ExpensoMeter

MINOR PROJECT REPORT

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF

BACHELOR OF TECHNOLOGY
(Computer Science and Engineering)

Submitted By:

Samridhi Sharma (2203551)
Sapna Maurya (2203557)
Shafneet Kaur (2203560)

Submitted To.:

Dr. Inderjit Singh
Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GURU NANAK DEV ENGINEERING COLLEGE
LUDHIANA, 141006
May, 2025

ABSTRACT

In the context of modern digital financial management, tracking personal and organizational expenses efficiently has become a critical challenge. Traditional expense management tools often suffer from limitations in user engagement, flexibility, and real-time data processing. To address these issues, ExpensoMeter, a full-stack expense tracking application, is introduced, combining a responsive user interface with scalable backend infrastructure to offer a robust solution for financial tracking.

ExpensoMeter leverages React.js for the frontend, ensuring an intuitive user experience, while the backend is powered by Node.js and Express.js, providing an efficient and modular framework for data processing. PostgreSQL is used for data persistence, ensuring high data integrity and optimized query performance for handling large volumes of financial records. The application supports secure user authentication and offers key features such as customizable budget categories, detailed expense reports, and graphical visualizations through bar and pie charts, making financial tracking both accessible and insightful.

A major challenge in traditional expense tracking tools is their inability to adapt to dynamic user needs and provide actionable insights. To overcome this, ExpensoMeter integrates future machine learning capabilities to analyze user spending patterns and deliver personalized suggestions for more efficient budgeting.

This work contributes to digital financial management by combining modern web technologies with intelligent financial analysis, offering a scalable, secure, and user-centric solution to expense tracking.

ACKNOWLEDGMENT

First and foremost, we are extremely thankful to Dr. Sehijpal Singh, Principal, Guru Nanak Dev Engineering College, Ludhiana, for providing the infrastructure and academic environment essential for research and development.

We extend our sincere thanks to Dr. Kiran Jyoti, Head of Department of Computer Science and Engineering, for her constant encouragement and support throughout the duration of the project.

A special note of appreciation goes to our guide, Dr. Inderjit Singh, for his dedicated mentorship, technical insights, and continuous support. His expertise in system architecture and web technologies helped us overcome challenges during backend integration and database design.

We are also grateful to Prof. Jasdeep Kaur and Prof. Harkomalpreet Kaur, Project Coordinators, for their guidance in aligning our implementation with academic and professional standards. This project was significantly enhanced by referring to open-source resources and collaborative development tools such as GitHub, where we maintained our codebase and managed version control.

Finally, we thank our families, peers, and friends who supported us unconditionally and encouraged us throughout this journey.

Samridhi Sharma
Sapna Maurya
Shafneet Kaur

LIST OF FIGURES

LIST OF TABLES

TABLE OF CONTENTS

Chapter 1: Introduction

1.1 Introduction to Project

Rightly said, “Financial health hinges on understanding where your money goes.” In today's fast-paced world, keeping track of expenses can be a daunting task. From daily coffee runs to unexpected bills, our finances often feel like a chaotic puzzle, making personal finance management crucial for achieving financial stability and goals.

By meticulously recording every expenditure, we gain insights into areas of overspending, enabling better financial awareness. Incorporating expense tracking into daily routines helps individuals understand spending habits, make informed decisions, and improve overall financial health.

This project, ExpensoMeter, is a user-friendly application designed to empower individuals to take control of their finances. It makes financial management easily accessible anytime, anywhere. By seamlessly integrating data entry, visualization, and insightful reporting, ExpensoMeter provides users with a powerful tool to monitor their expenses and work toward financial goals.

1.2 Project Category (Application Based)

This project falls under the Application category. It focuses on creating a practical, user-oriented software solution for personal and organizational finance management, using advanced technologies such as Machine Learning and Artificial Intelligence.

1.3 Problem Formulation

In an era where financial independence is a key to stability, individuals and businesses alike struggle with effective expense management. People often fail to track their spending accurately,

leading to poor financial decisions and a lack of awareness about their financial health. Furthermore, businesses may find it difficult to efficiently monitor operational costs, contributing to potential budget overruns. The need for an intuitive solution that can automate expense tracking, analyze spending patterns, and provide actionable insights is evident.

1.4 Identification/Recognition of Need

The importance of personal finance management is increasing as financial independence becomes a significant goal for many. People are often unaware of how their day-to-day expenses affect their long-term financial health. Businesses also face challenges in tracking large-scale expenditures and managing budgets efficiently. Thus, there is a need for an integrated tool like ExpensoMeter, which can assist users at both individual and organizational levels to monitor, analyze, and improve their financial health.

1.5 Existing System

Currently, many expense tracking solutions exist, including mobile apps and software programs. These typically allow users to input their expenses manually and categorize them. However, most existing systems fail to provide advanced insights, predictive analytics, and proactive financial recommendations. Additionally, some tools are either too complex for individual use or lack the scalability required for organizational finance management.

1.6 Objectives

The following objectives are achieved for this minor project:

i. To create a functional and user-friendly GUI application to track expenses on individual level and at large scale (i.e. for a company).

ii. To analyze expenses using Machine Learning and generate detailed reports, identify spending patterns and offer actionable insights for better financial management.

iii. To implement AI-powered suggestions and alerts when users exceed their budget.

1.7 Proposed System

The proposed system, ExpensoMeter, is an intelligent application designed to provide users with an easy-to-use interface for tracking expenses and managing finances. It will incorporate features such as:

- Expense entry and categorization to help users document their spending.
- Expense analysis powered by Machine Learning to detect spending patterns and predict future trends.
- Real-time alerts and suggestions from AI to help users stay within their budget and make informed decisions.
- Visualization tools such as charts and graphs to provide a clear view of users' financial standing.

1.8 Unique features of the proposed system

- **AI-Powered Insights:** The integration of AI and Machine Learning for personalized expense recommendations and predictive analytics.
- **Scalability:** The system will be designed to scale from individual users to large businesses, providing detailed financial management at all levels.
- **Real-time Alerts:** Users will receive timely notifications if they approach or exceed their budget, keeping their finances in check.

- **User-Friendly Interface:** The application will have a simple, intuitive interface, making it easy for users of all backgrounds to manage their finances without technical expertise.
- **Comprehensive Reporting:** Detailed expense reports will be generated to help users understand their spending habits, identify areas for improvement, and track financial progress over time.

‘

Chapter 2 Requirement Analysis and System Specification

2.1 Feasibility study

The feasibility study for the ExpensoMeter project examines its technical, operational, and economic feasibility to ensure successful development. It analyzes resource requirements, cost implications, and potential risks while confirming the system's ability to meet user demands. It includes an analysis of technical, operational, and economic feasibility to evaluate the overall viability of the ExpensoMeter project.

