EXPENSE TRACKER

A PROJECT REPORT

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE AWARD OF THE DEGREE OF

B.TECH. (COMPUTER ENGINEERING) TO

RK UNIVERSITY, RAJKOT

SUBMITTED BY

Name of Student	**Enrollment No.**
Radhesh S. Joshi	22SOECE11080
Vaibhav B. Goriya	22SOECE11077

UNDER THE GUIDANCE OF

Internal Guide
Prof. Jay Pithadiya
Assistant Professor,
RK University,
Rajkot

November 2025

SCHOOL OF ENGINEERING, RK UNIVERSITY, RAJKOT

DECLARATION

I/We hereby certify that I/We am/are the sole author(s) of this project work and that neither any part of this project work nor the whole of the project work has been submitted for a degree to any other University or Institution. I/We certify that, to the best of my/our knowledge, my/our project work does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my/our project document, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. I/We declare that this is a true copy of my/our project work, including any final revisions, as approved by my/our project review committee.

Signature of Student (S)

Radhesh S. Joshi (22SOECE11080),
Date: _____
Place: _____

Vaibhav B. Goriya (22SOECE11077)
Date: _____
Place: _____

CERTIFICATE

This is to certify that the work which is being presented in the Project Report entitled **"Expense Tracker",** in partial fulfillment of the requirement for the award of the degree of **B.Tech. (Computer Engineering)** and submitted to the School of Engineering, RK University, is an authentic record of my/our own work carried out during a period from **June 2025 to November 2025.**

The matter presented in this Project Report has not been submitted by me/us for the award of any other degree elsewhere.

Signature of Student (S)

Radhesh S. Joshi (22SOECE11080) Vaibhav B. Goriya (22SOECE11077)

This is to certify that the above statement made by the student(s) is correct to the best of my knowledge.

Internal Guide
Prof. Jay Pithadiya
Assistant Professor,
RK University,
Rajkot.

Head of Department
Dr. Chetan Shingadiya
CE / IT,
School of Engineering,
RK University, Rajkot.

June 2025

SCHOOL OF ENGINEERING, RK UNIVERSITY, RAJKOT

Acknowledgement

I would like to express my heartfelt gratitude to all the individuals and institutions who supported me in successfully completing this project. First and foremost, I am sincerely thankful to my project guide, Jay Pithadiya, for their continuous support, encouragement, and expert guidance throughout the development of this project. Their insights and feedback were invaluable at every stage of this work.

I also wish to thank the faculty members and peers at RK University for their constructive suggestions and motivation, which helped me stay focused and improve the quality of my work.

I would also like to acknowledge the open-source community, documentation, and forums that provided immense help in understanding and implementing technologies like MongoDB, Express.js, React.js, Node.js, and shadcn/ui. Their contributions played a vital role in enhancing my learning and technical skills.

Thank You

Your Sincerely

Radhesh S. Joshi

Vaibhav B. Goriya

Abstract

The primary goal of this project is to develop a Personal Expense Tracker web application using the MERN stack (MongoDB, Express.js, React.js, Node.js) integrated with modern UI components from shadcn/ui. The application enables users to securely log in, manage their income and expenses, and gain insights through visual dashboards.

Users can perform core functionalities such as adding, updating, deleting, and viewing financial records categorized by date, source, or type. Interactive charts and analytics provide a clear view of monthly savings, expense categories, and overall financial health.

This project demonstrates the use of full-stack JavaScript technologies to build a responsive, scalable, and user-centric application. It also emphasizes practical implementation of JWT-based authentication, RESTful APIs, and modern React design principles. The system is designed to be easily extendable with features like budget limits, PDF exports, and recurring transactions in the future.

Notations, Naming Convention, and Abbreviations

- **MERN** – MongoDB, Express.js, React.js, Node.js
- **JWT** – JSON Web Token (used for authentication)
- **API** – Application Programming Interface
- **DB** – Database (where income & expense data is stored)
- **UI** – User Interface
- **UX** – User Experience
- **CRUD** – Create, Read, Update, Delete (main operations in the app)
- **Chart.js** – JavaScript library for data visualization
- **Modal** – Popup window for adding/editing entries
- **Schema** – Database table/collection structure definition
- **Transaction** – A single record of income or expense
- **Category** – Label to group transactions (e.g., Food, Salary)
- **Balance** – Total funds available (Income – Expense)
- **API Endpoint** – Specific URL path for handling backend requests
- **Middleware** – Code that runs before/after route handlers (e.g., auth check)
- **Component** – Reusable UI building block in React
- **State** – React data that controls rendering
- **Hook** – Special React function for managing state/effects (e.g., useState, useEffect)

List of Figures

Index (Table of Contents)

CHAPTER 1: INTRODUCTION

1.1 Project Summary

The Expense Tracker is a full-stack web application designed to help users efficiently manage their personal finances. Built using the MERN stack (MongoDB, Express.js, React.js, Node.js), the system enables users to track incomes and expenses, visualize data through interactive charts, and analyse financial trends. The frontend interface uses modern UI components powered by shadcn/ui, offering a clean, intuitive, and responsive user experience.

1.1.1 Purpose: Goals & Objectives

Goal: To develop an Expense Tracker so Expanse can manage Easily with detail Report.

Objectives:

- To create a user-friendly platform to **record income and expense entries**
- To **visualize spending patterns** through bar and pie charts
- To provide a **secure login/signup system** using JWT authentication
- To enable **add, update, delete** financial records (CRUD)
- To maintain **category-wise financial statistics**
- To allow filtering, searching, and potentially exporting data

1.2 Scope

This project is intended for individual users seeking an organized approach to financial tracking. The system allows:

- Adding income/expense with details like source, amount, date, category (with emoji picker)

- View monthly statistics through dynamic charts
- User-specific data management with secured access

Can be extended for multi-user support, budgeting features, and reminders

• 1.3 Technology and Literature Review of Past Work

Technology	Description
MongoDB	NoSQL database used for storing user data and financial records
Express.js	Backend framework for building REST APIs
React.js	Frontend library for building dynamic UI
Node.js	JavaScript runtime for executing backend code
shadcn/ui	Component library for modern and accessible UI in React
JWT	JSON Web Tokens used for authentication
Chart.js / Recharts	Charting library used for income/expense visualizations

CHAPTER 2: PROJECT MANAGEMENT

2.1 Project Planning and Scheduling

Effective planning and scheduling are essential for the successful completion of any software project. The development of the Expense Tracker application was approached with clear phases including planning, design, development, testing, and deployment.

2.1.1 Development Approach

The **Agile Development Model** was used for this project. Agile allowed continuous feedback, short development cycles (sprints), and adaptability to changes. Each module (like Income, Expense, Dashboard, and Auth) was developed incrementally with regular testing and integration.

Key Agile Practices Used:

- Daily task lists and to-do boards (e.g., Trello/Notion)
- Weekly milestones
- User stories like:
  - *"As a user, I want to log in securely to access my financial records."*
  - *"As a user, I want to visualize my expenses monthly."*

2.1.2 Project Plan Including Milestones, Deliverables, Roles & Responsibilities

The project was divided into five main phases:

Phase	Duration	Key Activities
Planning	1 week	Define objectives, scope, technology stack
Design	1 week	Wireframing, ER diagrams, Use Case diagrams
Development	2–3 weeks	Backend APIs, frontend UI, Redux integration
Testing	1 week	Unit testing, integration testing
Finalization	1 week	Bug fixing, documentation, deployment

- **2.1.3 Schedule Representation**

  Here's the timeline represented using a **Gantt-like format**:

Week	Task
1	Requirement analysis, tech stack finalization, basic UI design
2	Backend setup (Node.js + Express + MongoDB), Auth APIs with JWT
3	Frontend setup (React with shadcn/ui), Income/Expense components
4	Charts (Recharts), Redux state management, dashboard setup
5	Testing, UI polish, report documentation, deployment

- ## 2.2 Risk Management

Details the identified risks, their potential impact on the project, and mitigation strategies.

Risk Identification: Identifying potential risks early helps avoid delays and maintain project quality.

- ### 2.2.1 Risk Identification

Risk ID	Description
R1	Integration issues between frontend and backend
R2	Incomplete data validation may cause errors
R3	React component state mismanagement
R4	Unexpected library compatibility issues
R5	Internet/API latency affecting deployment

- ### 2.2.2 Risk Analysis

Risk ID	Probability	Impact	Priority
R1	High	High	Critical
R2	Medium	Medium	Moderate
R3	Medium	Low	Low
R4	Low	Medium	Low
R5	Medium	Medium	Moderate

- **2.2.3 Risk Planning**

Risk ID	Mitigation Strategy
R1	Use Postman & testing tools to verify each API connection
R2	Add server-side + client-side validations
R3	Use Redux properly to manage global state
R4	Use stable and actively maintained packages
R5	Optimize backend performance and test on live servers

CHAPTER 3: SYSTEM REQUIREMENTS STUDY

3.1 User Characteristics

The system is designed for a wide range of users, primarily individuals who wish to manage their personal finances. These users may or may not have technical expertise, so the system prioritizes simplicity, usability, and clean UI/UX.

