard Page 27
7.5 Analysis Page 27
7.6 AI-Mitra 28
7.7 All Expenses with filters 28
7.8 Profile 29

List of Tables

6.1 Test Case Execution Summary for Expense Tracker Core Features . 24

CHAPTER 1

Introduction

1.1 Problem Statement and Objective

In today's fast-paced world, individuals often struggle to manage their daily expenses effectively, leading to untracked spending, poor budgeting habits, and financial stress. Despite the availability of several personal finance tools, many users find them either too complex or not tailored to their specific needs. The objective of this project is to develop a user-friendly, intuitive, and secure expense tracking system — **FINFLOW: The Expense Tracker**. This application enables users to record their expenses, categorize them, and visualize their financial health through charts and dashboards, empowering them to make informed financial decisions.

1.2 Literature Survey / Market Survey / Investigation and Analysis

Numerous personal finance applications like Walnut, Money Manager, and Spendee currently exist, offering varying degrees of financial tracking. However, many of these apps come with limitations such as paid features, lack of customization, or poor user experience. Additionally, some applications do not provide strong backend structures for data security or scalability. Through this project, we aim to address these gaps by providing a robust, secure, and modular expense tracking system with a clean UI and features tailored for Indian users. Our platform emphasizes usability, performance, and meaningful analytics, leveraging modern full-stack technologies.

1.3 Introduction to Project

FINFLOW: The Expense Tracker is a full-stack web application built to assist users in monitoring and managing their daily expenses with ease. It allows users to register, log in securely, add and edit expenses, categorize spending (e.g., food, travel, utilities), and generate reports for financial analysis. The application offers a responsive user interface, ensuring seamless access across devices. Key features include data visualization via charts, monthly summaries, and budget alerts. The inspiration for the project came from the growing need for digital tools that simplify personal finance management and encourage better financial habits.

1.4 Proposed Logic / Algorithm / Business Plan / Solution / Device

The system operates on a CRUD-based architecture where users can create, read, update, and delete expense entries. Each entry is tagged with a category, amount, date, and optional notes. A backend developed using Spring Boot handles business logic, data validation, and database transactions, while the Angular-based frontend ensures a dynamic and responsive user interface. Data is stored securely in a relational database (MySQL or PostgreSQL), and JWT-based authentication secures user sessions. A budget-monitoring algorithm flags users when spending in a category exceeds predefined limits, and visual charts offer insights into spending patterns. The system is modular, scalable, and follows clean coding practices.

1.5 Scope of the Project

The primary scope of **FINFLOW** is to serve individual users — including students, working professionals, and homemakers — who want to track their daily or monthly expenses efficiently. While the current focus is on individual finance tracking, the system can be scaled in the future to support collaborative budgeting (e.g., family or roommate groups), income tracking, and integration with banking APIs. The key deliverables include a secure and responsive web application with user authentication, expense categorization, visualization dashboards, alert mechanisms, and intuitive UX. The project not only encourages digital literacy but also promotes financial awareness among its users.

CHAPTER 2

Software Requirement Specification

2.1 Overall Description

The Expense Tracker Web Application is a full-stack solution designed to help users monitor and manage their daily, weekly, and monthly expenses. It allows users to log financial transactions, categorize them, and gain insights through visual reports. The system is role-based (User, Admin) and offers personalized dashboards for better financial decision-making. This section outlines the system's operational environment, how it interacts with other components, and what is expected from users and systems.

2.1.1 Product Perspective

2.1.1.1 System Interfaces

The application follows a modular architecture where frontend and backend interact via RESTful APIs:

- **Frontend Interface:** Developed using Angular. It communicates with backend services through HTTP protocols using Angular's HttpClient module.
- **Backend Services:** Built using Spring Boot, the backend exposes REST APIs for authentication, data handling, and business logic.
- **Database Layer:** Utilizes MySQL or PostgreSQL for storing structured data related to expenses, users, and categories.
- **Third-party Services:** Potential integration with APIs for email services (e.g., SendGrid), currency exchange rates, or cloud authentication providers.

2.1.1.2 User Interfaces

The user interface is implemented using Angular and styled with Tailwind CSS for a responsive, user-friendly experience across devices:

- **User Dashboard:** Allows users to manage their expenses, view analytics, set monthly budgets, and track financial patterns.
- **Admin Dashboard:** Enables monitoring of user activity, access control, and platform performance statistics.

2.1.1.3 Hardware Interfaces

The system does not require any specific hardware beyond standard computing devices:

- **Client Side:** Any modern browser on desktop, laptop, tablet, or mobile.
- **Server Side:** Cloud-deployed backend on Linux-based servers (e.g., AWS EC2, Heroku, or Render).

2.1.1.4 Software Interfaces

- **Frontend:** Angular, Tailwind CSS.
- **Backend:** Spring Boot with Java and RESTful APIs.
- **Database:** MySQL or PostgreSQL with Spring Data JPA.
- **Authentication:** OAuth 2.0 and JWT-based session management.

2.1.1.5 Communication Interfaces

The application communicates over the internet with secure and efficient protocols:

- HTTPS for secure data exchange between client and server.
- RESTful APIs for frontend-backend communication.
- SMTP or external API for sending emails and alerts (optional).

2.1.1.6 Memory Constraints

The application is optimized for cloud deployment. Approximate memory requirements include:

- **Client Side:** Minimum 4 GB RAM for optimal performance in browsers.
- **Server Side:** Minimum 8 GB RAM recommended for backend and database hosting.

2.1.1.7 Operations

The system is designed for continuous operation with minimal downtime. Key operational aspects include:

- Automated database backups on a scheduled basis.
- CI/CD pipelines for seamless code deployment and rollback.
- Role-based access and security mechanisms.

2.1.1.8 Project Functions

- User registration, login, and profile management.
- Expense creation, categorization, and deletion.
- Monthly expense summaries and graphical analysis.
- Notifications for overspending or budget thresholds.
- Admin panel for usage monitoring and user management.

2.1.1.9 User Characteristics

- **Users:** Individuals looking to track and manage their personal or household expenses.
- **Admin:** Responsible for maintaining the platform, user verification, and system analytics.

2.1.1.10 Constraints

- Currently available as a web-based application only.
- Requires a stable internet connection for full functionality.
- All financial data must be securely stored and validated.
- Email alerts depend on third-party services.

2.1.1.11 Assumptions and Dependencies

- Users provide correct and truthful information during registration.
- Hosting provider ensures reliable uptime (at least 99.9%).
- External APIs and services used (e.g., email, authentication) remain available and functional.
- Future scalability is considered; the codebase supports modular enhancements.

CHAPTER 3

System Design Specification

3.1 System Architecture

The Expense Tracker Web Application follows a three-tier architecture: **Presentation Layer (Frontend)**, **Application Layer (Backend/Business Logic)**, and **Data Layer (Database)**.

The frontend is built using Angular and styled with Tailwind CSS, delivering a responsive and dynamic user experience. The backend is developed with Spring Boot, exposing RESTful APIs that handle authentication, expense logic, and admin control functionalities. The data is stored in a relational database like MySQL or PostgreSQL using Spring Data JPA.

This architecture promotes modularity, scalability, and maintainability. Secure data exchange is ensured through HTTPS, with authentication handled using JWT or OAuth 2.0 protocols.

3.2 Module Decomposition Description

The system is divided into several key functional modules:

- **Authentication Module:** Manages secure user login, registration, and role-based access for users and administrators using JWT/OAuth.
- **User Dashboard Module:** Allows users to add, edit, categorize, and delete expense records and set monthly budgets.
- **Analytics and Reporting Module:** Generates graphical reports and summaries for daily, weekly, and monthly spending patterns.

