SDJ INTERNATIONAL COLLEGE - VESU

Affiliated To: Veer Narmad South Gujarat University, Surat

Project Report

AS A PARTIAL REQUIREMENT FOR THE DEGREE OF

BACHELOR OF COMPUTER APPLICATIONS (BCA)

BCA Sem V
A.Y. 2025-26

Project Report On Smart Expense Tracker

by

Exam No.	Roll No.	Name of Student
	158	Kalkani Prince Shaileshbhai
	208	Mangukiya Darsh Sanjaybhai
	102	Goti Neet Maheshbhai
	138	Jasoliya Manthan Chakrakantbhai

Project Guide by:
Prof. Vandana Prajapati

Acknowledgement

The success and outcome of this project required a lot of guidance and assistance from many people, and we are extremely fortunate to have received this throughout the completion of our project work. Whatever we have done is only due to their guidance and assistance.

We would not forget to thank Principal Dr. Aditi Bhatt, IQAC coordinator and trust representative Dr. Vaibhav Desai, In-charge of PG Courses Dr. Vimal Vaiwala, In-charge of UG and Head of the UG Department Prof. Nainesh Gathiyawala and my project guide _____ and all other Assistant professors of SDJ International College Vesu, who took keen interest in our project work and guided us all along, till the completion of our project work by providing all the necessary information for developing a good system.

We are extremely grateful to her/him for providing such nice support and guidance, though she/he had a busy schedule managing the college affairs.
We are thankful and fortunate enough to get support and guidance from all the Teaching staff of the IT Department, which helped us in completing our project work. Also, we would like to extend our sincere regards to all the non-teaching staff of the IT Department for their timely support.

Kalkani Prince Shaileshbhai ( )
Mangukiya Darsh Sanjaybhai ( )
Goti Neet Maheshbhai ( )
Jasoliya Manthan Chandrakantbhai ( )

INDEX

Table of Contents

Summary

The Expense Tracker is a web-based application designed to address the common challenge of managing personal finances. This project was developed in two parallel versions—one using a traditional **PHP server-side architecture** and another as a modern **React.js Single Page Application (SPA)**—to compare and understand different web development paradigms. Both versions provide a simple, yet powerful platform for users to monitor their expenditure. The system features a dual-module architecture, catering to both individual users and an administrator. Regular users can register, log in, and manage their daily expenses by categorizing them. The administrator has a holistic view of the system, with the ability to manage users and view comprehensive reports. This report details the objectives, scope, design, implementation, and future enhancements for both versions of the application.

Chapter 1: Introduction

1.1 Project Overview

This project is a web-based **Expense Tracker Application**. The main goal of this application is to provide users with a simple and effective tool to record and manage their daily expenses. The application is designed to be intuitive and user-friendly, ensuring that even users with minimal technical knowledge can use it effectively.

The application is divided into two main modules:

1. **User Panel:** This is for individual users where they can create an account, log in, and manage their own personal expenses.
2. **Admin Panel:** This is a special panel for an administrator who can view and manage the data of all registered users.

1.2 Problem Statement & Solution

- **The Problem:** Manually tracking expenses is inefficient, time-consuming, and prone to errors. It's hard to get a clear picture of spending habits, which makes it difficult to save money and create an effective budget.
- **Our Solution:** We have developed a centralized and easy-to-use web application. It provides a secure platform where users can register, log in, add expenses with categories, and see detailed reports.

Chapter 2: System Requirements

2.1 Software Requirements

- **Operating System:** Windows, Linux, or macOS
- **Web Server:** Apache (for PHP version)
- **Backend/Frontend:** PHP 7.4+, Node.js (for React version)
- **Database:** MySQL / MariaDB
- **Web Browser:** Google Chrome, Mozilla Firefox, or any modern browser.