2.1.1 Technical Feasibility

- **Technology Stack:** The project utilizes modern and scalable technologies like React[3] for the frontend, Python for Machine Learning integration, SQL for data storage, and HTML/CSS[2] for basic web structure. These tools ensure the application is reliable and maintainable.
- **Development Tools:** React provides an interactive and dynamic user interface, while Flask is used for the backend, offering a robust framework for handling user data and server-side logic.
- **Resources:** The project requires skilled developers proficient in web development, Python, and Machine Learning. Affordable hosting services will be sufficient for deploying the application.
- **Challenges:** Securing user data and optimizing performance, especially when generating graphical reports and handling notifications, are key challenges. These challenges will be addressed through best practices in development and performance optimization.

2.1.2 Operational Feasibility

- **Ease of Use:** The application is designed to be intuitive and user-friendly, ensuring that people with varying technical expertise can easily manage their finances.
- **Target Audience:** The app caters to individuals and small businesses looking for an efficient solution to track and manage their expenses.
- **Integration:** The system includes features like calendar notifications and third-party authentication for a seamless experience, ensuring users stay on track with their financial goals and can securely log in using existing credentials.

2.1.3 Economic Feasibility

- **Development Costs:** The project has low development costs due to the use of open-source tools (React, Python, etc.) and affordable cloud hosting services.
- **Revenue Model:** The application will offer a free version supported by ads, with an option for users to subscribe to a premium version that unlocks advanced features like detailed analytics, data backup, and enhanced reporting tools.
- **Market Demand:** As the demand for financial tools grows, the app's ability to offer a balanced combination of simplicity, security, and functionality positions it well in the market.

2.2 Software Requirement Specification Document

The ExpensoMeter project requires clear software specifications to guide development and ensure reliable, secure expense management. These specifications cover data handling, functionality, performance, and security aspects.

2.2.1 Data Requirements

ExpensoMeter will manage and store user-generated financial data, including transaction details, spending categories, and budget allocations. The application relies on PostgreSQL, ensuring that expense records are securely stored and managed within a centralized database system.

2.2.2 Functional Requirements

- **User Authentication:** The system will implement secure login and registration processes using email or third-party authentication methods. This ensures user data privacy and allows for personalized experiences.
- **Expense Management:** Users will have the ability to add, edit, and delete expenses. They will categorize transactions by date, amount, and payment method, enabling effective financial tracking.
- **Category Management:** Predefined and customizable spending categories will be available, along with budget allocation features for each category, helping users stay within their financial limits.
- **Expense Visualization:** The application will feature interactive charts and reports that provide monthly and yearly financial summaries, helping users visually track their spending.
- **Budgeting & Alerts:** Machine learning algorithms will provide spending insights, as well as notifications and alerts when users approach or exceed their budget limits.
- **Data Storage:** PostgreSQL enables persistent and reliable data storage, supporting efficient management, retrieval, and organization of expense records within the application.

2.2.3 Performance Requirements

- **Efficient Data Storage:** Using PostgreSQL ensures fast and reliable data storage, enabling quick read and write operations while maintaining performance and scalability.
- **Responsive Design:** The system will be designed to be accessible on both web and mobile platforms, ensuring smooth and intuitive navigation across devices.
- **Optimized Load Handling:** The application will efficiently handle expense tracking and report generation without significant delays, ensuring a smooth user experience.

2.2.4 Security Requirements

- **Data Encryption:** Financial data stored in PostgreSQL will be encrypted to prevent unauthorized access.
- **Authentication Mechanisms:** The login system will use password hashing and session-based authentication to secure user access.
- **User Privacy:** Since all data is securely stored in PostgreSQL, no financial information will be transmitted to external servers, ensuring complete control over user data.
- **Regular Updates:** Security patches and system enhancements will be applied regularly to maintain system security and protect against vulnerabilities.

2.3 SDLC model

The SDLC model will be implemented for the ExpensoMeter project. This iterative approach allows for regular feedback from stakeholders and continuous refinement of features. Each sprint will focus on specific modules or functionalities, such as expense tracking, budget alerts, or report generation. The model ensures that the development process remains flexible and responsive to changing requirements, ultimately leading to a product that closely aligns with user

needs and expectations. Additionally, it promotes close collaboration among cross-functional teams, enabling faster identification and resolution of issues. Regular sprint reviews and retrospectives will help improve team performance and product quality over time. By delivering incremental updates, stakeholders can see tangible progress and provide timely input. This approach significantly reduces the risk of project failure and enhances overall user satisfaction.