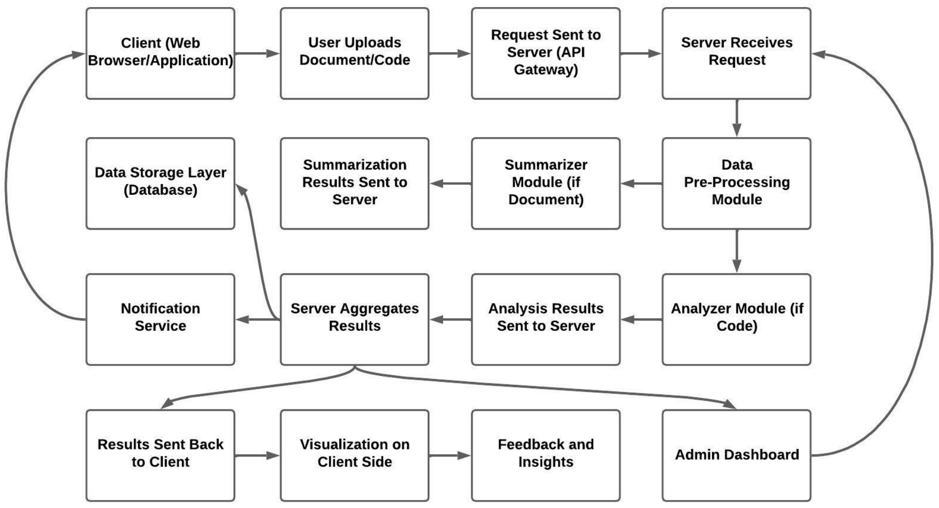

Client (Web Browser/Application)
User Uploads Document/Code
Request Sent to Server (API Gateway)
Server Receives Request
Data Storage Layer (Database)
Summarization Results Sent to Server
Summarizer Module (if Document)
Data Pre-Processing Module
Notification Service
Server Aggregates Results
Analysis Results Sent to Server
Analyzer Module (if Code)
Results Sent Back to Client
Visualization on Client Side
Feedback and Insights
Admin Dashboard

Figure 3.1: System Architecture

- **Budget Alert Module:** Sends alerts when a user is about to exceed or has exceeded their budget limit.
- **Admin Module:** Provides admin functionalities such as user activity monitoring, platform health checks, and basic analytics.
- **Notification Module:** Sends optional email notifications or alerts via integrated APIs (e.g., SendGrid).

3.3 High Level Design Diagrams

3.3.1 Use Case Diagram

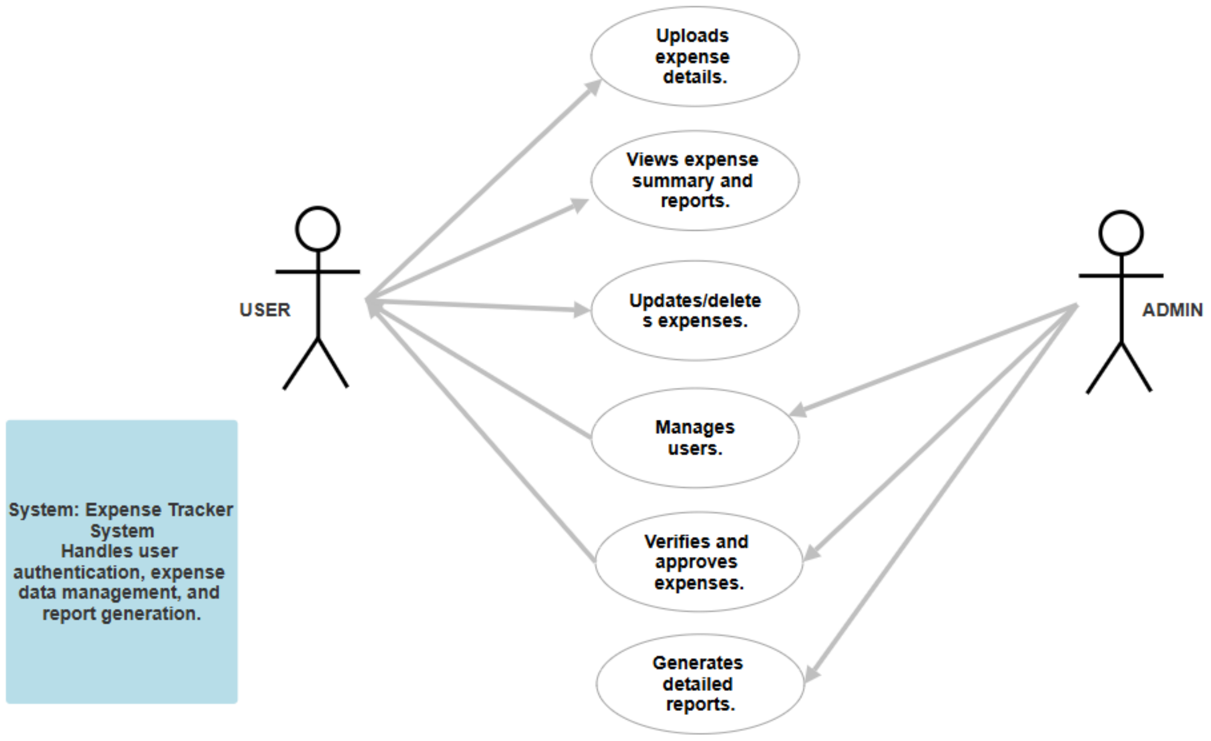

Uploads expense details.
Views expense summary and reports.
Updates/deletes expenses.
Manages users.
Verifies and approves expenses.
Generates detailed reports.
USER
ADMIN
System: Expense Tracker System
Handles user authentication, expense data management, and report generation.

Figure 3.2: Use Case Diagram for Expense Tracker

3.3.2 Sequence Diagram

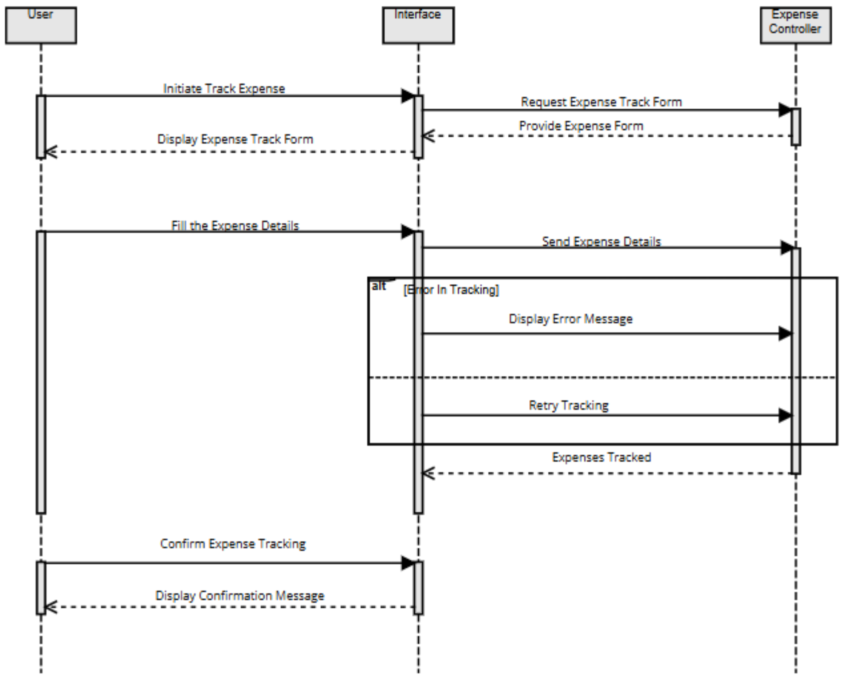


Figure 3.3: Sequence Diagram: User Adds Expense

3.3.3 Data Flow Diagram (DFD)

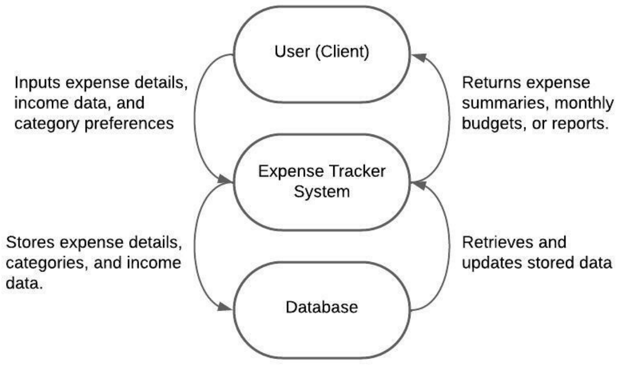

User (Client)
Inputs expense details, income data, and category preferences
Returns expense summaries, monthly budgets, or reports.
Expense Tracker System
Stores expense details, categories, and income data.
Retrieves and updates stored data
Database

