Phainance – Final Project Report

Submitted by:

Aashirwad Asmit Pradhan

24SUPMCAGL001
MCA – Sapthagiri NPS University

Chapter 1

Introduction

In today's fast-paced world, effective financial management has become an essential part of personal stability and growth. The ability to record, track, and analyze expenses allows individuals to make informed decisions about their spending habits and financial goals.

Phainance is a full-stack expense management system designed to help users efficiently monitor their daily expenses, manage monthly budgets, and visualize financial patterns through data-driven analytics. It combines modern technologies and a user-friendly interface to provide a secure and interactive financial tracking experience.

As the sole developer of this project, I, **Aashirwad Pradhan**, conceptualized and implemented Phainance as part of my MCA coursework at **Sapthagiri NPS University**. The project focuses on the development of a responsive and secure full-stack application using React for the frontend, Spring Boot for the backend, and MySQL for database management.

The goal behind Phainance is to simplify personal financial management — bridging the gap between complex tools and accessible design — ensuring usability, performance, and data integrity.

Chapter 2

Problem Statement

Many individuals struggle to maintain an organized record of their expenses and budgets. Existing tools are often complex, expensive, or lack customization.
Common challenges include:

- Lack of visual insights into spending habits
- Difficulty managing multiple categories efficiently
- No alert system for overspending
- Complicated user interfaces
- Limited options for secure self-hosted solutions

There was a need for a simple yet powerful platform to manage personal finances — one that is accessible, interactive, and visually informative.

Chapter 3

Proposed Solution

Phainance is designed as a comprehensive solution that simplifies expense management and budgeting through automation, visualization, and clean design.
It enables users to:

- Log in securely with JWT-based authentication
- Add, edit, and delete expenses by category and date
- Set monthly budgets and view remaining balance
- Access analytics via pie and line charts
- Switch between light and dark themes
- Receive real-time notifications for all key actions

This approach empowers users to control and analyze their finances with clarity and ease.

Chapter 4

Technology Stack and Decision Rationale

Layer	Technology	Rationale
Frontend	React (Vite)	Modular and high-performance component-based framework
Backend	Spring Boot	Secure and scalable REST API framework
Database	MySQL	Reliable and easy to integrate with Spring
Authentication	JWT	Ensures secure session management
Charts	Recharts	Interactive and responsive visualization library
Notifications	React Toastify	For rounded, real-time notifications
Styling	CSS3 + Responsive Layout	Ensures modern and adaptive design

These technologies were chosen to create a responsive, efficient, and user-focused application that supports scalability and performance.

Chapter 5

Database Design

The database for **Phainance** is implemented in **MySQL**. It follows a normalized structure with relationships between users, expenses, categories, and budgets.

Database Schema Overview

Table	Purpose
users	Stores user credentials and authentication details
categories	Predefined list of expense categories
expenses	Records all user expense entries
budgets	Manages user-set monthly budget limits

Schema Structure (SQL)

```
CREATE DATABASE IF NOT EXISTS expense_tracker;
USE expense_tracker;

CREATE TABLE IF NOT EXISTS users (
 id BIGINT AUTO_INCREMENT PRIMARY KEY,
 username VARCHAR(100) NOT NULL UNIQUE,
 email VARCHAR(150) NOT NULL UNIQUE,
 password VARCHAR(255) NOT NULL
);

CREATE TABLE IF NOT EXISTS categories (
 id BIGINT AUTO_INCREMENT PRIMARY KEY,
 name VARCHAR(100) NOT NULL UNIQUE
);

CREATE TABLE IF NOT EXISTS expenses (
 id BIGINT AUTO_INCREMENT PRIMARY KEY,
 amount DOUBLE,
 description VARCHAR(255),
 date DATE,
 category_id BIGINT,
 FOREIGN KEY (category_id) REFERENCES categories(id),
 user_id BIGINT,
 FOREIGN KEY (user_id) REFERENCES users(id)
);
```

```
CREATE TABLE IF NOT EXISTS budgets (
 id BIGINT AUTO_INCREMENT PRIMARY KEY,
 month VARCHAR(20) NOT NULL,
 limit_amount DOUBLE,
 user_id BIGINT,
 FOREIGN KEY (user_id) REFERENCES users(id)
);
```

Relationships

- **One User → Many Expenses**
- **One User → Many Budgets**
- **One Category → Many Expenses**

This ensures **data consistency**, prevents duplication, and allows easy aggregation for analytics.