2.2 Hardware Requirements

- **Processor:** Intel Core i3 or equivalent
- **RAM:** 4 GB or more
- **Hard Disk:** 10 GB of free space

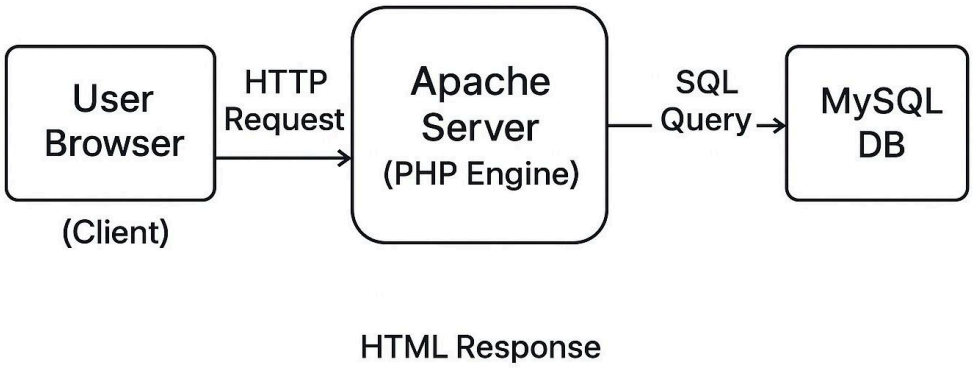
Chapter 3: System Design & Architecture

3.1 PHP Version Architecture (Client-Server)

The PHP version follows a traditional multi-page application (MPA) architecture.

- **Client:** The user's web browser sends a request for a page.
- **Server:** The Apache server receives the request. The PHP engine processes the script, interacts with the MySQL database, generates a complete HTML page, and sends it back to the client.
- Every action (like submitting a form or clicking a link) results in a new page request to the server.

Diagram :-

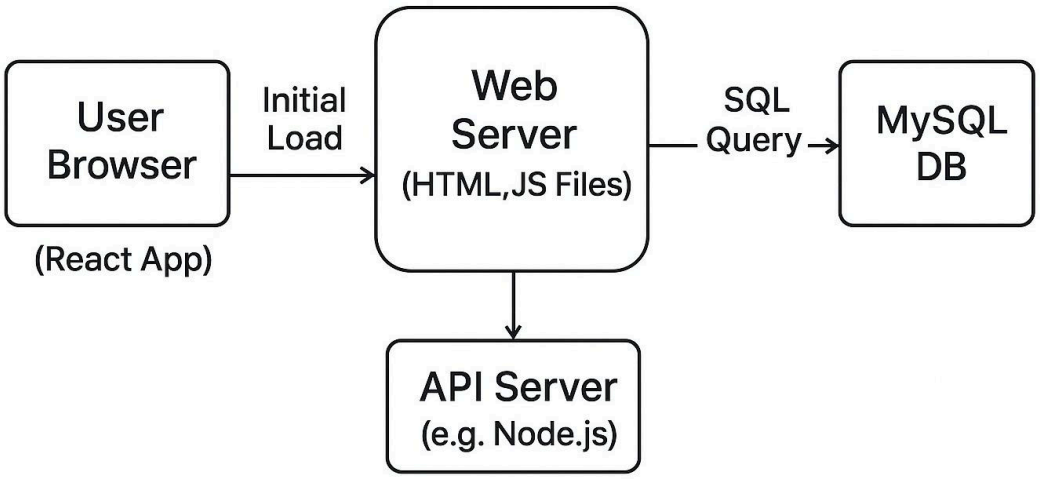


3.2 React.js Version Architecture (Single Page Application)

The React.js version is a modern Single Page Application (SPA).

- **Initial Load:** The server sends a single HTML file along with the JavaScript (React) code.
- **Client-Side Rendering:** React takes control of the page in the browser. When the user interacts with the app, React dynamically updates only the necessary parts of the page without a full page reload.
- **State Management:** Data is managed on the client-side. In a full-stack version, it would communicate with a backend API to fetch and send data.