Figure 3.4: DFD Level 0: Overview of Expense Tracker

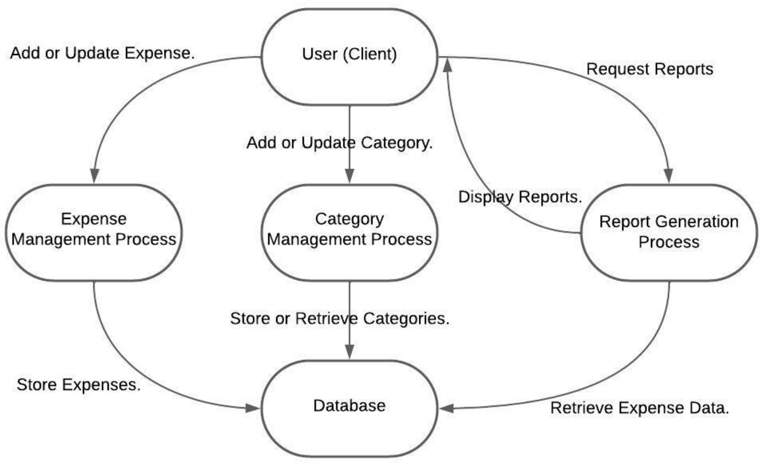


Figure 3.5: DFD Level 1: User Expense Management

3.3.4 Class Diagram

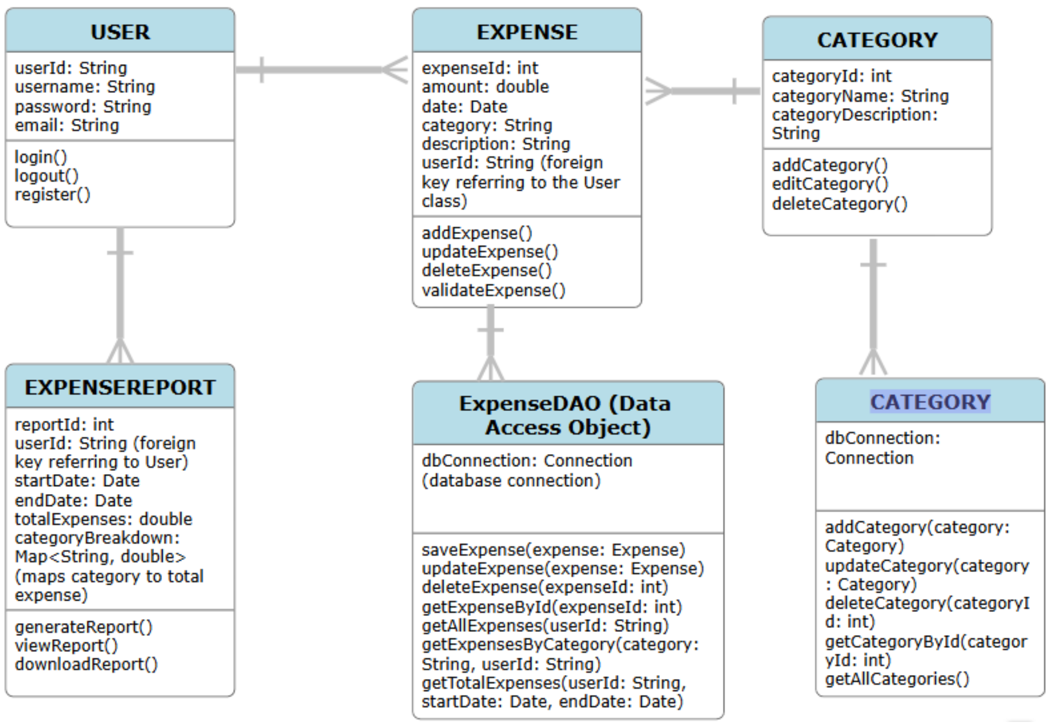

USER
userId: String
username: String
password: String
email: String
login()
logout()
register()
EXPENSE
expenseId: int
amount: double
date: Date
category: String
description: String
userId: String (foreign key referring to the User class)
addExpense()
updateExpense()
deleteExpense()
validateExpense()
CATEGORY
categoryId: int
categoryName: String
categoryDescription: String
addCategory()
editCategory()
deleteCategory()
EXPENSEREPORT
reportId: int
userId: String (foreign key referring to User)
startDate: Date
endDate: Date
totalExpenses: double
categoryBreakdown: Map<String, double> (maps category to total expense)
generateReport()
viewReport()
downloadReport()
ExpenseDAO (Data Access Object)
dbConnection: Connection (database connection)
saveExpense(expense: Expense)
updateExpense(expense: Expense)
deleteExpense(expenseId: int)
getExpenseById(expenseId: int)
getAllExpenses(userId: String)
getExpensesByCategory(category: String, userId: String)
getTotalExpenses(userId: String, startDate: Date, endDate: Date)
CATEGORY
dbConnection: Connection
addCategory(category: Category)
updateCategory(category : Category)
deleteCategory(categoryId: int)
getCategoryById(categoryId: int)
getAllCategories()

Figure 3.6: Class Diagram: Entity Relationships

3.3.5 ER Diagram

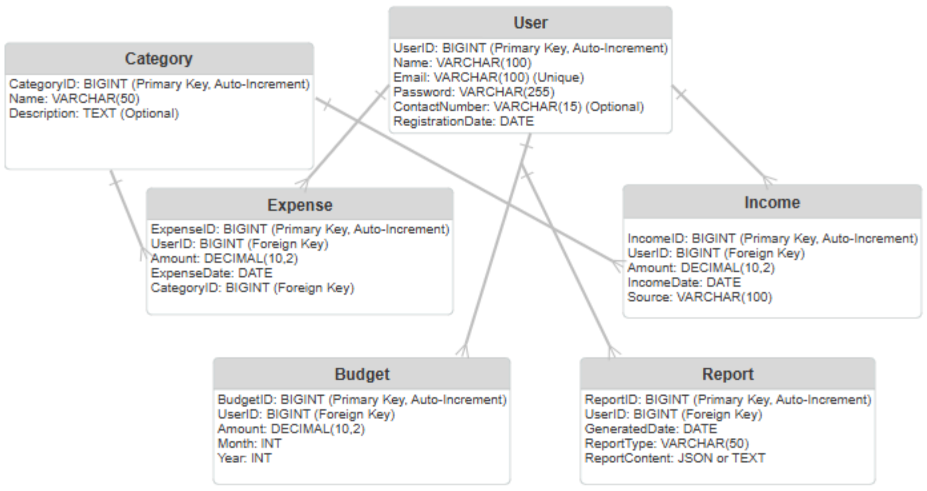

Category
CategoryID: BIGINT (Primary Key, Auto-Increment)
Name: VARCHAR(50)
Description: TEXT (Optional)
User
UserID: BIGINT (Primary Key, Auto-Increment)
Name: VARCHAR(100)
Email: VARCHAR(100) (Unique)
Password: VARCHAR(255)
ContactNumber: VARCHAR(15) (Optional)
RegistrationDate: DATE
Expense
ExpenseID: BIGINT (Primary Key, Auto-Increment)
UserID: BIGINT (Foreign Key)
Amount: DECIMAL(10,2)
ExpenseDate: DATE
CategoryID: BIGINT (Foreign Key)
Income
IncomeID: BIGINT (Primary Key, Auto-Increment)
UserID: BIGINT (Foreign Key)
Amount: DECIMAL(10,2)
IncomeDate: DATE
Source: VARCHAR(100)
Budget
BudgetID: BIGINT (Primary Key, Auto-Increment)
UserID: BIGINT (Foreign Key)
Amount: DECIMAL(10,2)
Month: INT
Year: INT
Report
ReportID: BIGINT (Primary Key, Auto-Increment)
UserID: BIGINT (Foreign Key)
GeneratedDate: DATE
ReportType: VARCHAR(50)
ReportContent: JSON or TEXT