Sample Data (Categories Table)

id	name
1	Food
2	Rent
3	Utilities
4	Entertainment
5	Other

Key Characteristics

- Referential integrity enforced via **foreign keys**.
- **Unique constraints** on usernames and category names.
- Designed for **monthly analytics and expense tracking**.
- Scalable for multi-user operations.

Chapter 6

System Workflow

The overall workflow of **Phainance** follows a seamless full-stack data flow from the user interface to the backend and database. Each layer is designed to ensure security, scalability, and responsiveness.

Frontend Layer (React)

The frontend is built using **React (Vite)**. It manages all user interactions through forms, charts, and dynamic components.

- The user interacts with components such as **AddExpense**, **ExpenseList**, **BudgetForm**, and **Dashboard**.
- When a user submits data (for example, adding a new expense), it triggers an **Axios HTTP request** to the backend API.

Backend Layer (Spring Boot)

The backend handles API requests through **Spring Boot REST controllers**.

- Requests from the frontend are received at routes like /api/expenses, /api/budget, /api/auth/login, etc.
- The **controller layer** validates and processes the request, then calls the **service layer** for business logic.
- The **service layer** uses **repositories** (DAO) to interact with the database using Spring Data JPA.

Example flow for adding an expense:

1. **Frontend:** Sends a POST request with expense details via Axios.
2. **Controller:** Receives the request (ExpenseController), extracts user info from JWT, and maps data into an Expense object.
3. **Service Layer:** Calls ExpenseRepository.save() to persist the record.
4. **Repository:** Spring JPA translates this into an SQL INSERT query.
5. **Database:** MySQL stores the record in the expenses table.
6. **Response:** The saved record is returned as JSON to the frontend and displayed on the dashboard.

Data Flow Summary

User → React (Form Input)

↓

Axios Request (JSON)

↓

Spring Boot Controller (ExpenseController)

↓

ExpenseService → ExpenseRepository

↓

MySQL Database (Tables: users, expenses, categories, budgets)

↓

Response → React (Updated UI + Notification)

Authentication and Security

The authentication system uses **JWT (JSON Web Tokens)** for secure access:

- When a user logs in, the backend issues a token.
- The token is stored in localStorage on the frontend.
- All subsequent requests include this token in the Authorization header.
- The backend validates the token before allowing any data access.

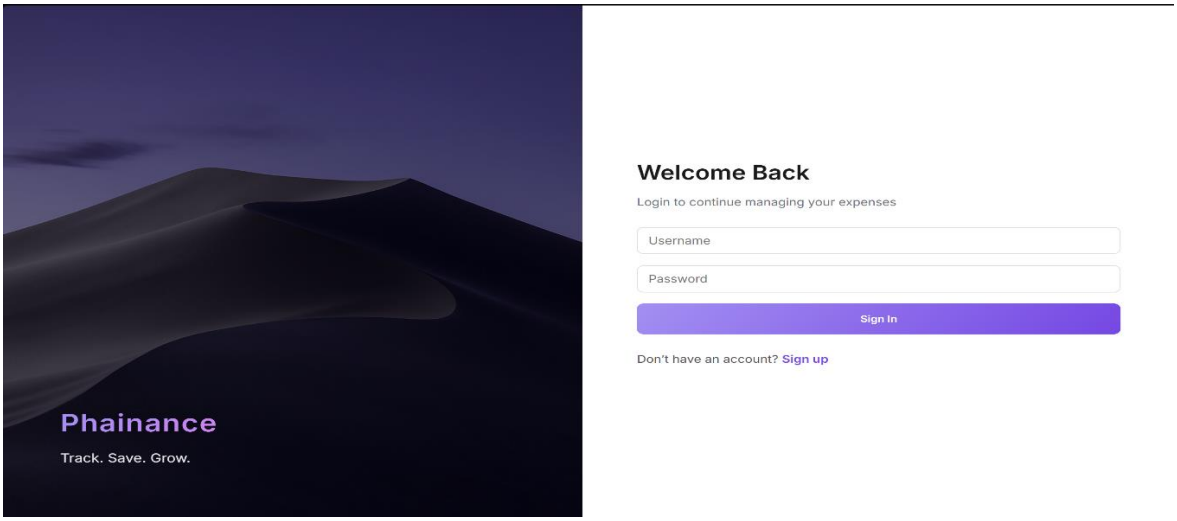
This ensures that only authenticated users can perform CRUD operations on their data.

