

EXPENSEMATE – AN EXPENSE TRACKER

A PROJECT REPORT

Submitted by

CHAUHAN MALHAR JIGNESHKUMAR

(210130107035)

In partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

in

Computer Engineering

Government Engineering College, Sector-28, Gandhinagar



Gujarat Technological University, Ahmedabad

April, 2025



Government Engineering College, Sector-28

Gandhinagar

CERTIFICATE

This is to certify that the project report submitted along with the project entitled **ExpenseMate - An Expense Tracker** has been carried out by **Chauhan Malhar Jigneshkumar** under my partial fulfilment for the degree of Bachelor of Engineering in Computer Engineering, 8th Semester of Gujarat Technological University, and Ahmadabad during the academic year 2025-2026.

Prof. Nirnanjan Prajapati

Internal Guide

Dr. D A Parikh

Head of Department

CONFIRMATION LETTER



INTERNSHIP ALLOTMENT

Date: - 2/01/2025

TO WHOMSOEVER IT MAY CONCERN

This is to state that **Malhar Jigneshkumar Chauhan**, student representing **Government Engineering College Gandhinagar, Sect-28, Gandhinagar** is assigned Industry Internship as per norms.

We wish him/her all the best to perform in this internship which is to be conducted from 20th January 2025 to 19th April 2025.

For, Grownited Private Limited

Rahul Kirpekar
(Authorised Signature)

301-305, 3rd Floor, Surbhi Complex, Nr. Municipal Market, Chimanlal Girdharlal Road,
Navrangpura, Ahmedabad Gujarat- 380009, 7874014621, hr@grownited.com



Government Engineering College, Sector-28

Gandhinagar

DECLARATION

We hereby declare that the Internship / Project report submitted along with the Internship / Project entitled **ExpenseMate - An Expense Tracker** submitted in partial fulfillment for the degree of Bachelor of Engineering in Computer Engineering to Gujarat Technological University, Ahmedabad, is a bonafide record of original project work carried out by me at **Grownited Private Limited** under the supervision of **Prof. Niranjan Prajapati** and that no part of this report has been directly copied from any students' reports or taken from any other source, without providing due reference.

Name of the Student

Sign of Student

Chauhan Malhar Jigneshkumar

Acknowledgement

I am pleased to present this project report entitled **ExpenseMate – An Expense Tracker**.

It is indeed a great pleasure and a moment of immense satisfaction for me to express my profound gratitude and indebtedness towards my guide **Prof. Niranjan Prajapati**, whose enthusiasm has been a constant source of inspiration for me.

I would also like to extend my sincere thanks to **Dr. D. A. Parikh**, Head of the Department for their valuable guidance and for providing the necessary facilities to successfully complete this project report. My heartfelt thanks to the entire teaching and non-teaching staff for their support in data collection and software assistance, which played a crucial role in bringing this project to completion.

I am deeply grateful to my parents for their unwavering support and encouragement, which has been a prime source of motivation in completing this project without any obstacles.

Lastly, I would like to express my gratitude to my colleagues for their cooperation, useful suggestions, and all those who have directly or indirectly contributed to the successful completion of this project.

Regards,

Chauhan Malhar Jigneshkumar (210130107035)

Abstract

An expense tracker is a tool or application that helps individuals or businesses monitor and manage their financial expenditures. The primary goal of an expense tracker is to provide a clear and comprehensive overview of spending patterns, enabling users to make informed financial decisions.

Typically, it involves logging income and expenses, categorizing transactions, setting budget limits, and generating reports. Advanced trackers may include features such as automated expense entry via bank integrations, visualizations like charts and graphs and alerts for overspending.

By offering insights into financial behavior, an expense tracker promotes better budgeting and financial planning.

Table of Contents

CERTIFICATE.....	ii
CONFIRMATION LETTER.....	iii
DECLARATION	iv
Acknowledgement	v
Abstract	vi
List of Figures	x
List of Tables	xi
Abbreviations.....	xii
Chapter 1 : Overview of Company	1
1.1 Services	1
1.2 Mission and Vision	1
Chapter 2 : Introduction to ExpenseMate.....	2
2.1 Overview	2
2.2 Purpose and Importance	2
2.3 Key Features.....	2
2.4 Scope of Project.....	3
2.5 Technology Review	3
2.6 Literature Review	4
2.7 Summary	4
Chapter 3 : Software & Hardware Requirements.....	5
3.1 Functional Requirements.....	5
3.1.1 Hardware Requirements	5
3.1.2 Software Requirements	5
3.2 Non-Functional Requirements.....	6
Chapter 4 : Process Model.....	7
4.1 Incremental Process Model	7
4.2 Advantages	8
4.3 Disadvantages.....	8
4.4 When to Use This Model?	8
Chapter 5 : System Analysis	9
5.1 Feasibility Analysis	9
5.1.1 Technical Feasibility:.....	9
5.1.2 Economic Feasibility:.....	10
5.1.3 Schedule Feasibility:	10
5.1.4 Operational Feasibility:.....	11
5.1.5 Implementation Feasibility:	11

Chapter 6 : Project Planning	12
6.1 About ExpenseMate	12
6.2 Features of ExpenseMate	13
6.3 Need of an ExpenseMate	13
6.4 Mechanism of ExpenseMate	14
6.5 Transaction List	14
6.6 Add, Edit and Delete Expenses	14
6.7 Feedback System:	15
6.8 Contact us	15
Chapter 7 : System Design	16
7.1 Use Case Diagrams	16
7.1.1 Use Case Diagram of User	16
7.1.2 Use Case Diagram of Admin	17
7.2 Class Diagram	18
7.3 Sequence Diagram	19
7.3.1 Sequence Diagram of User	19
7.3.2 Sequence Diagram of Admin	20
7.4 Activity Diagram	21
7.5 Flow Chart	22
7.6 Entity-Relationship Diagram	23
7.7 Data Flow Diagrams	24
7.7.1 Data Flow Diagram level 0	24
7.7.2 User DFD Level 1	24
7.7.3 Admin DFD Level 1	24
7.8 Abstract Design	25
7.9 Details	26
7.9.1 Frontend Design	26
7.9.2 Backend Design (API Design)	28
7.9.3 Database Design	33
Chapter 8 : Implementation	36
8.1 React Environment Setup	36
8.2 FastAPI Environment Setup	38
8.3 MongoDB Environment Setup	39
Chapter 9 : User Manual	41
Chapter 10 : Testing	49
10.1 Testing Strategy	49
10.1.1 White Box testing	49

10.1.2	Black Box Testing	49
10.2	Test Cases (User):	50
10.2.1	Login Tests:.....	50
10.2.2	Registration Tests:.....	51
10.2.3	Transactions Tests:	51
10.2.4	Budget Tests:.....	52
Chapter 11 : Limitations		53
Chapter 12 : Conclusion and Future Enhancements		54
12.1	Conclusion.....	54
12.2.	Future Enhancements	55
References		56

List of Figures

Figure 4.1	Incremental Model	7
Figure 6.1	ExpenseMate Architecture	12
Figure 7.1.1	Use Case Diagram of User	16
Figure 7.1.2	Use Case Diagram of Admin	17
Figure 7.2	Class Diagram	18
Figure 7.3.1	Sequence Diagram of User.....	19
Figure 7.3.2	Sequence Diagram of Admin	20
Figure 7.4	Activity Diagram.....	21
Figure 7.5	Flow Chart.....	22
Figure 7.6	Entity-Relationship Diagram	22
Figure 7.7.1	DFD Level 0.....	24
Figure 7.7.2	User DFD Level 1	24
Figure 7.7.2	Admin DFD Level 1	24
Figure 7.8	High Level Design ExpenseMate.....	25
Figure 9.1	Landing Page of ExpenseMate.....	41
Figure 9.2	Sign Up Page.....	42
Figure 9.3	Sign In Page	42
Figure 9.4	Verify OTP	433
Figure 9.5	User Dashboard.....	43
Figure 9.6	Transactions Page.....	44
Figure 9.7	Add transaction(Expense/Income)	45
Figure 9.8	Add Category	45
Figure 9.9	Budget Page	46
Figure 9.11	Financial Dashboard Page.....	47
Figure 9.10	Profile Page	47
Figure 9.11	Settings Page	48

List of Tables

Table 0 Abbreviations	xii
Table 7.9.3.1 User Model	33
Table 7.9.3.2 Transactions Model.....	34
Table 7.9.3.3 Category Model	34
Table 7.9.3.4 Budget Model	35
Table 10.2.1 Login Tests.....	50
Table 10.2.2 Registration tests.....	51
Table 10.2.2 Transactions Tests	51
Table 10.2.4 Budget Tests.....	52

Abbreviations

Table 0 Abbreviations

Abbreviation	Full Form
UX	User Experience
UI	User Interface
API	Application Programming Inteface
CRUD	Create, Read, Update, Delete
JSON	JavaScript Object Notation
OTP	One Time Password
FARM	FastAPI, React, MongoDB
FastAPI	Fast Asynchronous Python API
CORS	Cross-Origin Resource Sharing
SPA	Single-Page Application
REST	Representational State Transfer

Chapter 1 : Overview of Company

Grownited Private Limited is a startup cultivation firm based in Ahmedabad, Gujarat, dedicated to transforming innovative ideas into tangible realities. The company emphasizes the significance of youth entrepreneurship in shaping a modern and futuristic India.

1.1 Services

- **Mobile App Development:** Grownited brings concepts to life by developing Android mobile applications, combining passion and creativity with technical expertise.
- **E-Commerce Development:** The company provides e-commerce solutions for both B2B and B2C services, aiming to maximize the benefits of online selling for clients.
- **Website Design:** Understanding the importance of effective communication, Grownited's design team ensures that clients' messages are conveyed clearly and effectively through well-designed websites.
- **Digital Marketing:** Grownited's dedicated digital marketing team strives to deliver top-notch services, ensuring clients receive the best in the industry.
- **Software Development:** Combining design and technology, Grownited offers clients optimal UI/UX experiences and compatible websites and applications, recognizing the importance of a lasting first impression.

1.2 Mission and Vision

- Grownited serves as an open-source organization for individuals and groups who think differently, valuing ideas and striving to bring them to fruition. The company aims to become a hub of creativity and coherence by collaborating with idea cultivators, believing that nurturing the future is a shared responsibility. To support this vision, Grownited offers internships in over 15 categories for individuals aged 14–26, emphasizing the importance of experience in today's global scenario.
- By fostering a culture of innovation and providing comprehensive services, Grownited Private Limited plays a pivotal role in shaping the future of startups in modern India.

Chapter 2 : Introduction to ExpenseMate

2.1 Overview

ExpenseMate is a digital expense tracking solution designed to simplify financial management for individuals and organizations. It provides an intuitive platform to record, monitor, and analyze financial transactions efficiently. By offering features such as categorization, budgeting, and detailed reporting, ExpenseMate enhances financial awareness and helps users make informed financial decisions.

2.2 Purpose and Importance

Managing expenses manually can be tedious and error-prone. Traditional methods lack real-time insights, making it difficult to identify spending patterns and potential savings. ExpenseMate addresses these challenges by offering an automated and streamlined approach to financial tracking. The primary goal is to reduce manual effort, increase accuracy, and improve financial planning for users.

2.3 Key Features

- **Expense Recording:** Users can easily add, edit, and delete expenses with relevant details such as date, category, and amount.
- **Categorization:** Expenses can be classified into predefined categories like food, transportation, rent, and entertainment, allowing for better financial organization.
- **Budgeting:** Users can set monthly or category-based budgets to control their spending and avoid overspending.
- **Visual Analytics:** Interactive charts and reports provide a clear understanding of spending habits and financial trends.
- **Security and Privacy:** Data encryption and secure authentication ensure user privacy and data protection.

2.4 Scope of Project

Managing and tracking daily expenses manually can be time-consuming and prone to errors. Users often struggle to maintain records, categorize expenses, and analyze spending patterns efficiently. With the **ExpenseMate**, users can digitally record and monitor their financial transactions with ease.

The scope of the project includes:

- **User Management:** Providing secure authentication and role-based access to users.
- **Expense Tracking:** Enabling users to log, categorize, and manage expenses efficiently.
- **Budget Management:** Allowing users to set budgets and receive alerts when exceeding limits.
- **Reporting and Analytics:** Generating insightful reports to help users analyze spending patterns.
- **Data Security:** Implementing encryption and security measures to protect user data.

2.5 Technology Review

ExpenseMate leverages modern web technologies to provide a seamless and efficient experience.

- **FastAPI:** A high-performance Python-based framework, offering fast response times and easy API development.
- **React:** A component-based JavaScript library that provides a dynamic and interactive UI for users.
- **MongoDB:** A NoSQL database designed for scalability and flexibility in storing expense records.
- **RESTful API:** Ensures smooth communication between the frontend and backend services.