Key User Characteristics:

- Comfortable using web-based applications.
- Basic understanding of financial terms like income, expense, balance.
- Expect secure login and personalized data access.
- Prefer clean, responsive, and intuitive design (achieved using shadcn/ui).

3.2 Hardware and Software Requirements

Category	Requirements
Frontend	React.js, shadcn/ui, HTML5, CSS3, JavaScript
Backend	Node.js, Express.js
Database	MongoDB (cloud via MongoDB Atlas or local)
Authentication	JSON Web Tokens (JWT)
Visualization	Chart.js or Recharts for data graphs
Operating System	Windows/Linux/macOS
IDE/Text Editor	VS Code or any JavaScript-compatible IDE
Browser	Chrome, Firefox, or any modern browser
RAM	Minimum 4 GB
Disk Space	Minimum 500 MB for development files

- ## 3.3 Constraints

Technical Constraints:

- The app must support modern browsers and be responsive (mobile & desktop).
- Must maintain secure communication (JWT & HTTPS recommended in production).
- Data must be persistent and secured in the database.
- The application should be deployable on platforms like Vercel/Netlify (frontend) and Render/Heroku (backend).

Functional Constraints:

- Only authenticated users can access income/expense features.
- No data sharing or multi-user dashboard in this version.
- Internet connection required to access cloud database and hosted APIs.

CHAPTER 4: SYSTEM ANALYSIS

- ## 4.1 Study of Current System

  Currently, most individuals and small businesses track their income and expenses manually or using basic spreadsheet tools. These methods are prone to errors, lack automation, and do not provide visual insights into financial data. They also do not support instant access from multiple devices or offer advanced features like category-wise expense tracking and reports.

- ## 4.2 Problems and Weaknesses

  The current manual or basic tracking methods suffer from the following issues:

  - **Data Inaccuracy** – Manual entry often leads to errors and inconsistencies.
  - **Lack of Visual Reports** – Users cannot easily visualize spending patterns.
  - **No Categorization** – Transactions are not properly grouped into categories.
  - **Limited Accessibility** – Data is stored locally and cannot be accessed remotely.
  - **No Security** – Sensitive financial data is not protected with authentication.

- ## 4.3 Requirements of the New System

  - A web-based platform accessible from any device.
  - The new system aims to address the above problems with:
  - Secure user authentication with JWT.
  - CRUD operations for income and expense transactions.
  - Category management for better organization.
  - Real-time balance calculation.
  - Data visualization using charts and graphs.
  - Responsive UI for both desktop and mobile.

4.4 Feasibility Study

A web-based platform accessible from any device.

- **Technical Feasibility** – Built using the MERN stack (MongoDB, Express.js, React.js, Node.js) which supports scalability and modern UI/UX features.

- **Operational Feasibility** – User-friendly interface ensures ease of use for non-technical users.

- **Economic Feasibility** – Open-source technologies minimize development costs.

4.5 Requirements Validation

The requirements were validated by ensuring:

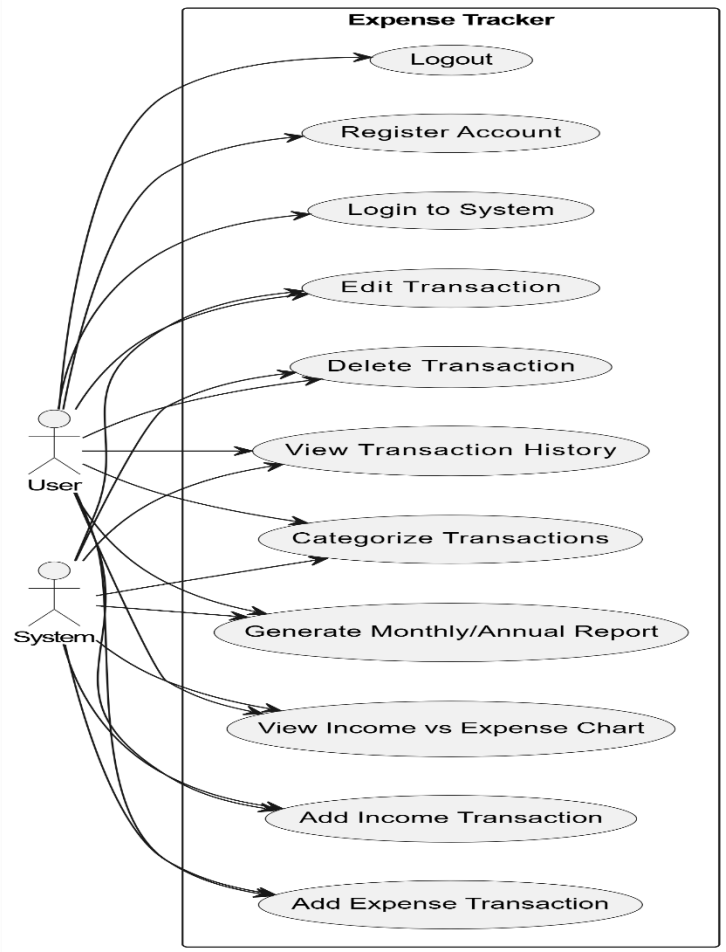
- Each feature aligns with solving current system problems.
- Data security is maintained using authentication and authorization.
- Reports and analytics match user expectations.

4.6 Functions of the System

- **User Management** – Registration, login, and authentication.
- **Transaction Management** – Add, edit, delete, and view transactions.
- **Category Management** – Create and manage transaction categories.
- **Report Generation** – Display monthly/annual summaries.
- **Data Visualization** – Graphs for income vs expenses.

- **4.6.1 Use Case Diagram**

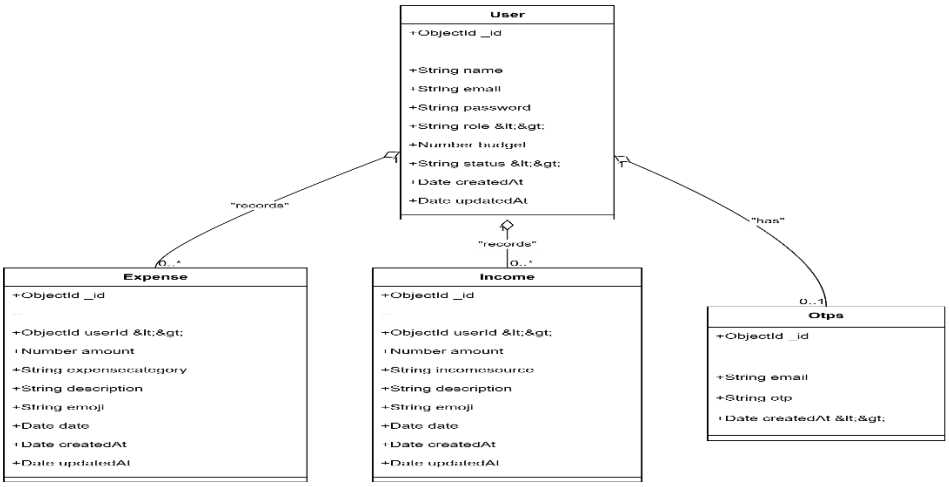
    - Shows how different users (e.g., registered users) interact with the system's functionalities.



- # 4.7 Data Modelling

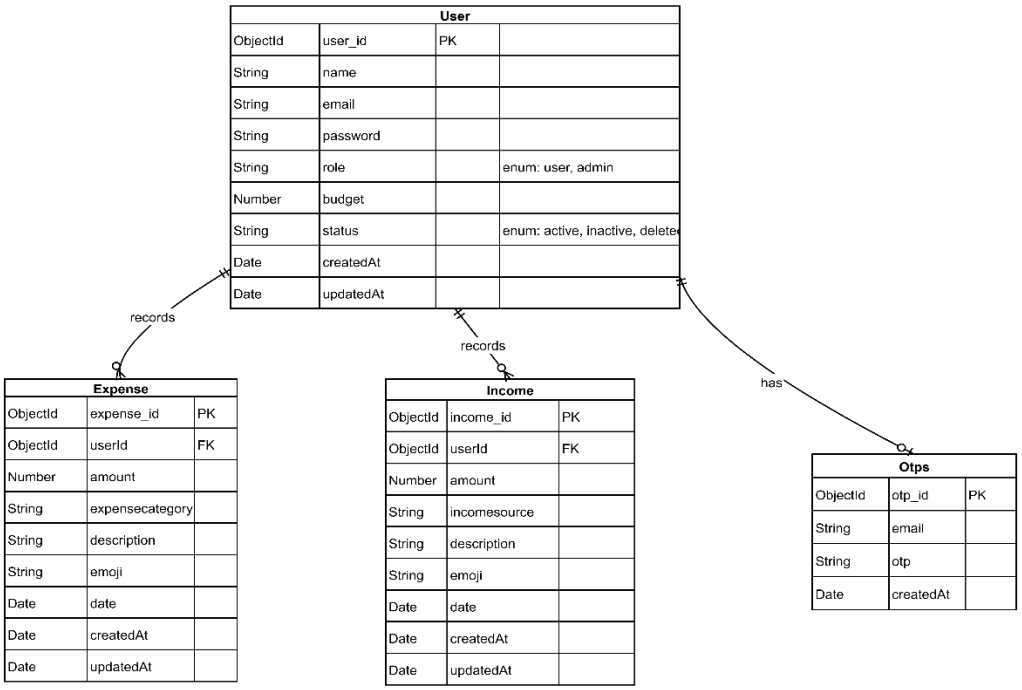
4.7.1 Data Dictionary

- Contains definitions of important terms like **Transaction**, **Category**, **Balance**, and **User**.