Diagram :-



3.3 Data Flow Diagram (DFD)

This diagram is applicable to both versions as it represents the logical flow of data.

- **Level 0:** A user interacts with the Expense Tracker System. The system interacts with the Database.

- **Level 1 (User):** User logs in/registers, manages expenses, and views reports. Data flows between the user, the system, and the database tables (`user`, `expense`).
- **Level 1 (Admin):** Admin logs in, views all data, and manages users.

Chapter 4: Technology Stack

4.1 PHP Version (Server-Side Rendering)

- **Frontend:** HTML, Tailwind CSS, Vanilla JavaScript (for small interactions).
- **Backend:** PHP (for all server-side logic and database communication).
- **Database:** MySQL.
- **Charts:** Chart.js.

4.2 React.js Version (Client-Side Rendering)

- **Frontend:** React.js (for building the user interface), Tailwind CSS.
- **State Management:** React Hooks (`useState`, `useContext`).
- **Charts:** Chart.js.
- **Backend (Simulated):** All data is currently stored and managed within the React application's state for this prototype.

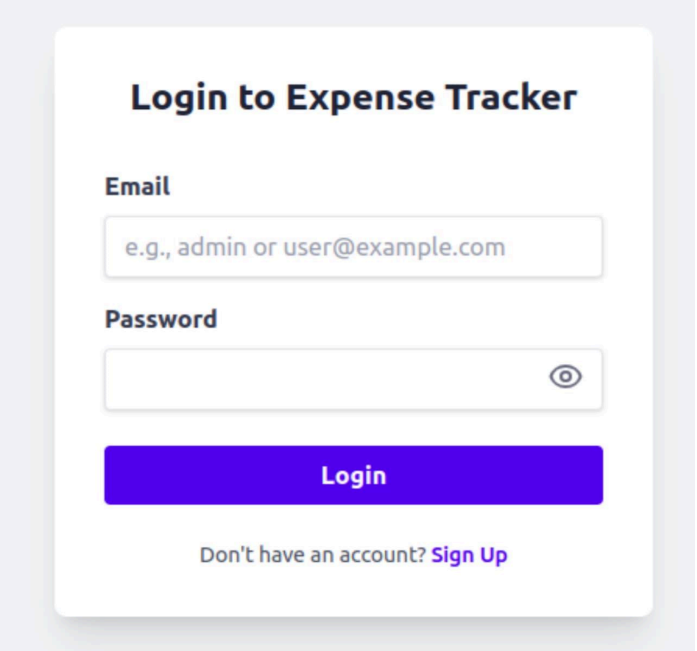
Chapter 5: Implementation & Modules

This chapter provides a look at the key features, which are functionally identical in both the PHP and React versions.

5.1 Login & Authentication
5.2 New User Registration
5.3 The User Dashboard
5.4 Adding & Managing Expenses
5.5 The Admin Panel