2.6 Literature Review

Several studies highlight the importance of digital expense tracking in modern financial management:

- **Financial Planning Tools:** Research indicates that automated financial tracking solutions enhance budgeting and financial awareness by up to 40% compared to traditional methods.
- **User Engagement:** Studies show that mobile-friendly and cloud-based financial management tools improve user engagement and encourage proactive spending control.
- **Security in Financial Apps:** Literature emphasizes the role of encryption and secure authentication in maintaining user trust and data confidentiality.
- **Impact of Data Visualization:** Interactive dashboards and real-time analytics help users better understand their financial behaviors, leading to improved decision-making.

2.7 Summary

ExpenseMate is a comprehensive and modern expense tracker designed to improve financial management. By leveraging automation, real-time data insights, and user-friendly features, it empowers users to take control of their finances effortlessly. Whether for personal use or business management, ExpenseMate serves as a reliable tool to ensure financial stability and growth.

Chapter 3 : Software & Hardware Requirements

3.1 Functional Requirements

Functional requirements outline the core functionalities of the Expense Tracker website, ensuring seamless data processing, user interactions, and system operations. These requirements define how the system should handle transactions, manage expenses, generate reports, and authenticate users.

Additionally, these functionalities are supported by non-functional requirements, which establish performance, security, and usability constraints to enhance system efficiency and user experience.

3.1.1 Hardware Requirements

- Intel Core i5 processor or higher
- 8GB RAM or more
- 512GB SSD storage
- Windows 11 operating system
- Stable internet connection for hosting and testing

3.1.2 Software Requirements

- Visual Studio Code / PyCharm (for coding and development)
- Postman (for API testing)
- Command Prompt / Terminal (for executing commands and scripts)
- Web Browser (for application testing and debugging)

3.2 Non-Functional Requirements

Non-functional requirements define the quality attributes and constraints of the system, ensuring efficiency, security, and scalability. Unlike functional requirements, which specify what the system does, non-functional requirements determine how well the system performs.

Non-functional requirements include:

- **Reliability** – The system must operate consistently without failures.
- **Scalability** – The system should efficiently handle increased user loads.
- **Performance** – Fast response times and optimized resource utilization are essential.
- **Security** – Robust measures must be in place to prevent unauthorized access and data breaches.
- **Usability** – The interface should be intuitive and user-friendly.
- **Maintainability** – The system should be easy to update, modify, and enhance as needed.

Chapter 4 : Process Model

4.1 Incremental Process Model

The **Incremental Process Model** is a combination of one or more **Waterfall Models**. In this model, the project requirements are divided into multiple **modules**, and each module is developed separately. Once developed, these modules are integrated with other modules to form the complete system.

During the development of each module, the **Waterfall Model** is followed individually. Each developed module is a **standalone feature** and can be delivered to end users for use. Additional modules are integrated incrementally as new features in subsequent releases.

One of the key advantages of the Incremental Model is that **there is no need to wait for all modules to be developed and integrated**. Since each module is independent and has no dependencies on others, the project can be delivered in phases. The process continues until all requirements are met, and the entire system is fully developed.

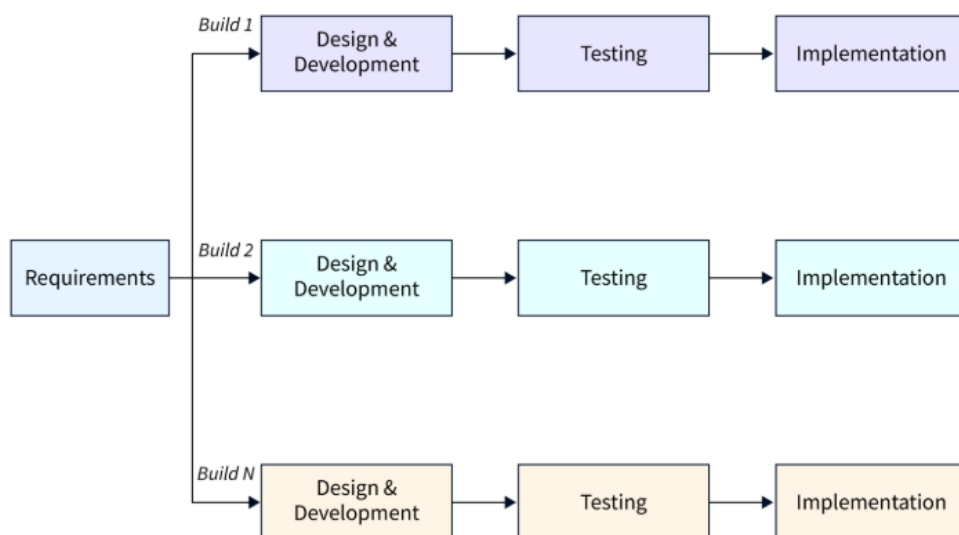


Figure 4.1 Incremental Model

4.2 Advantages

- Enables early delivery of working software within the development lifecycle.
- Provides flexibility, making it less costly to modify scope and requirements.
- Simplifies testing and debugging through smaller iterations.
- Allows customers to provide feedback after each build.
- Reduces initial delivery costs.
- Enhances risk management by identifying and addressing high-risk components in iterations.

4.3 Disadvantages

- Requires good planning and design.
- Demands a well-defined system structure before dividing it into increments.
- Typically incurs higher overall costs compared to the Waterfall model.

4.4 When to Use This Model?

- Core requirements are established, but some details may evolve over time.
- Early market release is a priority.
- The project involves new or emerging technologies.
- Skilled resources are not readily available.
- High-risk features and objectives must be addressed.
- Suitable when system requirements are well-defined and clearly understood.

Chapter 5 : System Analysis

5.1 Feasibility Analysis

The primary goal of the feasibility study is to assess whether developing the **Expense Tracker** system is financially and technically viable. This study involves analyzing the core problem, gathering relevant information, and determining the necessary inputs, processing methods, and expected outputs while considering system constraints.

A preliminary investigation evaluates whether the system will effectively assist users in managing their financial transactions.

The feasibility study consists of three key aspects, each playing a crucial role in determining the success of the system:

5.1.1 Technical Feasibility:

The **Expense Tracker (ExpenseMate)** system is technically feasible as we have the required technology for its implementation. The system is developed using **React.js** for the frontend and **FastAPI** for the backend.

For development, we used **VS Code** and **PyCharm** as primary tools. The backend is powered by **FastAPI**, ensuring fast and efficient API communication. **MongoDB** is used as the database to store all user expenses and transaction records. Authentication and session management are handled securely using token-based authentication.

By leveraging these technologies, the system ensures seamless functionality, data security, and high performance.

5.1.2 Economic Feasibility:

The **Expense Tracker(ExpenseMate)** system is economically feasible as the cost of development is minimal, making it a good investment for users

The system is a **one-time investment** that does not require expensive hardware or software. It is developed using **open-source technologies** like **React.js, FastAPI, and MongoDB**, reducing licensing costs.

Since users can access the system through a **web application**, there is **no need for special tools or devices**. The system improves efficiency by automating expense tracking, **reducing manual effort** and **saving time**, making it a cost-effective solution.

5.1.3 Schedule Feasibility:

Schedule feasibility ensures that the project can be completed within the given time frame. The **Expense Tracker** project is planned for **3 months**, divided into the following phases:

- **Month 1** – Requirement gathering, feasibility study, and system design.
- **Month 2** – Development and implementation.
- **Month 3** – Testing, debugging, and deployment.

The timeline is **realistic and achievable**, ensuring efficient completion within the given period.

5.1.4 Operational Feasibility:

The **Expense Tracker** system is designed to be **user-friendly and efficient**. It allows users to **add, edit, and delete expenses**, track financial records, and categorize expenses easily. The **Admin** has full control over user management, reports, and system settings.

- **No extra training** is required to use the system.
- The system ensures **easy operation with enhanced features**.
- Provides **quick access to financial information** for better expense tracking and management.

5.1.5 Implementation Feasibility:

The organization provides all necessary resources and software for developing the ExpenseMate WebApplication. The required tools and technologies are readily available, making implementation smooth. Additional software and libraries were downloaded from the internet as needed, ensuring feasibility in development.

Chapter 6 : Project Planning

6.1 About ExpenseMate

ExpenseMate is an intuitive and efficient expense tracking application designed to help users manage their personal and business finances with ease. Developed using **FastAPI**, **React.js**, and **MongoDB**, it allows users to **record transactions, categorize expenses, set budgets**, and visualize financial insights through interactive dashboards.

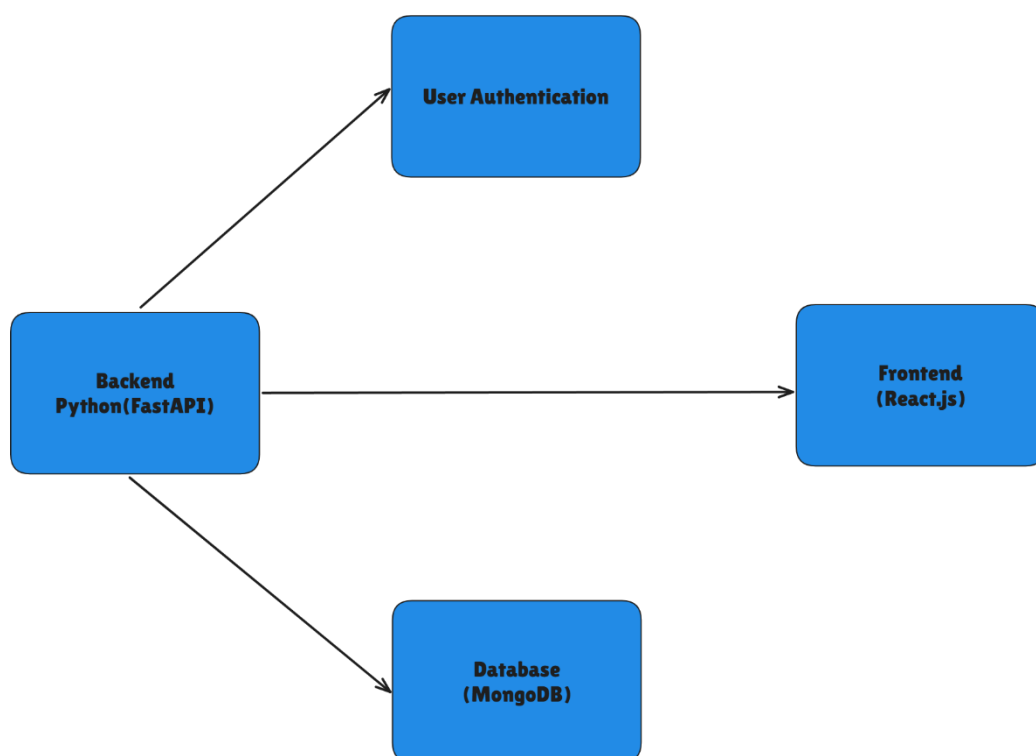


Figure 6.1 ExpenseMate Architecture

With FastAPI for backend, React.js for frontend, and MongoDB for storage, this system ensures fast, secure, and efficient expense tracking.

6.2 Features of ExpenseMate

- **Secure Access:** Only registered users can access the system.
- **Expense & Income Management:** Users can add, edit, and delete their income and expense records.
- **Category Customization:** Users can create and manage categories for expenses and income.
- **Budget Tracking:** Users can set monthly budgets and monitor their spending against them.
- **Visual Analytics:** A dashboard with visual reports helps users analyze financial data.
- **Data Privacy & Security:** Secure authentication mechanisms ensure user data is protected.
- **Expense Report Exporting:** Users can export reports for better financial planning.

6.3 Need of an ExpenseMate

In today's fast-paced digital world, managing personal and business finances effectively is crucial. Many individuals struggle with tracking their income, expenses, and savings, leading to financial mismanagement.

The Expense Tracker Application provides an intelligent, automated solution for recording, categorizing, and analyzing expenses in real-time. Built using FastAPI (backend), React.js (frontend), and MongoDB (database), the application enables users to track financial transactions seamlessly and maintain control over their budget.

Key Benefits:

- **Time-saving:** Automates expense tracking, reducing manual effort.
- **Reduced financial stress:** Helps users maintain financial discipline.
- **Better decision-making:** Provides clear insights into spending habits.

6.4 Mechanism of ExpenseMate

Users can log into the application and record their income and expenses in different categories. The system automatically analyzes spending patterns and provides insights through visual charts and reports.

Core Functionalities:

- **User Registration & Login:** Secure authentication for user access.
- **Transaction Management:** Users can add, edit, and delete transactions.
- **Category-wise Tracking:** Expenses are categorized for better management.
- **Dashboard & Reports:** Users can view spending trends through interactive charts.
- **Admin Control:** The admin can manage user accounts and monitor the system.

By leveraging **FastAPI, React.js, and MongoDB**, the system ensures fast, secure, and efficient expense tracking.

