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
	138	Jasoliya Manthan Chandrakantbhai
	102	Goti Neet Maheshbhai

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 modern frontend application developed using React.js, designed to provide a fast and interactive experience for managing personal finances. This project demonstrates the power of a Single Page Application (SPA) architecture for creating dynamic and user-friendly interfaces. 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 in a fluid interface without page reloads. The administrator has a holistic view of the system, with the ability to manage users and view comprehensive, interactive reports.

The application is built using React.js for the user interface, with state management handled by React Hooks. For this prototype, all data is managed on the client side, simulating a full-stack experience. This report details the project's objectives, scope, system design, implementation, testing, and future enhancements.

Chapter 1: Introduction

1.1 Project Motivation

In today's fast-paced world, tracking personal expenses is often overlooked, leading to poor financial decisions. The lack of a simple and

accessible tool to monitor spending can result in overspending and missed savings opportunities. This project was motivated by the need for a modern, fast, and interactive digital solution that empowers individuals to take control of their finances.

1.2 Problem Statement

To design and develop a web-based application that allows users to register, manage their daily expenses, and view reports, while also providing an admin panel for complete system oversight, using modern frontend technologies like React.js.

1.3 Project Objectives

- To create a secure user registration and login system on the client side.
- To build a reusable component-based UI for managing expenses.
- To provide users with functionality to perform CRUD (Create, Read, Update, Delete) operations on their expenses without page reloads.
- To develop a secure admin panel with privileges to view all user data.
- To generate dynamic and interactive charts for data visualization.

1.4 Scope of the Project

The project is designed for individual users who want to track their personal finances. The admin panel makes it suitable for a small organization or a financial advisor to manage multiple users. The application is a Single Page Application (SPA) accessible through any modern browser.

Chapter 2: System Analysis

2.1 Existing System & Limitations

Currently, many people rely on traditional methods like:

- **Manual Ledgers/Diaries:** Prone to human error, difficult to analyze, and not easily accessible.
- **Spreadsheets (e.g., Excel):** Requires manual data entry, lacks real-time updates, and is not user-friendly for non-technical users. These methods lack the interactivity and speed of modern web applications.

2.2 Proposed System & Advantages

Our proposed React.js application overcomes these limitations by offering:

- **Fast & Interactive UI:** As a Single Page Application, the interface is extremely fast and responsive.

- **Component-Based:** The code is modular and easy to maintain.
- **Real-time Updates:** Changes are reflected instantly on the screen without waiting for the server.
- **Admin Oversight:** A dedicated admin can monitor the entire system's health and user activity.
- **Modern User Experience:** The application provides a smooth, app-like experience in the browser.

2.3 Feasibility Study

- **Technical Feasibility:** The project is technically feasible as it uses React.js, a popular and well-documented JavaScript library.
- **Economic Feasibility:** The development cost is minimal as it relies on open-source software (Node.js, React).
- **Operational Feasibility:** The application is easy to use and requires minimal training for end-users.

Chapter 3: System Requirements

3.1 Functional Requirements

- **User Module:**

    - Users shall be able to register with a unique email.
    - Users shall be able to log in with their credentials.
    - Users shall be able to add a new expense with a name, amount, category, and date.
    - Users shall be able to view a list of all their expenses.
    - Users shall be able to edit or delete an existing expense.
- **Admin Module:**
    - Admin shall have a separate login.
    - Admin shall be able to view all expenses from all users.
    - Admin shall be able to view a list of all registered users.
    - Admin shall be able to delete any user account.
    - Admin shall be able to view visual reports (charts) of expenses.

3.2 Non-Functional Requirements

- **Performance:** The application should be highly responsive with minimal loading times between views.
- **Usability:** The interface should be intuitive and easy to navigate.

- **Maintainability:** The component-based structure should make the code easy to update and debug.

3.3 Hardware & Software Requirements

- **Development Environment:**
  - Code Editor: VS Code
  - Runtime: Node.js 14+
- **Client Side:**
  - Any modern web browser (Chrome, Firefox, Safari).

Chapter 4: System Design & Architecture

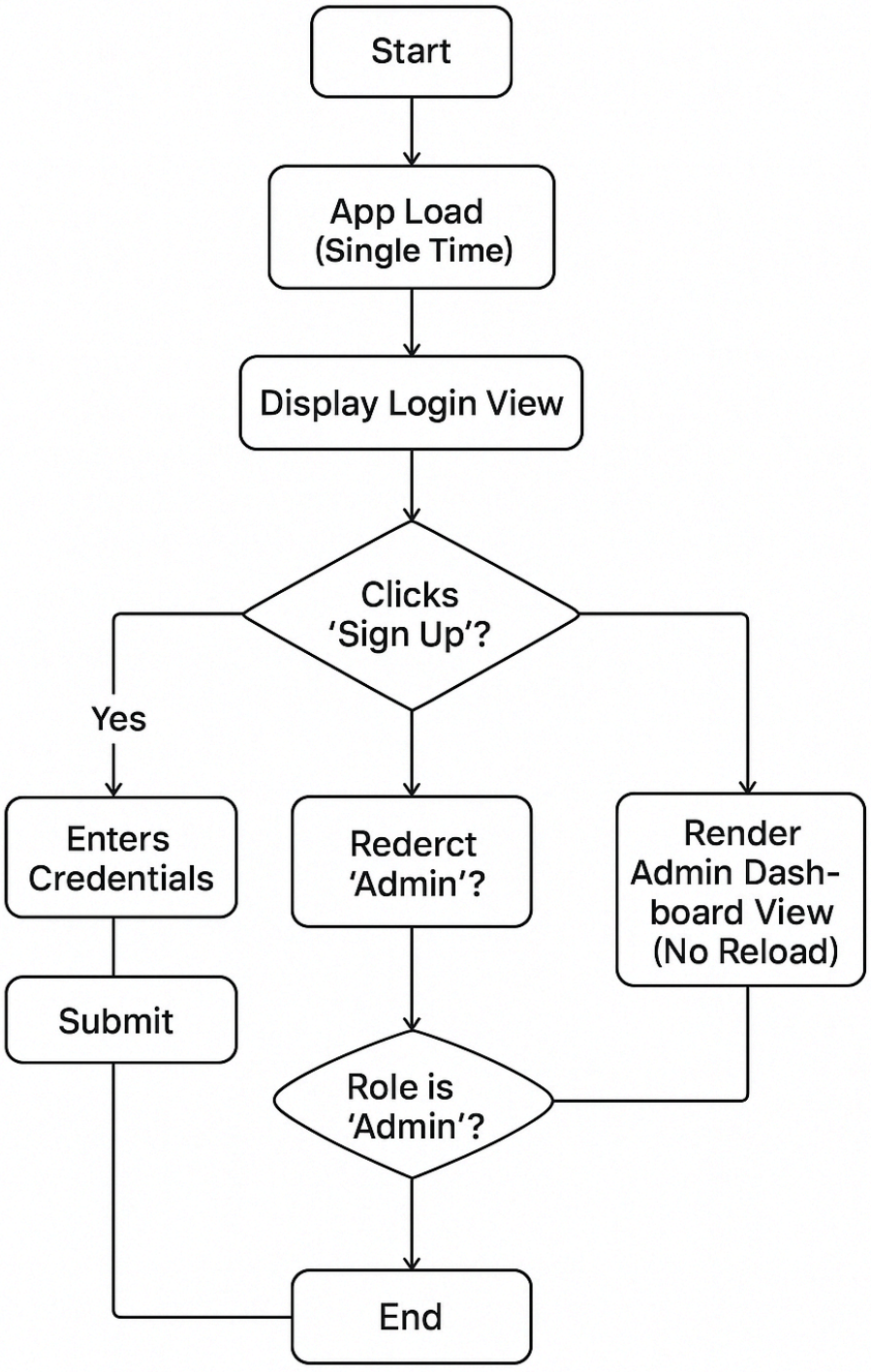
4.1 System Architecture (SPA)

This application is built as a **Single Page Application (SPA)**.