Chapter 7

Modules Implemented

1. Authentication Module

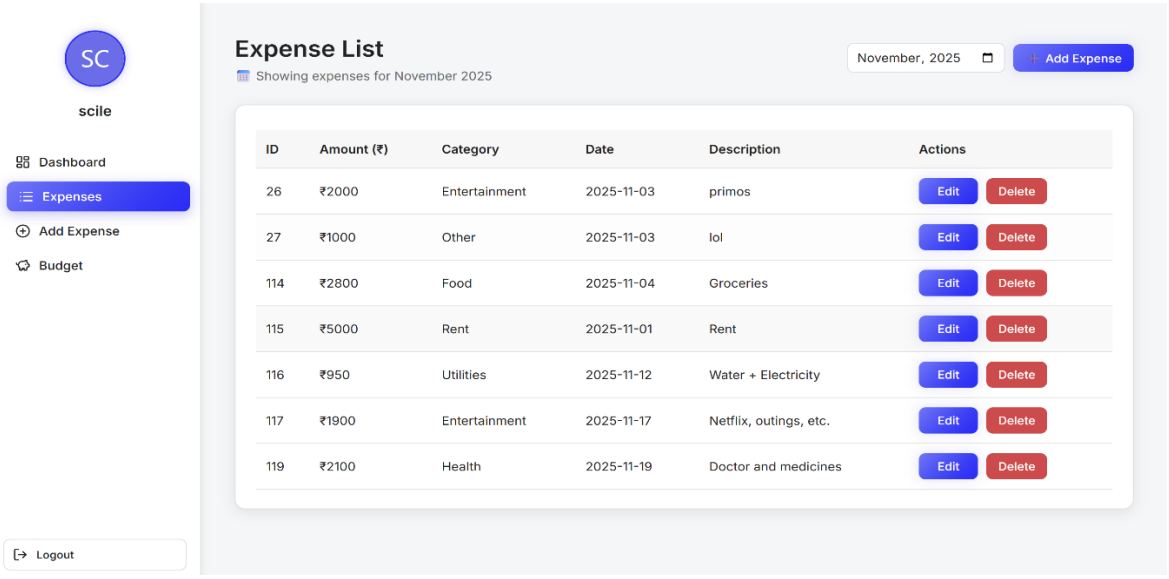
Handles secure login and signup using Spring Security and JWT tokens.
It ensures that only authorized users can access the dashboard and expense management features.



---

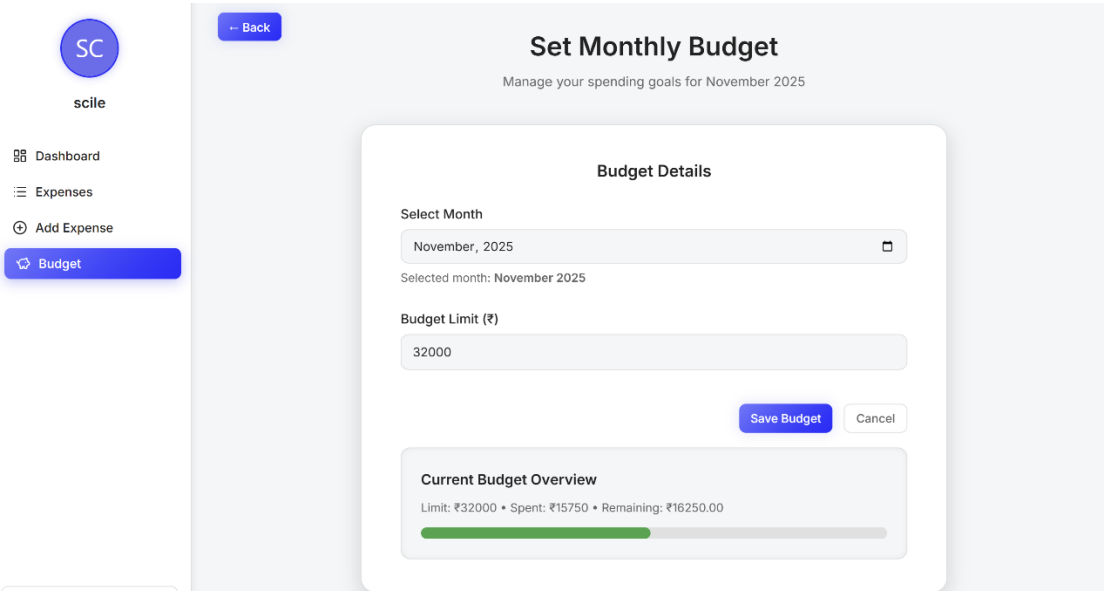
2. Expense Management Module

Allows users to add, edit, and delete expenses directly from the dashboard.
Inline editing and smart category selection make it efficient for quick updates.