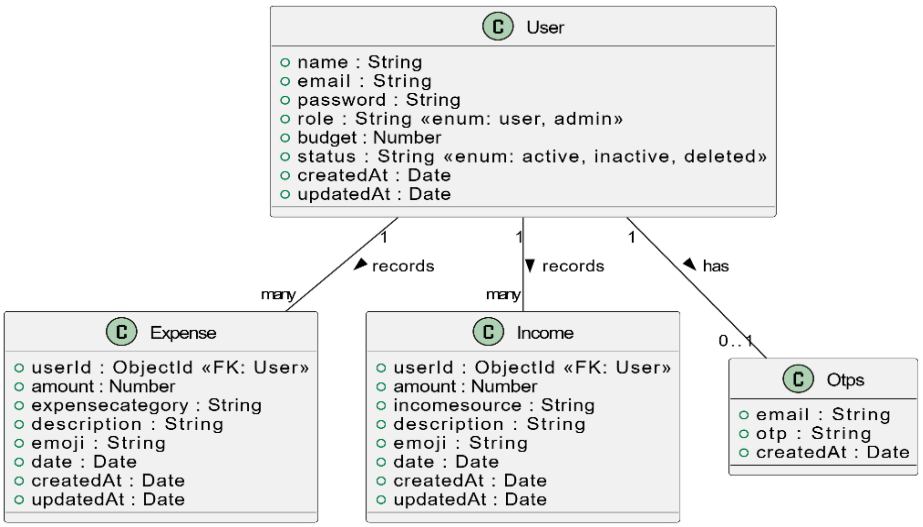
4.7.2 ER Diagram

- Represents relationships between entities such as **User**, **Transaction**, and **Category**.

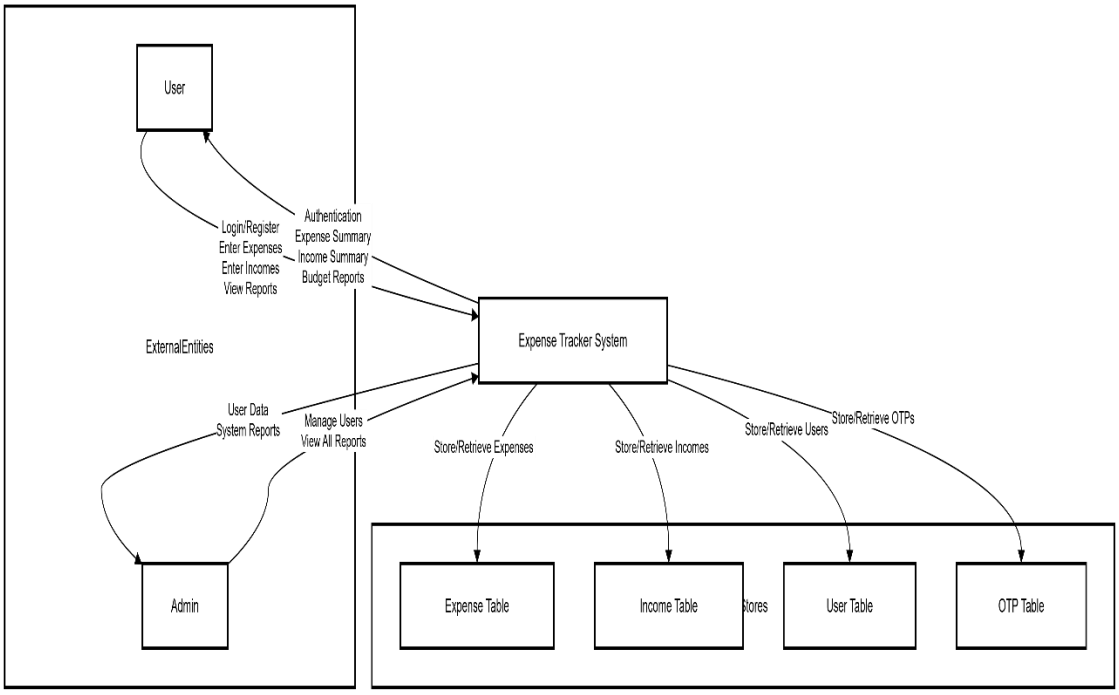


4.7.3 Class Diagram

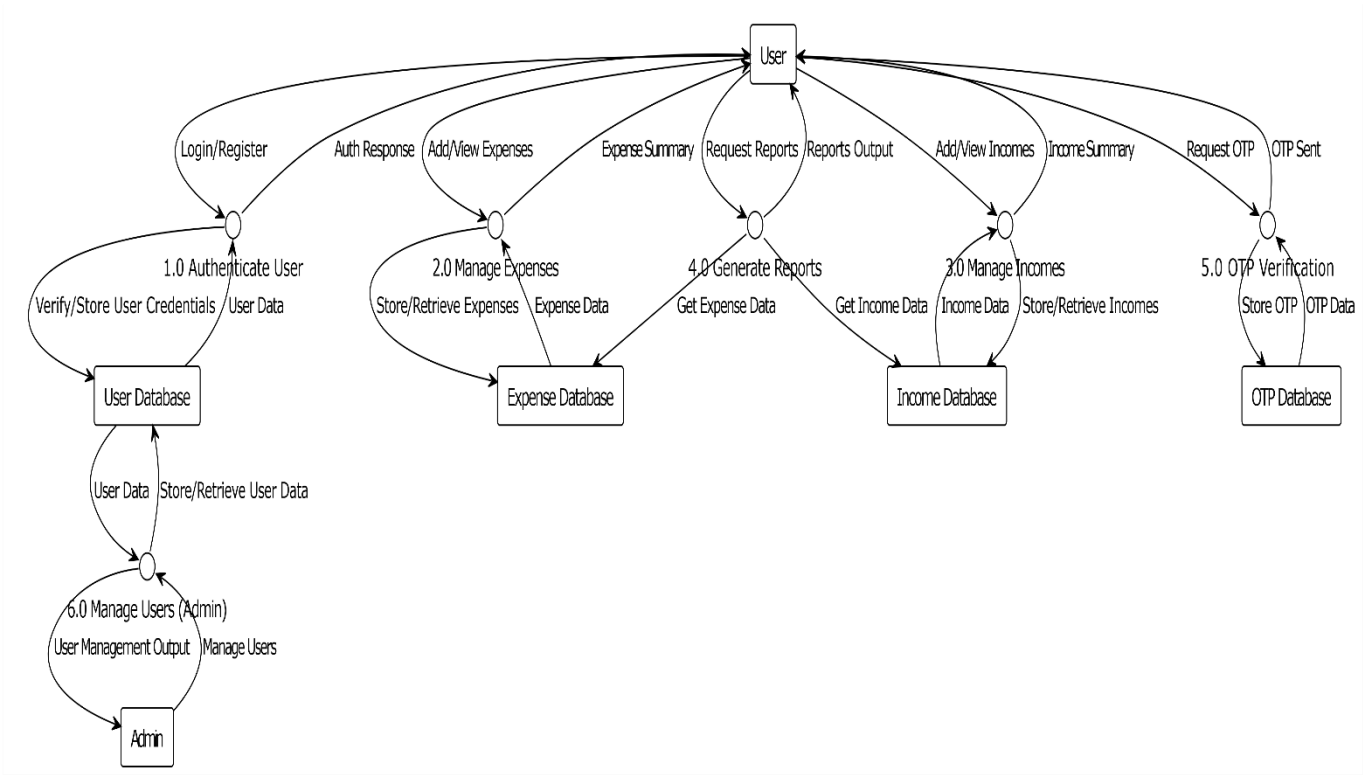
- Shows the structure of the project in terms of classes/components like **User**, **Transaction**, **Category**, and **Auth Controller**.



- ## 4.8 Functional and Behavioral Modeling
- ### 4.8.1 Data Flow Diagram (0 and 1 Level)
  - **Level 0 DFD** – Shows overall system as a single process.



- **Level 1 DFD** – Breaks the system into processes like **Login**, **Manage Transactions**, and **Generate Reports**.



- ## 4.9 Main Modules of New System
    - **Authentication Module** – Handles login, registration, and security.
    - **Income Module** – Manages income entries.
    - **Expense Module** – Manages expense entries.
    - **Category Module** – Allows categorization of transactions.
    - **Reports Module** – Generates charts and summaries.