- **Flow:** When a user first visits the site, the server sends a single HTML file along with all the necessary JavaScript (the React app). After this initial load, the application runs entirely in the user's browser. When the user clicks a link, React doesn't request a new page from the server; instead, it dynamically updates the content of the current page. This results in a much faster and smoother user experience.

4.2 Workflow Diagram

This diagram illustrates the typical journey of a user through the SPA.



4.3 Component-Based Architecture :

The entire application is broken down into small, reusable pieces called **components**.

- **`App.js`**: The main parent component.
- **`LoginPage.js`**, **`RegisterPage.js`**: Components for authentication.
- **`Dashboard.js`**: A container component that holds the sidebar, header, and main content area.
- **`Sidebar.js`**, **`Header.js`**: Reusable layout components.
- **`ExpenseList.js`**, **`AdminDashboard.js`**: Components responsible for displaying data.

4.4 State Management

We have used **React Hooks** for state management.

- **`useState`**: Used for managing local component state (e.g., input values in a form).
- **`useContext`**: Used for managing global state. We created an `AppContext` that holds all the important data like the list of users, expenses, and the currently logged-in user. This allows any component in the application to access this data without passing it down through many levels of props.

4.5 Data Schema (Client-Side)

The data is stored in JavaScript arrays of objects.

- **`users` array:** `[{ id, name, email, password, role }]`
- **`categories` array:** `[{ id, name }]`
- **`expenses` array:** `[{ id, userId, name, amount, categoryId, date }]`

Chapter 5: Implementation & Modules

5.1 Technology Stack

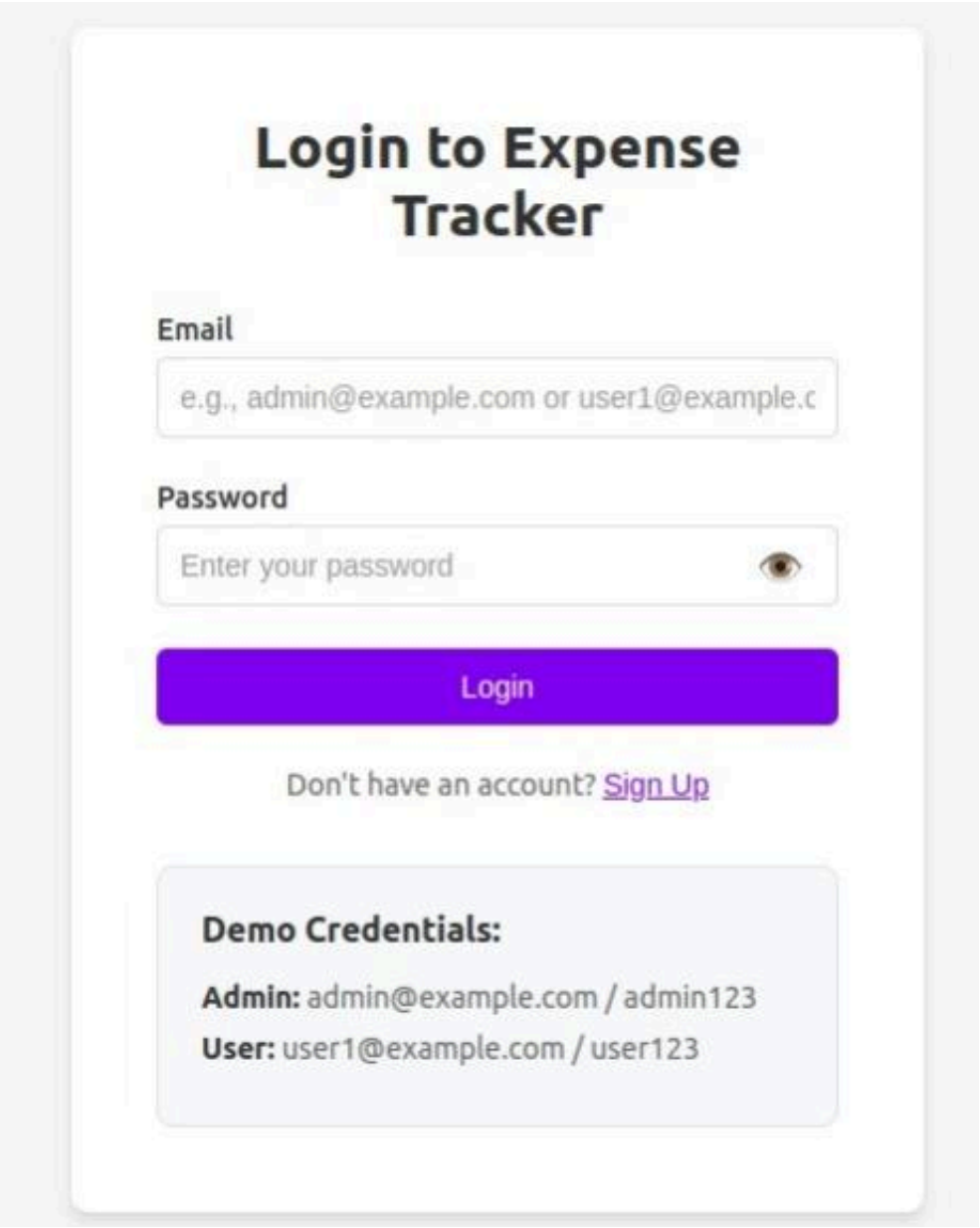
- **React.js:** A JavaScript library for building user interfaces. We used functional components and hooks.
- **Tailwind CSS:** A utility-first CSS framework for building a responsive and modern UI.
- **JavaScript (ES6):** Used for all the application logic, including state management and data manipulation.
- **Chart.js:** A JavaScript library used to create the dynamic Pie Chart in the admin reports.