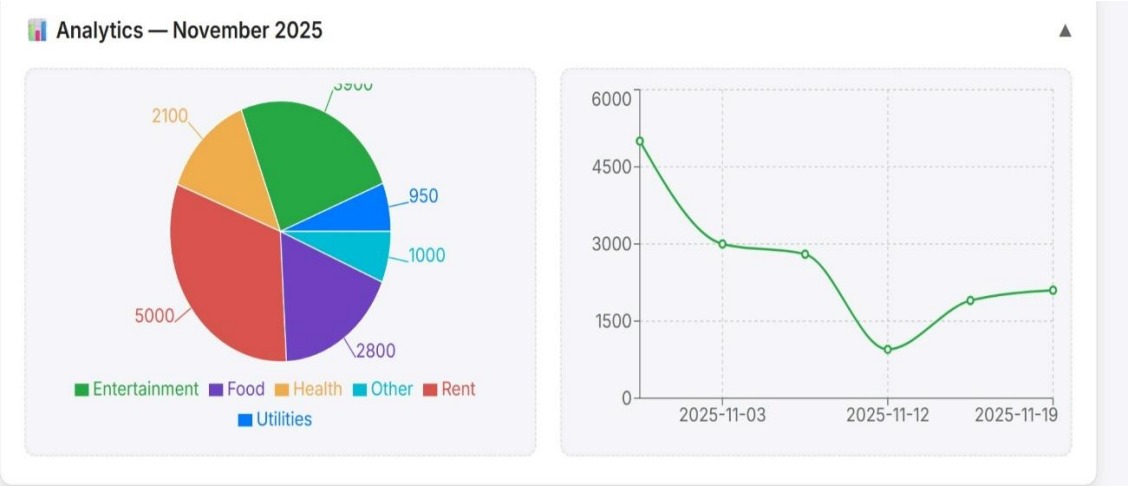
3. Budget Module

Lets users set and modify monthly budget limits.
Displays total spent, remaining balance, and visual indicators for budget health.