CHAPTER 5: SYSTEM DESIGN

- ## 5.1 Database Design

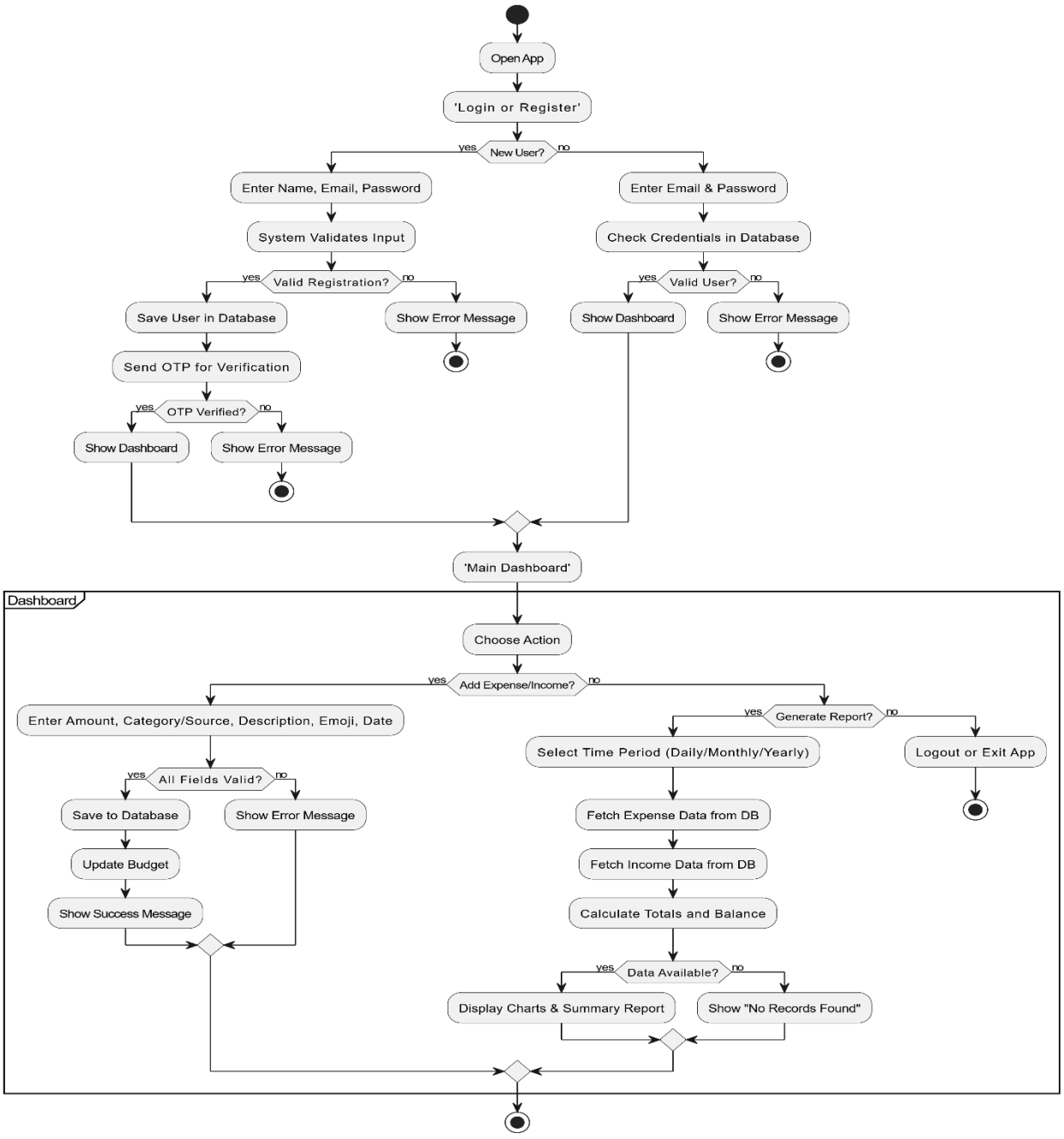
  The Expense Tracker application uses MongoDB as its primary database. Mongoose schemas define collections, their fields, and relationships.

  - ### 5.1.1 Tables and Relationships
    - **User Table**
      - **Fields:** name, email, password, role, budget, status
      - **Relationships:** A user can have multiple expenses and incomes.
    - **Expense Table**
      - **Fields:** userId (FK), amount, expenseCategory, description, emoji, date
      - **Relationships:** Linked to a user.
    - **Income Table**
      - **Fields:** userId (FK), amount, incomeSource, description, emoji, date
      - **Relationships:** Linked to a user.
    - **OTP Table**
      - **Fields:** email, otp, createdAt (expires after 10 minutes)
      - **Relationships:** Used for user authentication and verification.

- ## 5.2 System Procedural Design

  - ### 5.2.1 Flowchart or Activity Diagram

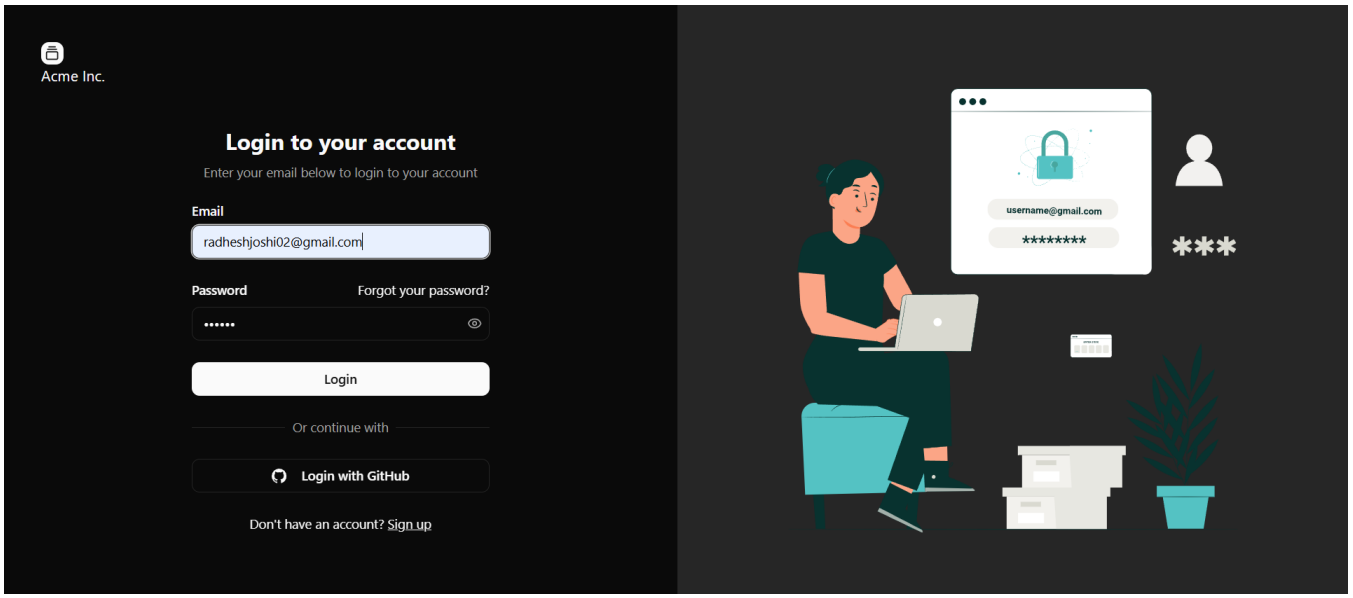
    - Login/Register Process
    - Add Expense/Income Process
    - Generate Report Process