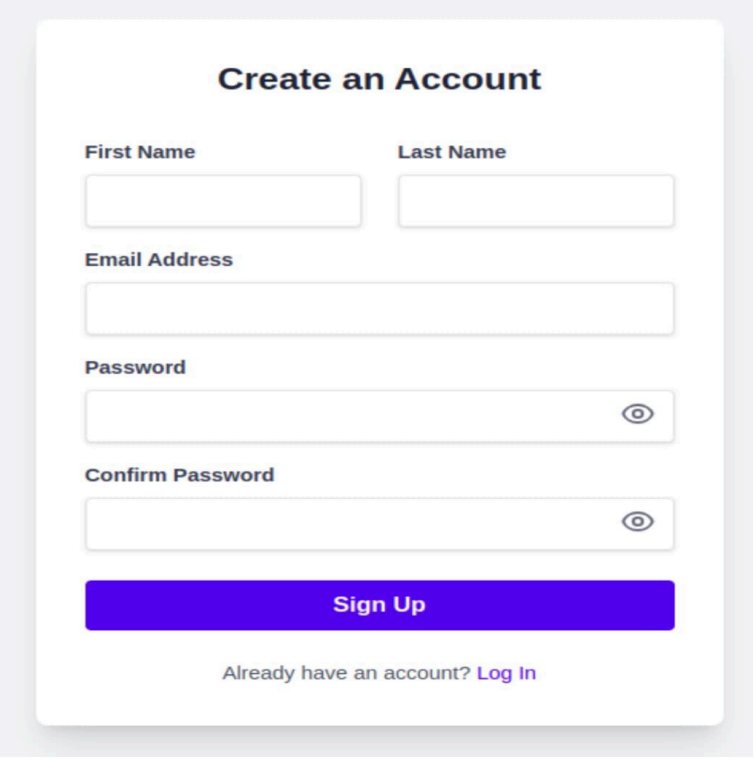
5.1 Login & Authentication

A secure entry point for the application allowing role-based access.



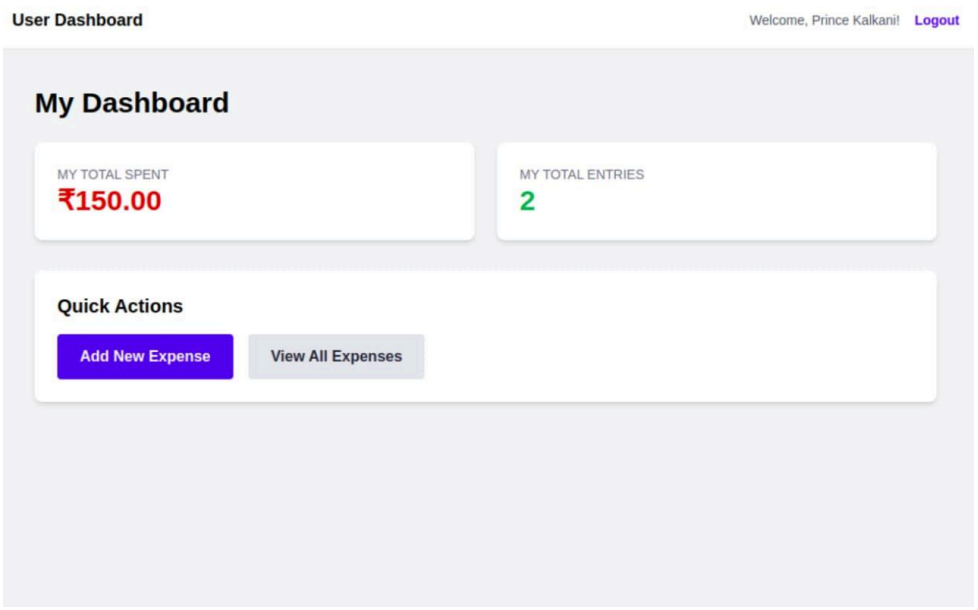
5.2 New User Registration

A simple form for new users to create an account.