Figure 3.7: ER Diagram for Expense Tracker

3.3.6 Communication Diagram

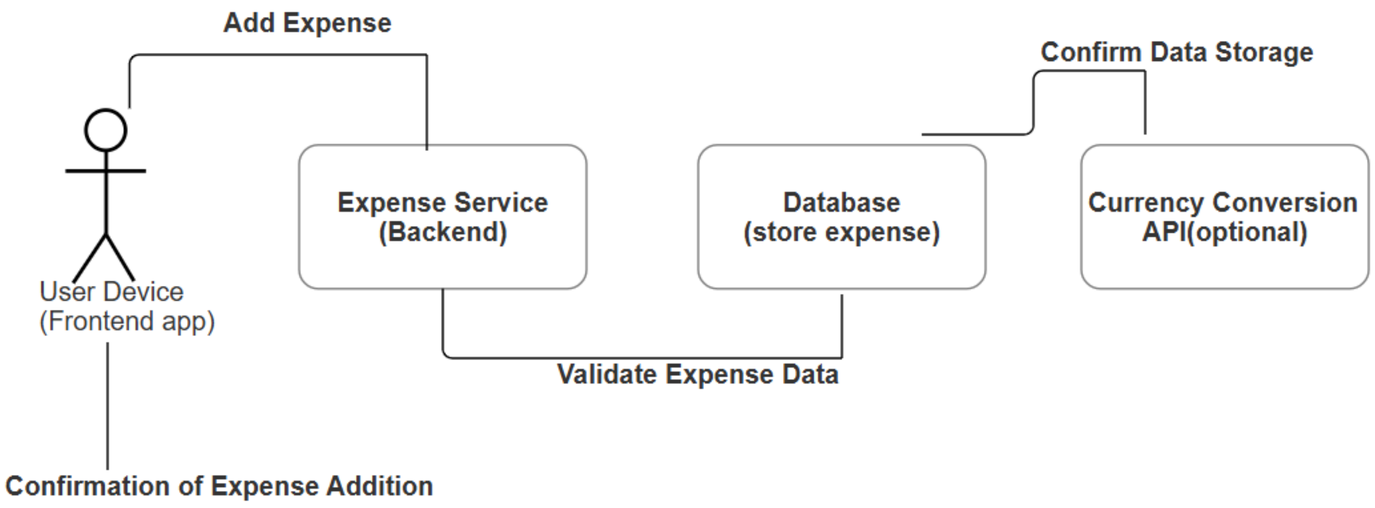


Figure 3.8: Communication Diagram: User and System Interaction

3.3.7 Deployment Diagram

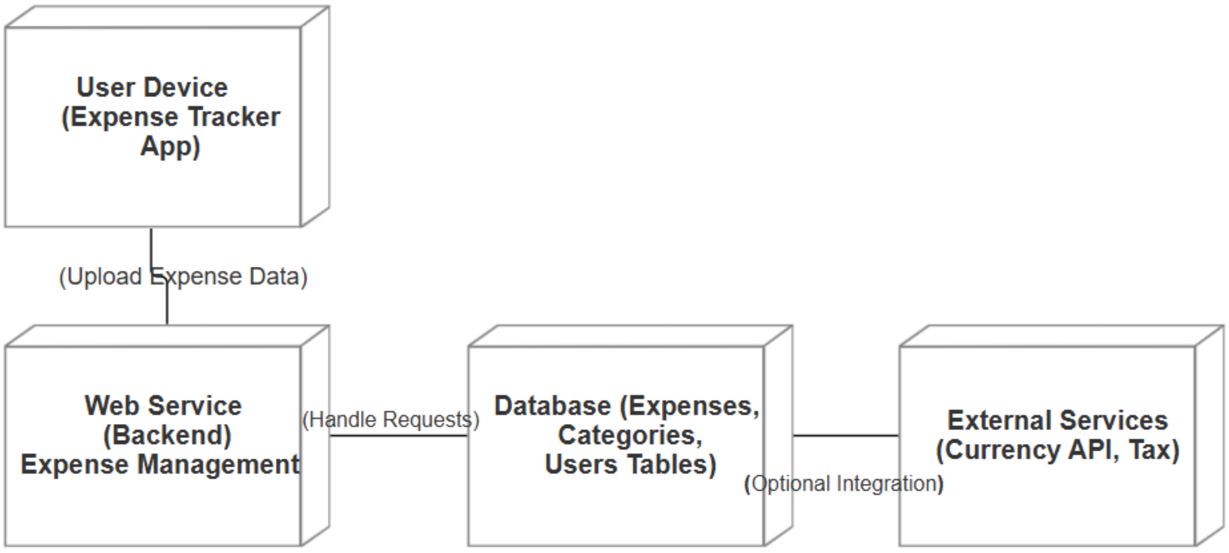

User Device (Expense Tracker App)
(Upload Expense Data)
Web Service (Backend) Expense Management
(Handle Requests)
Database (Expenses, Categories, Users Tables)
(Optional Integration)
External Services (Currency API, Tax)

Figure 3.9: Deployment Diagram for Expense Tracker Web App

CHAPTER 4

Methodology and Team

4.1 Introduction to Waterfall Framework

The Waterfall Model is a traditional software development methodology that follows a linear and sequential approach. Each phase must be completed before the next one begins, making it highly structured and easy to manage.

The development process follows a top-down approach through phases including: Requirements, Design, Implementation, Testing, Deployment, and Maintenance. This model is particularly effective for projects where requirements are well-defined and unlikely to change during development.

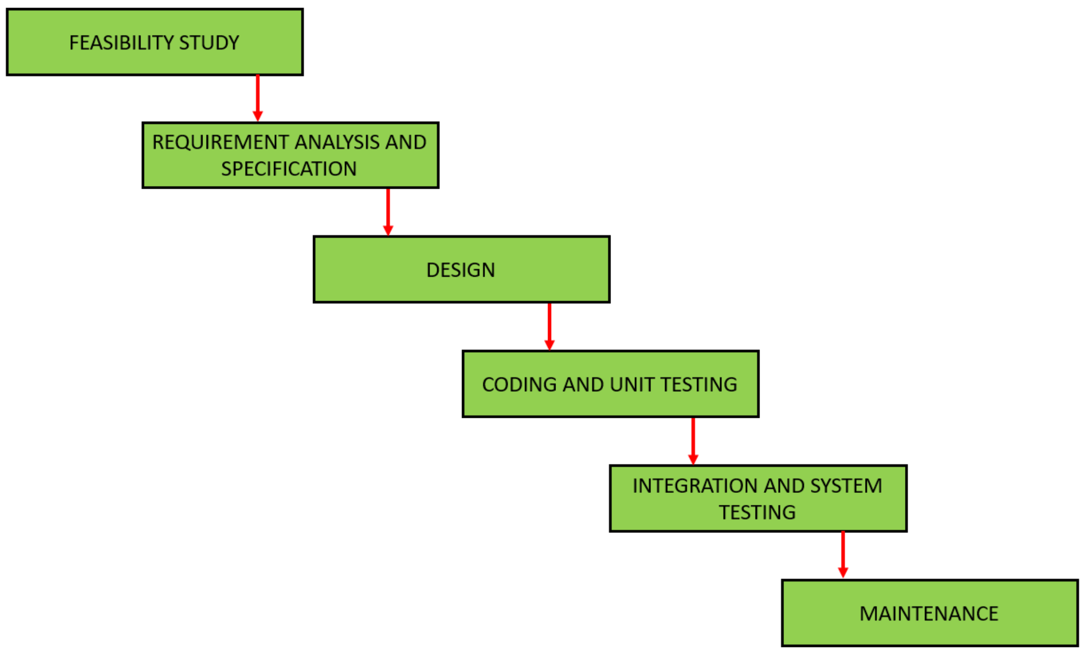

FEASIBILITY STUDY
REQUIREMENT ANALYSIS AND SPECIFICATION
DESIGN
CODING AND UNIT TESTING
INTEGRATION AND SYSTEM TESTING
MAINTENANCE

Figure 4.1: Waterfall Model

The sequential phases of the Waterfall model as applied to our Expense Tracker project include:

1. **Requirement Gathering and Analysis:**

   Gathered specific requirements such as expense categorization, monthly budget setting, analytics, and user authentication. These were documented in the Software Requirements Specification (SRS) document.

2. **System Design:**

   Created detailed architecture including frontend-backend interaction, REST API planning, database schema (ER diagram), and component-based UI layout.

3. **Implementation:**

   Developed the frontend using Angular and Tailwind CSS, and backend using Spring Boot. Implemented user authentication, CRUD operations for expenses, and alert notifications.

4. **Integration and Testing:**

   Integrated frontend with backend APIs using HTTPClient and performed unit and integration testing of modules using tools like JUnit, Postman, and Angular testing libraries.

5. **Deployment of System:**

   Deployed the application to a web server for demonstration and usage. The backend was deployed on a local or cloud-based Spring Boot runtime with connected relational database.

6. **Maintenance:**

   Monitored the application post-deployment for bug fixing, UI enhancements, performance tuning, and new feature additions.

Each phase in the Waterfall model strictly follows completion of the previous phase, ensuring traceability and documentation at each level.

Advantages of Waterfall Model:

- Allows for structured planning and progress tracking.
- Clearly defined stages with well-documented deliverables.
- Easy to manage due to its rigid model structure.
- Suitable for short-term projects with fixed requirements.

Disadvantages of Waterfall Model:

- Difficult to accommodate changes once a phase is completed.
- Late discovery of bugs as testing is performed only after implementation.
- Not ideal for complex, long-term projects where requirements may evolve.

4.2 Team Member, Roles & Responsibilities

This project was individually developed and managed by a single contributor, taking full responsibility for all phases including planning, development, testing, and documentation. The following roles were handled:

- **Project Leader:**

  Planned the project structure, selected the technology stack (Angular for frontend and Spring Boot for backend), and outlined the system architecture and development timeline.