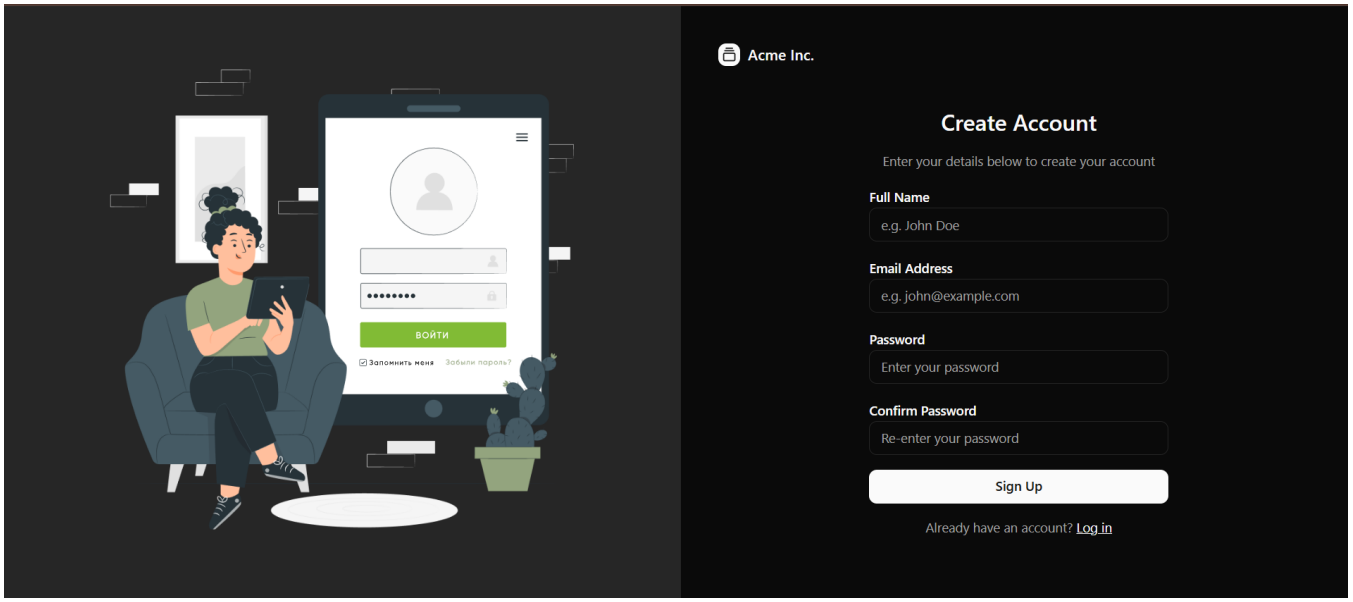


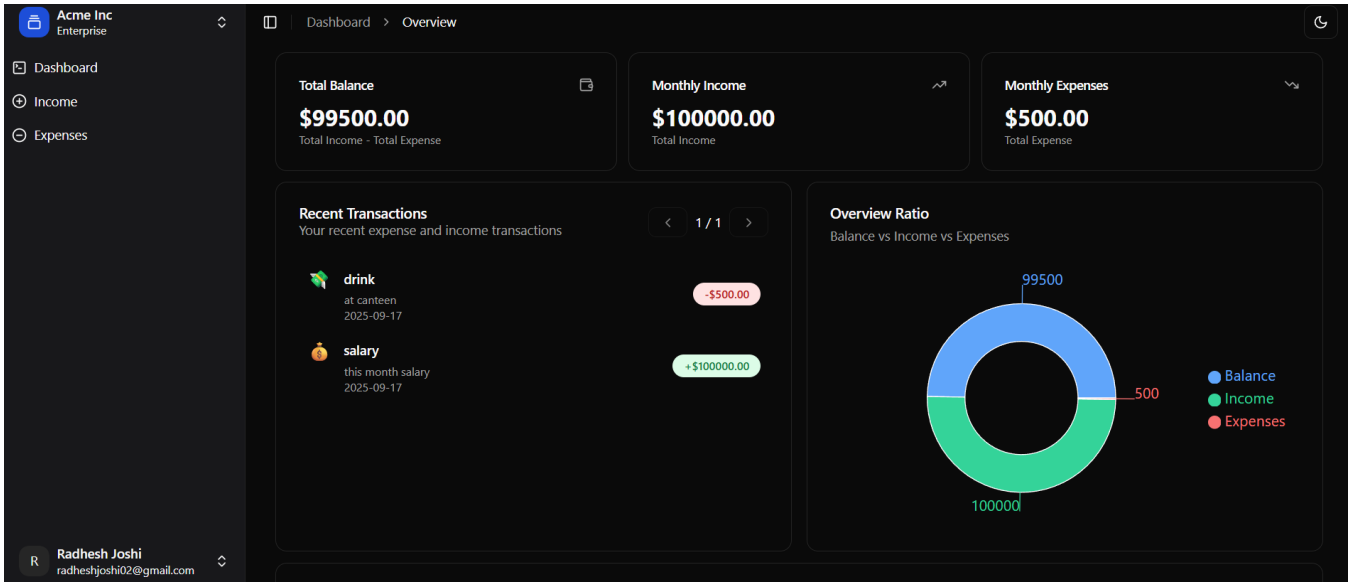
- ## 5.3 Input/Output and Interface Design

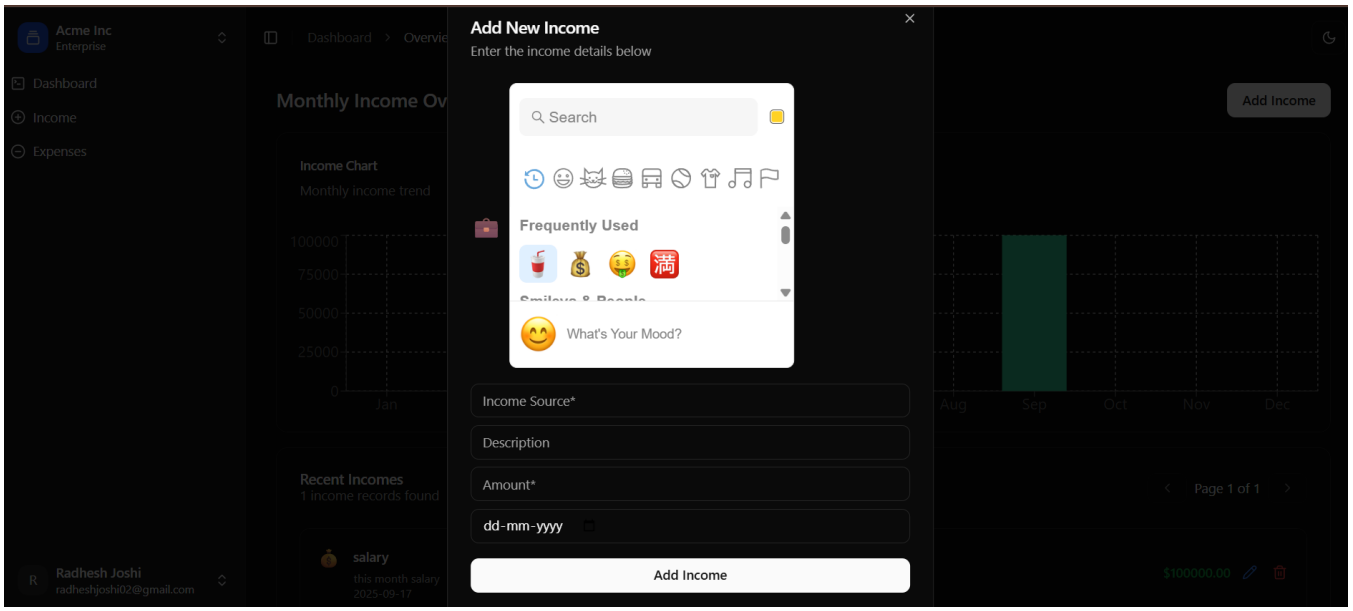
  - **Input**: User details (signup/login), expense amount, category, income source, description.

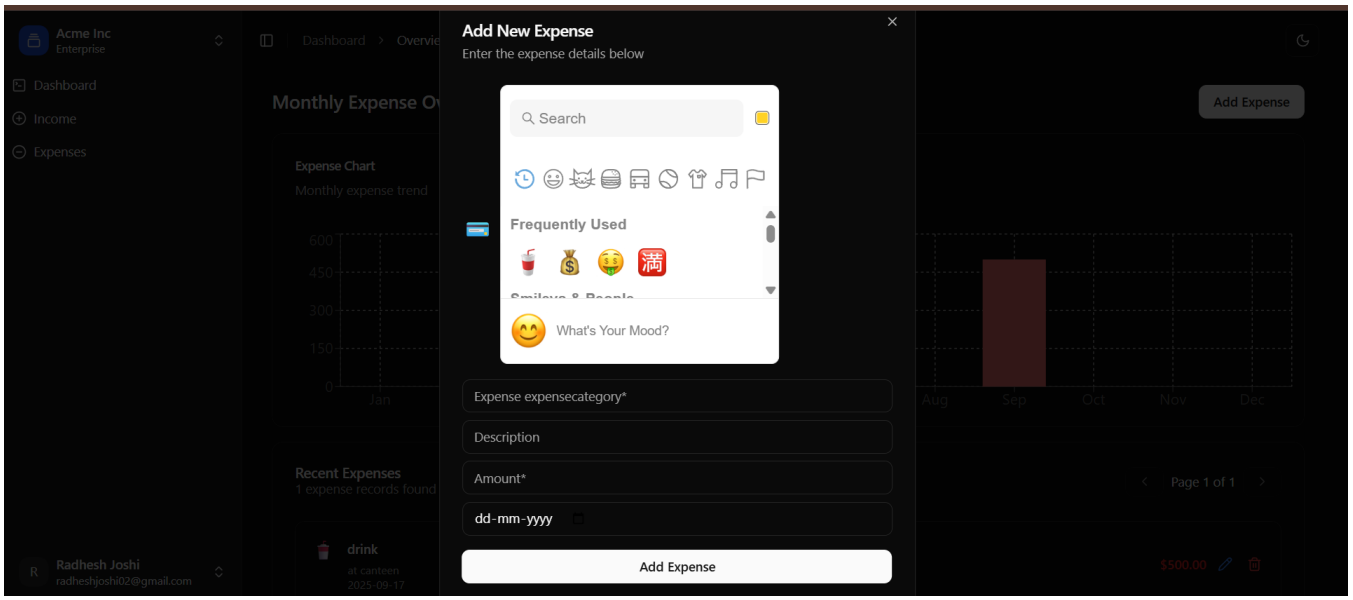
- **Output**: Reports, dashboards, expense lists, income lists, charts.
- **Interface**:
  - Login/Signup Screen
  - Dashboard (Summary of income vs expenses)
  - Expense & Income Entry Forms