5.2 Module Description

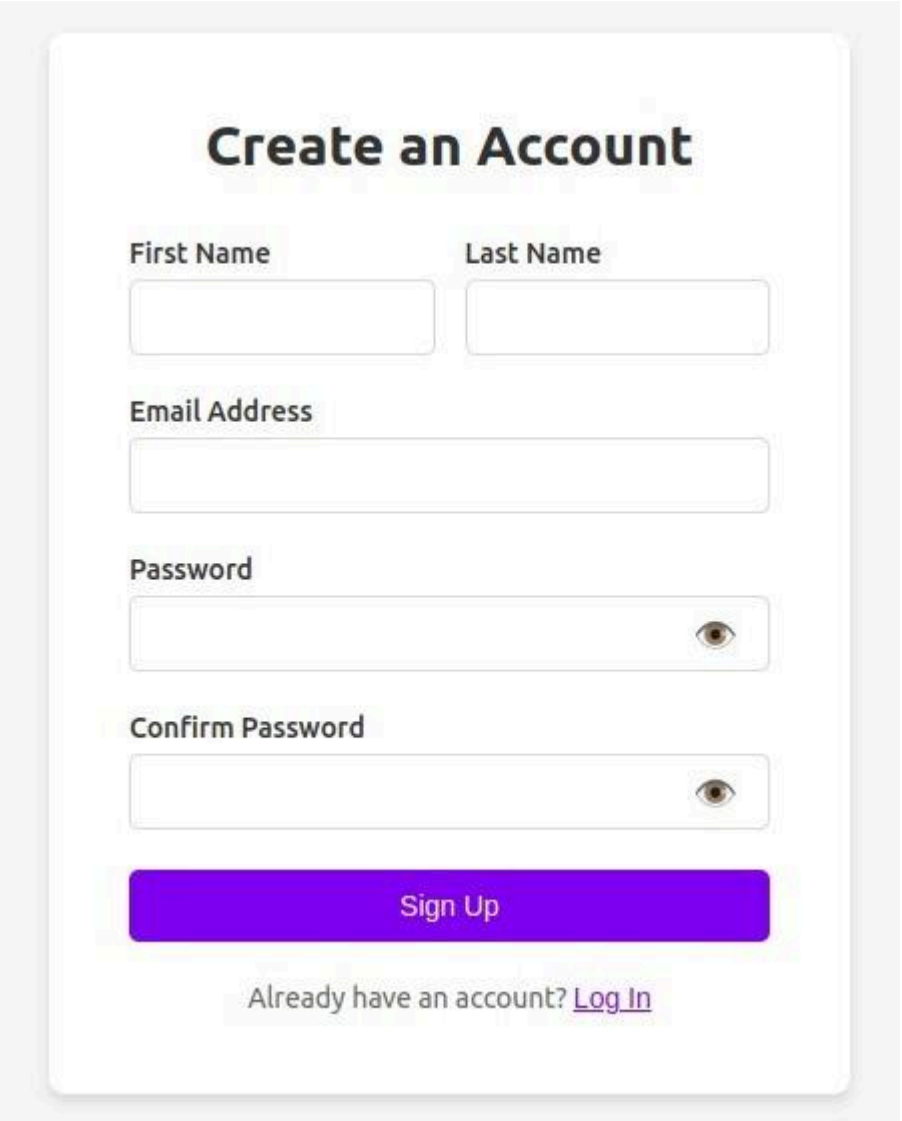
This section describes the key features of the application.

- **Module 1: Authentication (Login & Register)**
  - This module provides a secure entry point. The registration component collects user details and adds them to the application's state. The login component authenticates users against this state.

Login Page :-

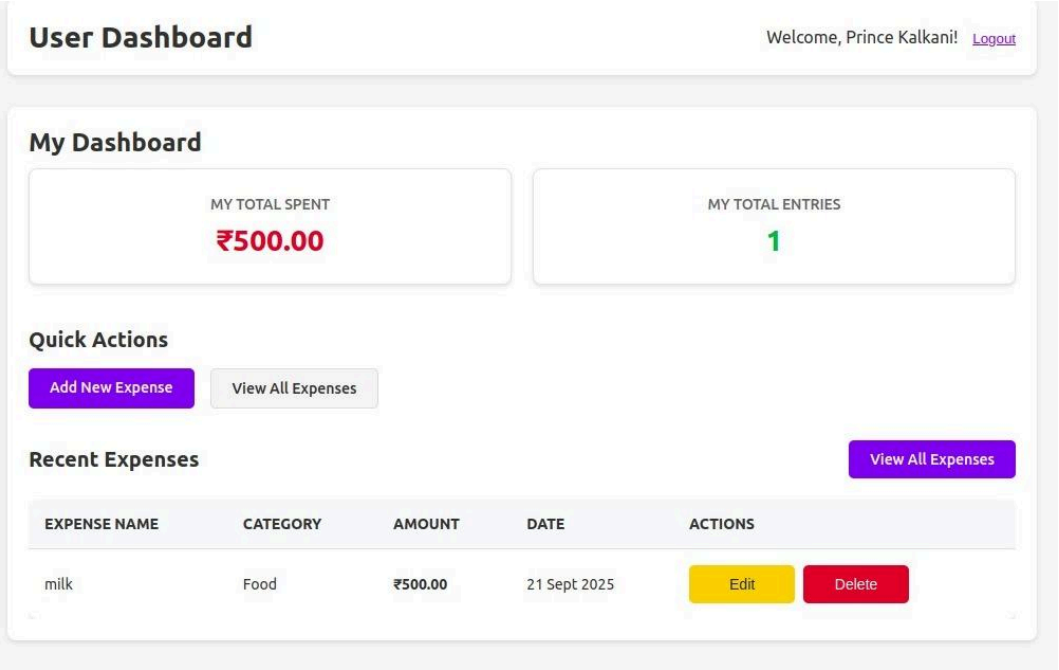


Register Page :-



Module 2: User Dashboard

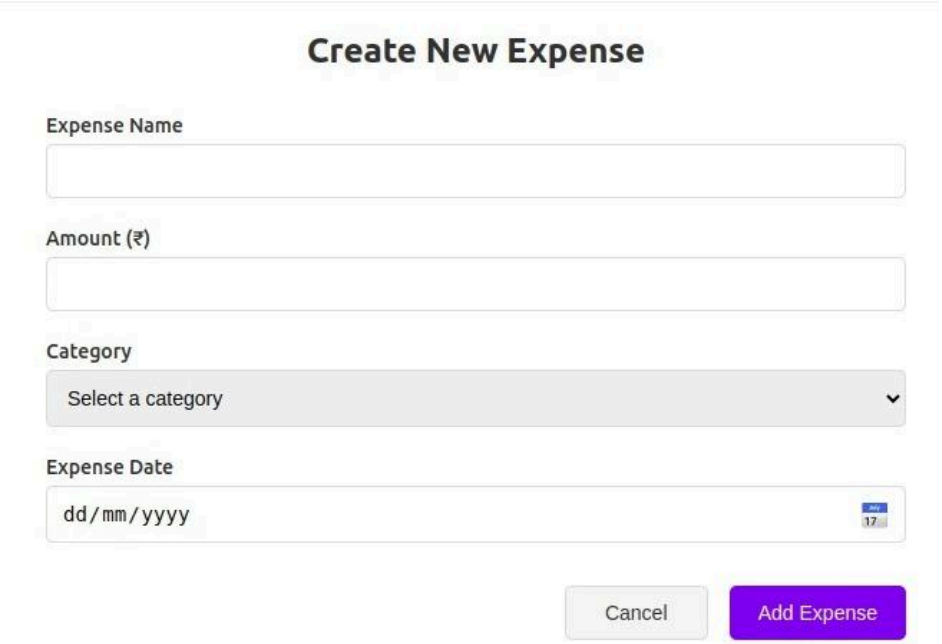
- This is the main landing page for a logged-in user, showing summary cards like "My Total Spent" and "My Total Entries".