- **Frontend Developer:**

  Designed and implemented responsive UI components using Angular and Tailwind CSS. Managed routing, form validation, state handling, and integration with backend APIs.

- **Backend Developer:**

Developed RESTful APIs using Spring Boot, implemented JWT-based authentication, handled CRUD operations, and managed the relational database schema.

- **Tester:**

  Performed unit testing and integration testing using Postman, JUnit, and Angular testing tools. Ensured the system worked as expected across different scenarios.

- **Documentation Author:**

  Created the Software Requirements Specification (SRS), detailed system design documentation, and user manual, ensuring clarity and traceability throughout the development cycle.

CHAPTER 5

System Testing

The developed Expense Tracker system was thoroughly tested to ensure it satisfies all functional and non-functional requirements. The testing process was focused on identifying bugs, verifying correct behavior, and confirming the system's performance under typical usage scenarios.

5.1 Functionality Testing

The following features of the application were tested to validate their correctness and user experience.

1. **Links**

   (a) **Internal Links:**

   All internal navigation links such as Dashboard, Add Expense, View Transactions, and Categories were tested. Each route was verified for correct redirection and data rendering specific to the logged-in user.

   (b) **External Links:**

   No external links were included in the initial version. Future releases may incorporate links to financial tools, APIs, or educational budgeting resources.

   (c) **Broken Links:**

   Manual and automated testing confirmed there are no broken links in the system. All routes are functioning as intended.

2. **Forms**

(a) **Error Message for Wrong Input:**

Real-time error messages appear when incorrect or missing values are entered in forms (e.g., missing amount, invalid email format). These validations improve the overall user experience.

(b) **Optional and Mandatory Fields:**

Mandatory fields such as ”Amount”, ”Category”, and ”Date” are clearly marked. The system prevents form submission until all required fields are filled, with corresponding error notifications displayed.

3. **Database**

   - CRUD operations such as adding a new expense, editing existing entries, and deleting records were tested through the integrated UI. Data persistence was verified across user sessions.
   - Backend validations and database constraints were tested to ensure referential integrity and prevention of duplicate or invalid entries.

5.2 Performance Testing

Performance tests were conducted to validate responsiveness and resource usage under realistic user loads.

- **Page Load Speed:**

  All core pages including login, dashboard, and expense forms load within 1–2 seconds under normal internet conditions.

- **API Response Time:**

  Spring Boot REST APIs typically responded within 150–300 milliseconds under test data load.

- **Data Scalability:**

The system was populated with over 500 sample expense records to evaluate how data retrieval and filtering performed. No lags or crashes were observed.

- **Session Handling:**

  Proper session management ensured that users remained authenticated during active use, with session expiration handled securely.

5.3 Usability Testing

Usability testing focused on how user-friendly and accessible the system is for typical users managing their expenses.

- **Ease of Navigation:**

  Users can quickly access core functions like adding an expense or viewing reports with minimal clicks. The navigation is simple and intuitive.

- **Design Consistency:**

  Tailwind CSS was used to maintain a consistent visual design across all screens. Buttons, form inputs, and labels follow a uniform layout.

- **Feedback and Error Messaging:**

  The system provides immediate feedback on successful actions (e.g., ”Expense added”) and displays clear messages on input errors or failed operations.

- **Accessibility:**

  The interface was tested on both desktop and mobile devices. Font sizes, button placements, and color contrast were optimized for readability.

- **User Feedback:**

  Informal feedback from peers and mentors indicated that the app was simple to understand and aesthetically pleasing, with all features functioning as expected.

CHAPTER 6

Test Execution Summary

This chapter provides a summary of the testing process executed during the development lifecycle of the Expense Tracker application. The test plan was outlined at the initial phase, while this summary report was compiled upon completion of the system to evaluate its functionality, reliability, and user experience. The report outlines each key scenario tested, their outcomes, and the effort involved, helping assess the overall system quality.

As the sole developer and tester of this project, all test cases were designed, executed, and documented independently. This summary aims to ensure the application behaves as intended under normal user workflows.

The report includes:

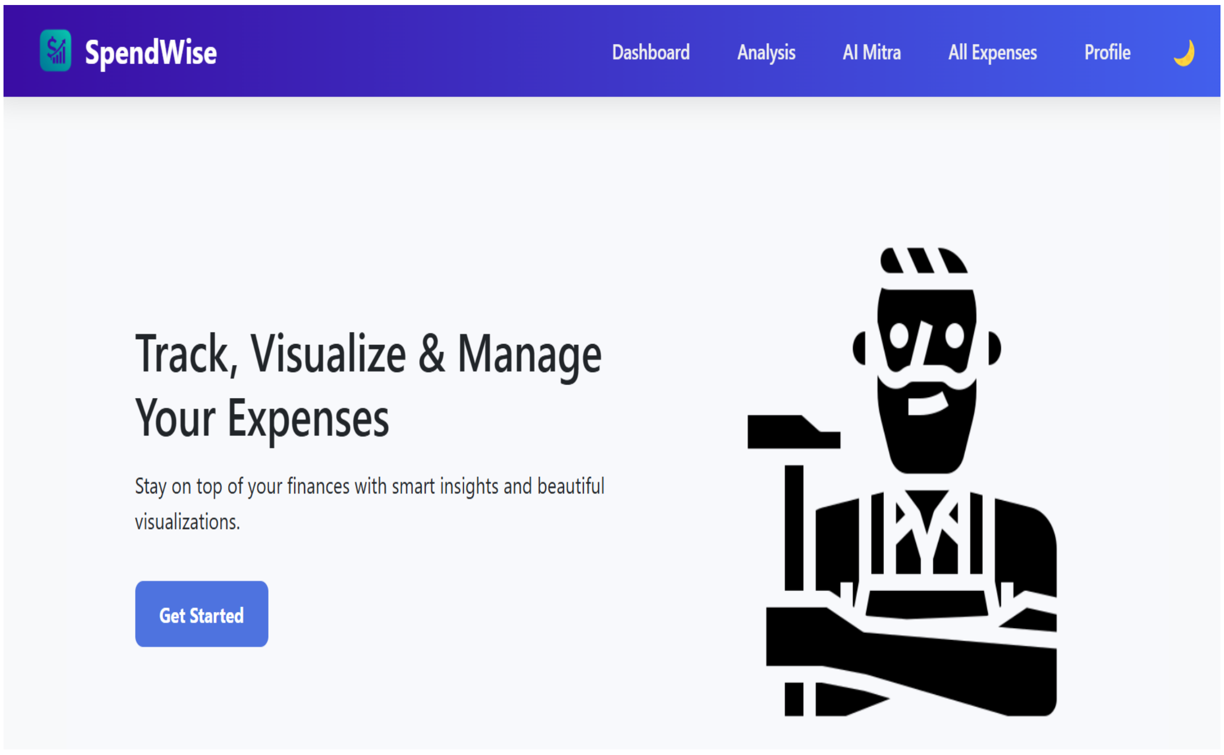
- Unique Test Case IDs and scenario mapping
- Description of each test scenario
- Outcome of test execution (Pass/Fail)
- Resources used (time and tools involved)

S.No	Test Case ID	Test Case Description	Status	Resources Used
1	TC001	User login with valid and invalid credentials	Pass	Manual testing, 8 mins
2	TC002	Add expense with complete and incomplete data	Pass	Manual testing, 10 mins
3	TC003	View and filter expense history by date/category	Pass	Manual testing, 12 mins
4	TC004	Category creation, update, and deletion workflows	Pass	Manual testing, 15 mins
5	TC005	Responsive design check across devices (mobile/desktop)	Pass	Chrome DevTools, 20 mins
6	TC006	Data persistence after refresh/logout	Pass	Spring Boot MySQL backend, 7 mins
7	TC007	Validation error display on missing fields	Pass	Form testing with dummy data, 9 mins
8	TC008	API response time and error handling for Add Expense	Pass	Postman and browser console, 10 mins

Table 6.1: Test Case Execution Summary for Expense Tracker Core Features

CHAPTER 7

Project Screen Shots



Track, Visualize & Manage Your Expenses

Stay on top of your finances with smart insights and beautiful visualizations.