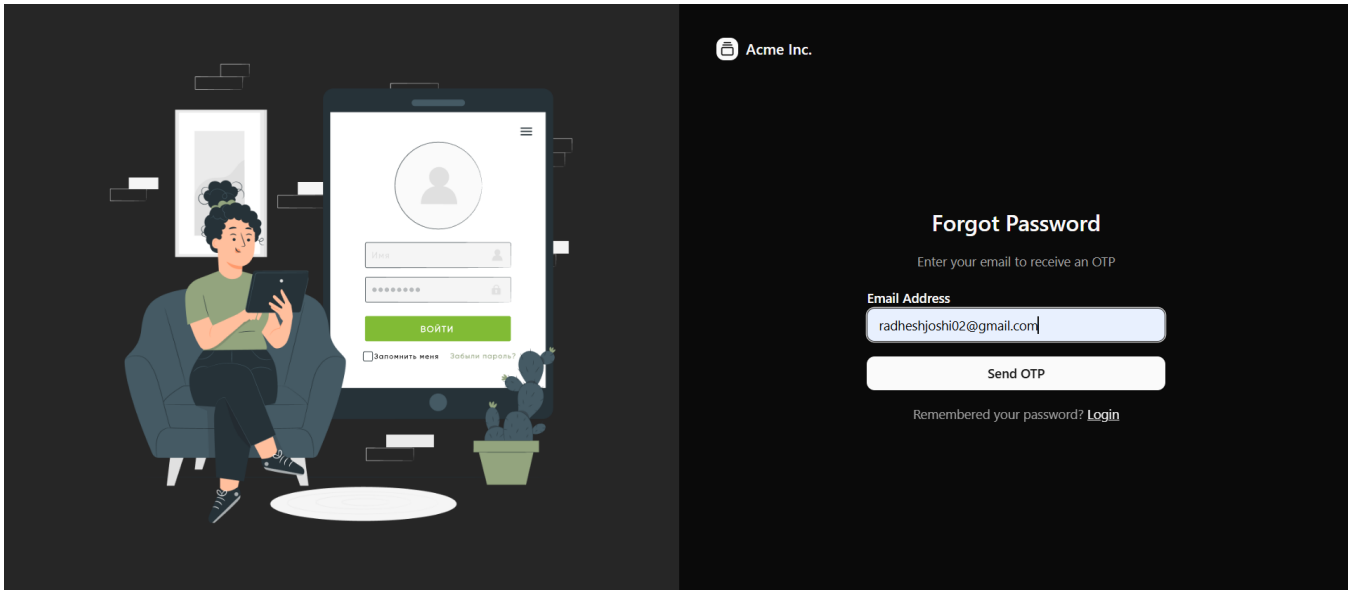


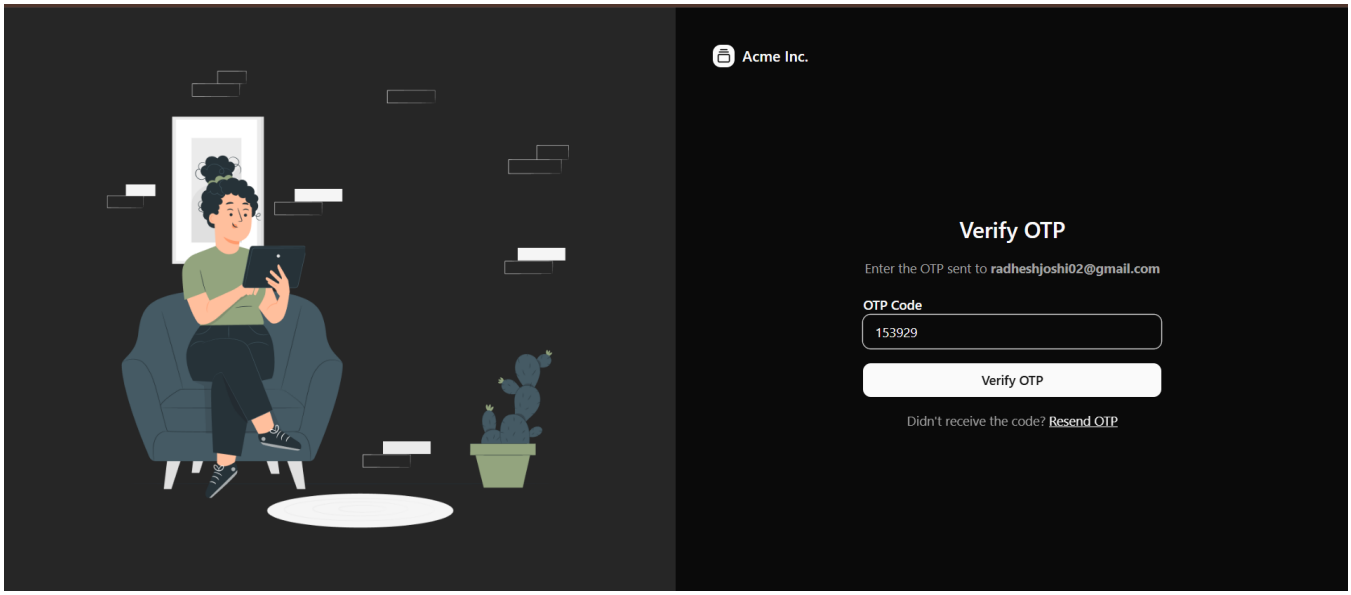


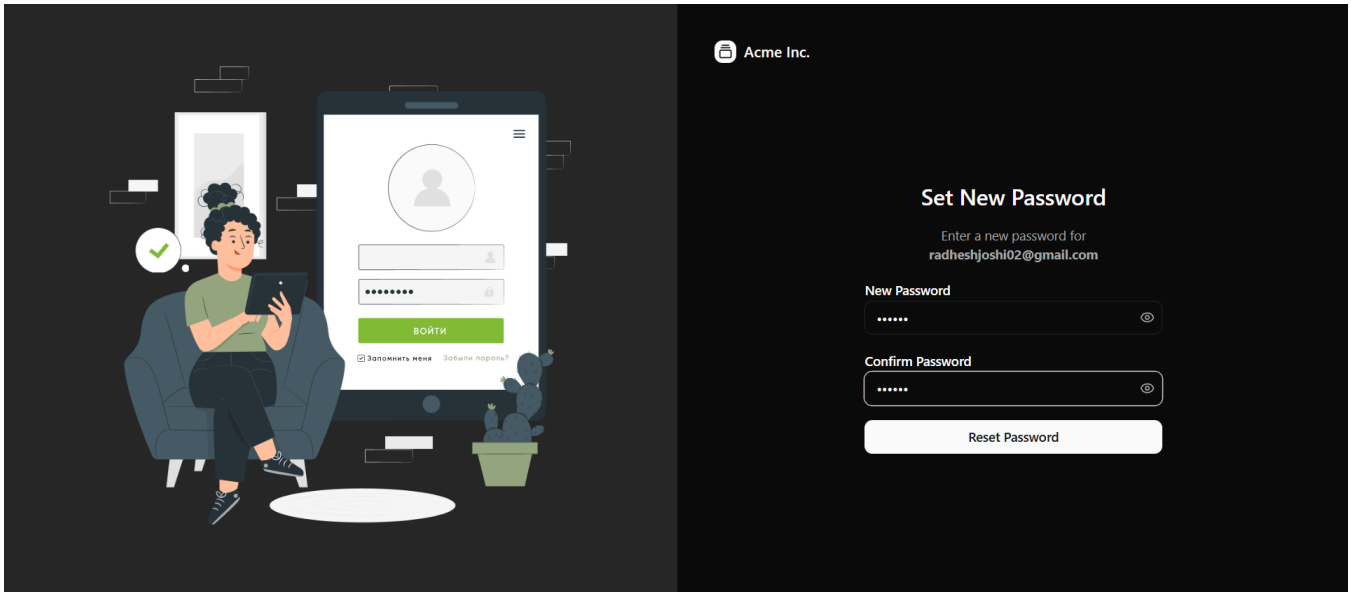

Acme Inc
Enterprise
Dashboard
Income
Expenses
Dashboard > Overview
Total Balance
$99500.00
Total Income - Total Expense
Monthly Income
$100000.00
Total Income
Monthly Expenses
$500.00
Total Expense
Recent Transactions
Your recent expense and income transactions
1 / 1
drink
at canteen
2025-09-17
-$500.00
salary
this month salary
2025-09-17
+$100000.00
Overview Ratio
Balance vs Income vs Expenses
99500
500
100000
Balance
Income
Expenses
Radhesh Joshi
radheshjoshi02@gmail.com


Add New Income
Enter the income details below
Search
Frequently Used
What's Your Mood?
Income Source*
Description
Amount*
dd-mm-yyyy
Add Income
Add Income
Page 1 of 1


Add New Expense
Enter the expense details below
Search
Frequently Used
What's Your Mood?
Expense expensecategory*
Description
Amount*
dd-mm-yyyy
Add Expense
Add Expense
Page 1 of 1


Acme Inc.
Forgot Password
Enter your email to receive an OTP
Email Address
radheshjoshi02@gmail.com
Send OTP
Remembered your password? Login


Acme Inc.
Verify OTP
Enter the OTP sent to radheshjoshi02@gmail.com
OTP Code
153929
Verify OTP
Didn't receive the code? Resend OTP


Acme Inc.
Set New Password
Enter a new password for
radheshjoshi02@gmail.com
New Password
Confirm Password
Reset Password

CHAPTER 6: IMPLEMENTATION

- ## 6.1 Implementation Environment
    - **Frontend:** React.js, Tailwind CSS
    - **Backend:** Node.js, Express.js
    - **Database:** MongoDB with Mongoose
    - **Authentication:** JWT & OTP verification
    - **Hosting:** Can be deployed on Vercel (frontend) and Render/Heroku (backend).