6.5 Transaction List

The Transaction List displays all financial transactions recorded by users. Each transaction includes:

- Transaction Type : Expense or Income
- Amount
- Category
- Date

6.6 Add, Edit and Delete Expenses

- Users can **add** expenses with multiple details, including amount, category, and date.
- They can **edit** or **delete** expenses when necessary.
- The **admin** has the ability to manage all expenses added by users.

6.7 Feedback System:

Through the feedback system, users can contact the Admin. When the Admin receives a feedback message, they can respond directly to the users. This enables smooth communication between users and the Admin. Additionally, users can provide reviews based on their experience with expenses or financial management features. Admin can view and analyze the feedback given by users for continuous improvement.

6.8 Contact us

Through this page, users can contact the admin by providing suggestions or reporting any issues related to the application. This ensures smooth communication and helps in improving the user experience.

Chapter 7 : System Design

7.1 Use Case Diagrams

A Use Case Diagram represents the interactions between users and the system, showing different functionalities available to users and admins.

7.1.1 Use Case Diagram of User

Use Case Diagram (User)

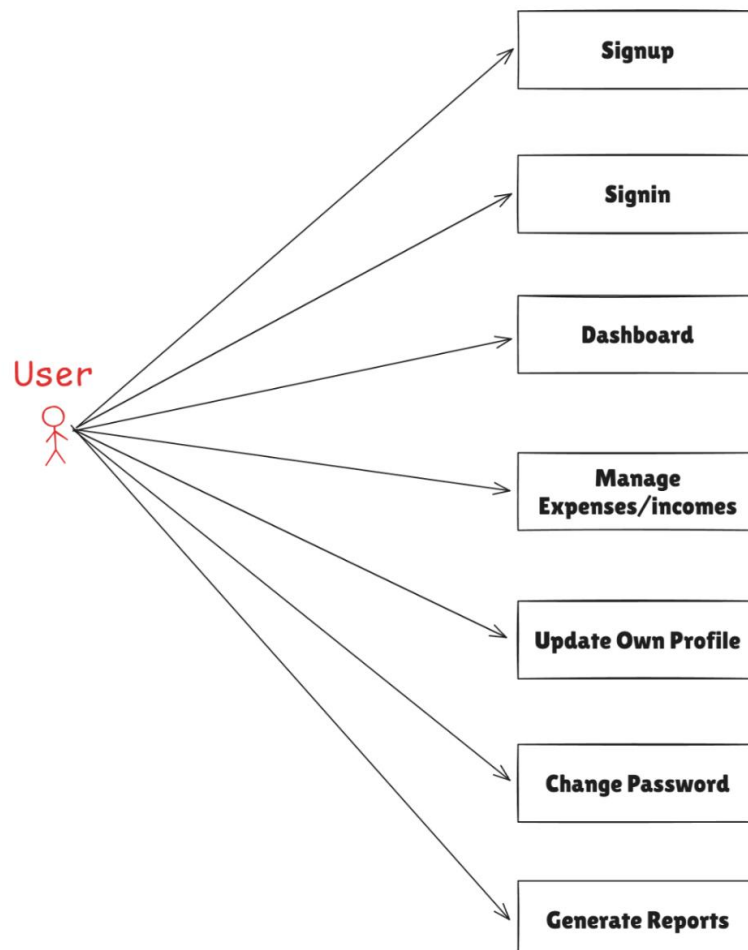


Figure 7.1.1 Use Case Diagram of User

7.1.2 Use Case Diagram of Admin

Use Case Diagram (Admin)

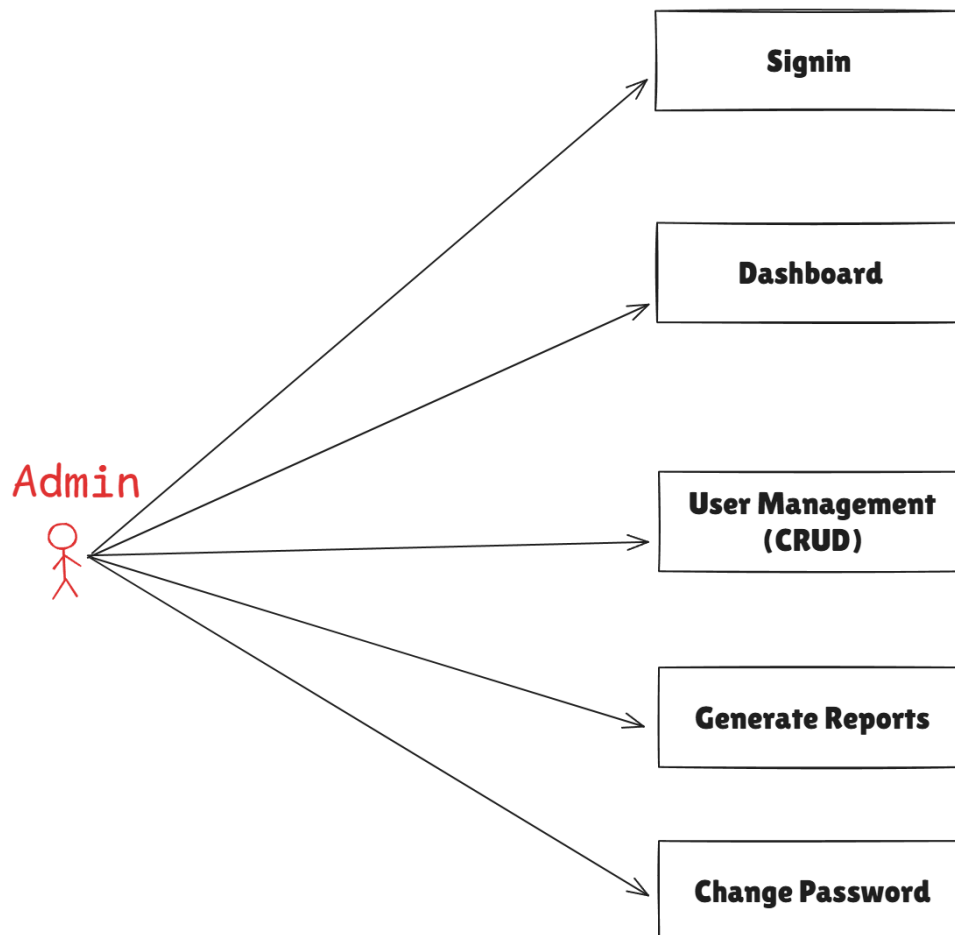


Figure 7.1.2 Use Case Diagram of Admin

7.2 Class Diagram

A Class Diagram defines the structure of ExpenseMate by showing classes, attributes, methods, and relationships between different entities.

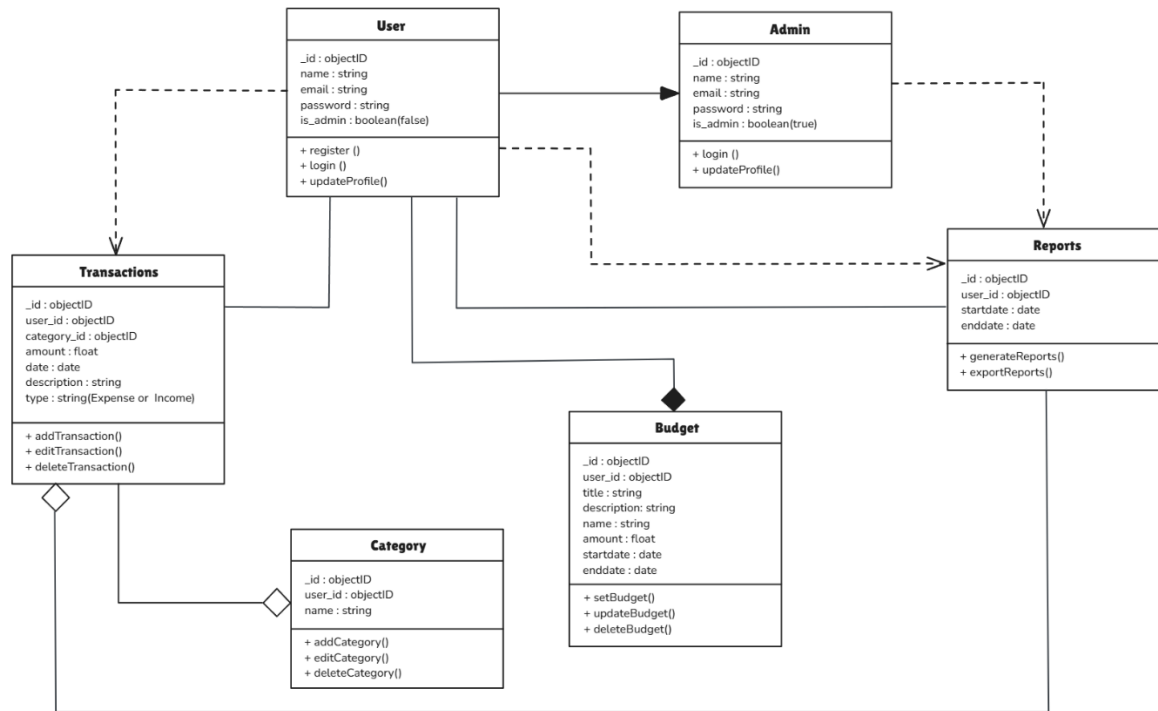


Figure 7.2 Class Diagram

7.3 Sequence Diagram

A Sequence Diagram illustrates how interactions occur over time.

7.3.1 Sequence Diagram of User

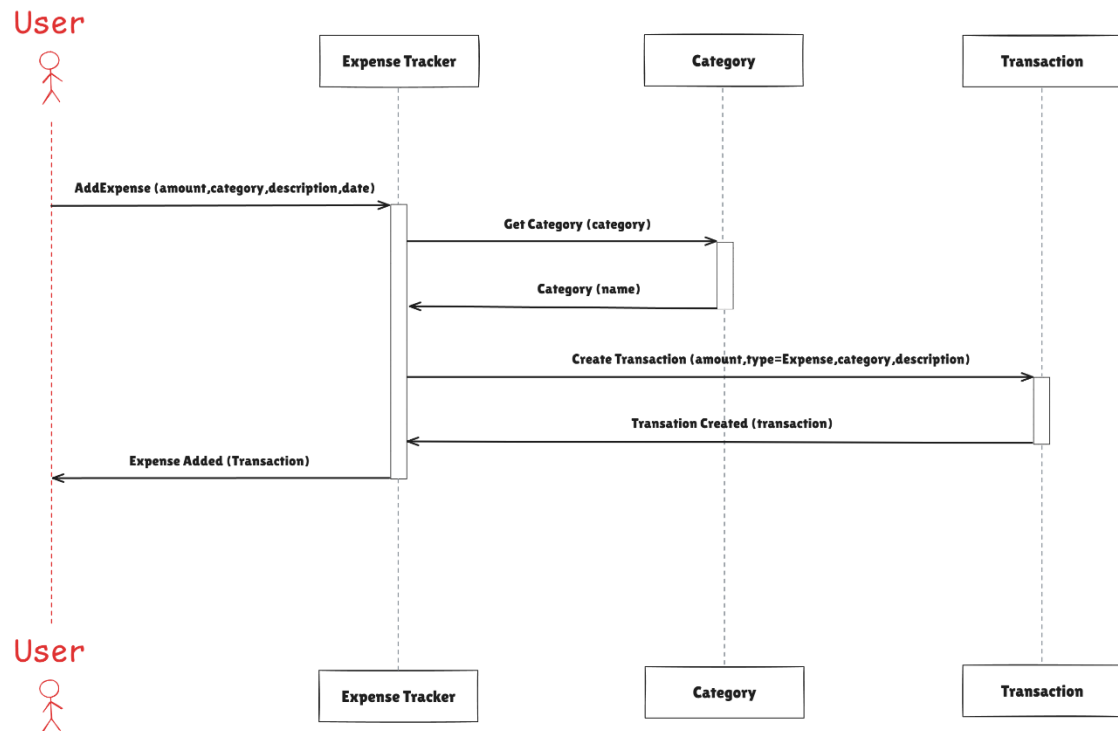


Figure 7.3.1 Sequence Diagram of User

7.3.2 Sequence Diagram of Admin

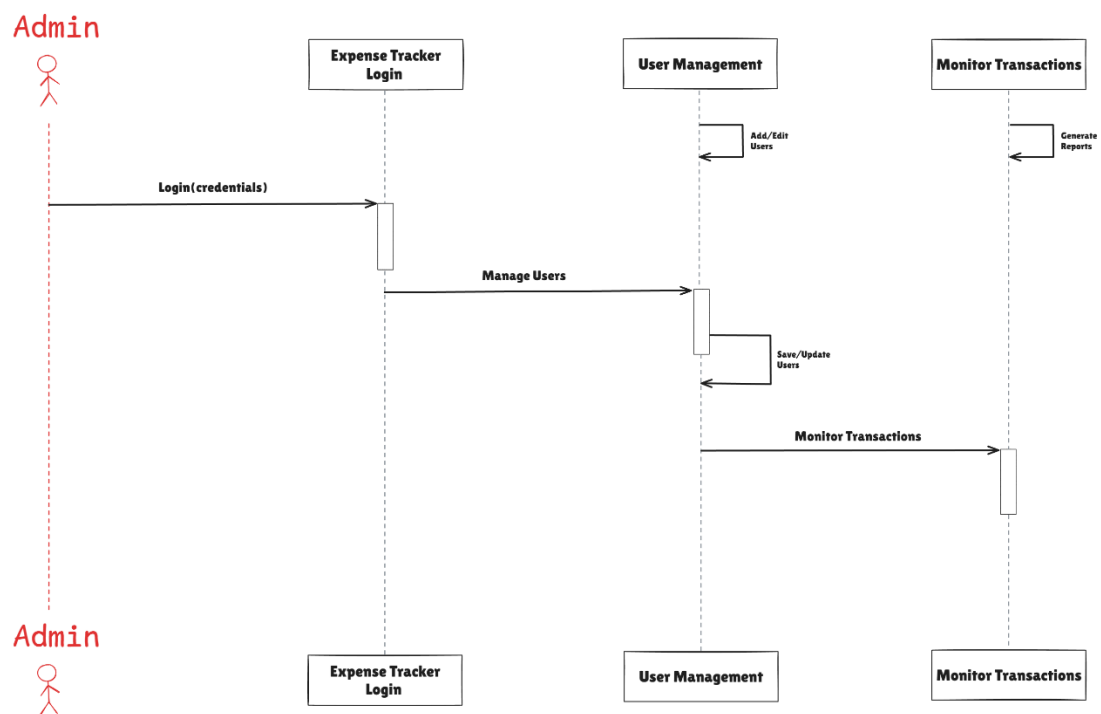


Figure 7.3.2 Sequence Diagram of Admin

7.4 Activity Diagram

An Activity Diagram shows the flow of user actions in the system.