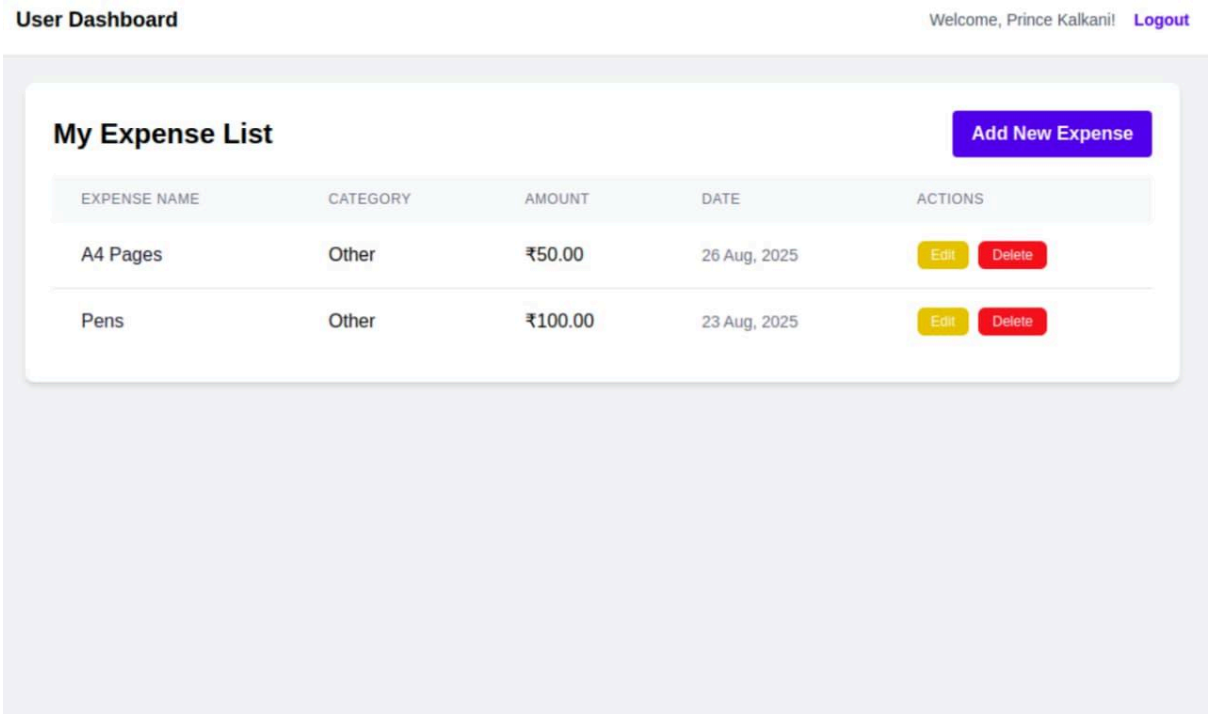
5.3 The User Dashboard

The main landing page for a user, showing summary cards and quick actions.