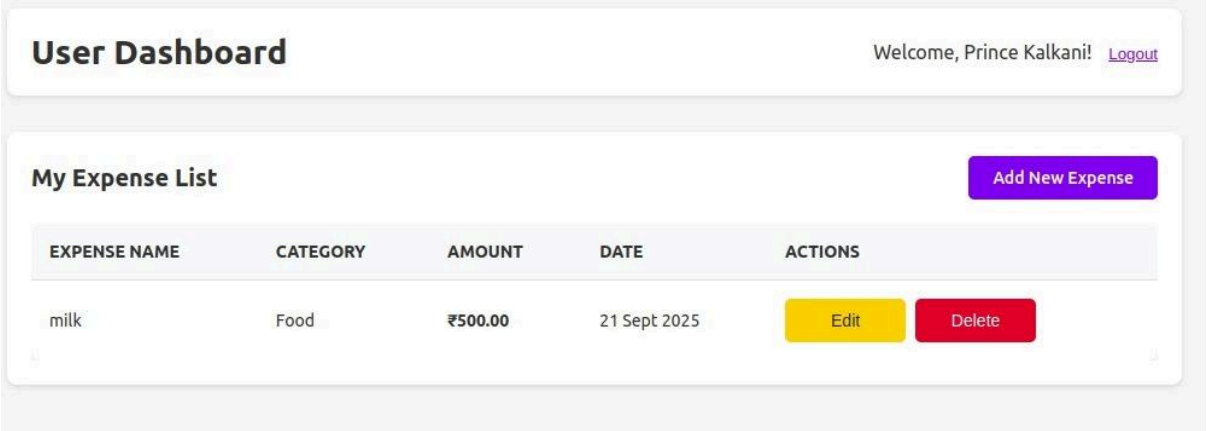
Module 3: Expense Management (CRUD)

- This is the core feature for users. They can add, view, update, and delete their expense records through a fast, interactive interface without any page reloads.