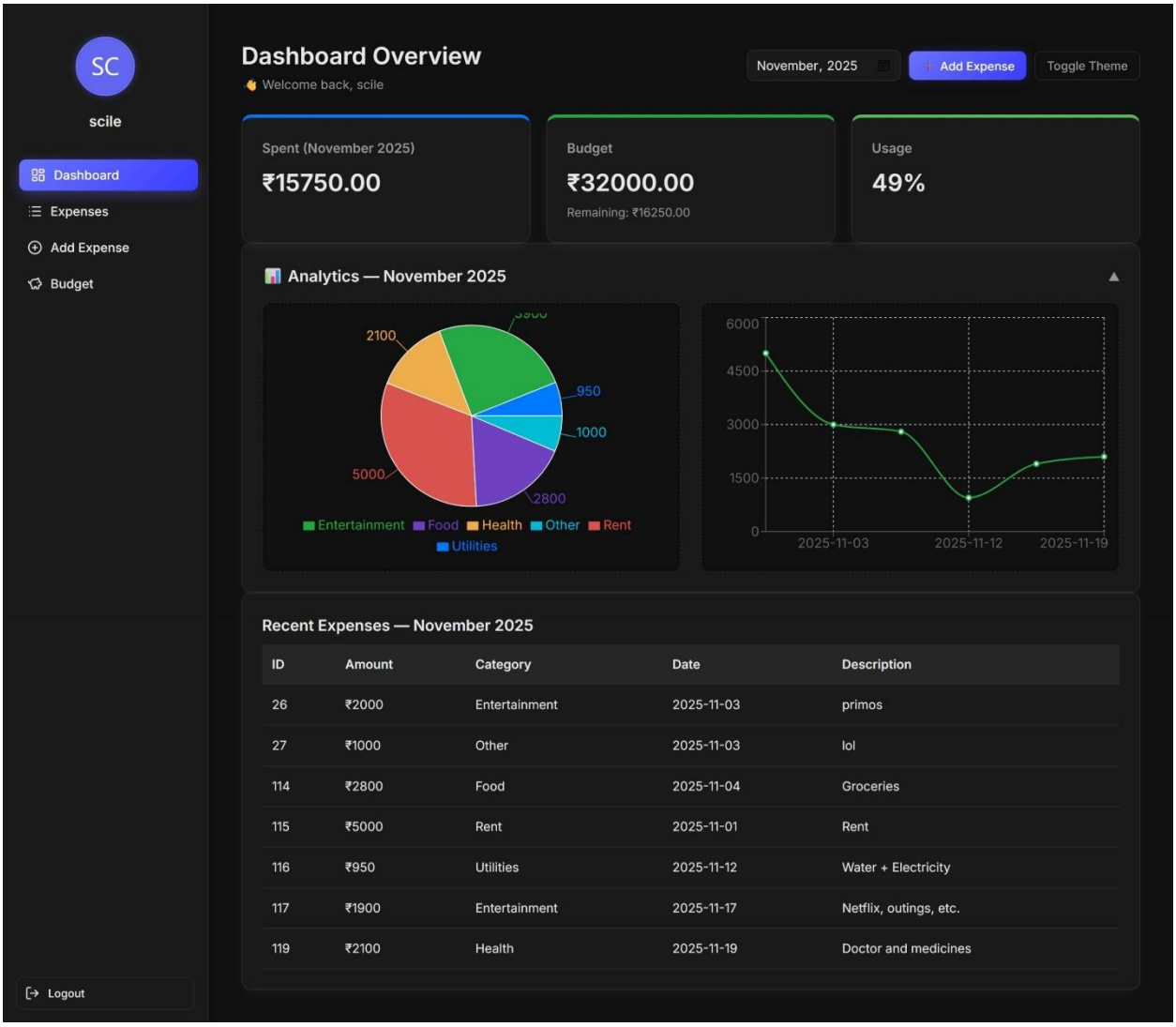
4. Analytics Module

Provides visual insights using **Pie Charts** (category-wise expenses) and **Line Charts** (daily spending trends).
Helps users analyze patterns and make better financial decisions.



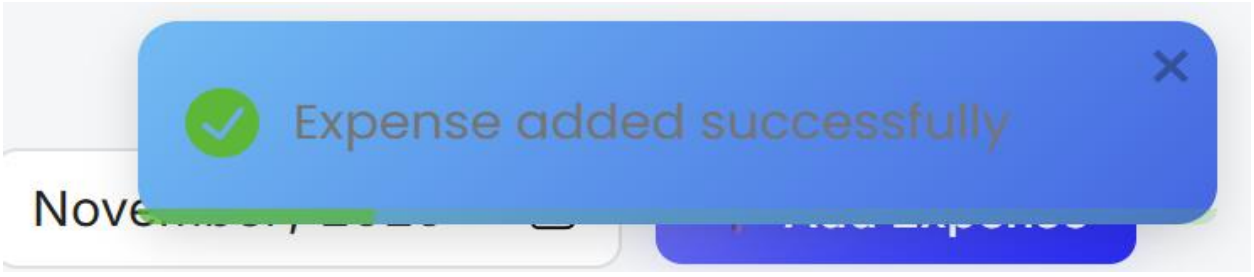
5. User Interface and Theme Module

Implements a responsive layout with sidebar navigation.
Includes dark and light mode support, ensuring consistent accessibility and design appeal.



6. Notification Module

Integrates **React Toastify** to display success and error alerts for all user operations.
Rounded notifications enhance feedback clarity and aesthetics.



Chapter 8

Future Enhancements

- **Predictive Expense Analysis:** Implement machine learning for future expense forecasting.
- **Data Export:** Generate PDF and Excel summaries for financial reports.
- **Collaborative Budgets:** Allow multiple users to manage shared budgets.
- **Smart Recommendations:** Suggest saving or investment plans based on patterns.
- **Mobile App Integration:** Sync data between web and mobile clients using APIs.

Chapter 9

Conclusion

The development of **Phainance** demonstrates the complete design and deployment of a modern, scalable financial management system.

This project enhanced my technical expertise in **React**, **Spring Boot**, **MySQL**, and **RESTful API development**, while also improving my understanding of **UI/UX principles** and **system design**.

Phainance effectively simplifies the process of expense tracking and budgeting, offering users clarity, control, and insights over their finances. It stands as a strong foundation for future improvements, combining both technical excellence and user-centric design.