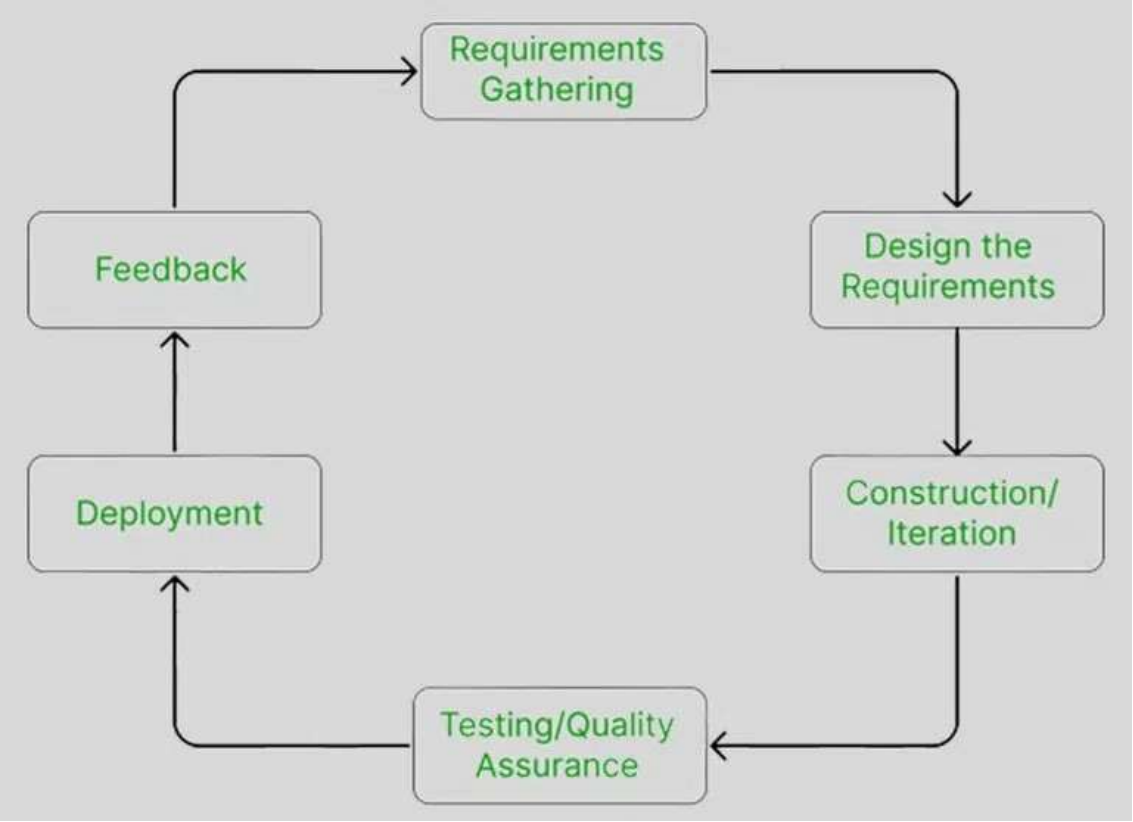

Figure 2.1: SDLC Model

Chapter 3 System Design

3.1 Design Approach

The ExpensoMeter application uses an object-oriented design approach. This methodology ensures modularity, scalability, and code reusability. Key components like User, Expense, and Category are structured as independent objects, each with defined properties and behaviors. This makes the application easier to maintain and extend.

3.2 Detail Design

The system design includes multiple structured analysis and design tools to meet the objectives of the project.

3.2.1 Flowcharts

Flowcharts depicting workflow and system architecture are shown in this section.

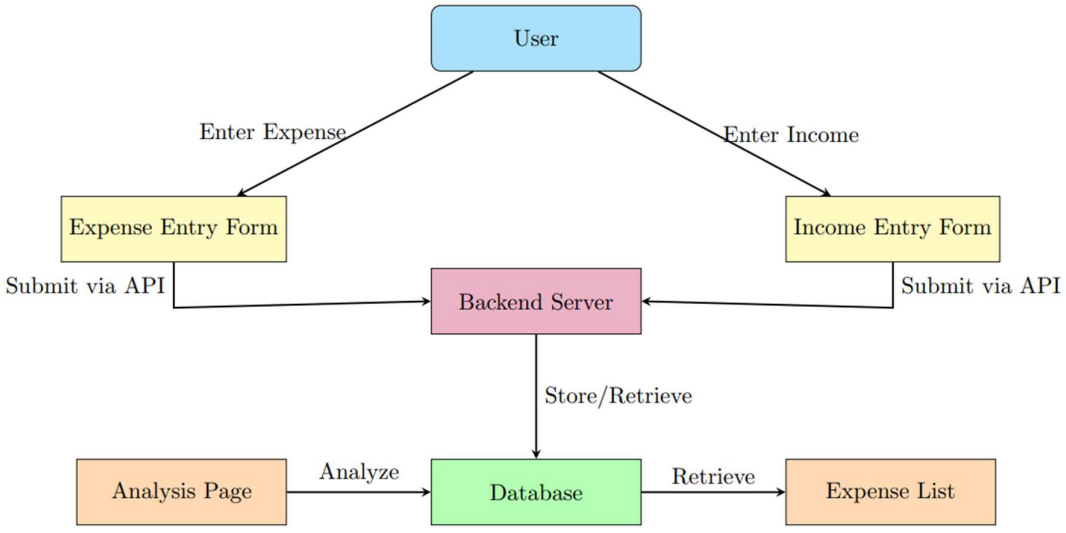

Figure 3.1: Workflow

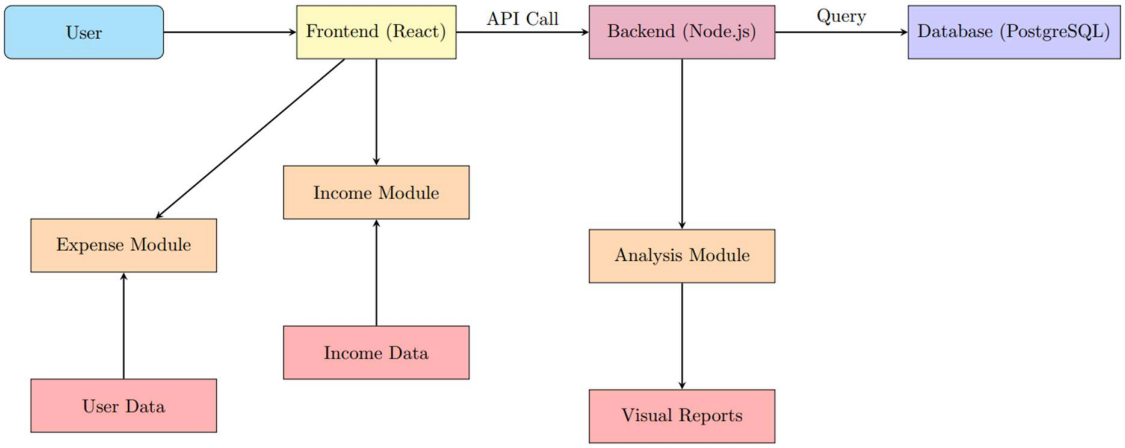

Figure 3.2: System Architecture

3.2.2 Data Flow Diagrams (DFDs)

DFDs are used to represent the flow of data within the system at different levels of detail. Different levels of DFDs help in understanding the system from a high-level overview down to detailed process flows. Level 0 DFD gives a context-level view of the system interacting with the user. Level 1 DFD breaks down into smaller modules like authentication, expense entry, visualization, and alert generation.

- **Level 0 DFD (Context Level)**

  The Level 0 DFD provides a high-level overview of the ExpensoMeter system, showing how the user interacts with it. It captures the primary data flow, where users input expenses or view reports and receive summaries or alerts in response. This context-level diagram helps in understanding the system boundaries and the main exchange of information between the external entity and the system.

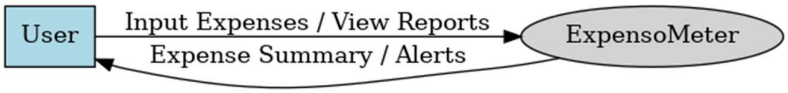

Figure 3.3: Level 0 DFD

- **Level 1 DFD**

  The Level 1 Data Flow Diagram (DFD) for the "Expensometer" expense tracker project illustrates the main processes that the system performs, such as user input, expense categorization, and report generation. It shows how data flows between external entities (like users or external systems) and internal processes, focusing on major functions such as adding expenses, viewing expense reports, and updating the database. This level provides a more detailed view than the context diagram, breaking down the system's core operations.