Create Expense :-

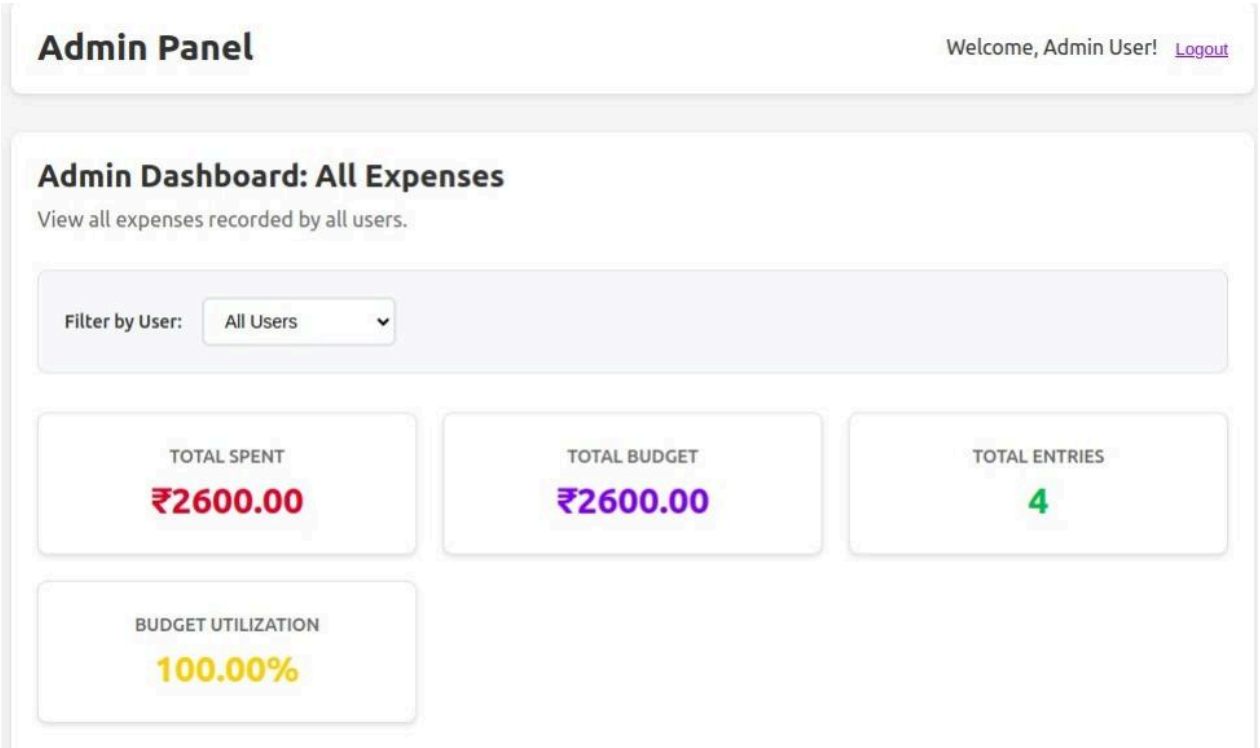


Expense List :-

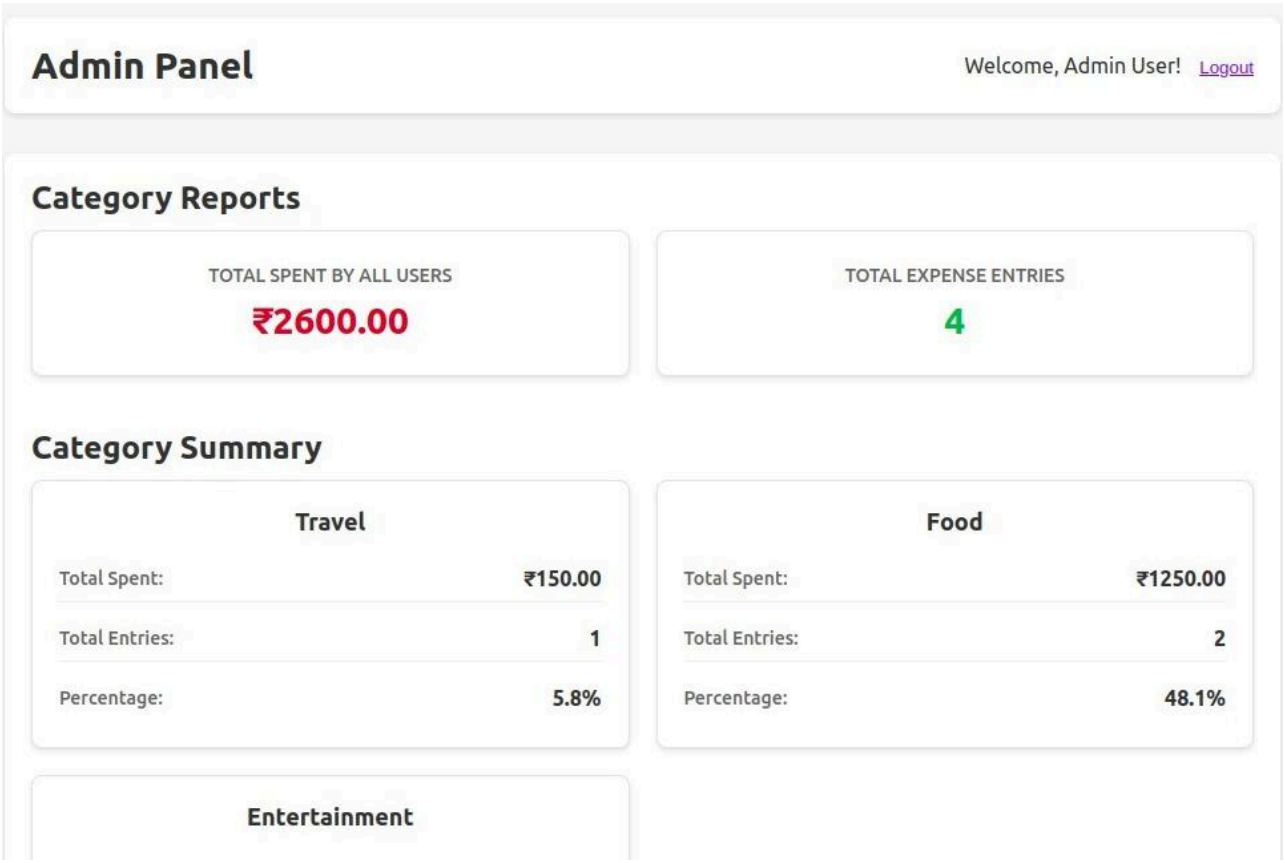


Module 4: Admin Dashboard & Reports

- The admin has a centralized dashboard to view all expenses from all users. They can also see dynamic visual reports, like a Pie Chart that updates in real-time.



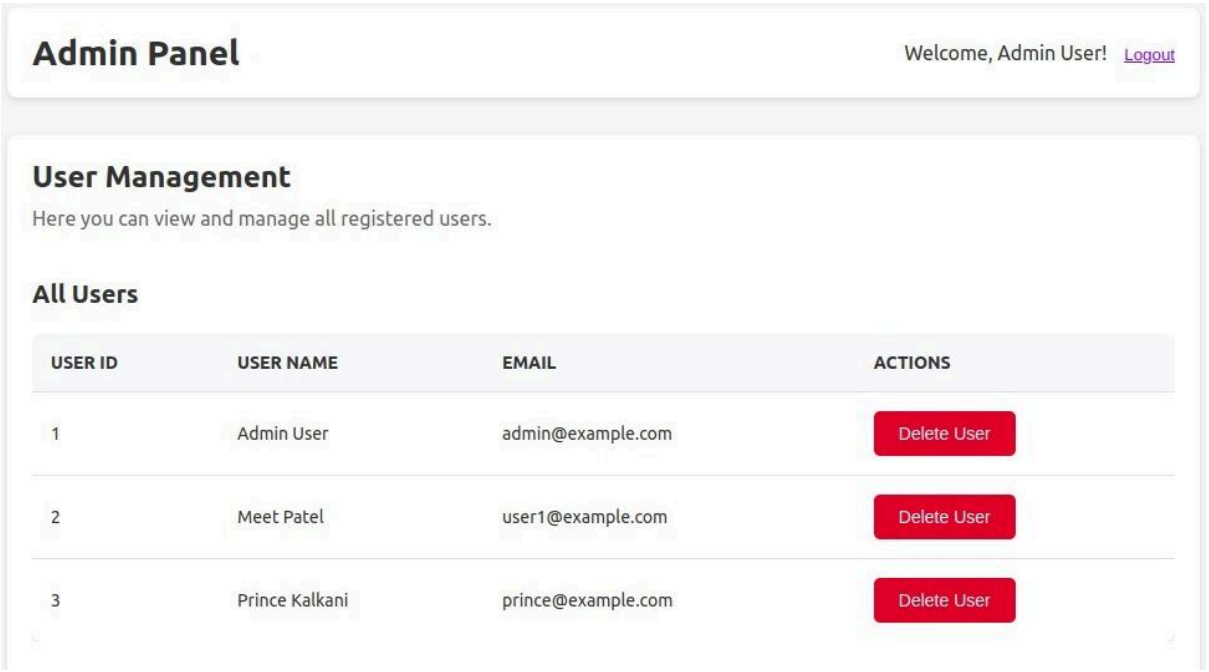
Category Reports :-



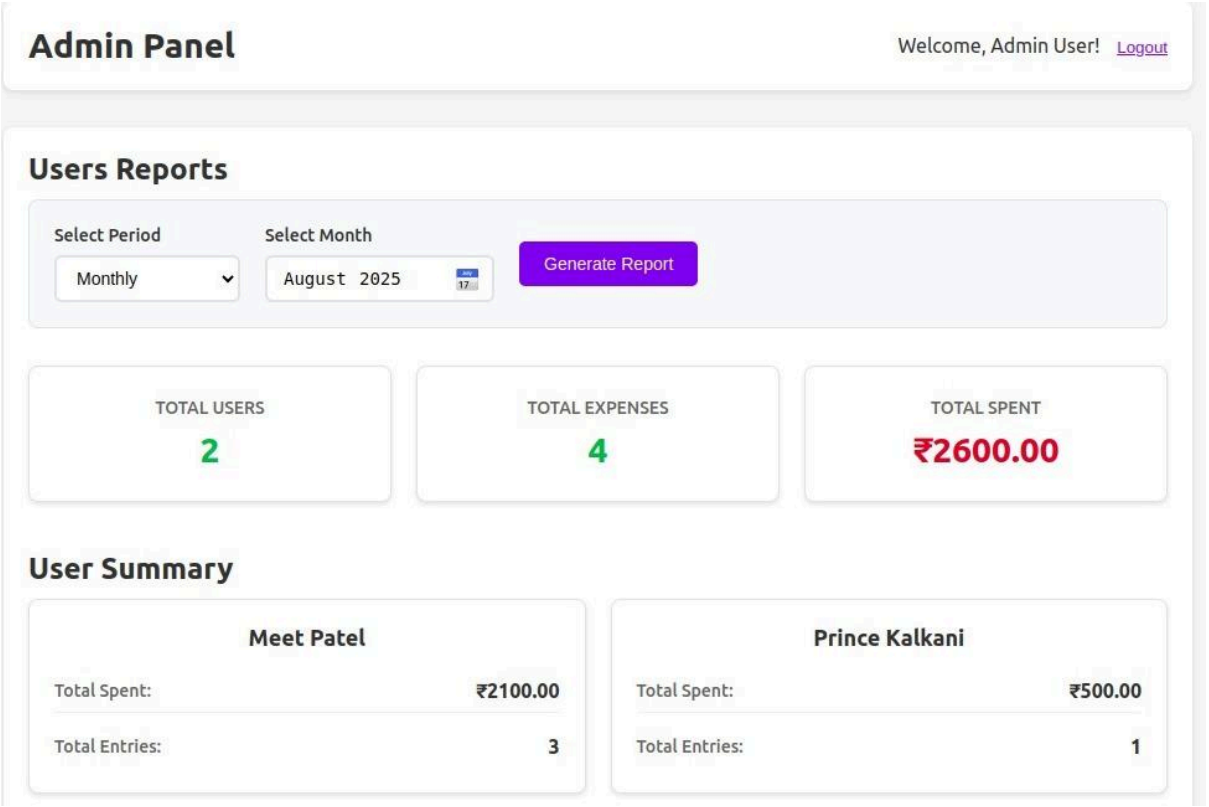
Module 5: User Management (Admin)

- The admin can view a list of all registered users and has the authority to delete any user account from the system, with the changes reflected instantly on the screen.