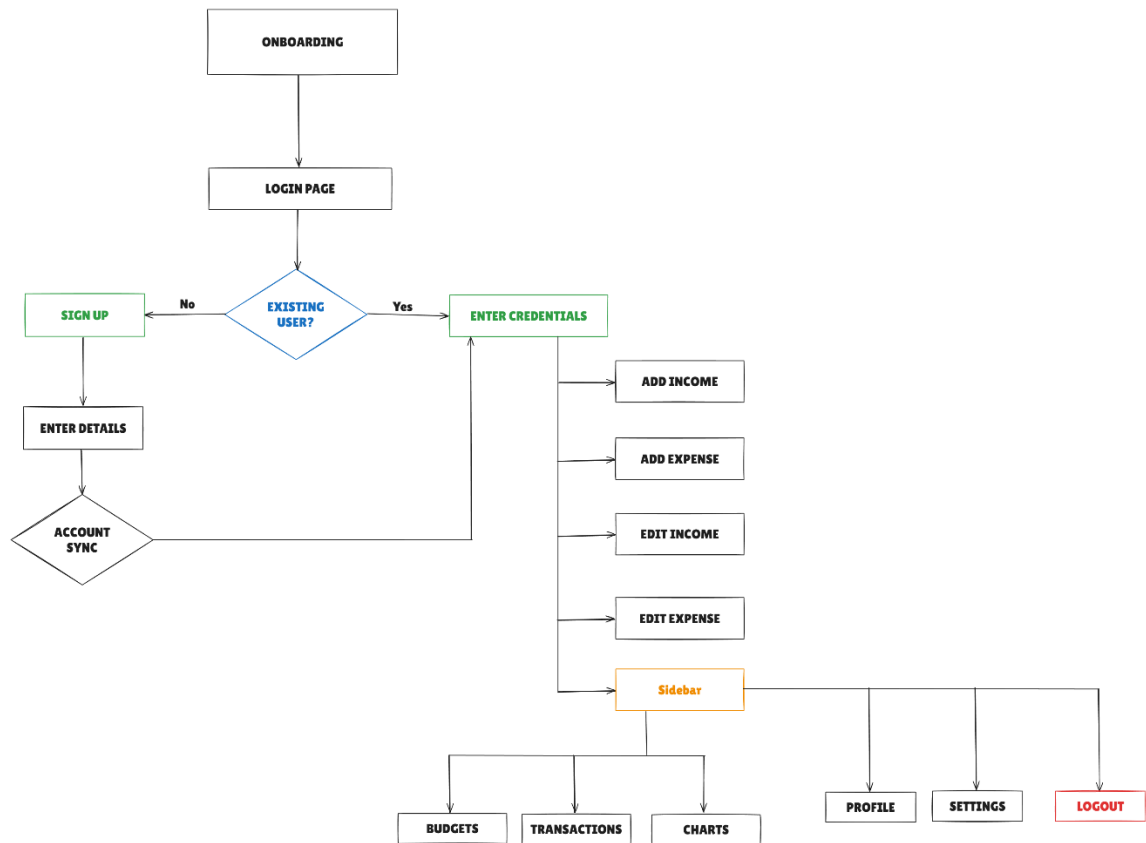


Figure 7.4 Activity Diagram

7.5 Flow Chart

A Flow Chart is a step-by-step representation of system operations.

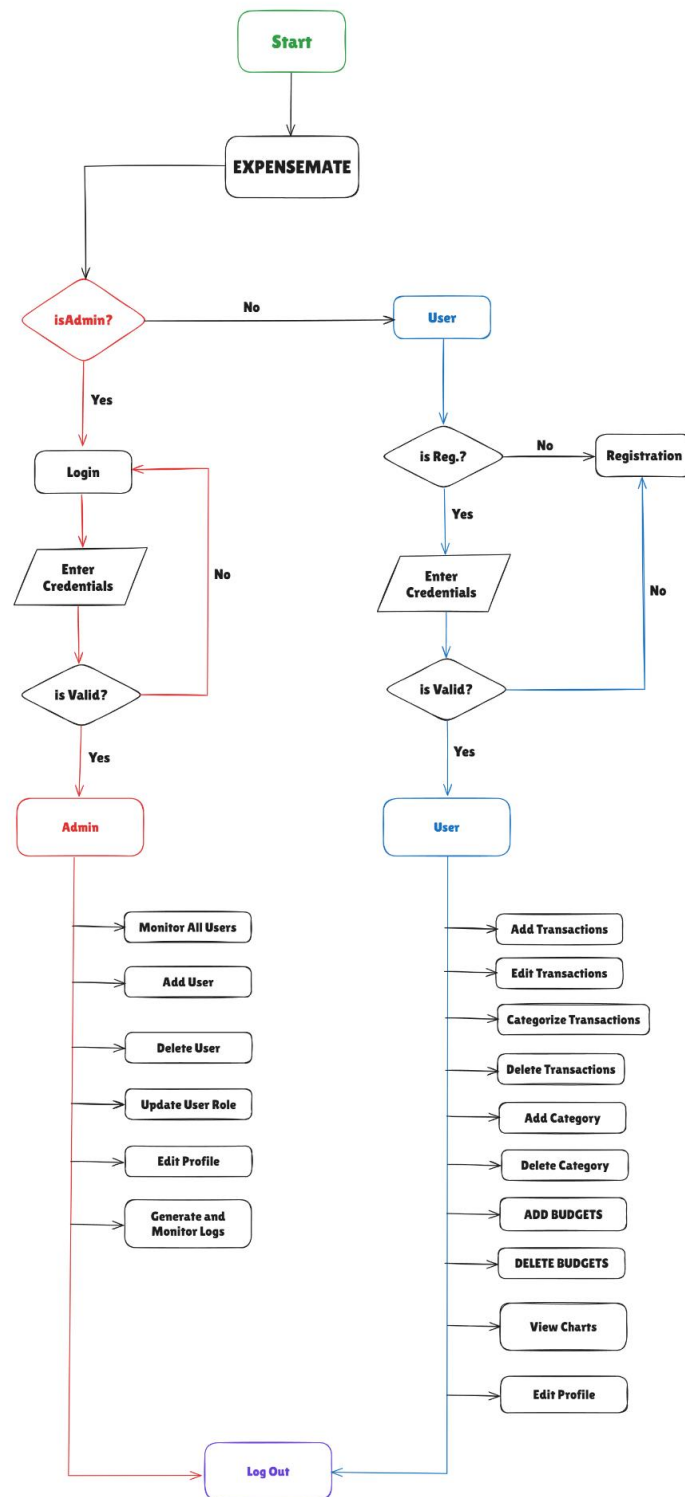


Figure 7.5 Flow Chart

7.6 Entity-Relationship Diagram

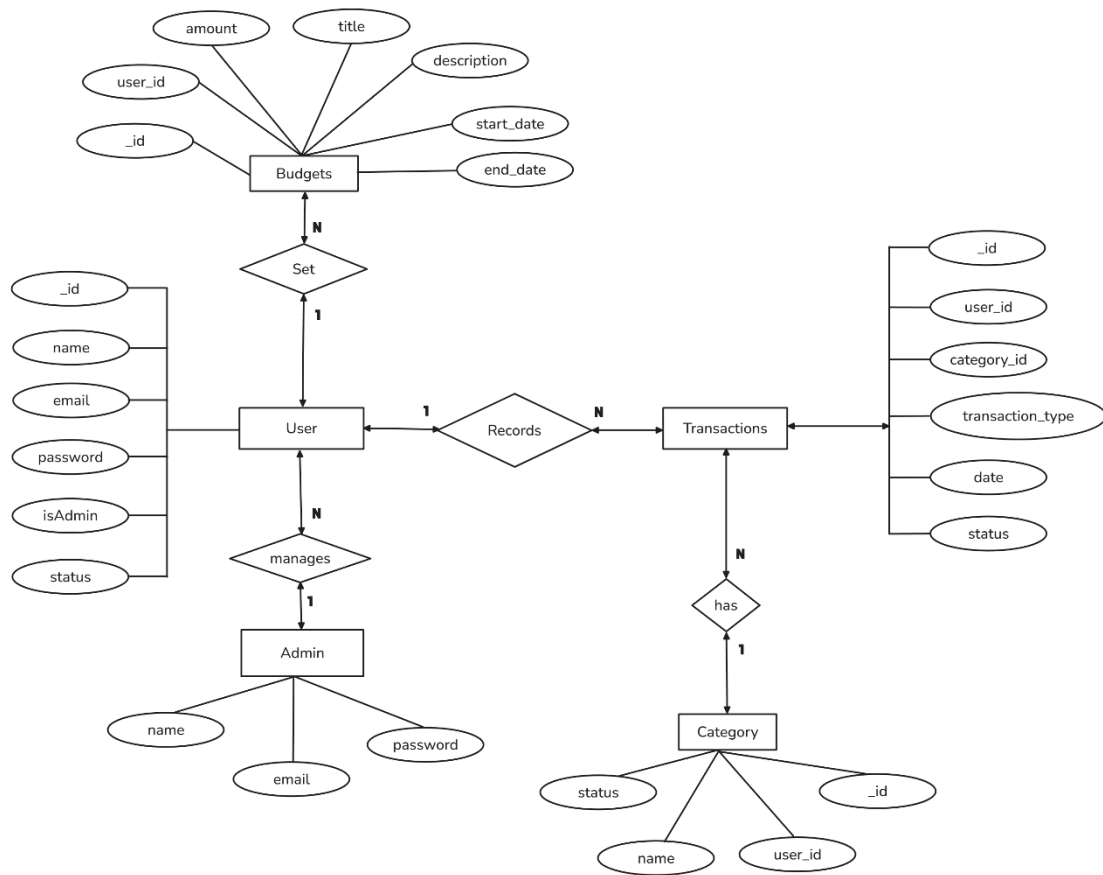


Figure 7.6 Entity Relationship Diagram

7.7 Data Flow Diagrams

A Data Flow Diagram (DFD) represents how data flows within the system.