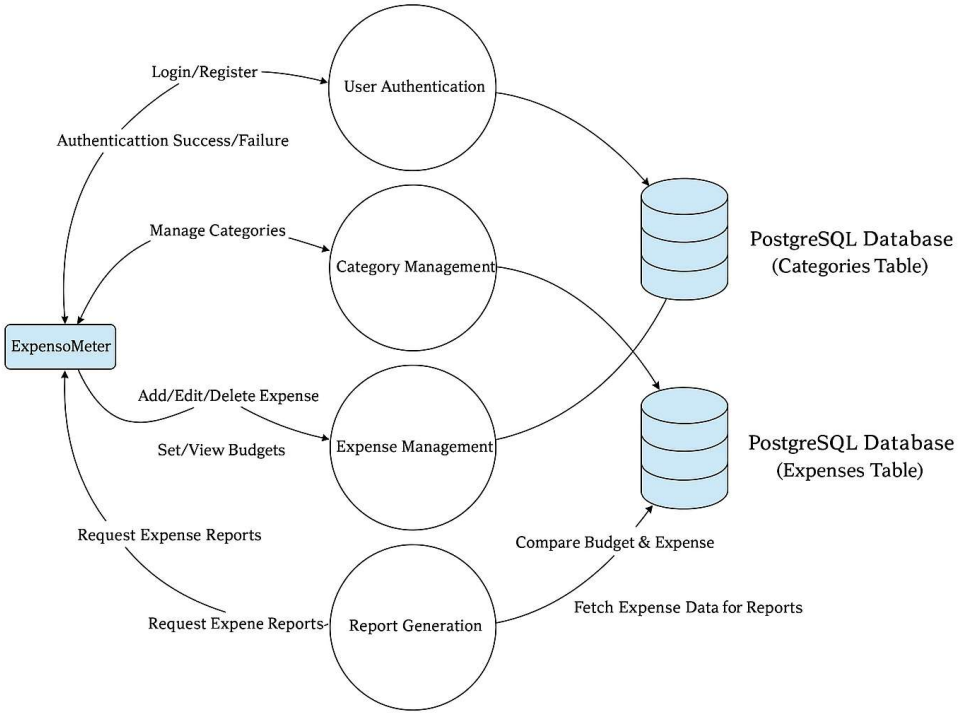

Figure 3.4: Level 1 DFD

3.2.3 Database Design

The database is designed using a relational model to store and manage project data efficiently. It includes well-structured tables with primary and foreign keys to ensure data integrity and support

seamless querying and operations. The design supports system requirements while maintaining efficiency and scalability.

- **E-R Diagram**

Figure 3.5: E-R Diagram

The database for the ExpensoMeter project consists of the following tables:

- **BusinessExpenses**: Stores details of expenses related to business activities, including the type of expense, amount, date, and category.
- **BusinessIncome**: Records income generated by the business, including income amount, date, and source.
- **Expenses**: Tracks personal or general expenses, capturing the amount, category, date, and any associated notes.
- **Incomes**: Holds information about personal income, including source, amount, and date.

These tables are structured to ensure efficient tracking and management of both business and personal financial data, allowing for seamless categorization and reporting within the system.

3.3 User Interface Design

The interface is built using React.js and is component-based for better structure and maintenance. Key screens include navigation, home page, authentication forms, expense entry form, expense listing, and the analysis section. The design is fully responsive, making it usable across devices.

3.4 Methodology

Phase 1: Requirement Gathering

In this phase, the primary focus was to identify the core features needed by end users. This included:

- Expense recording with date, amount, and category
- Notifications for budget thresholds
- Monthly and yearly visual summaries
- A clean and responsive UI for all platforms

Feedback from potential users and a review of existing applications helped shape the feature set.

Phase 2: System Design

This stage involved designing the layout and structure of the application:

- A component-based UI was structured using React.js for modularity and reuse

- Backend logic and data storage strategy were decided (using PostgreSQL for data storage).
- Initial design diagrams such as use case, activity, and E-R diagrams were created
- Emphasis was placed on an intuitive user flow from login to analysis dashboard

Phase 3: Development

Development was divided into frontend and backend tasks:

- Frontend: Built using React.js, HTML, CSS, and JavaScript for interactivity and responsiveness.
- Backend (database handling): Developed using PostgreSQL for efficient data storage and retrieval.

Phase 4: Testing

Multiple testing strategies ensured stability and usability:

- Unit Testing: Verified individual functions like add, update, and delete expense
- Integration Testing: Ensured UI components worked seamlessly with localStorage
- User Acceptance Testing: Real users tested the app and gave feedback on performance, clarity, and feature usefulness

Phase 5: Deployment

Deployment was done using platforms like GitHub Pages and Firebase Hosting:

- Codebase was optimized for static hosting
- Proper routing and page loading were ensured
- Lightweight bundling helped in fast loading across devices

Phase 6: Post-Deployment Monitoring & Feedback

User feedback was actively collected through shared beta versions:

- Minor UI bugs and inconsistencies were identified
- User feedback was collected after release.
- Performance issues were monitored and improved in real-time

Phase 7: Maintenance

Ongoing maintenance includes:

- Adding new features based on user demand (e.g., ML-powered alerts)
- Bug fixing and UI enhancements
- Regular performance checks and optimization

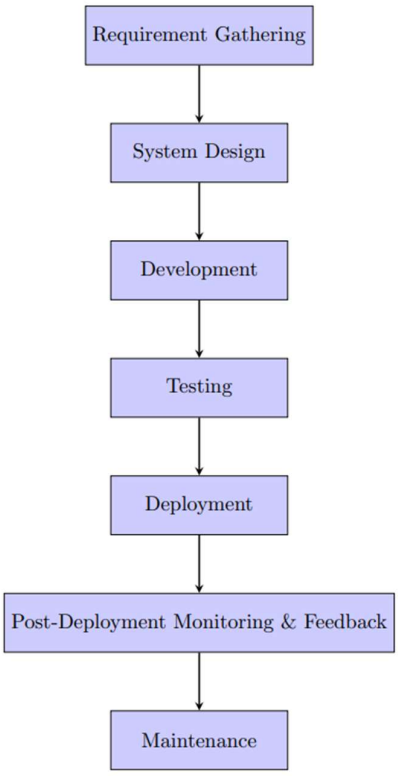

Figure 3.6: Methodology

Chapter 4 Implementation and Testing

4.1 Introduction to Languages, IDE's, Tools and Technologies used for Project work

4.1.1 Frontend:

- **Language: JavaScript**

  JavaScript was used to implement dynamic functionality and interactivity within the frontend components of the application.

Figure 4.1: JavaScipt

- **Framework: React.js**

  React.js provided a component-based architecture to efficiently manage the user interface and state of the application.

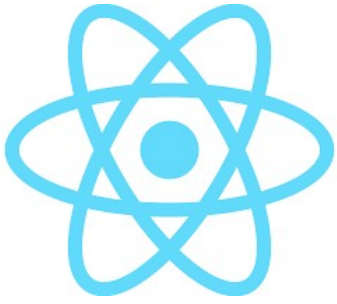

Figure 4.2: React.js

- **IDE: Visual Studio Code**

  Visual Studio Code offered an efficient and customizable development environment for writing and debugging frontend code.