Figure 7.1: HOMEPAGE

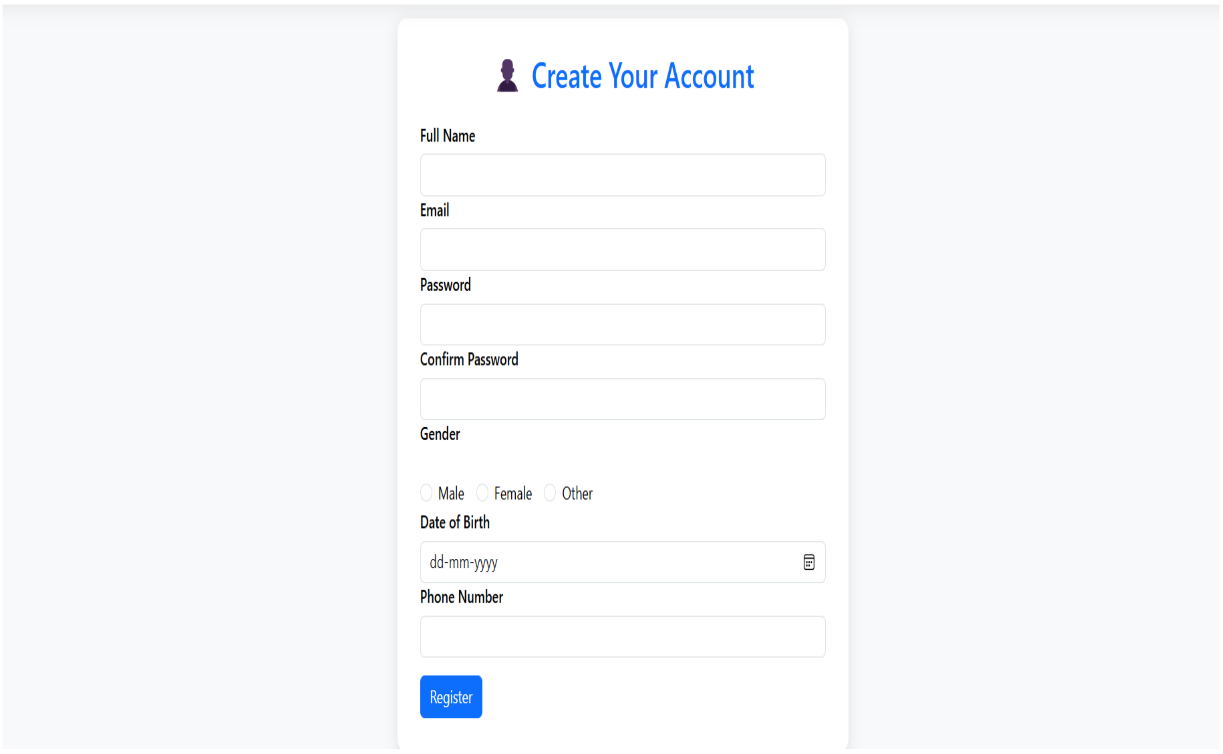


Figure 7.2: Registration Page

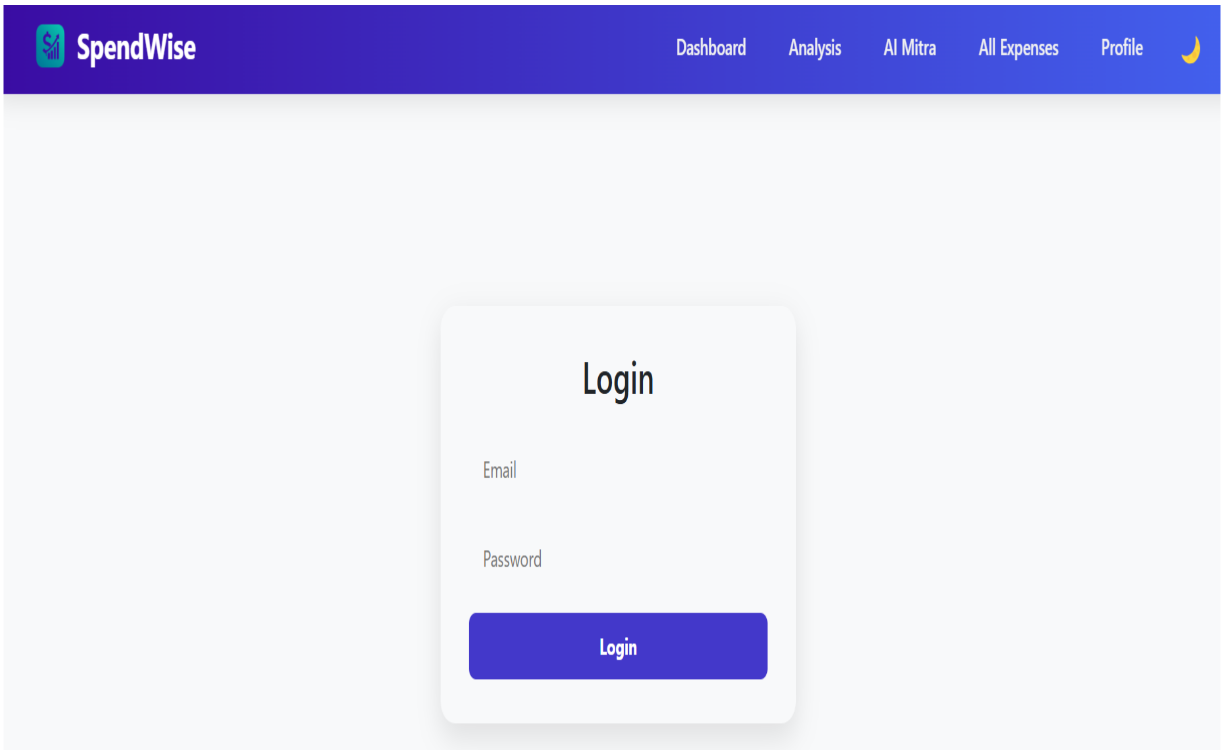
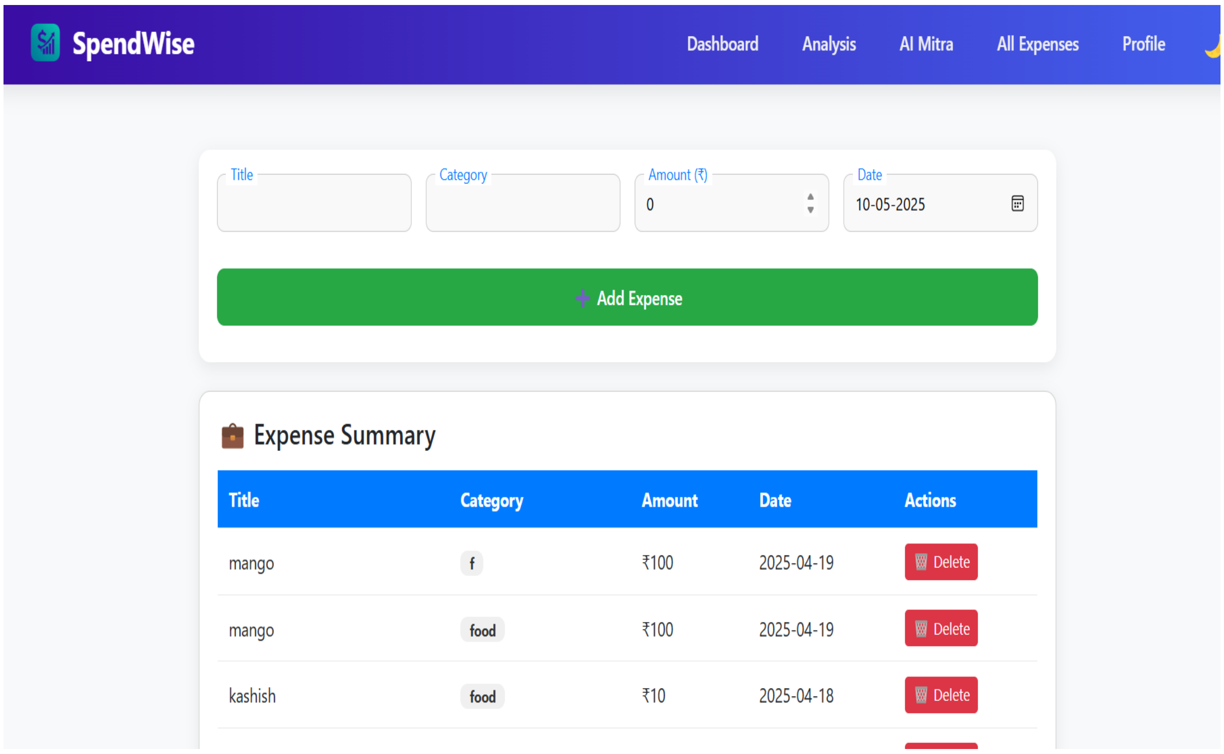