User Management :-


Admin Panel
Welcome, Admin User! Logout
User Management
Here you can view and manage all registered users.
All Users
USER ID
USER NAME
EMAIL
ACTIONS
1
Admin User
admin@example.com
Delete User
2
Meet Patel
user1@example.com
Delete User
3
Prince Kalkani
prince@example.com
Delete User

User Report :-


Admin Panel
Welcome, Admin User! Logout
Users Reports
Select Period
Monthly
Select Month
August 2025
Generate Report
TOTAL USERS
2
TOTAL EXPENSES
4
TOTAL SPENT
₹2600.00
User Summary
Meet Patel
Total Spent:
₹2100.00
Total Entries:
3
Prince Kalkani
Total Spent:
₹500.00
Total Entries:
1

Chapter 6: System Testing

6.1 Testing Strategy

A multi-level testing strategy was used to ensure the application's quality. This included component testing, integration testing, and end-to-end testing.

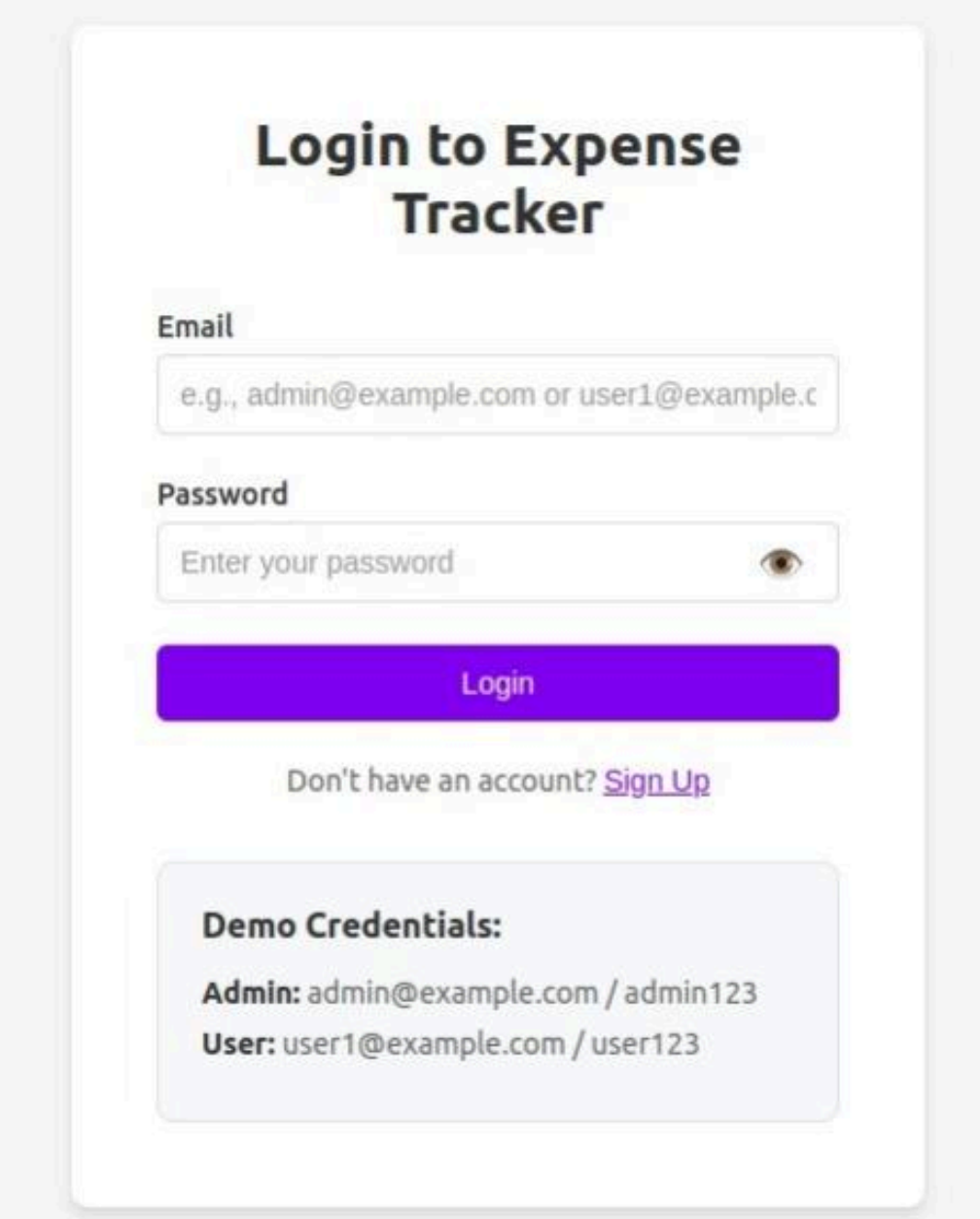
6.2 Test Cases

Test Case ID	Description	Expected Result	Actual Result	Status
TC-01	Register with a new email.	New user is added to the state, and the view changes to login.	Success	Pass
TC-02	Login with valid user credentials.	The `currentUser` state is updated, and the user dashboard is rendered.	Success	Pass
TC-03	User adds a new expense.	The new expense appears in the expense list instantly, without a page reload.	Success	Pass
TC-04	Admin filters expenses by a specific user.	The table updates to show only the expenses of that user.	Success	Pass
TC-05	Log out from the application.	The `currentUser` state is set to null, and the login page is rendered.	Success	Pass