7.7.1 Data Flow Diagram level 0

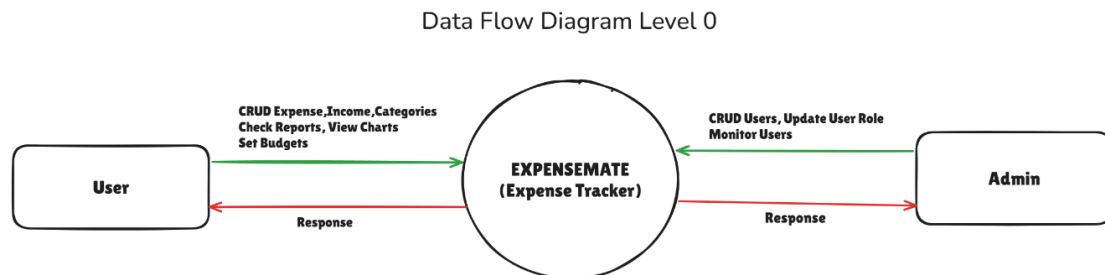


Figure 7.7.1 DFD Level 0

7.7.2 User DFD Level 1

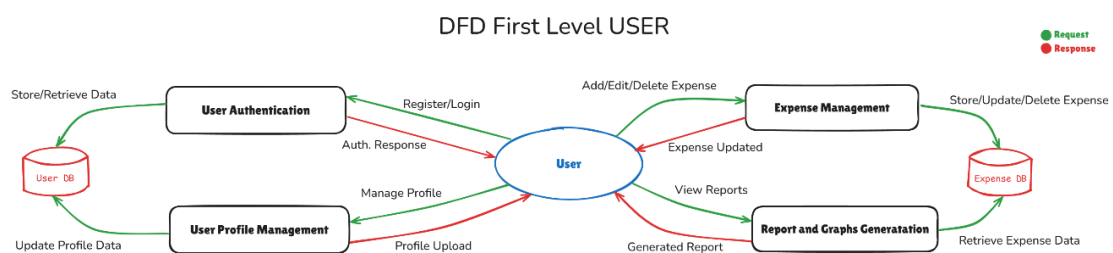


Figure 7.7.2 User DFD Level 1

7.7.3 Admin DFD Level 1

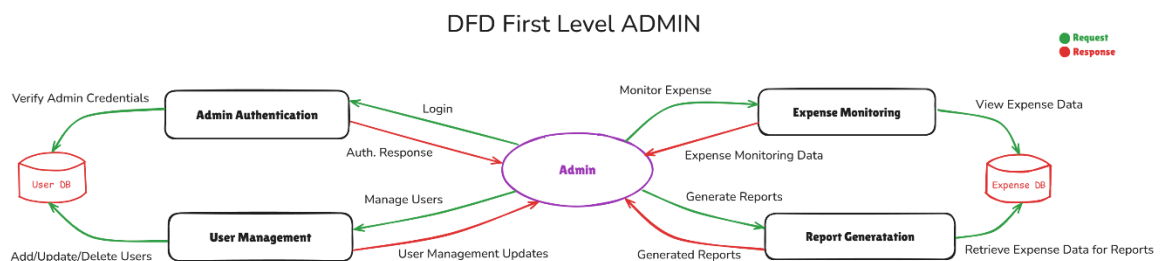


Figure 7.6.3 Admin DFD Level 1

7.8 Abstract Design

ExpenseMate follows a structured three-tier architecture, ensuring modularity, scalability, and security.

1. Presentation Layer (Frontend)

- Built with React and Tailwind CSS for a dynamic, responsive, and user-friendly interface.

2. Business Logic Layer (Backend)

- Developed using **FastAPI**, handling authentication, API requests, and transaction processing efficiently.

3. Data Layer (Database)

- Powered by MongoDB, securely storing user data, expenses, and budgets with optimized retrieval and indexing.

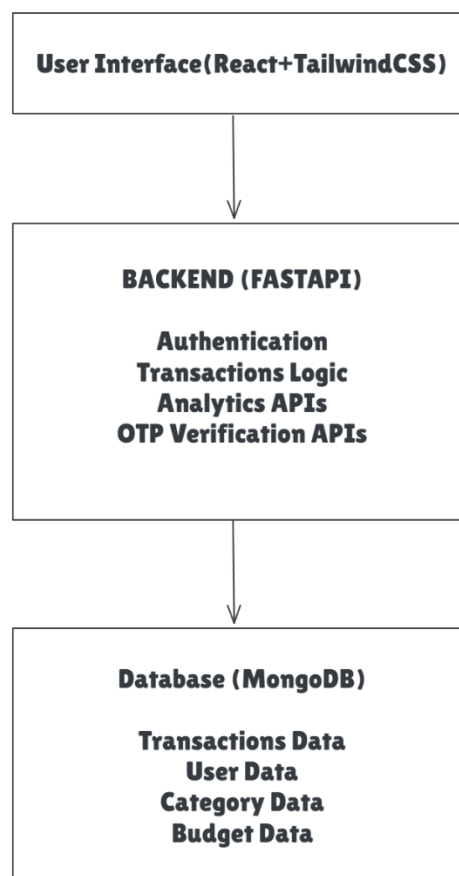


Figure 7.8 High Level Design ExpenseMate

7.9 Details

7.9.1 Frontend Design

Technology Used : React.js (v19.0.0), TailwindCSS(v4.0.9)

Key Components of Frontend:

1. User Components:

- **User Dashboard :** Displays an overview of expenses, budgets, and financial trends, Serves as the main interface for users.
- **Transactions :** Displays a list of all transactions, Supports filtering and sorting by date, category, or amount.
- **Add Expense(Income) :** Pop-up Component used to add Transaction (Expense or Income)
- **Add Category :** Pop-up Component used to add Category, Enables users to create custom spending categories.
- **Add Budget :** Provides input fields for budget amount and category.
- **UserProfile and Settings :** Allows users to update their personal information, Manages user settings, profile picture, and preferences, Allows users to change password.
- **Charts & Reports :** User can access viewing charts, supports filtering.

2. Admin Components :

- **Admin Dashboard :** Displays key metrics such as total users, total expenses, and budget reports.
- **User Management :** Allows the admin to view, edit, and manage all registered users, Enables searching, filtering, and sorting users for easy management.
- **Admin Profile & Settings :** Allows the admin to manage their profile details.

3. Sidebar :

- The Sidebar is a key component of the ExpenseMate application, providing a structured navigation experience for users and admins.
- Enhances user experience by allowing users to expand or collapse the menu.
- Role Based Display : Regular user see expense related components, Admins get User Management, Admin Dashboard etc.
- Sidebar consists of this Navigation Links :
User : Dashboard, Transactions, Charts, Profile, Settings, Budgets
Admin : Dashboard, User Management, Charts, Profile, Settings
- Sidebar also include logout button for logout application.

4. Sign In & Sign Up :

- **Sign In** : The Sign In component allows users to securely log into the ExpenseMate platform, Email & Password Authentication (validated using FastAPI backend) with Error Handling (e.g., incorrect credentials, inactive accounts), Also Includes OTP Based Verification. OTP will send to your entered Email.
- **Sign Up** : The Sign Up component allows new users to create an account on ExpenseMate, Includes User Registration Form (Name, Email, Password), **Redirection to Sign In** after successful registration.

5. Landing Page :

- The Landing Page serves as the first impression of ExpenseMate, showcasing its features and benefits to potential users. It is designed to be engaging, informative, and user-friendly to encourage sign-ups.

7.9.2 Backend Design (API Design)

For ExpenseMate, We'll design RESTful APIs that are intuitive, efficient, and scalable.

RESTful Apis :

A RESTful API (Representational State Transfer API) is a web service that follows REST principles to enable seamless communication between client and server applications. It allows CRUD (Create, Read, Update, Delete) operations using standard HTTP methods:

- **GET** – Retrieve data from the server.
- **POST** – Send new data to the server.
- **PUT/PATCH** – Update existing data on the server.
- **DELETE** – Remove data from the server.

RESTful APIs use **JSON (JavaScript Object Notation)** or **XML** as data exchange formats, ensuring lightweight and efficient communication.

RESTful API Structure in ExpenseMate:

1. Login/Signup API EndPoints:

- **POST** /api/login → User authentication and Admin login
- **POST** /api/signup → New user registration.
- **GET** /api/send-otp/{user_email} → Send OTP mail to user mail
- **POST** /api/verify-otp → OTP verification.

Sample Login Request (JSON):

```
{  
  "email": "user@gmail.com",  
  "password": "user@123*"  
}
```


Sample Login Response (JSON):

```
{
  "success": "true",
  "message": "User Login Successfully",
}
```

2. User APIs and Admin API Endpoints:**For User:**

- **GET** /api/users → Fetch all users.
- **POST** /api/users → Create a new user.
- **GET** /api/users/{user_id} → Retrieve user by ID.
- **PUT** /api/users/{user_id} → Update user details.
- **DELETE** /api/users/{user_id} → Remove user.

Sample Create User Request (JSON):

```
{
  "id": "string",
  "name": "string",
  "email": "string",
  "password": "string",
  "verifyOTP": 0,
  "isAdmin": false,
  "status": true,
  "updated_at": "2025-03-22T11:22:13.023Z",
  "created_at": "2025-03-22T11:22:13.023Z"
}
```

For Admin:

- **GET** /api/admin/users/{admin_id} → Fetch users by admin.
- **PUT** /api/admin/{user_id}/{admin_id} → Update user details

Note: The creation, updating, and deletion of an Admin follow the same APIs as the User APIs. The Admin APIs are specifically used for user management tasks.

3. Transactions API Endpoints:

- **GET** /api/transactions → Fetch all transactions.
- **POST** /api/transactions → Create a transaction.
- **GET** /api/transactions/{transaction_id} → Fetch a transaction by ID.
- **PUT** /api/transactions/{transaction_id} → Update a transaction.
- **DELETE** /api/transactions/{transaction_id} → Delete a transaction.
- **GET** /api/transactions/user/{user_id} → Fetch a transaction by User ID.

Sample Create Transactions Request (JSON):

```
{
  "_id": "string",
  "user_id": "string",
  "category_id": "string",
  "transaction_type": "string",
  "amount": 0,
  "description": "string",
  "date": "2025-03-22T11:27:50.064Z",
  "status": true,
  "updated_at": "2025-03-22T11:27:50.064Z",
  "created_at": "2025-03-22T11:27:50.064Z"
}
```

4. Category API Endpoints:

- **GET** /api/category → Fetch all categories.
- **POST** /api/category → Create a category.
- **GET** /api/category/{category_id} → Fetch a category by ID.
- **PUT** /api/category/{category_id} → Update a category.
- **DELETE** /api/category/{category_id} → Delete a category.
- **GET** /api/category/user/{user_id} → Fetch a category by User ID.

Sample Create Category Request (JSON):

```
{
  "_id": "string",
  "user_id": "string",
  "name": "string",
  "status": true,
  "updated_at": "2025-03-22T12:20:20.915Z",
  "created_at": "2025-03-22T12:20:20.915Z"
}
```

5. Budget Api Endpoints:

- **GET** /api/budget → Fetch all budgets.
- **POST** /api/budget → Create a budget.
- **GET** /api/budget/{budget_id} → Fetch a budget by ID.
- **PUT** /api/budget/{budget_id} → Update a budget.
- **DELETE** /api/budget/{budget_id} → Delete a budget.
- **GET** /api/budgets/user/{user_id} → Fetch a budget by User ID.

Sample Create Budget Request (JSON):

```
{
  "_id": "string",
  "title": "string",
  "user_id": "string",
  "amount": 0,
  "description": "string",
  "start_date": "2025-03-22T12:26:40.012Z",
  "end_date": "2025-03-22T12:26:40.012Z",
  "updated_at": "2025-03-22T12:26:40.012Z",
  "created_at": "2025-03-22T12:26:40.012Z"
}
```

6. Analytics API Endpoints:

- **GET** /api/analytics/transactions/user/{user_id} → Retrieves analytics data for a specific user based on their transactions.
- **GET** /api/analytics/users/ → Fetches analytics data for all users.
- **GET** /api/analytics/transactions/ → Retrieves analytics data for all transactions.
- **GET** /api/budgets/analytics/{user_id} → Fetches budget analytics for a specific user.

Sample Budget Analytics by User ID Response (JSON):

```
{
  "budget_id": "67deae702c11ee9584887d27",
  "title": "March Budget",
  "budget_limit": 1500,
  "total_spent": 150.25,
  "remaining": 1349.75,
  "percentage_used": 10.02,
  "status": "normal",
  "message": "Budget is on track",
  "period": {
    "start": "2025-03-01T00:00:00",
    "end": "2025-03-31T00:00:00"
  }
}
```

7.9.3 Database Design

Why MongoDB for ExpenseMate ?

ExpenseMate is a financial tracking application that requires flexible and efficient data storage. MongoDB, a document-based NoSQL database, is an ideal choice due to the following reasons:

- **Flexible Schema** – ExpenseMate deals with diverse data structures, including user transactions, budgets, and analytics. MongoDB allows dynamic schema changes without altering the database structure.
- **High Performance** – Since MongoDB stores data in JSON-like documents, querying and retrieving user-specific transactions is fast and efficient.
- **Efficient Data Modeling** – Collections in MongoDB can store user transactions, budget analytics, and categories in a way that minimizes data redundancy.