Figure 7.3: Login Page



Expense Summary

Title	Category	Amount	Date	Actions
mango	f	₹100	2025-04-19	Delete
mango	food	₹100	2025-04-19	Delete
kashish	food	₹10	2025-04-18	Delete

Figure 7.4: DashBoard Page

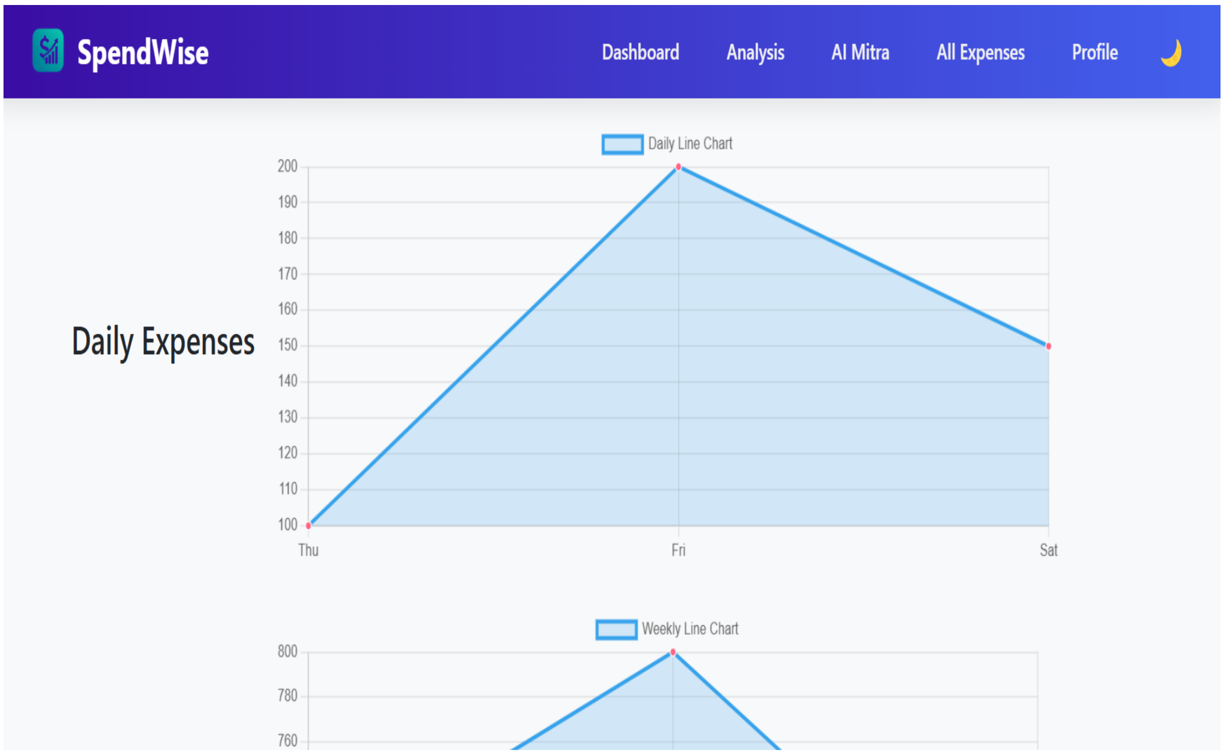


Figure 7.5: Analysis Page

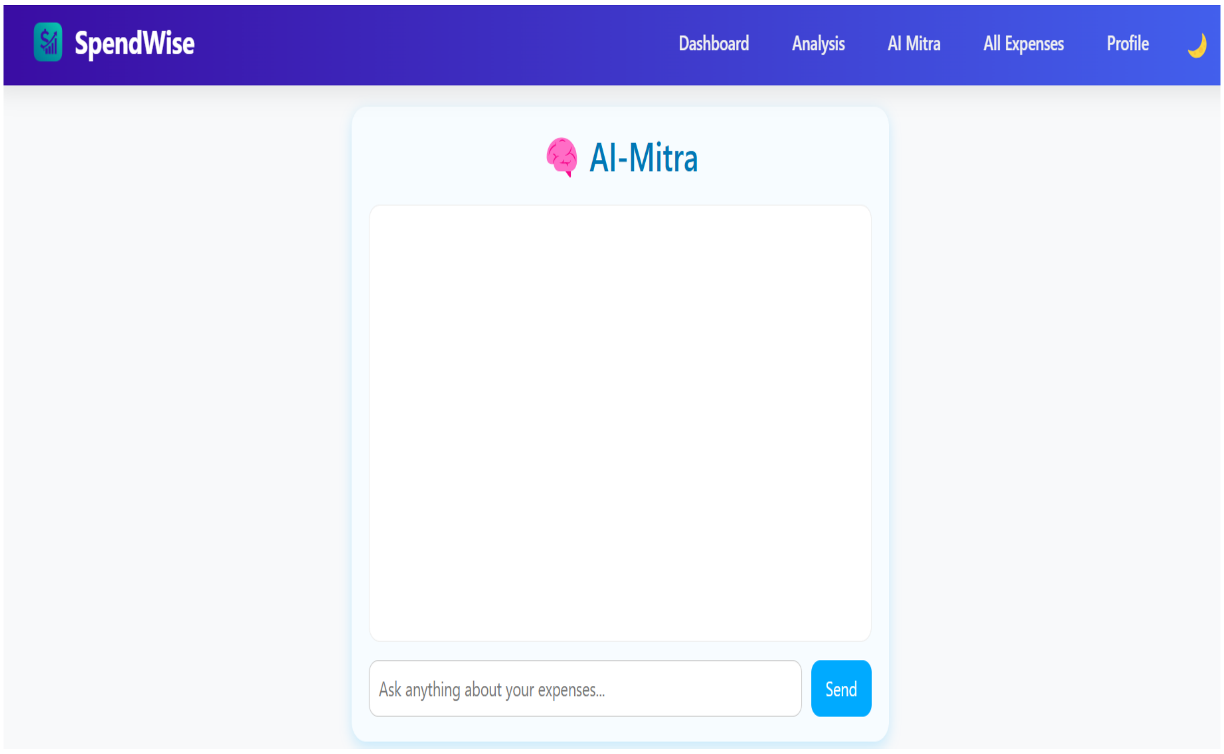
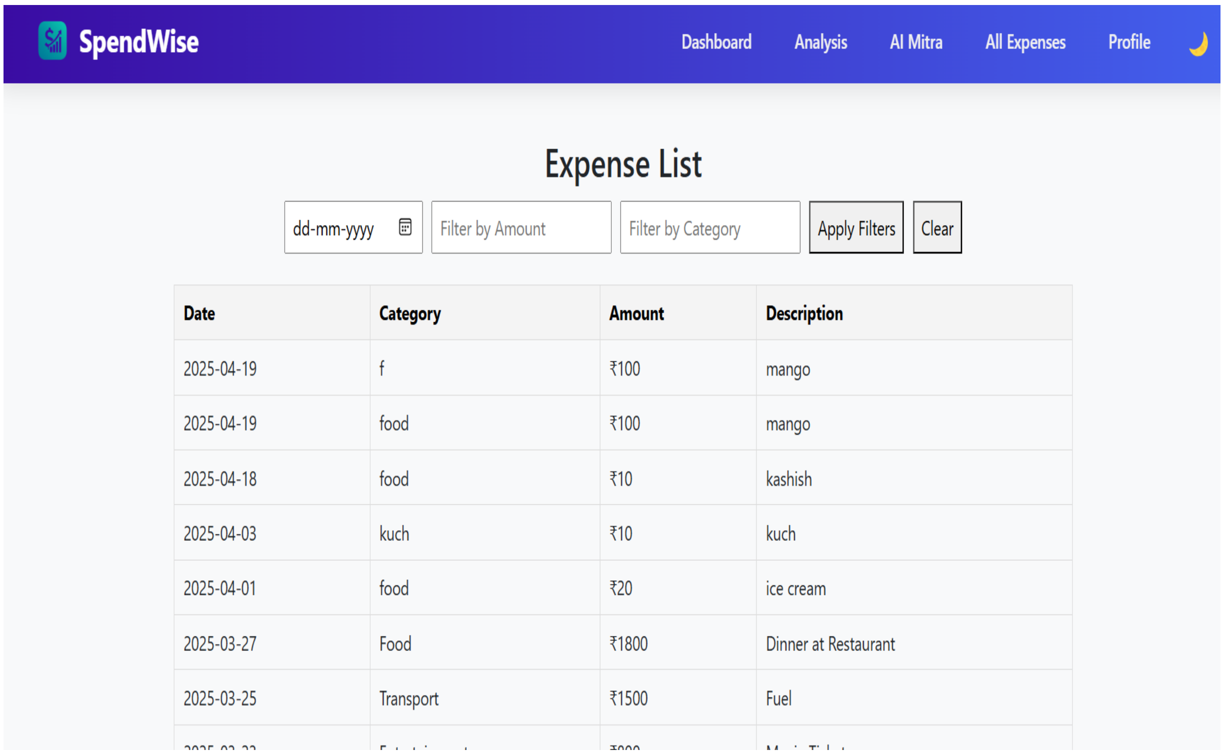