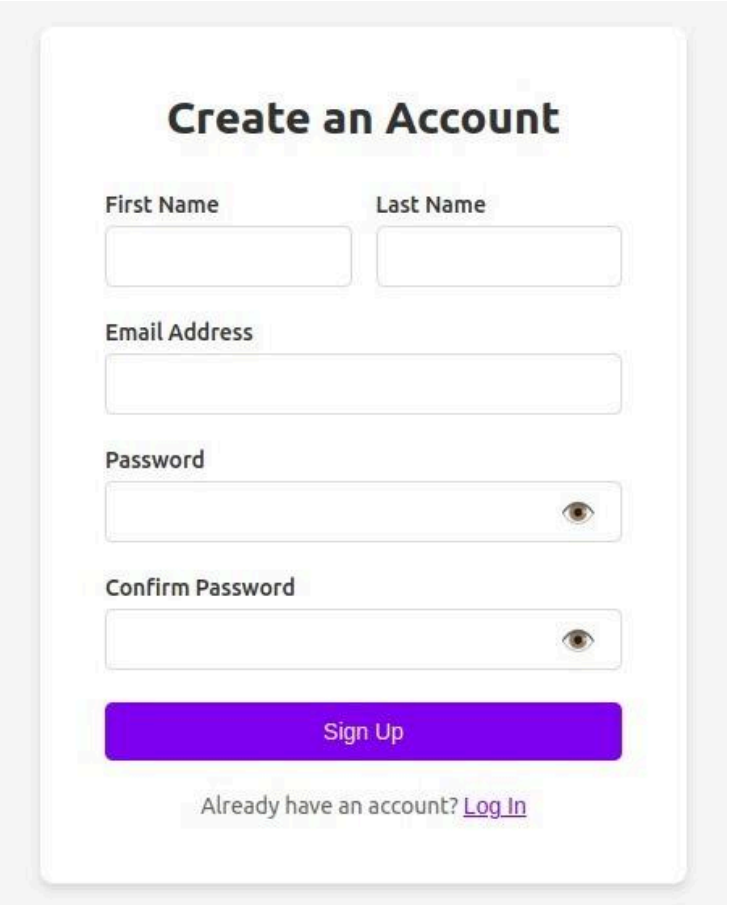
Chapter 7: Results & Snapshots

This section contains a collection of screenshots showcasing the final working application's user interface.