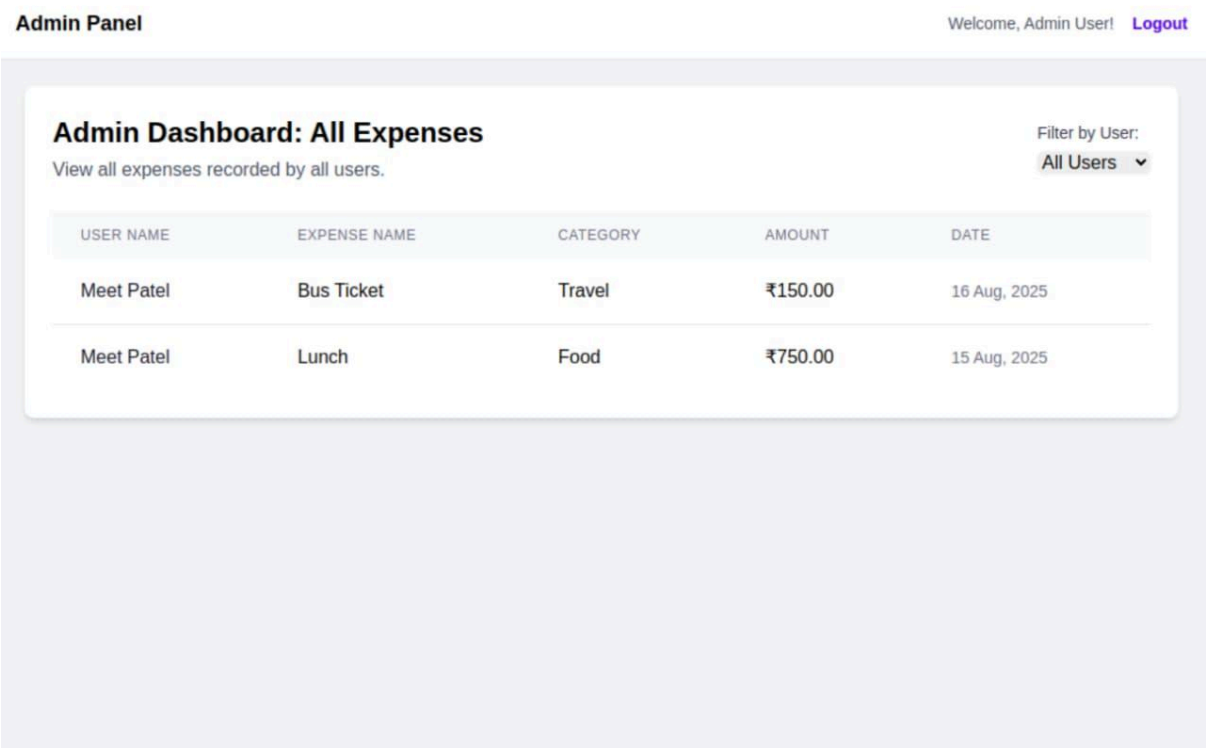
5.4 Adding & Managing Expenses

The core functionality where a user can perform CRUD (Create, Read, Update, Delete) operations on their expenses.

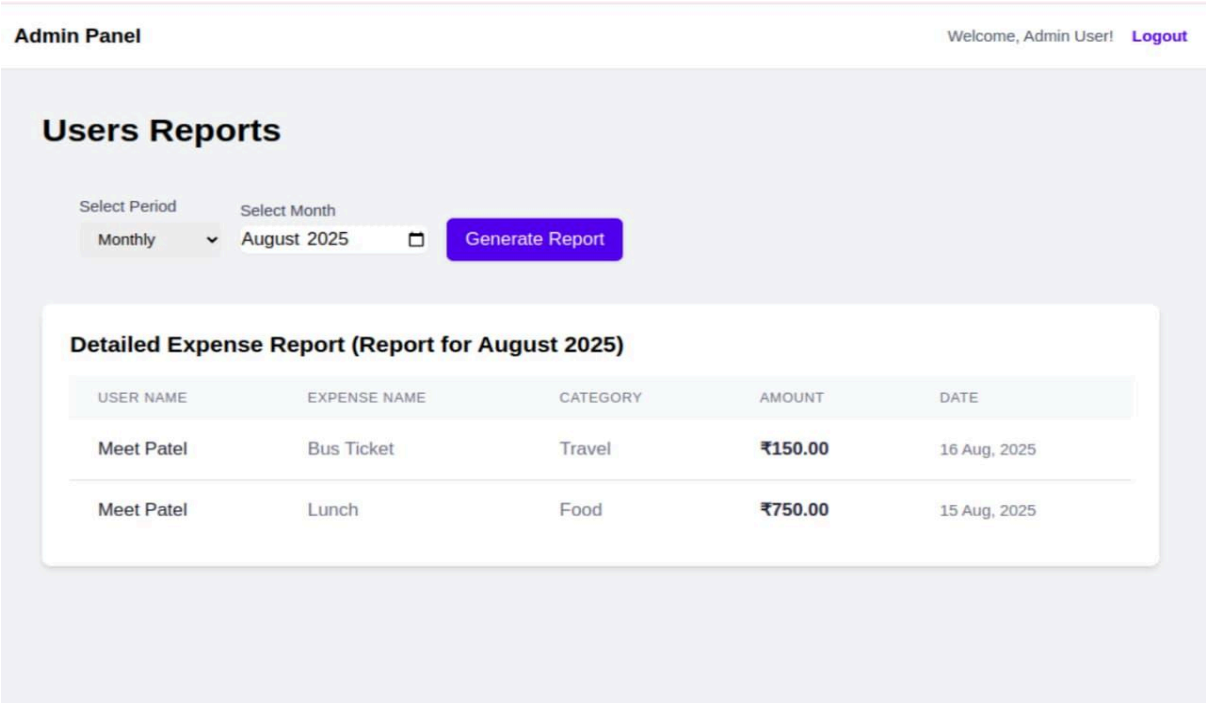


5.5 The Admin Panel

A powerful overview of the entire system for the administrator, including viewing all user expenses, visual reports with charts, and user management.