Database Schemas (Data-Dictionary):

1. User Model

Table 7.9.3.1 User Model

Column Name	Data Type	Constraints	Description
id	OBJECT ID	PRIMARY KEY	Unique Identifier for each user
name	STRING	NOT NULL	Full name of User
email	STRING	UNIQUE, NOT NULL	User's Email
password	STRING	NOT NULL	Hashed Password for Authentication
isAdmin	BOOLEAN	DEFAULT FALSE	To Indicate user is a admin
status	BOOLEAN	DEFAULT TRUE	Active(true) or InActive(false)
updated_at	TIMESTAMP	OPTIONAL	Last Update Timestamp
created_at	TIMESTAMP	OPTIONAL	Account Creation TimeStamp

2. Transactions Model

Table 7.9.3.2 Transactions Model

Column Name	Data Type	Constraints	Description
id	OBJECT ID	PRIMARY KEY	Unique Identifier for each Transaction
user_id	OBJECT ID	FOREIGN KEY, NOT NULL	Ref. id of user who created transaction
amount	FLOAT	NOT NULL	Transaction amount
category_id	OBJECT ID	FOREIGN KEY, NOT NULL	Ref. id of category, Category for Transactions
description	STRING	NOT NULL	A brief note for Transaction
transaction_type	STRING	INCOME or EXPENSE, NOT NULL	Used to define its Expense or Income
Date	TIMESTAMP	NOT NULL	Date and Time for Transaction
status	BOOLEAN	DEFAULT TRUE	Used to indicate Transaction Deleted or not
updated_at	TIMESTAMP	OPTIONAL	Last Update Timestamp
created_at	TIMESTAMP	OPTIONAL	Transaction Creation TimeStamp

3. Category Model

Table 7.9.3.3 Category Model

Column Name	Data Type	Constraints	Description
id	OBJECT ID	PRIMARY KEY	Unique Identifier for each user
user_id	OBJECT ID	FOREIGN KEY, NOT NULL	Ref. id of user who created Category
name	STRING	NOT NULL	Name of Category (Food,Rent,Bills)
status	BOOLEAN	DEFAULT TRUE	Used to indicate Category Deleted or not
category_type	STRING	NOT NULL	Category type : Expense or Income
updated_at	TIMESTAMP	OPTIONAL	Last Update Timestamp
created_at	TIMESTAMP	OPTIONAL	Category Creation TimeStamp

4. Budget Model

Table 7.9.3.4 Budget Model

Column Name	Data Type	Constraints	Description
id	OBJECT ID	PRIMARY KEY	Unique Identifier for each Budget
user_id	OBJECT ID	FOREIGN KEY, NOT NULL	Ref. id of user who created Budget
amount	FLOAT	NOT NULL	Name of Category (Food,Rent,Bills)
status	BOOLEAN	DEFAULT TRUE	Used to indicate Category Deleted or not
start_date	TIMESTAMP	NOT NULL	Start date of Budget
end_date	TIMESTAMP	NOT NULL	End date of Budget
updated_at	TIMESTAMP	OPTIONAL	Last Update Timestamp
created_at	TIMESTAMP	OPTIONAL	Budget Creation TimeStamp

Chapter 8 : Implementation

8.1 React Environment Setup

The frontend of **ExpenseMate** is built using **React.js**, a component-based UI framework that enhances modularity and reusability.

Step-by-Step React Environment Setup

1. Initialize Project:

- Install Node.js for javascript runtime environment
- Execute command in Command Prompt:
`npm create vite@latest frontend --template react`

2. Install Dependencies:

- **UI Styling:** tailwindcss
- **Routing:** react-router-dom
- **API Requests:** axios
- **Forms & Validation:** react-hook-form
- **Animations:** framer-motion
- **Charts:** recharts
- **Date Format:** date-fns

Command for Installing Dependencies:

```
npm install tailwindcss @tailwindcss/vite
npm install react-router-dom axios react-hook-form
framer-motion recharts date-fns
```

3. Routing & Navigations:

- Implemented Using **react-router-dom**
- Protected Routes for **authenticated users**

4. Connecting to Backend:

- Apis Calls Handle using **axios**
- Uses **useEffect** and **useState** hooks for data fetching

5. Folder Structure:

```
src/  
| — components/  
|   | — admin  
|   | — common  
|   | — layouts  
|   | — pages  
|   | — user  
| — hooks  
| — App.jsx  
| — index.css  
| — main.jsx
```

6. Running Frontend Server:

- `npm run dev`
- server running on port : `http://localhost:5173`

8.2 FastAPI Environment Setup

The backend is developed using FastAPI, a high-performance Python web framework for APIs.

Step-by-Step FastAPI Environment Setup

1. Create Virtual Environment:

Execute this commands in CMD:

- `python -m venv venv`
- `venv\Scripts\activate`

2. Install Dependencies:

- `pip install fastapi uvicorn motor bcrypt pydantic`

3. Folder Structure:

```
backend/  
|— config  
|— controllers  
|— models  
|— routes  
|— utils  
|— venv  
|— main.py
```

4. Creating Apis (prefix = “/api”):

- User Authentication (/login,/signup, etc.)
- Transactions (/transactions,/transactions/{user_id} etc.)
- Category (/category,/category/{user_id}, etc.)
- Budgets (/budgets,/budgets/{user_id}, etc.)

5. Authentication & Security:

- OTP based Login
- Password hashed via bcrypt

6. Running FastAPI Server:

- `uvicorn main:app --reload`

8.3 MongoDB Environment Setup

The database for ExpenseMate is MongoDB, a NoSQL database that provides flexibility and high performance.

Step-by-Step MongoDB Environment Setup

1. Database Setup:

- **Local:** Install MongoDB Compass
- **Cloud:** Create account in MongoDB atlas for remote storage

2. Install MongoDB Driver:

- `pip install motor` (FastAPI's async MongoDB driver)

3. Database Structure:

Collections :

- `users` (User Details)
- `transactions` (Transactions records)
- `category` (Category details)
- `budgets` (Budget Details)

4. Data Schema (Using Pydantic):

User Schema Example:

```
class User(BaseModel):  
    name:str  
    email:str  
    password:str  
    verifyOTP:Optional[int]  
    isAdmin:Optional[bool] = False  
    status:Optional[bool] = True  
    updated_at:Optional[timestamp]  
    created_at:Optional[timestamp]
```

5. Connecting MongoDB with FastAPI (Local):

```
from motor.motor_asyncio import AsyncIOMotorClient  
client = AsyncIOMotorClient("mongodb://localhost:27017")  
db = client["expensemate-db"]  
users_collection = db["users"]  
transactions_collection = db["transactions"]  
category_collection = db["category"]  
budgets_collection = db["budgets"]
```

Chapter 9 : User Manual

1. Splash screen or Landing page:

A Splash Screen or Landing Page is an introductory screen that appears when an or websapp is launched. It is the first page users see when they visit a website, used to redirect to signup page for new user or login page for regirested users

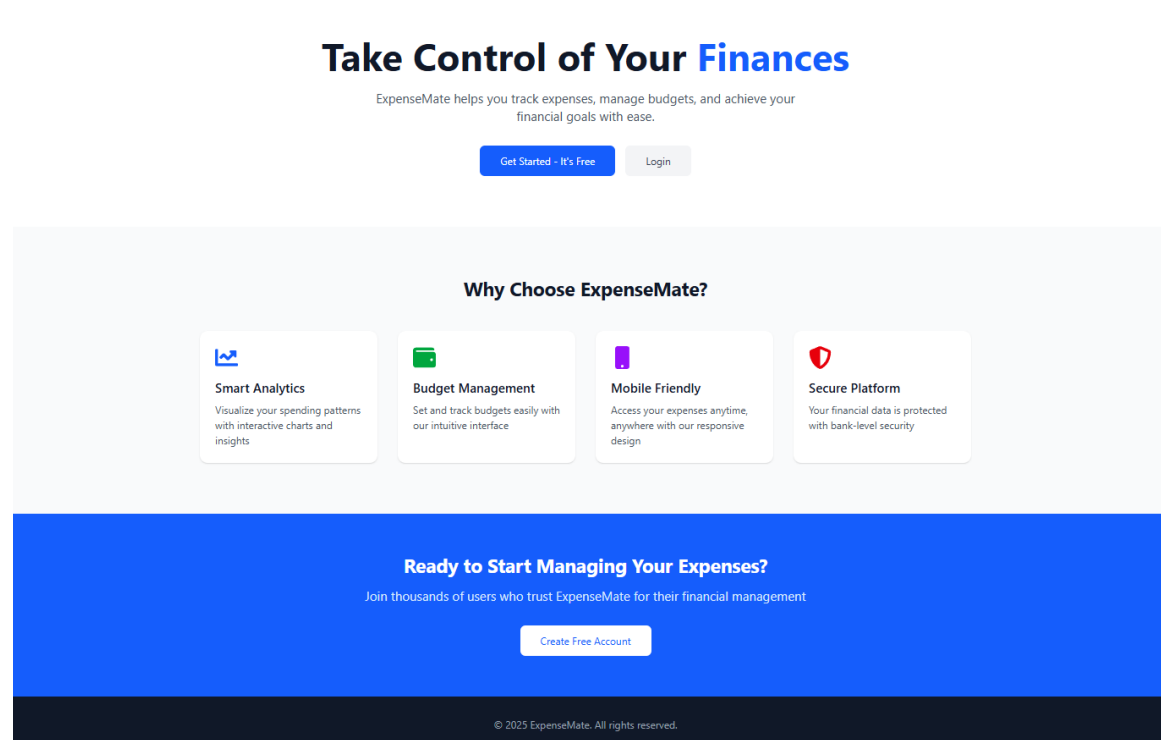


Figure 9.1 Landing Page of ExpenseMate

Features of Landing Page:

- **Clear Headline:** Communicates the main message or purpose.
- **Call-to-Action (CTA):** Encourages users to take a specific action (e.g., "Sign Up," "Get Started").
- **Brief and Focused Content:** Provides essential information without distractions.
- **Mobile Responsiveness:** Ensures a smooth experience across all devices.

2. Sign up Page:

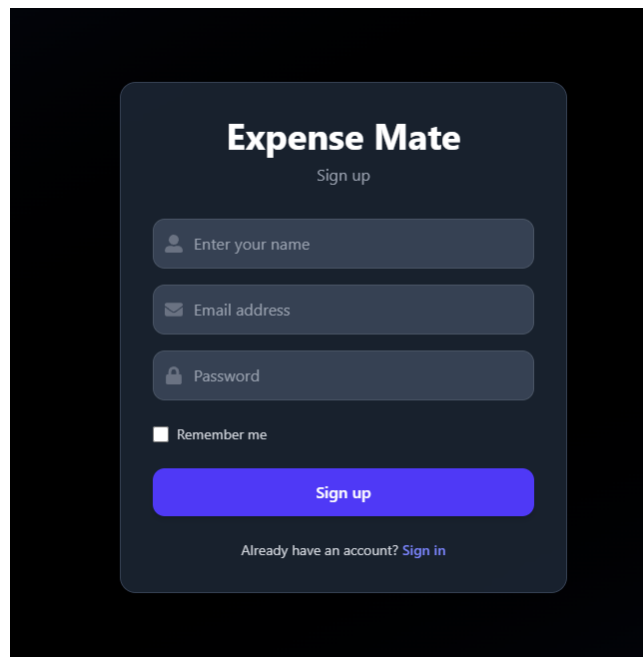
The image shows a dark-themed sign-up page for 'Expense Mate'. At the top, the title 'Expense Mate' is displayed in white, with 'Sign up' written below it in a smaller font. There are three input fields: 'Enter your name' with a person icon, 'Email address' with an envelope icon, and 'Password' with a lock icon. Below these fields is a checkbox labeled 'Remember me'. A large blue button with the text 'Sign up' is positioned below the checkbox. At the bottom, there is a link that says 'Already have an account? Sign in'.

Figure 9.2 Sign Up Page

A Sign-Up Page allows new users to create an account by providing details like name, email, password. If user is already registered, user can sign in by clicking on Sign In button.

3. Sign In Page:

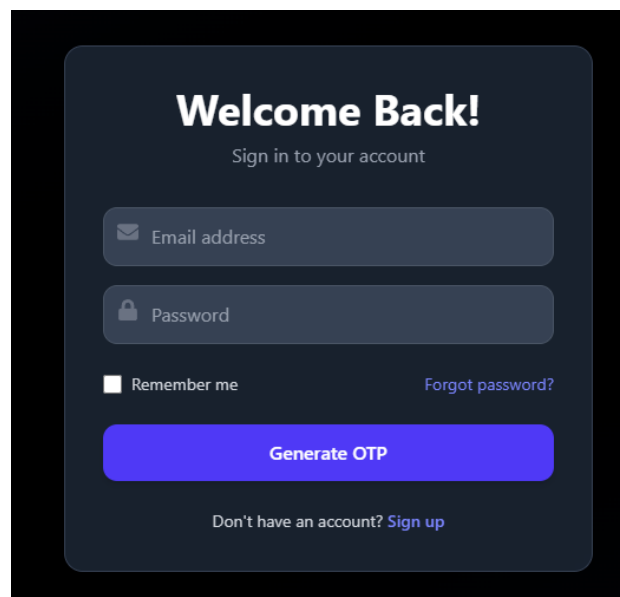
The image shows a dark-themed sign-in page for 'Expense Mate'. At the top, the title 'Welcome Back!' is displayed in white, with 'Sign in to your account' written below it in a smaller font. There are two input fields: 'Email address' with an envelope icon and 'Password' with a lock icon. Below these fields is a checkbox labeled 'Remember me' and a link labeled 'Forgot password?'. A large blue button with the text 'Generate OTP' is positioned below the checkbox. At the bottom, there is a link that says 'Don't have an account? Sign up'.