Figure 4.3: Visual Studio Code

4.1.2 Backend:

- **Language: Python**

  Python was used to handle server-side logic and API development for managing data and business logic.

Figure 4.4: Python

- **IDE: Visual Studio Code**

  Visual Studio Code served as the main development environment for backend coding, offering extensions and debugging support.

- **Testing Tool: Postman**

  Postman was utilized to test and validate API endpoints, ensuring correct data handling and response formatting.

Figure 4.5: Postman

4.1.3 Database:

- **Database System: PostgreSQL**

  PostgreSQL, a powerful open-source relational database system, was used to store, manage, and retrieve structured data for the application efficiently and securely.

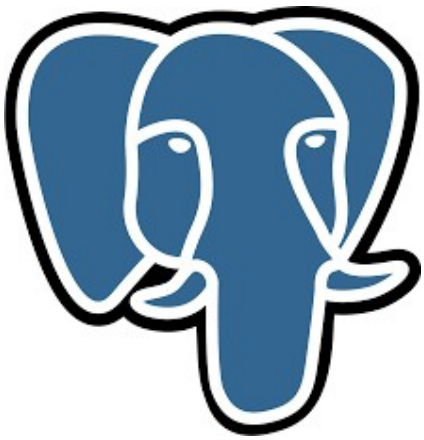

Figure 4.6: PostgreSQL

4.1.4 Version Control:

- **Tool: Git**

  Git was used for tracking changes, managing versions, and collaborating effectively throughout the development process.

- **Repository Hosting: GitHub**

  GitHub[5] provided a remote repository to host the project code and enabled version control, issue tracking, and team collaboration.

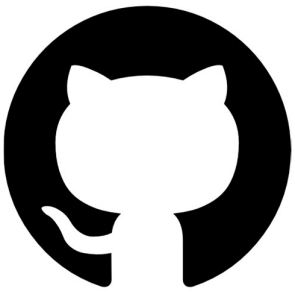

Figure 4.7: GitHub

4.1.5 Deployment:

- **Platform: GitHub Pages for frontend deployment**

  The frontend application was deployed on GitHub Pages[5], enabling it to be accessed online via a public URL.

4.2 Algorithm/Pseudocode Used

4.2.1 Algorithms Used

The expense analysis employs K-means Clustering[4] to analyze spending patterns and Linear Regression[4] to predict future expenses.

- **K-means Clustering**
    - **Purpose**

      Group transactions into similar clusters to identify spending patterns.
    - **Algorithm**
        i. Choose 3 clusters to group transactions into.
        ii. Randomly pick 3 starting points (called centroids) in the data.
        iii. For each transaction:
            - Find the closest centroid by calculating the distance.
            - Assign the transaction to that centroid's cluster.
        iv. For each cluster:
            - Calculate a new centroid by averaging the positions of all transactions in the cluster.
        v. Repeat steps 3–4 until the centroids no longer move.
        vi. Output the final clusters, showing which transactions belong to each group.
- **Linear Regression**
    - **Purpose**

      Predict the next month's total expenses based on past spending trends.
    - **Algorithm**

        i. Gather past data, like monthly expenses and the month number (e.g., February is 2).
        ii. Find a straight line that best matches the data by minimizing the difference between actual expenses and the line's predictions.

iii. Use the line's equation (e.g., expenses = a × month_index + b × month_number + c) to calculate the next month's expenses.

iv. If the predicted expense is too high, cap it at ₹40,000 to keep it realistic.

v. Output the predicted expense for the next month (e.g., June 2025).

4.2.2 Pseudocode Used

- **Fetching Transactions**

  Function fetchTransactions():

    Send GET request to '/api/v1/transactions'

  If response is successful:

    Set transactions state with data from response

- **Adding a New Transaction**

  Function handleAddTransaction(transaction):

    Send POST request to '/api/v1/transactions' with transaction data

  If response is successful:

    Append the new transaction (from response) to transactions state

- **Deleting a Transaction**

  Function handleDeleteTransaction(id):

    Send DELETE request to '/api/v1/transactions/:id'

  If response is successful:

    Remove the transaction with the given ID from transactions state

- **Calculating Income, Expense, and Balance**

  Function calculateSummary(transactions):

    Initialize income = 0

```
Initialize expense = 0
For each transaction in transactions:
        If amount > 0:
            income += amount
        Else if amount < 0:
            expense += amount
balance = income + expense  // since expense is negative
Update UI with income, expense, and balance
```

4.3 Testing Techniques: in context of Project Work

4.3.1 Frontend Testing

- **Manual Testing:** Conducted using Chrome DevTools to inspect elements, monitor console logs, and test responsiveness.
- **Functional Testing:** Ensured that all UI components (forms, buttons, charts) behave as expected.
- **Cross-Browser Testing:** Verified application functionality across different browsers like Chrome, Firefox, and Edge.

4.3.2 Backend Testing

- **Tool Used:** Postman
- **Testing Approach:**
    - **Unit Testing:** Tested individual API endpoints for correct responses.

- **Integration Testing:** Ensured that the frontend and backend communicate seamlessly.
- **Error Handling:** Verified that appropriate error messages are returned for invalid inputs or requests.

4.3.3 Testing Scenarios in Postman

- **GET /expenses:** Retrieve all expenses.
- **POST /expenses:** Add a new expense.
- **PUT /expenses/{id}:** Update an existing expense.
- **DELETE /expenses/{id}:** Delete an expense.

4.4 Test Cases Designed for the Project Work

Table 4.1: Test Cases for Expense Management Functionalities

Test Case ID	Description	Input Parameters	Expected Result	Actual Result	Status
TC01	Add a new expense	Amount: 100, Category: Food	Expense is added and displayed in the list	As Expected	Pass
TC02	Retrieve all expenses	None	List of all expenses is displayed	As Expected	Pass
TC03	Update an existing expense	ID: 1, New Amount: 150	Expense amount is updated to 150	As Expected	Pass
TC04	Delete an expense	ID: 1	Expense is removed from the list	As Expected	Pass
TC05	Add expense with missing fields	Amount: '', Category: Food	Error message prompting for required fields	As Expected	Pass
TC06	Access application offline	Disconnect internet	Application functions without errors	As Expected	Pass

Chapter 5 Results and Discussions

5.1 User Interface Representation

ExpensoMeter features a clean and responsive interface built with React.js. It follows a component-based design to maintain modularity and improve user experience.