6.2 Program/Module Specification

- **User Module**: Handles authentication, roles, budget.
- **Expense Module**: Add, update, delete, view expenses.
- **Income Module**: Add, update, delete, view incomes.
- **OTP Module**: Email-based OTP verification.

6.3 Sample Coding

- **User Model**

```
const mongoose = require("mongoose");
const bcrypt = require("bcryptjs");

const userSchema = new mongoose.Schema(
  {
    name: {
      type: String,
      required: true,
      trim: true,
    },
    email: {
      type: String,
      required: true,
      unique: true,
      lowercase: true,
    },
    password: {
      type: String,
      required: true,
      select: false,
    },
    role: {
      type: String,
      enum: ["user", "admin"],
      default: "user",
    },
    budget: {
      type: Number,
      min: 0,
    },
    status: {
      type: String,
      enum: ["active", "inactive", "deleted"],
      default: "active",
```

```
    },
  },
  { timestamps: true }
);

// Hash password before saving
userSchema.pre("save", async function (next) {
  if (!this.isModified("password")) return next();
  this.password = await bcrypt.hash(this.password, 12);
  next();
});

// Compare passwords (for login)
userSchema.methods.comparePassword = async function
(candidatePassword) {
  // Add validation
  if (!candidatePassword) {
    throw new Error("Password is required");
  }
  if (!this.password) {
    throw new Error("User password not found");
  }
  return await bcrypt.compare(candidatePassword, this.password);
};

module.exports = mongoose.model("User", userSchema);
```

- **Expense Model**

```
const mongoose = require("mongoose");

const expenseSchema = new mongoose.Schema(
  {
    userId: {
      type: mongoose.Schema.Types.ObjectId,
      ref: "User",
      required: true,
    },
    amount: {
      type: Number,
      required: true,
      min: 0,
    },
    expensecategory: {
      type: String,
      required: true,
```

```
        trim: true,
      },
      description: {
        type: String,
        trim: true,
      },
      emoji: {
        type: String,
        default: "💰",
      },
      date: {
        type: Date,
        default: Date.now,
      },
    },
    { timestamps: true }
  );

  module.exports = mongoose.model("Expense", expenseSchema);

```

- **Income Model**

```
const mongoose = require("mongoose");

const incomeSchema = new mongoose.Schema(
  {
    userId: {
      type: mongoose.Schema.Types.ObjectId,
      ref: "User",
      required: true,
    },
    amount: {
      type: Number,
      required: true,
      min: 0,
    },
    incomesource: {
      type: String,
      required: true,
      trim: true,
    },
    description: {
      type: String,
      trim: true,
    },
    emoji: {
```

```
        type: String,
        default: "₹",
      },
      date: {
        type: Date,
        default: Date.now,
      },
    },
    { timestamps: true }
);

module.exports = mongoose.model("Income", incomeSchema);

```

- **Otp Model**

```
// models/Otp.js
const mongoose = require("mongoose");
const { Schema } = mongoose;

const otpSchema = new Schema({
  email: {
    type: String,
    required: true,
    unique: true,
  },
  otp: {
    type: String,
    required: true,
  },
  createdAt: {
    type: Date,
    default: Date.now,
    expires: 600, // Document expires after 10 minutes (600 seconds)
  },
});

module.exports = mongoose.model("Otps", otpSchema);

```

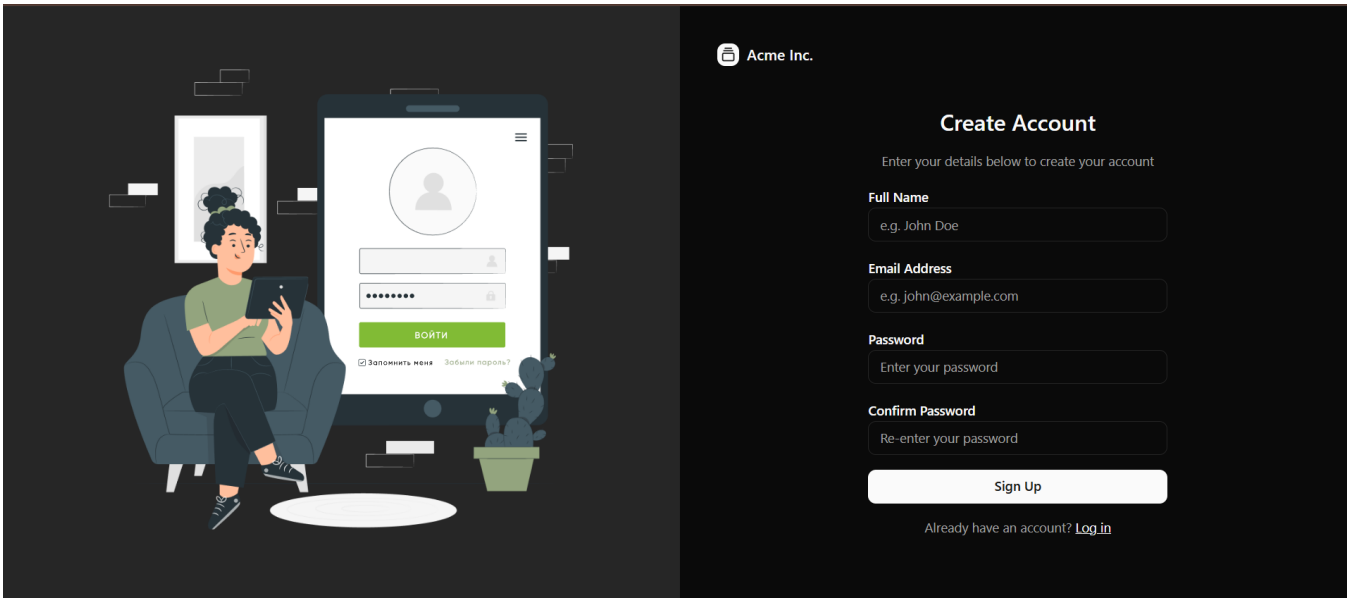
CHAPTER 7: TESTING

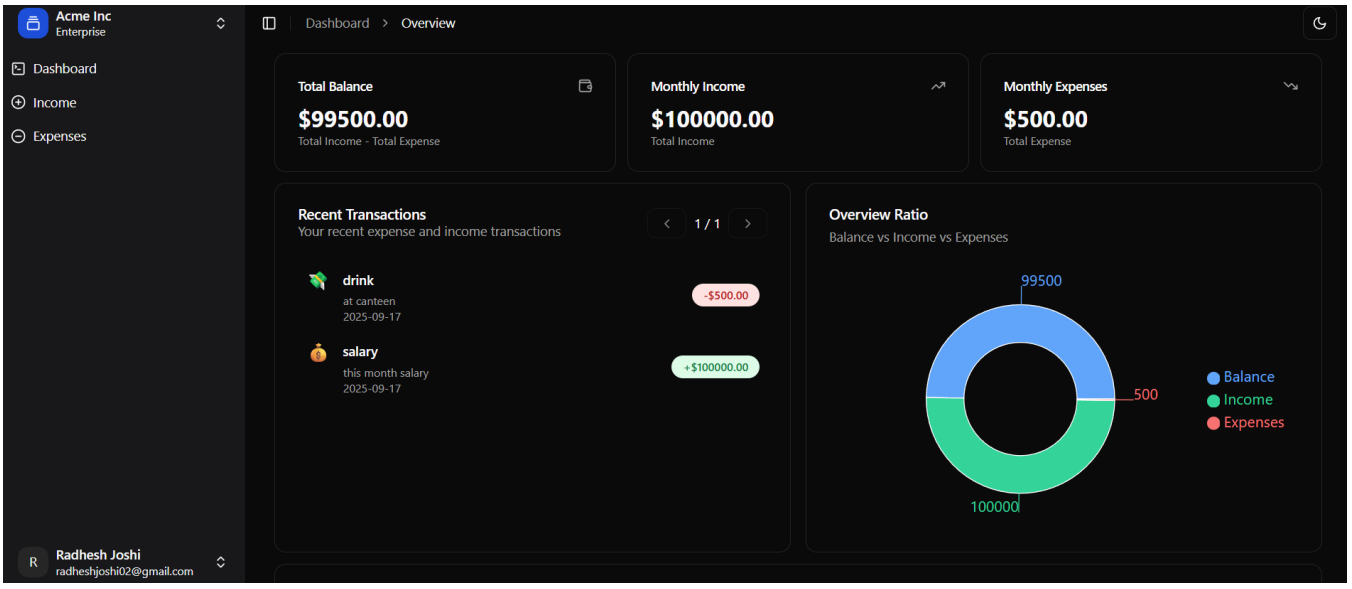
- ## 7.1 Testing Plan
    - Unit Testing (Jest, Mocha for backend functions).
    - Integration Testing (API endpoints tested with Postman).
    - User Acceptance Testing (ensures UI works as expected).
- ## 7.2 Testing Strategy
    - **Black Box Testing:** Test user input without internal code knowledge.
    - **White Box Testing:** Validate API logic.
    - **Regression Testing:** After bug fixes.
- ## 7.3 Test Cases