Figure 9.3 Sign In Page

When a user enters their credentials, an OTP is generated and sent to their registered email. A VerifyOTP pop-ups and the user enter the OTP.

4. Verify OTP:

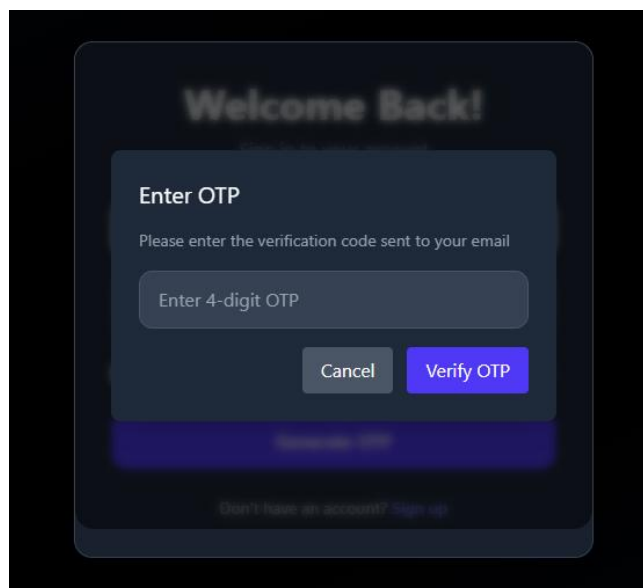


Figure 9.4 Verify OTP

If the OTP is correct, the user is successfully logged in and redirected to the dashboard.

5. User Dashboard:

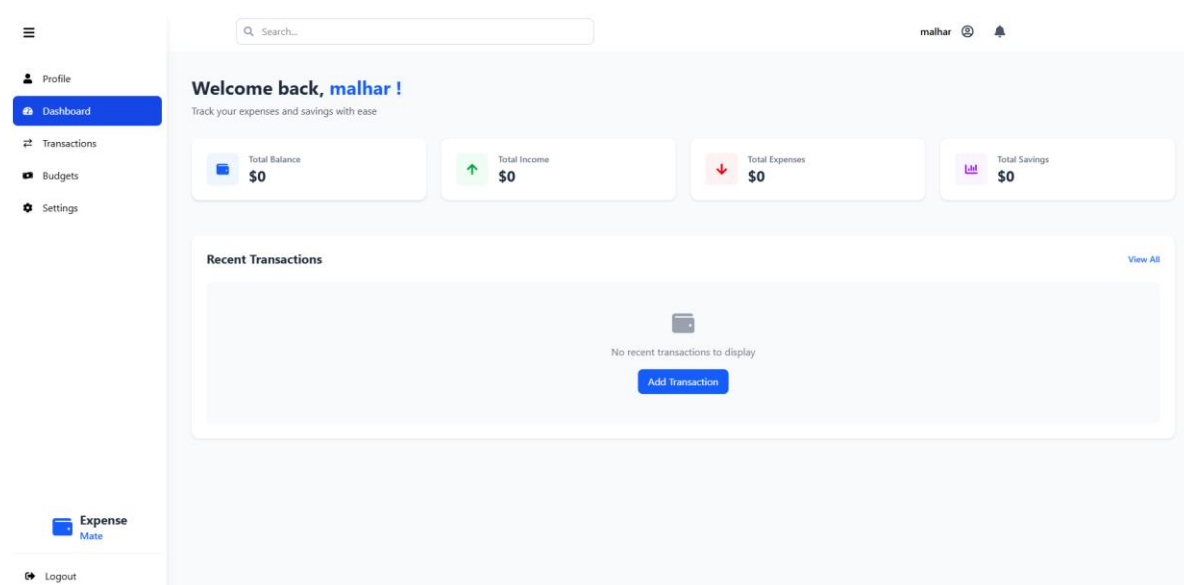


Figure 9.5 User Dashboard

It provides an overview of key financial metrics such as **Total Balance**, **Total Income**, **Total Expenses**, and **Total Savings**. The interface also displays recent transactions, with an option to **add a new transaction**. Sidebar provides quick access to essential sections, including **Transactions**, **Budgets**, **Profile**, and **Settings**.

6. Transactions Page:

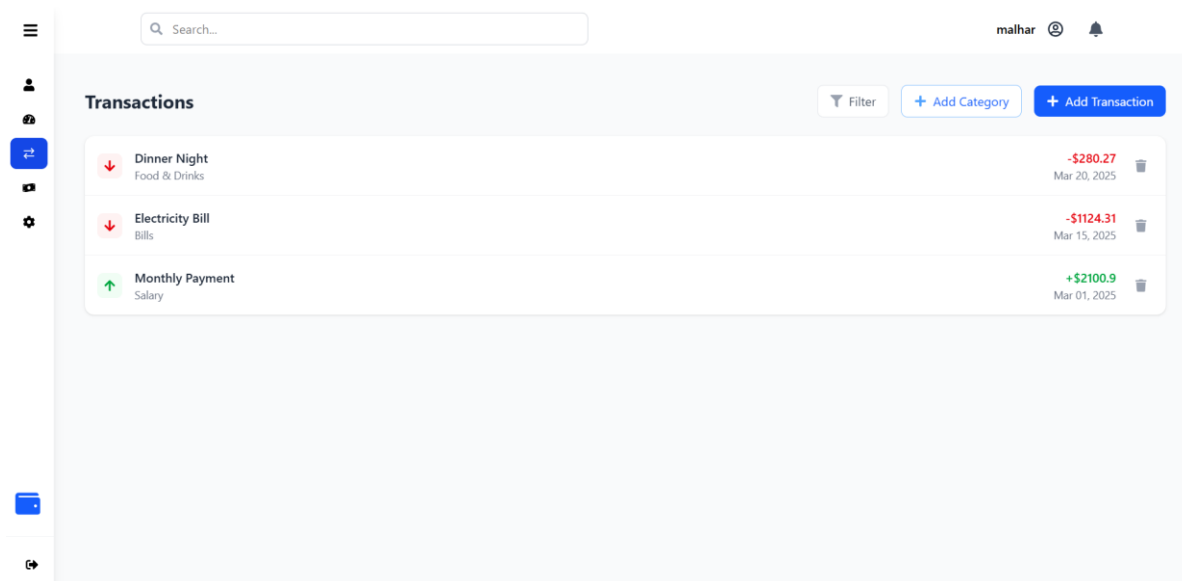
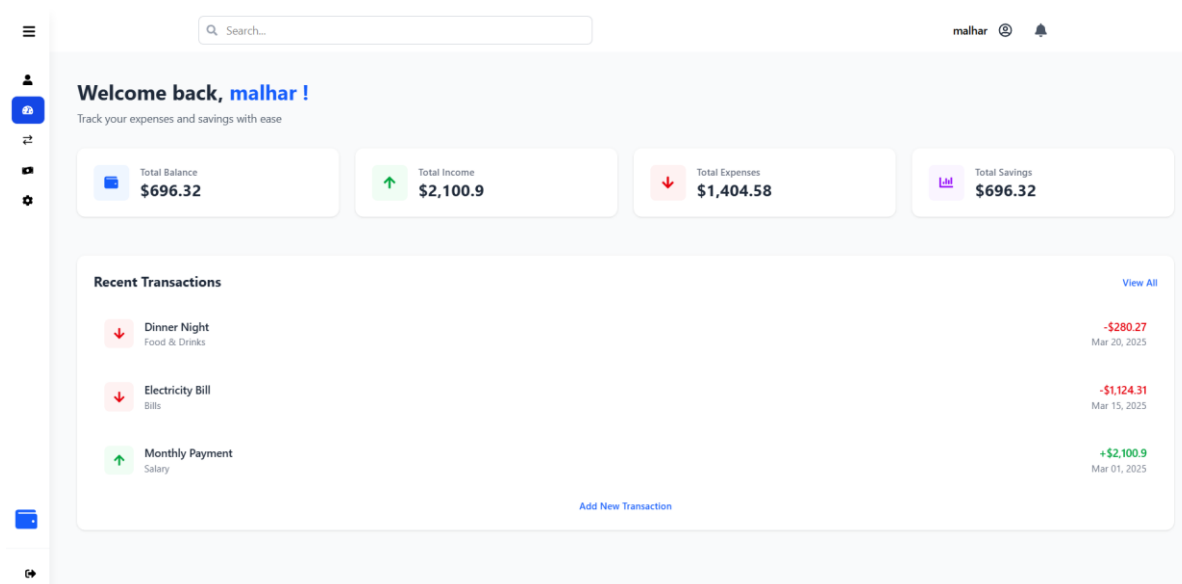


Figure 9.6 Transactions Page

It displays a list of recorded transactions, categorized based on their type, such as **Food & Drinks**, **Bills**, and **Salary**. It has Add Transaction and Add Category to Add new Transactions and Categories.

Dashboard after adding Transactions:



7. Add Transactions (Expense or Income):

The image shows two side-by-side "Add Transaction" modals. The left modal is for "Income" (green) and the right is for "Expense" (red). Both modals have a close button (X) in the top right corner. The left modal has a green "Income" button and a green "Add Income" button at the bottom. The right modal has a red "Expense" button and a red "Add Expense" button at the bottom. Both modals have input fields for Amount, Category, Description, and Date.

Field	Income Modal (Left)	Expense Modal (Right)
Type	Income	Expense
Amount	3500	250.25
Category	Salary	Entertainment
Description	Monthly Payment	Movie Night
Date	01-03-2025	08-03-2025
Action Button	Add Income	Add Expense

Figure 9.7 Add transaction(Expense/Income)

The "Add Transaction" modal allows users to log income (green) or expenses (red) by entering the amount, category, description, and date.

8. Add Category:

The image shows an "Add Category" modal. It has a close button (X) in the top right corner. At the top, it says "Your Categories" and lists "Food" (Expense) and "Salary" (Income). Below this, there is a "Category Name" input field with "Entertainment" entered. Below that is a "Category Type" dropdown menu with "Expense" selected. At the bottom is a blue "Add Category" button.

Figure 9.8 Add Category

The "Add Category" modal allows users to add category by name of category.

9. Create Budget and Current Budgets:

Current Budgets

TITLE	AMOUNT	PERIOD	ACTIONS
No budgets found			

Create New Budget

Budget Title

March Budget

Budget Amount

₹ 2500

☒ Use Month Selection

Select Month

March, 2025

Description

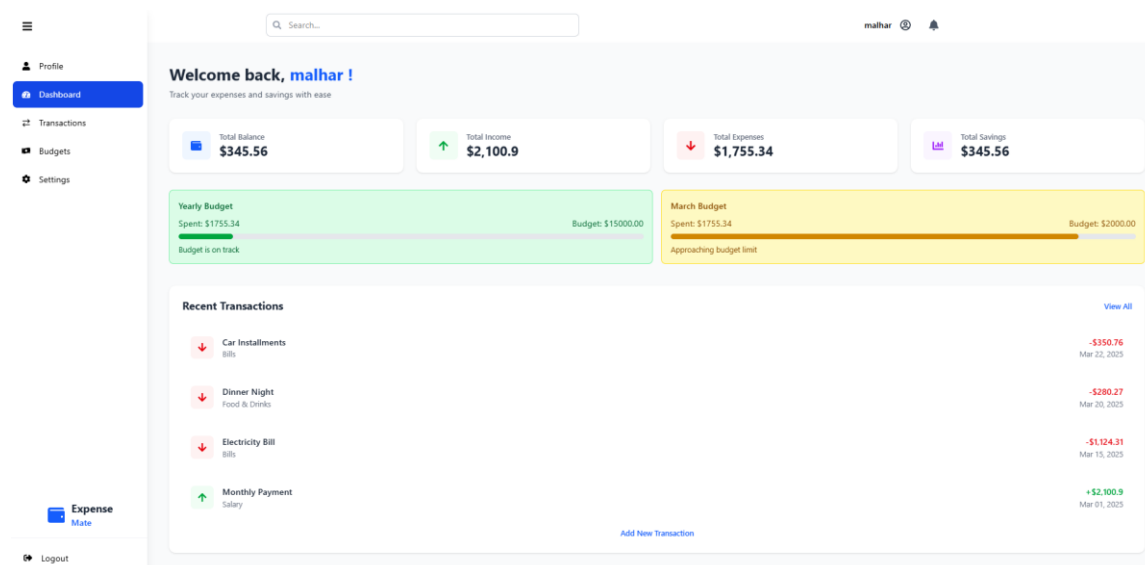
March Budget

Create Budget

Figure 9.9 Budget Page

The **Budget Page** allows users to **create new budgets** and **view existing budgets**. It helps in tracking expenses and managing finances efficiently by setting spending limits for different categories.