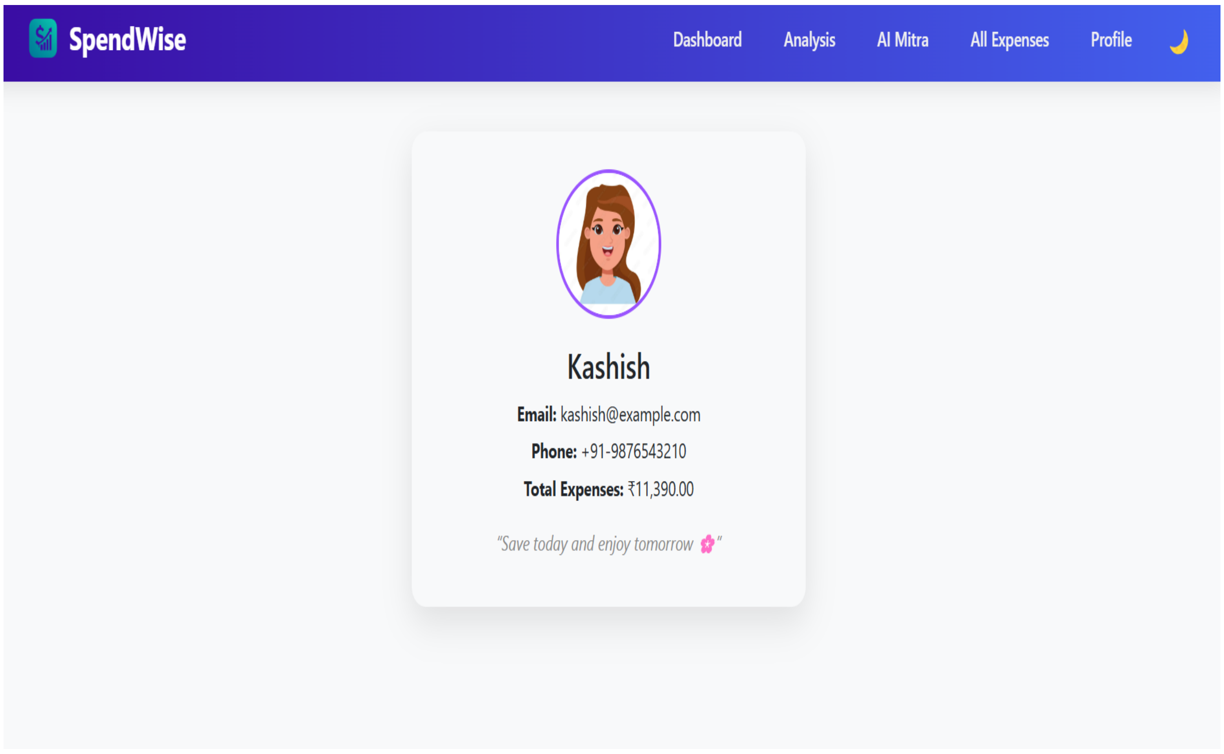
Figure 7.6: AI-Mitra



Expense List

Date	Category	Amount	Description
2025-04-19	f	₹100	mango
2025-04-19	food	₹100	mango
2025-04-18	food	₹10	kashish
2025-04-03	kuch	₹10	kuch
2025-04-01	food	₹20	ice cream
2025-03-27	Food	₹1800	Dinner at Restaurant
2025-03-25	Transport	₹1500	Fuel

Figure 7.7: All Expenses with filters



Kashish

Email: kashish@example.com

Phone: +91-9876543210

Total Expenses: ₹11,390.00

"Save today and enjoy tomorrow 🌸"

Figure 7.8: Profile

CHAPTER 8

Project Summary and Conclusions

8.1 Conclusion

The development of the Expense Tracker application has addressed the key challenge of managing personal expenses efficiently, offering users a streamlined way to track, categorize, and analyze their daily spending. The platform provides essential features like expense entry, category management, filtering by date or category, and secure data storage, making it an invaluable tool for individuals seeking better financial management.

Throughout the project, I successfully implemented core functionalities such as user authentication, expense tracking, category management, and real-time data persistence. These features ensure that users can seamlessly log and monitor their expenses, helping them stay within their budget and make informed financial decisions.

The application leverages modern technologies like React.js, Tailwind CSS for responsive design, and Spring Boot with MySQL for backend management, ensuring both scalability and a smooth user experience. Comprehensive testing validated its functionality, usability, and performance, confirming the application's reliability.

In conclusion, the Expense Tracker meets its primary goals of simplifying expense management while providing a solid foundation for future enhancements. Potential future features include integration with external financial APIs, AI-powered budget suggestions, multi-user support, and mobile app compatibility. This project lays the groundwork for a versatile, scalable tool that can support users in achieving their financial goals.

CHAPTER 9

Future Scope

The current implementation of the Expense Tracker application provides users with an efficient way to manage their finances. However, to enhance its functionality, scalability, and user experience, several key improvements and new features can be integrated in future iterations of the platform. These enhancements will increase user engagement, broaden its usage, and provide additional tools for more comprehensive financial management.

- **AI-Based Expense Categorization:** Future versions can include AI algorithms that automatically categorize expenses based on user behavior, transaction details, and spending patterns. This will reduce manual input and improve accuracy over time.
- **Mobile Application Development:** Developing native mobile apps for Android and iOS would allow users to track their expenses on-the-go, providing a seamless experience across devices and ensuring higher user engagement.
- **Cloud Sync and Backup:** Integrating cloud synchronization will enable users to access their data from multiple devices and ensure data security with regular backups. This would also support real-time updates across all platforms.
- **Integration with Banking APIs:** Future versions can integrate with banking APIs to automatically import transaction data from user bank accounts, reducing manual data entry and providing a more seamless financial management experience.
- **Real-Time Financial Insights:** A real-time analytics dashboard could provide users with insights into their spending trends, budget adherence, and financial

goals. Users can track their progress and receive alerts if they are overspending in any category.

- **Bill Reminders and Payment Tracking:** The addition of bill reminders and payment tracking would help users stay on top of recurring payments such as utility bills, subscriptions, and loan repayments, helping prevent missed payments.
- **Multi-Currency Support:** As the app scales globally, incorporating multi-currency support would make it more accessible for users worldwide, allowing them to track their expenses in different currencies.
- **Financial Goal Setting and Tracking:** A module for setting financial goals (e.g., saving for a vacation, buying a gadget) could be added. Users could track their progress and receive recommendations to reach their goals more effectively.
- **Collaborative Expense Tracking:** Implement a feature allowing users to share expenses with family members or friends, enabling easy tracking of shared expenses and settlements, especially for group activities or joint projects.
- **Investment Portfolio Tracking:** Adding support for tracking investments, such as stocks, mutual funds, or cryptocurrencies, will allow users to have a more holistic view of their financial status and performance.
- **Data Encryption and Security Enhancements:** As user data will be increasingly sensitive, adopting end-to-end encryption and implementing two-factor authentication (2FA) for additional security will further safeguard user information.
- **Premium Features and Monetization:** A freemium model could be introduced, where basic features remain free, and premium features such as advanced analytics, multi-user collaboration, and priority customer support are available as part of a paid plan.

References

[1] *Spring Boot Documentation*, Available at: `https://spring.io/projects/spring-boot` [Accessed: March 2025].

[2] *Angular Documentation*, Available at: `https://angular.io/docs` [Accessed: March 2025].

[3] *Tailwind CSS Documentation*, Available at: `https://tailwindcss.com/docs` [Accessed: March 2025].

[4] *MySQL Official Documentation*, Available at: `https://dev.mysql.com/doc/` [Accessed: March 2025].

[5] *Firebase Documentation*, Available at: `https://firebase.google.com/docs` [Accessed: March 2025].

[6] *How to Build an Expense Tracker Web Application*, Available at: `https://www.freecodecamp.org/news/how-to-build-an-expense-tracker-web-application/` [Accessed: March 2025].

[7] *Research Methodology: Definitions and Concepts*, Available at: `http://research-methodology.net/` [Accessed: March 2025].