5.1.1 Brief Description of Various Modules of the System

The application consists of multiple UI components:

- **Navigation Bar –** Provides navigation to different sections of the application.
- **Welcome Page –** Introduces users to the application.
- **Dashboard –** Displays a summary of financial data.
- **Login & Register Pages –** Handles user authentication.
- **Expenses Page –** Allows users to manage expenses.
- **Income Page –** Manages income entries.
- **Analysis Page –** Displays expense trends and predictions.

5.2 Snapshots of system

5.2.1 Welcome Page

The Welcome Page is the landing page of the expense analysis application. It introduces users to the system with a clean and inviting interface, typically featuring a greeting message (e.g., "Welcome to Expense Tracker"), a tagline or brief overview of the app's purpose (e.g., "Track and Analyze Your Expenses with Ease"), and prominent buttons or links to either log in or register.

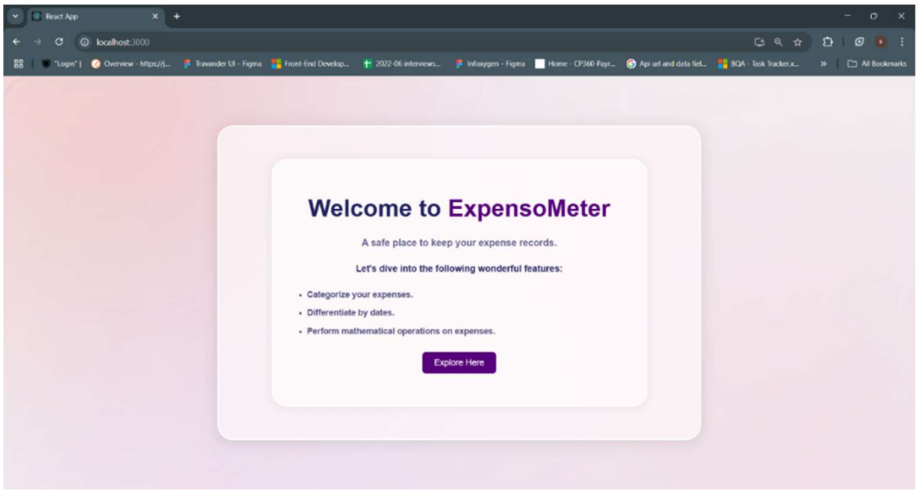

Figure 5.1: Welcome Page

5.2.2 Account Type Page

After clicking "Explore Here" on the Welcome Page, the next page displayed is the Account Type Page, which allows users to specify their account preference. This page features a simple form with the title "Select Account Type," prompting the user to choose between two options: "Business" for professional use or "Personal" for individual use, followed by a "submit" button to proceed.

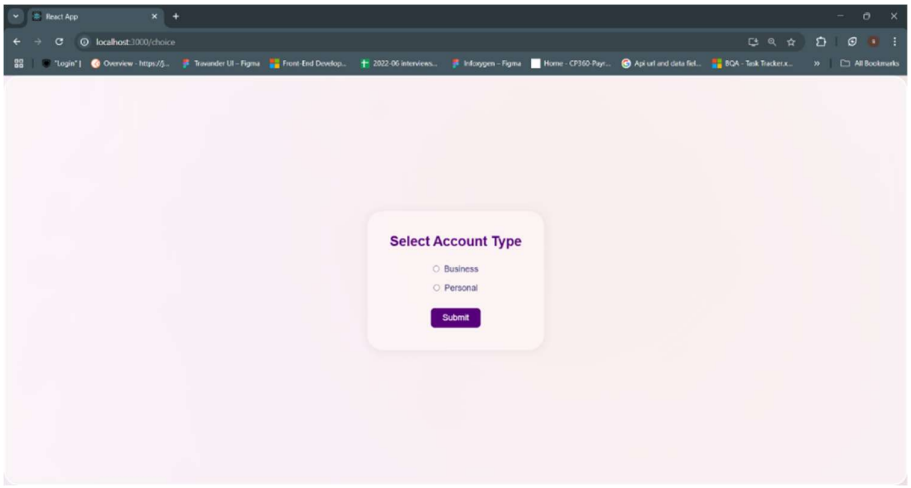

Figure 5.2: Account Type Page

5.2.3 Register Page

After selecting an account type and clicking "Submit" on the Account Type Page, the user reaches the Register Page. It includes a form for entering a username, email, password, and password confirmation, a "Register" button, and a "Already have an account? Login" link for existing users.

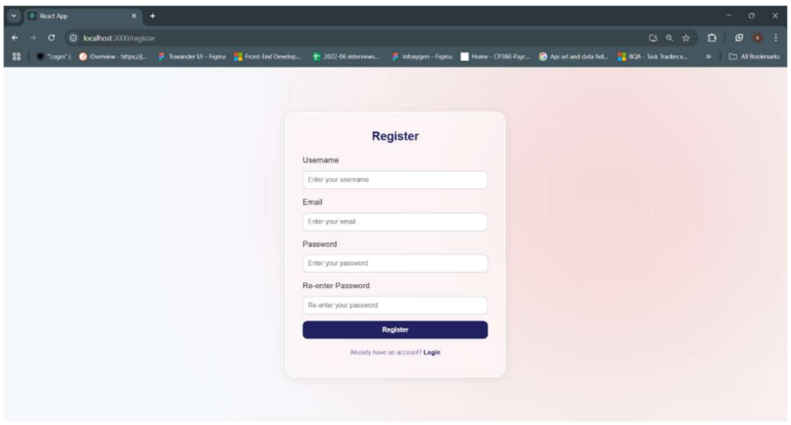

Figure 5.3: Register Page

5.2.4 Login Page

Clicking "Already have an account? LOGIN" on the Register Page leads to the Login Page, titled "ExpensoMeter Login." It features a form for entering an email and password, a "Login" button, and a link "New to ExpensoMeter? Create an account" for new users.

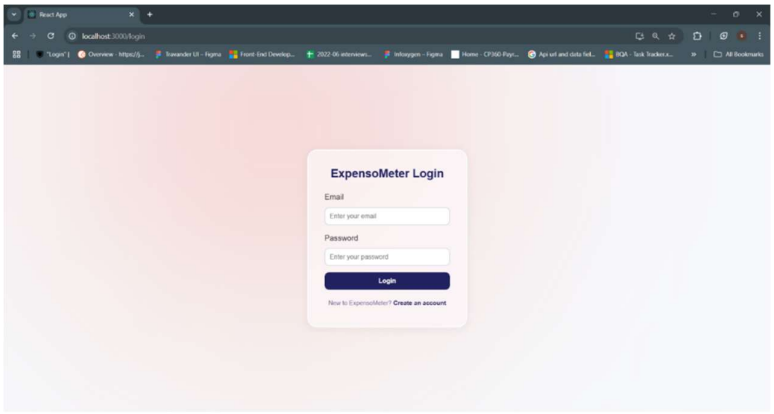

Figure 5.4: Login Page

5.2.5 Expenses Page

For personal accounts, it appears as "Expenses" and displays the total personal expense, a form to add new entries (amount, date, category), and an "Add Expense" button. For business accounts, the same page is titled "Business Expenses" and shows the total business expense along with a form that includes title, amount, date, and category, plus an "Add Business Expense" button. In both cases, a consistent sidebar provides navigation options like Dashboard, Transactions, Incomes, and Analysis.