Dashboard after adding Budgets:



10. Financial Dashboard and Charts:

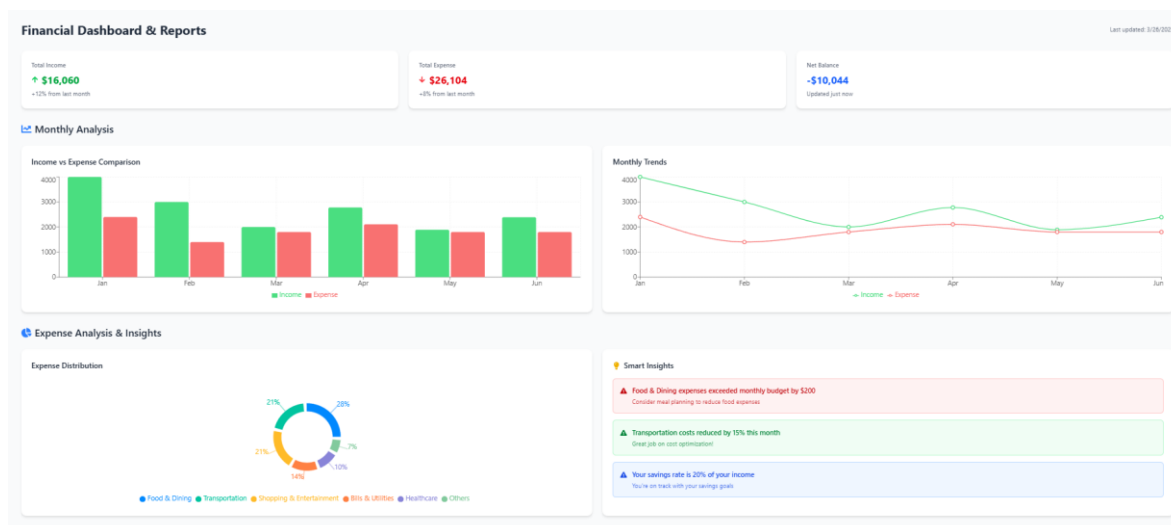


Figure 9.11 Financial Dashboard Page

This Page allows users to view financial insights, user can see Bar Charts and Line Charts of monthly expenses and income. Category wise Pie charts and Budget Insights.

11. Profile Page:

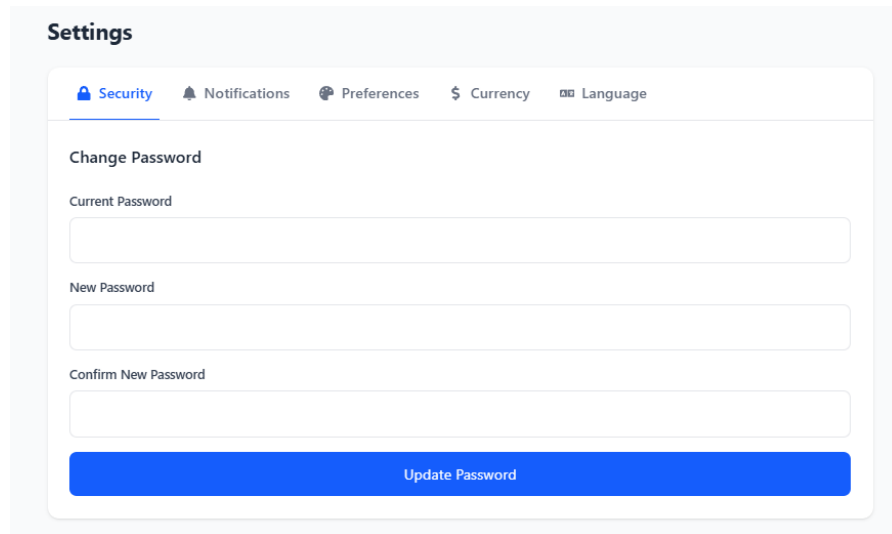
The Profile Page displays user information and allows updates to personal details. It features a user profile card with a placeholder for a profile picture, followed by input fields for Name (malhar), Email (malharchauhan02@gmail.com), and Member Since (Mar 22, 2025). An 'Edit Profile' button is located below these fields. Below the profile card, the 'Account Statistics' section shows three key metrics: Total Transactions (156), Total Savings (\$2,450), and Member Since (Mar 22, 2025).

Metric	Value
Total Transactions	156
Total Savings	\$2,450
Member Since	Mar 22, 2025

Figure 9.10 Profile Page

Profile Page displays user information and allows updates to personal details.

12. Settings Page:



The screenshot displays the 'Settings' page with a navigation bar at the top containing 'Security' (active), 'Notifications', 'Preferences', 'Currency', and 'Language'. Below the navigation bar, the 'Change Password' section is visible, featuring three input fields: 'Current Password', 'New Password', and 'Confirm New Password'. A blue 'Update Password' button is positioned at the bottom of the form.

Figure 9.11 Settings Page

Settings Page provides options to customize preferences, manage security settings, and configure notifications.

Chapter 10 : Testing

Testing is the process of evaluating the **ExpenseMate** system, both manually and through automation, to ensure it meets specified requirements. The goal is to identify and resolve errors, verify smooth functionality, ensure accurate expense tracking, and maintain secure transactions for users.

10.1 Testing Strategy

10.1.1 White Box testing

White-box testing (also known as structural or clear-box testing) involves analyzing the internal structure, code, and logic of ExpenseMate. This approach helps detect hidden errors by testing individual functions, conditions, and loops.

- Performed mainly at the unit level to validate code correctness.
- Ensures proper execution of algorithms, data validation, and security checks.
- May not identify missing requirements or unimplemented features.

10.1.2 Black Box Testing

Black-box testing evaluates the functionality of **ExpenseMate** without examining its internal code. It focuses on validating user requirements and system behavior to ensure a seamless user experience.

- Conducted at different levels, including unit, integration, system, and acceptance testing.
- Ensures correct handling of transactions, budget creation, profile updates, and settings.
- Validates expected outcomes based on real user scenarios and interactions.

10.2 Test Cases (User):

A typical test case includes:

- Test Case ID (Unique identifier)
- Test Scenario (Description of what is being tested)
- Test Steps (Step-by-step instructions)
- Test Data (Input values)
- Expected Result (What should happen)
- Actual Result (What happens during testing)

10.2.1 Login Tests:

Table 10.2.1 Login Tests

Test Case ID	TEST Scenario	Test Steps	Test Data	Expected Result
TC001	Login with valid credentials	1. Open login page 2. Enter valid email & password 3. Click login	Valid email & password	Redirect to dashboard
TC002	Login with incorrect password	1. Open login page 2. Enter valid email but wrong password 3. Click login	Valid email, wrong password	Show error: "Invalid password"
TC003	Login with Empty mail	1. Open login page 2. leave email field empty 3. Click login	Empty email	Show error: "Enter your Email"

10.2.2 Registration Tests:

Table 10.2.2 Registration tests

Test Case ID	TEST Scenario	Test Steps	Test Data	Expected Result
TC004	Register with valid details	1. Open sign up page 2. Enter valid details 3. Click submit	Name, Email, Password	Redirect to Sign in Page
TC005	Register with missing fields	1. Open sign up page 2. Leave fields empty 3. Click submit	Empty fields	Show error: "Fill all required fields"

10.2.3 Transactions Tests:

Table 10.2.2 Transactions Tests

Test Case ID	TEST Scenario	Test Steps	Test Data	Expected Result
TC006	Add an Transaction (Expense or Income)	1. Open "Add Transaction" page 2. Enter amount, category, date 3. Click submit	Amount: \$50, Category: Food Title: Lunch	Alert: "Transaction added successfully"
TC007	Delete an Transaction	1. Open Transactions Page 2. Click delete on an Transaction	Selected Transaction	Alert: "Transaction deleted successfully"

10.2.4 Budget Tests:

Table 10.2.4 Budget Tests

Test Case ID	TEST Scenario	Test Steps	Test Data	Expected Result
TC009	Set budget limit	1. Open Budget page 2. Enter budget amount,title,date 3. Click Create Budget	Amount: \$500 Title: Monthly Date: 15-03-2025	Alert: "Budget created successfully"
TC010	Exceed budget limit	1. Add expenses exceeding budget limit	Budget: \$500, Expenses: \$600	Warning: "You have exceeded your budget!"

Chapter 11 : Limitations

While ExpenseMate is a useful tool for tracking expenses, managing budgets, and generating reports, it does have some limitations that could impact its usability for certain users. Below are some key limitations, along with their potential impact:

- 1. No Bank Integration:** Users must manually enter transactions, as ExpenseMate does not sync with bank accounts or credit cards, Time-consuming for users who make frequent transactions.
- 2. Limited User Roles & Collaboration:** The system may not support multiple users or shared financial tracking, making it less effective for families or businesses.
- 3. No Mobile App:** ExpenseMate is accessible only via the web, with no dedicated mobile application, Mobile users may experience a less optimized UI compared to dedicated apps.
- 4. Basic Security Measures:** ExpenseMate may lack robust security features such as multi-factor authentication (MFA).
- 5. No AI-Based Financial Insights:** The system does not provide AI-driven financial insights, automated expense categorization, or spending recommendations. Lacks AI-powered alerts for unusual spending patterns or budget overruns.

Chapter 12 : Conclusion and Future Enhancements

12.1 Conclusion

Starting the journey of building **ExpenseMate** and gaining practical experience through an internship offers incredible learning opportunities in software development, problem-solving, and financial technology. The process of developing ExpenseMate really showcases the significance of understanding what users need, creating effective financial management features, and constantly improving along the way.

Practicing and committing to continuous learning are vital for honing your skills in web and software development. Getting your hands dirty with real-world projects, collaborating with teams, and tackling real-life challenges will not only enhance your technical skills but also give you a richer understanding of software design, security, and user experience. Utilizing version control, exploring new frameworks, and following industry best practices will pave the way for your long-term growth as a developer.

Lastly, keep in mind that every project and experience serves as a stepping stone toward success. The challenges, setbacks, and learning moments are all part of the ride.

12.2 Future Enhancements

- 1. Multi-Currency Support:** Future versions of ExpenseMate will incorporate support for multiple currencies, allowing users to track expenses in different currencies with real-time exchange rate updates. This will enhance usability for international users and frequent travelers.
- 2. Bank Account Integration:** To streamline transaction tracking, ExpenseMate will integrate with bank APIs, enabling users to automatically sync transactions instead of manually entering them. This will improve accuracy and efficiency.
- 3. Enhanced Security Features:** Security will be strengthened by incorporating multi-factor authentication (MFA), end-to-end encryption, and biometric authentication. These features will safeguard user data and enhance trust in the platform.
- 4. Mobile App Development:** A dedicated mobile application for iOS and Android will be developed, providing on-the-go expense tracking and a seamless user experience across devices. Offline functionality will also be introduced for added convenience.
- 5. Cross-Platform Compatibility:** Enhancing ExpenseMate's compatibility across web, mobile, and desktop platforms will provide a unified and synchronized experience, ensuring users can manage their finances seamlessly across devices.
- 6. AI-Powered Financial Insights:** Use AI to categorize expenses automatically and provide smart financial insights. Identifies unusual transactions and alerts users about potential fraud.
- 7. Community Engagement & Open-Source Contributions:** Future enhancements will include an interactive financial community, allowing users to share financial tips, budgeting strategies, and insights. Open-source contributions will be encouraged to drive innovation and continuous improvement.

References

1. **FastAPI Documentation:** <https://fastapi.tiangolo.com/>
2. **Python Documentation:**
 - <https://www.python.org/doc/>
 - <https://www.w3schools.com/python/>
3. **React Documentation:** <https://react.dev/learn>
4. **TailwindCSS Documentation:** <https://tailwindcss.com/docs>
5. **RestAPIs:** <https://www.geeksforgeeks.org/rest-api-introduction/>
6. **MongoDB:** <https://www.mongodb.com/docs/manual/>
7. **Authentication and Authorization:** <https://www.geeksforgeeks.org/difference-between-authentication-and-authorization/>
8. **Dependencies Documentations:**
 - **react-router-dom:** <https://reactrouter.com/home>
 - **react-hook-form:** <https://www.react-hook-form.com/get-started/>
 - **Framer-motion:** <https://www.npmjs.com/package/framer-motion>
 - **React-icons:** <https://react-icons.github.io/react-icons/>
 - **Pydantic:** <https://docs.pydantic.dev/latest/>
9. **Research Papers:**
 - https://www.researchgate.net/publication/383056881_EXPENSE_TRACKER_MANAGEMENT_SYSTEM_PROJECT_REPORT
 - https://www.researchgate.net/publication/388499664_EXPENSE_TRACKER?enrichId=rgreq-ed4925dde461c3dde3f1aff392fadf7e-XXX&enrichSource=Y292ZXJQYWdlOzM4ODQ5OTY2NDtBUzoxMTQzMTI4MTMwNjI3NjQ4MkAxNzM4MjMjMjk3NjE2&el=1_x_2&esc=publicationCoverPdf
10. **Stack Over Flow:** <https://stackoverflow.com/>