Category Reports



Chapter 6: Testing

To ensure the application is reliable, robust, and bug-free, a comprehensive testing strategy was employed. We focused on different levels of testing to cover all aspects of the application, from individual functions to the complete user workflow.

- **Unit Testing:** In this phase, we tested the smallest individual parts of the code. For example, in the PHP version, we tested the database connection function separately to ensure it connects successfully. We also tested individual functions like the password hashing function during registration to confirm that it works as expected.
- **Integration Testing:** After unit testing, we tested how different modules work together. A key test case was the **Login-to-Dashboard flow**. We tested if a newly registered user could immediately log in and be redirected to the correct user dashboard, and not the admin panel. We also checked if an expense added by a user was correctly reflected in the admin's dashboard.
- **Validation Testing:** This was crucial for ensuring data integrity. We tested all input forms with various kinds of data:
- **Empty Submissions:** Checked if the system shows appropriate error messages when a user tries to submit an empty form.
- **Invalid Data Formats:** Tested the registration form with an invalid email format.
- **Data Type Mismatches:** Tried to enter text in the "Amount" field, which should only accept numbers. The system successfully handled all these cases by displaying user-friendly error messages.
- **Browser Compatibility:** The application was manually tested on the latest versions of Google Chrome and Mozilla Firefox to ensure that the layout, functionality, and styling are consistent across different browsers. We confirmed that the Tailwind CSS classes rendered correctly and the JavaScript for the charts and password toggles worked flawlessly on both platforms.

- **Testing Outcome:** All test cases were passed successfully. The application was found to be stable, secure, and performed as per the requirements. No critical bugs were found before deployment.

Chapter 7: Conclusion & Future Scope

7.1 Conclusion

This project was a great learning experience. Building two versions using PHP and React.js provided deep insights into both server-side and client-side rendering paradigms. The final applications are practical and useful tools that can genuinely help people manage their finances more effectively. Both versions successfully achieve the project's objectives.

7.2 Future Scope

This project has a strong foundation and offers many possibilities for future enhancements to make it even more powerful and feature-rich.

- **Budgeting Feature:** A key future enhancement would be to allow users to set a monthly budget for different categories (e.g., a budget for 'Food'). The system could then track their spending against this budget and send them alerts or notifications when they are close to exceeding it. This would move the application from a simple tracker to a proactive financial planning tool.
- **API for React Version:** Currently, the React version uses simulated data. The next logical step is to develop a proper backend API (e.g., using Node.js/Express or a PHP-based framework like Laravel). This would allow the React application to connect to a real database, making it a fully functional, full-stack application.
- **Advanced Reports & Export:** We can enhance the reporting feature by adding more types of charts, such as bar charts to show monthly spending trends. A highly requested feature would be the

ability for users to **export their reports as PDF or Excel files** for their personal records or for sharing.

- **Mobile App:** To provide users with easier access to their financial data on the go, a dedicated mobile application could be developed. This could be done using **React Native**, which would allow us to reuse some of the logic from our existing React.js web application.

8. References

Website :-

PHP Official Documentation: https://www.php.net

React.js Official Documentation: https://react.dev

MySQL Official Documentation: https://www.mysql.com

Tailwind CSS: https://tailwindcss.com

Chart.js for Data Visualization: https://www.chartjs.org

Stack Overflow Community: https://stackoverflow.com