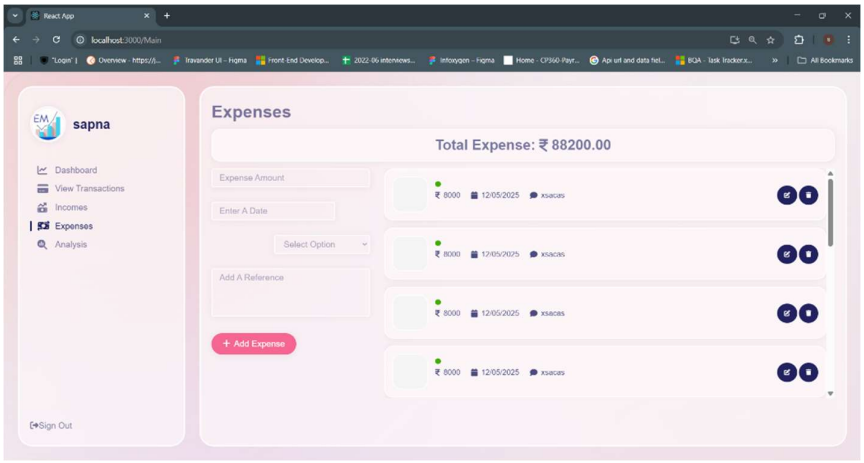

Figure 5.5: Personal Expenses Page

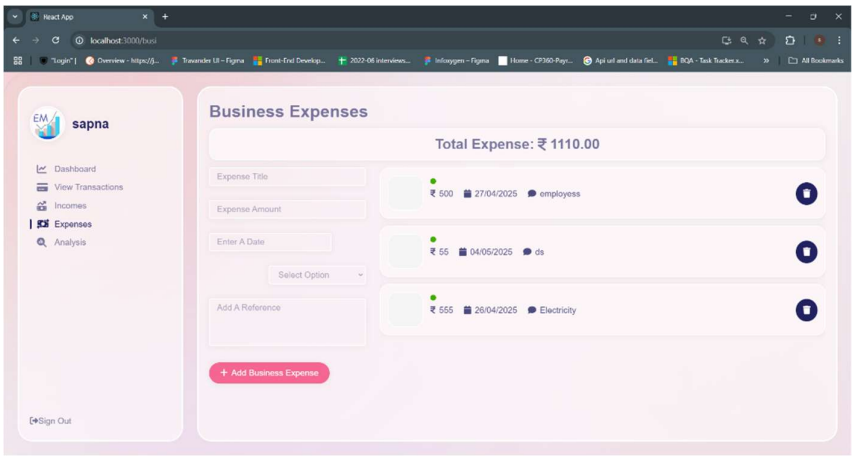

Figure 5.6: Business Expenses Page

5.2.6 Income Page

For personal accounts, it is titled "Income" and displays the total personal income along with a form to add new entries (amount, date, source), and an "Add Income" button. For business accounts, the page is labeled "Business Income" and shows the total business income with a form that includes title, amount, date, and source, accompanied by an "Add Business Income" button. A shared sidebar remains consistent across both views, offering navigation options such as Dashboard, Transactions, Expenses, and Analysis.

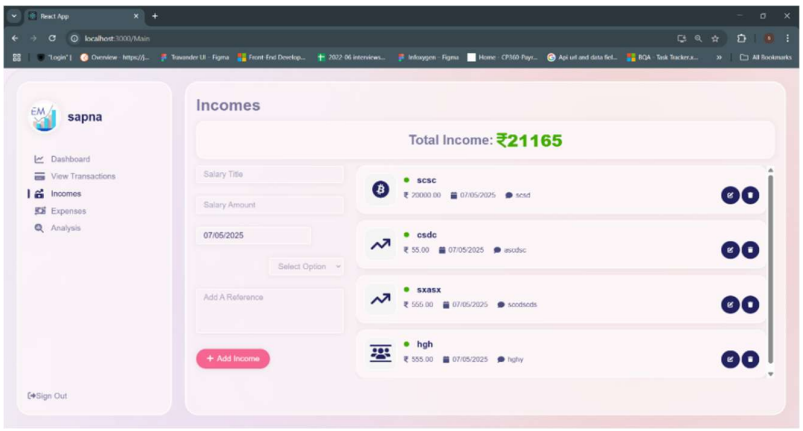

Figure 5.7: Personal Income Page

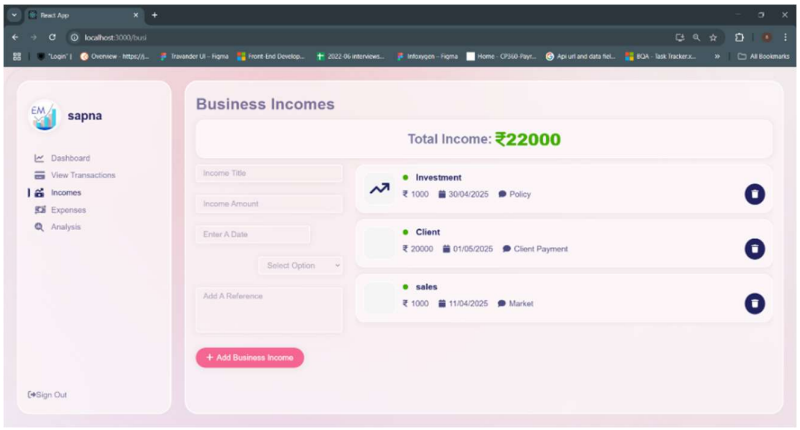

Figure 5.8: Business Income Page

5.2.7 Analysis Page

The Analysis page provides a comprehensive overview of expense trends and predictions. It features visual insights such as a pie chart for Expense Distribution by Category (INR) and a bar graph for Monthly Spending by Category (INR) to analyze spending patterns over time. Additionally, a Prediction for June 2025 highlights the anticipated total expenses, and K-means Clustering Insights categorize user transactions into clusters, offering deeper behavioral analysis. A consistent sidebar allows easy navigation across Dashboard, Transactions, Incomes, and Expenses sections.