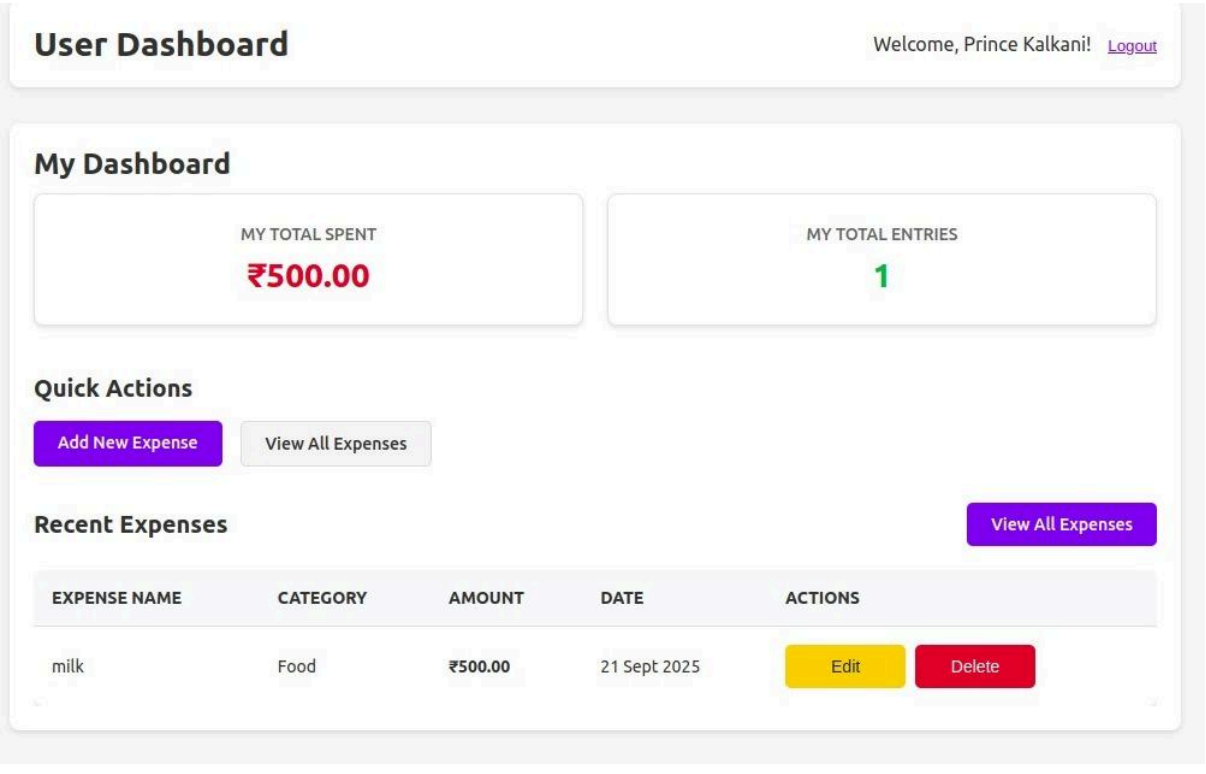
Login Page :-



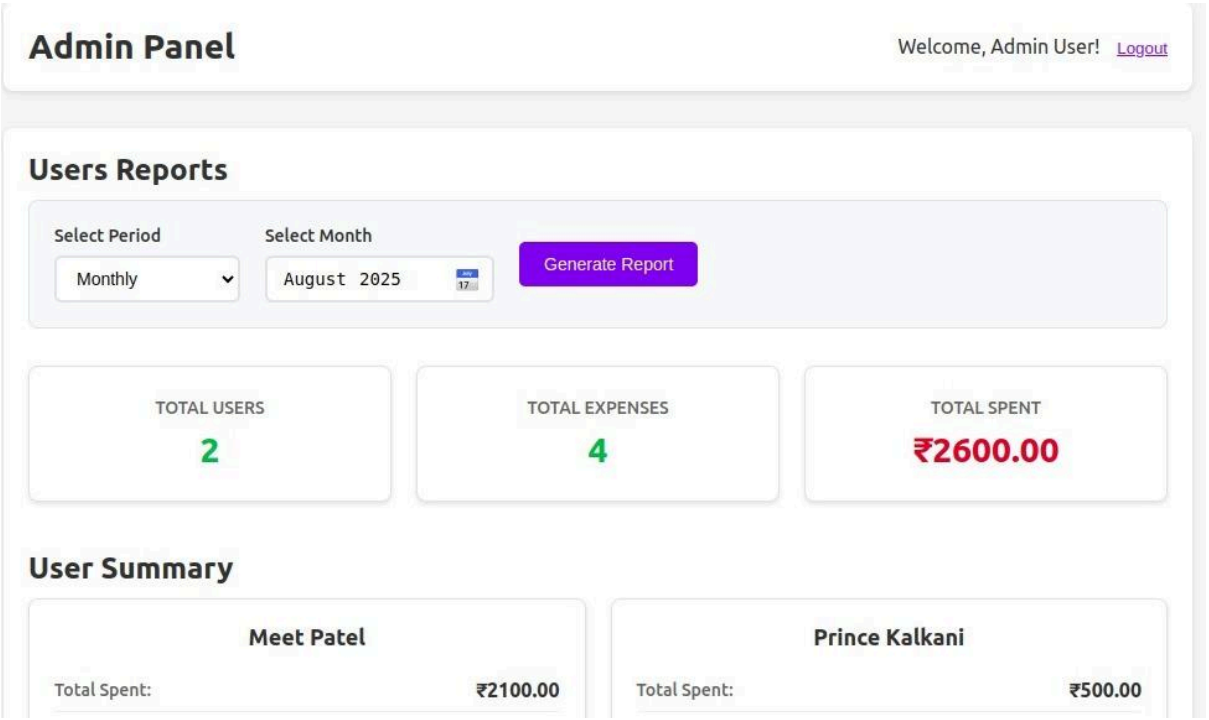
Register Page :-



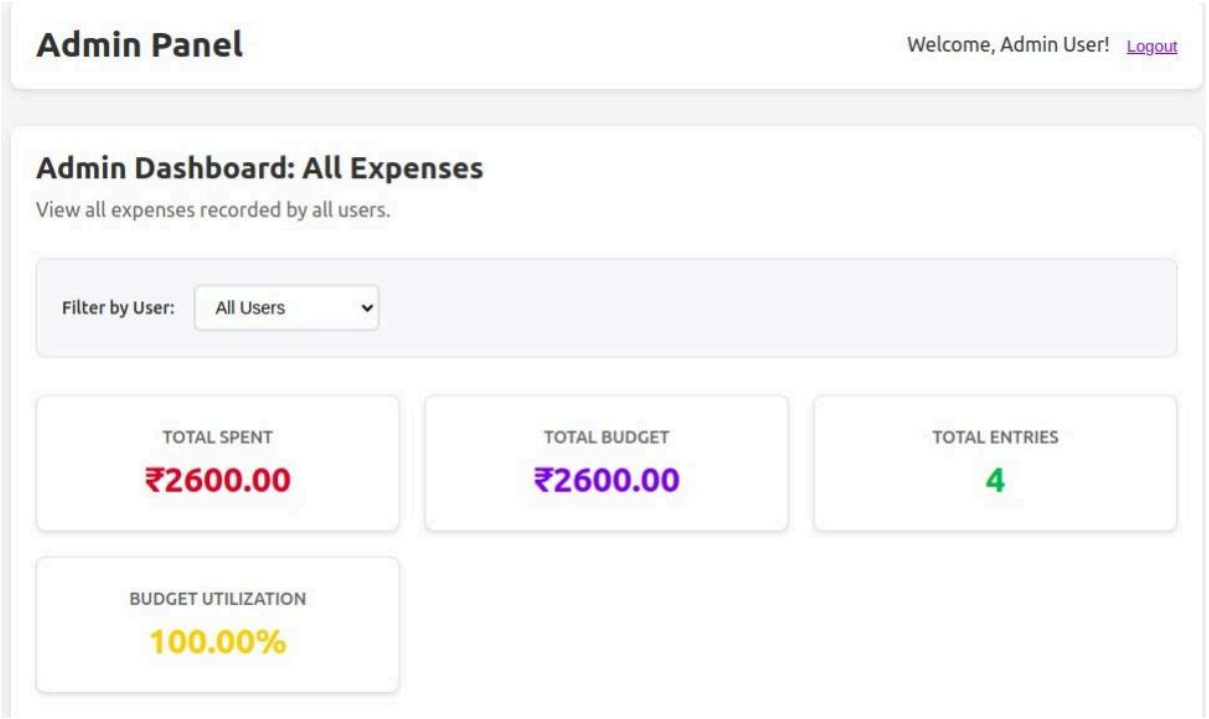
User Dashboard :-



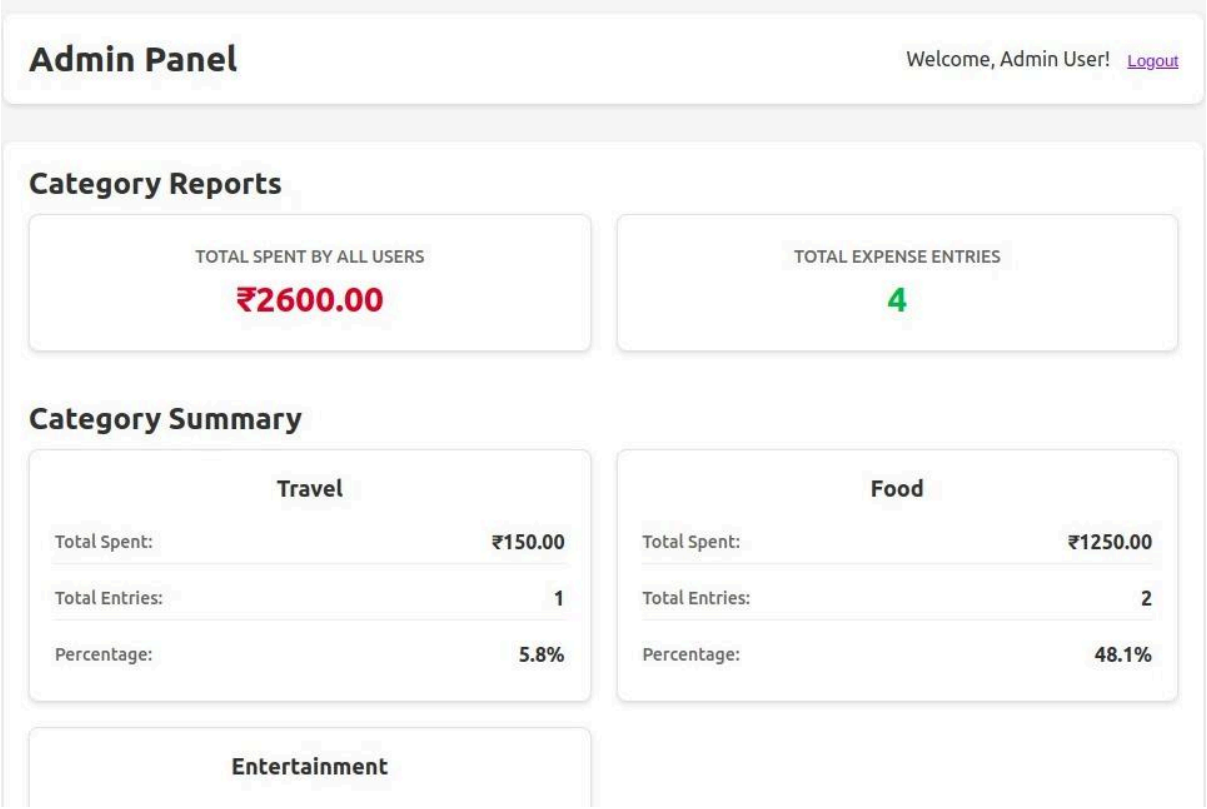
Expense List Table :-



Admin Dashboard



Admin's Category Report with Chart :-



Chapter 8: Conclusion & Future Scope

8.1 Conclusion

This project successfully demonstrates the development of a modern, interactive web application using React.js. We have successfully implemented all the core features in a Single Page Application architecture, which provides a fast and smooth user experience. This project has provided us with invaluable hands-on experience in component-based UI development, client-side state management, and modern frontend workflows.

8.2 Future Scope

- **Backend API Integration:** The next logical step is to develop a proper backend API (e.g., using Node.js/Express) and connect this React application to a real database.
- **Advanced Reports:** Add more charts and allow users to export reports as PDF.
- **Mobile App:** Develop a mobile application for easier access.

Bibliography (References)

- **React.js Official Documentation:** https://react.dev
- **Node.js Official Documentation:** https://nodejs.org
- **Tailwind CSS:** https://tailwindcss.com
- **Chart.js for Data Visualization:** https://www.chartjs.org