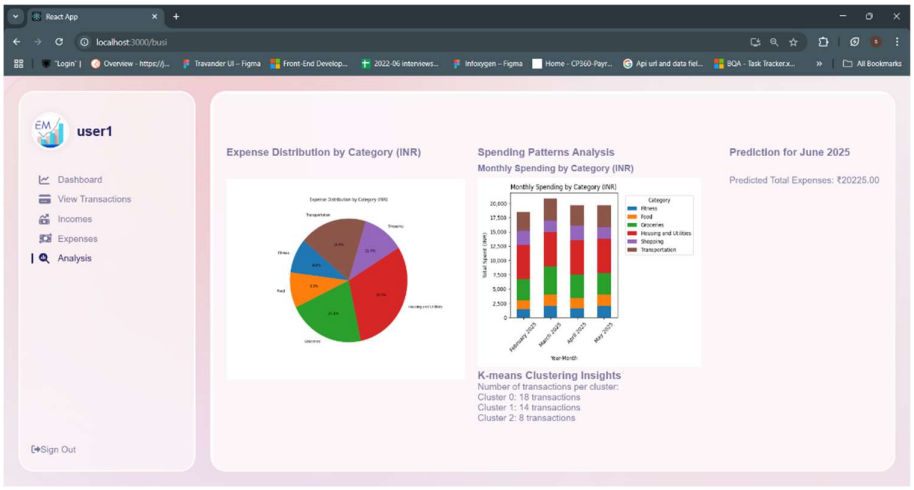

Figure 5.9: Analysis Page

5.2.8 Alerts

The Alert section on the Dashboard page notifies users when they exceed predefined financial limits. Prominently displayed in the top-right corner, it includes warning messages like “You have exceeded your total monthly limit of 10000!” to help users stay within budget. Alongside these alerts, the page also presents a graph of All Transactions, summaries of Total Income, Total Expense, and Total Balance, a Recent History of transactions, and ranges for Salary and Expense.

The sidebar remains consistent with navigation links to key sections like Transactions, Incomes, Expenses, and Analysis.

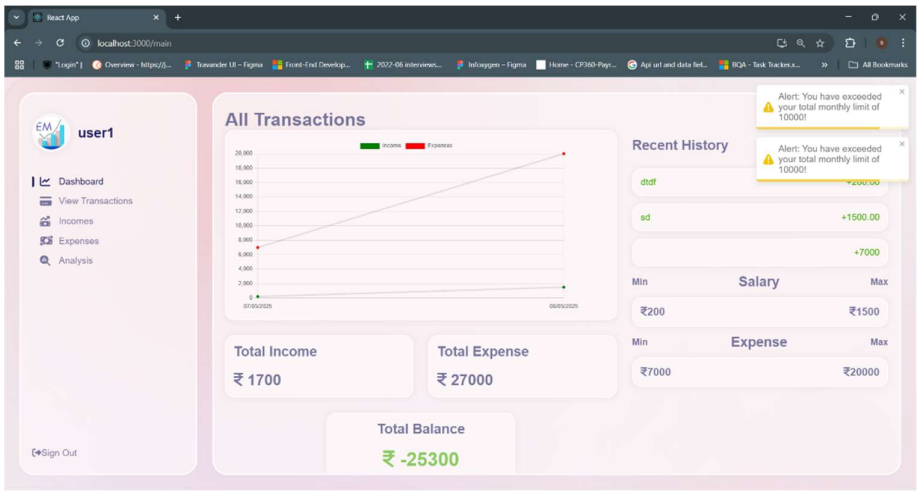

Figure 5.10: Alerts

5.3 Backend Representation

ExpensoMeter uses PostgreSQL as its backend database to manage user data efficiently. This approach enables reliable data storage, robust querying capabilities, and supports scalability for handling complex financial records.

5.3.1 Snapshot of Storage with Brief Description

This snapshot illustrates the database structure of ExpensoMeter, implemented in PostgreSQL within the "expenseTrackerMinorProject" database under the "public" schema. It reveals five key tables: "business" for storing business account details, "businessexpenses" for tracking business-related expenditures, "expenses" for recording personal expenses, "incomes" for managing income entries, and "users" for storing user profile information such as usernames and emails. This relational structure supports efficient data organization, enables seamless querying for

financial analysis, and ensures scalability by separating distinct data categories, facilitating robust management of both personal and business financial records within the application.

Figure 5.11: Database Structure of ExpensoMeter in PostgreSQL

Chapter 6 Conclusion and Future Scope

6.1 Conclusion

The ExpensoMeter project successfully addresses the need for an intuitive and efficient expense tracking system. By leveraging modern web technologies such as React.js and PostgreSQL, it provides users with a responsive, reliable, and user-friendly platform for managing their personal finances.

The application enables users to record and categorize expenses, view detailed summaries, and gain insights through graphical reports. With a focus on simplicity and performance, ExpensoMeter delivers key features such as real-time updates, offline storage, and personalized financial tracking without the complexity of traditional database systems.

Its component-based architecture ensures easy maintenance and scalability, while the use of PostgreSQL ensures reliable data access, persistence, and secure management of user information.

6.2 Future Scope

While the core features of ExpensoMeter are fully functional, there are several enhancements planned for future development to expand the system's capabilities:

- **Machine Learning-Based Financial Insights**

  Implementation of ML algorithms to analyze user spending habits, forecast future expenses, and offer personalized budgeting recommendations.

- **AI-Powered Suggestions and Alerts**

  Integration of intelligent alerts to notify users when they approach or exceed budget limits, and suggest cost-cutting measures based on historical data.

- **Cloud Integration**

  Option to sync data with cloud storage for backup and multi-device access while still offering local mode for privacy-focused users.

- **Enhanced Security Features**

  Addition of encrypted storage, two-factor authentication (2FA), and session management to further strengthen data security.

- **Mobile App Version**

  Development of a cross-platform mobile version using React Native for better accessibility and real-time tracking on the go.

- **Export and Reporting Tools**

  Ability to export expenses to PDF/CSV format and generate downloadable financial reports for tax or personal use.

In conclusion, ExpensoMeter has laid a solid foundation for intelligent personal finance management and holds promising potential for future expansion and user adoption.

References

[1] Google, "Google Search," Google. [Online]. Available: https://www.google.com. [Accessed: Jan. 21, 2025].

[2] W3Schools, "HTML, CSS, JavaScript Tutorials," W3Schools. [Online]. Available: https://www.w3schools.com/. [Accessed: Jan. 21, 2025].

[3] React, "React: A JavaScript library for building user interfaces," React. [Online]. Available: https://reactjs.org/. [Accessed: Jan. 21, 2025].

[4] Stack Overflow, "Developer Community," Stack Overflow. [Online]. Available: https://stackoverflow.com/. [Accessed: Jan. 21, 2025].

[5] GitHub, "GitHub: Where the world builds software," GitHub. [Online]. Available: https://github.com/. [Accessed: Jan. 21, 2025].

Plagiarism Check