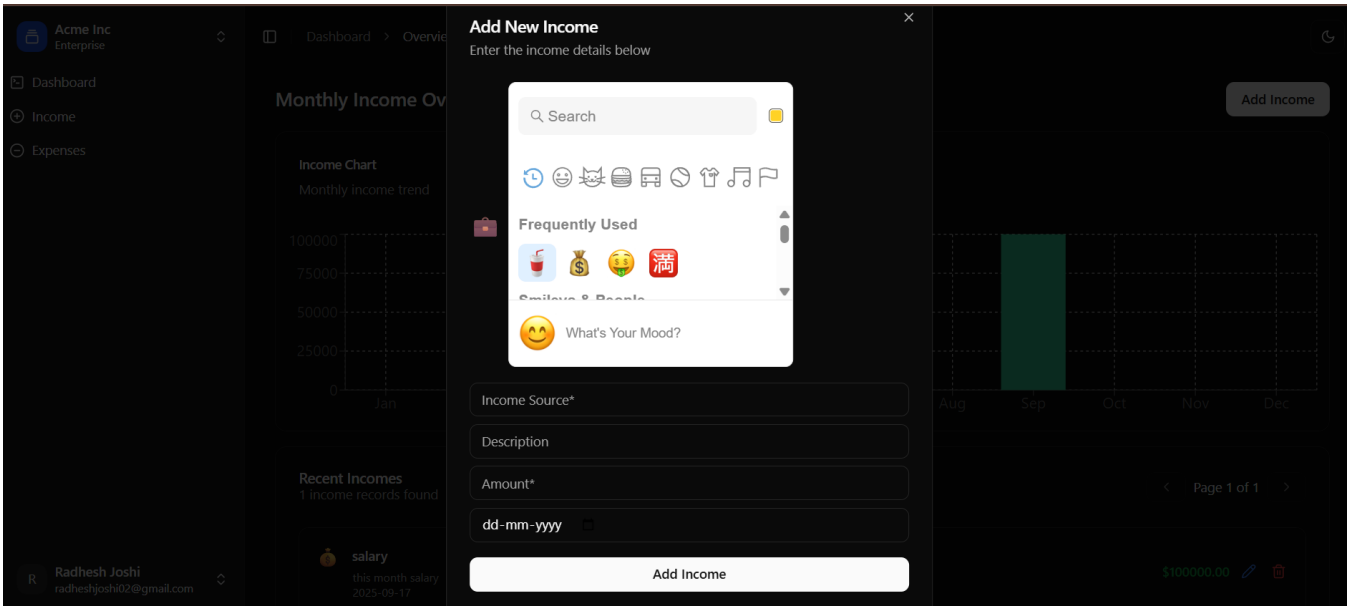
Test Case ID	Description	Input	Expected Output	Result
TC01	User Login	Email + Password	Dashboard	Pass
TC02	Add Expense	Amount + Category	Expense stored	Pass
TC03	Generate Report	User request	Report summary	Pass

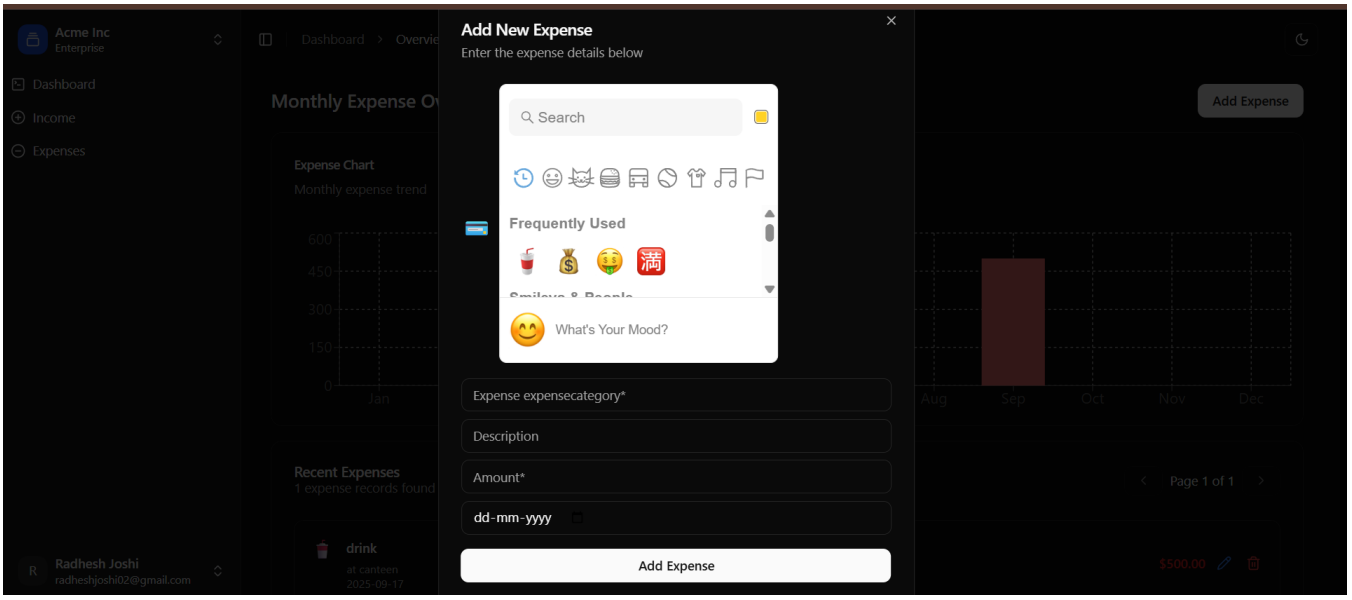
CHAPTER 8: SCREENSHOTS AND USER MANUAL

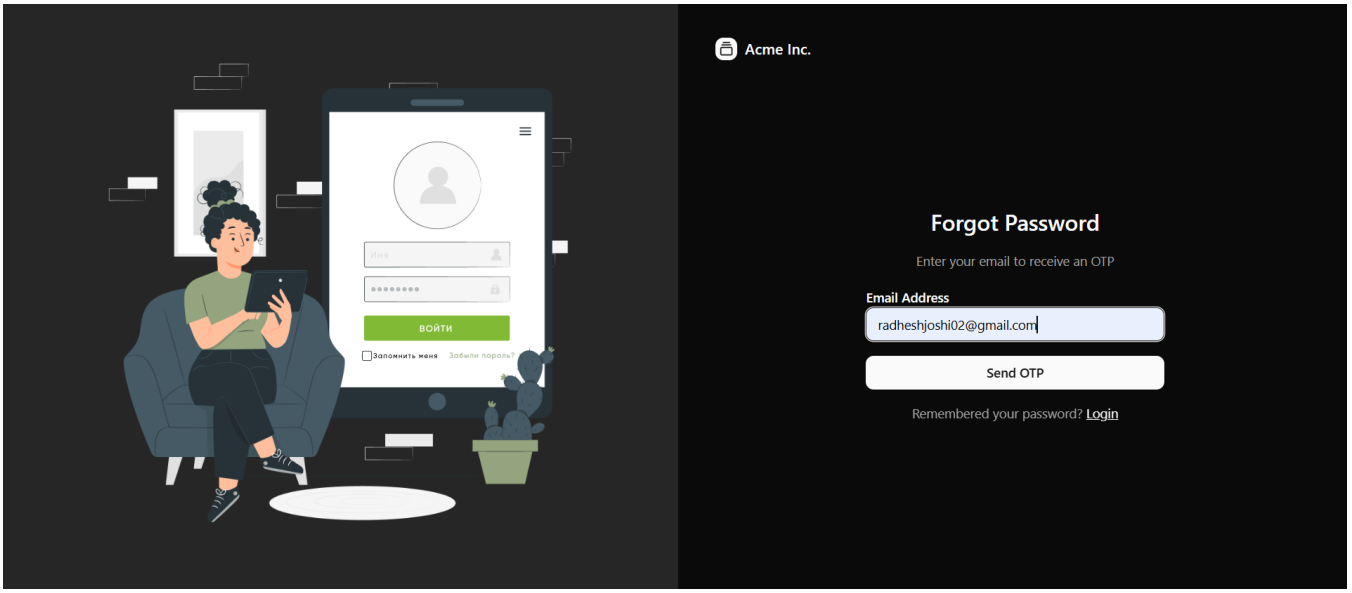
- **Screenshots of Login, Dashboard, Expense Form, Income Form,** Reports.

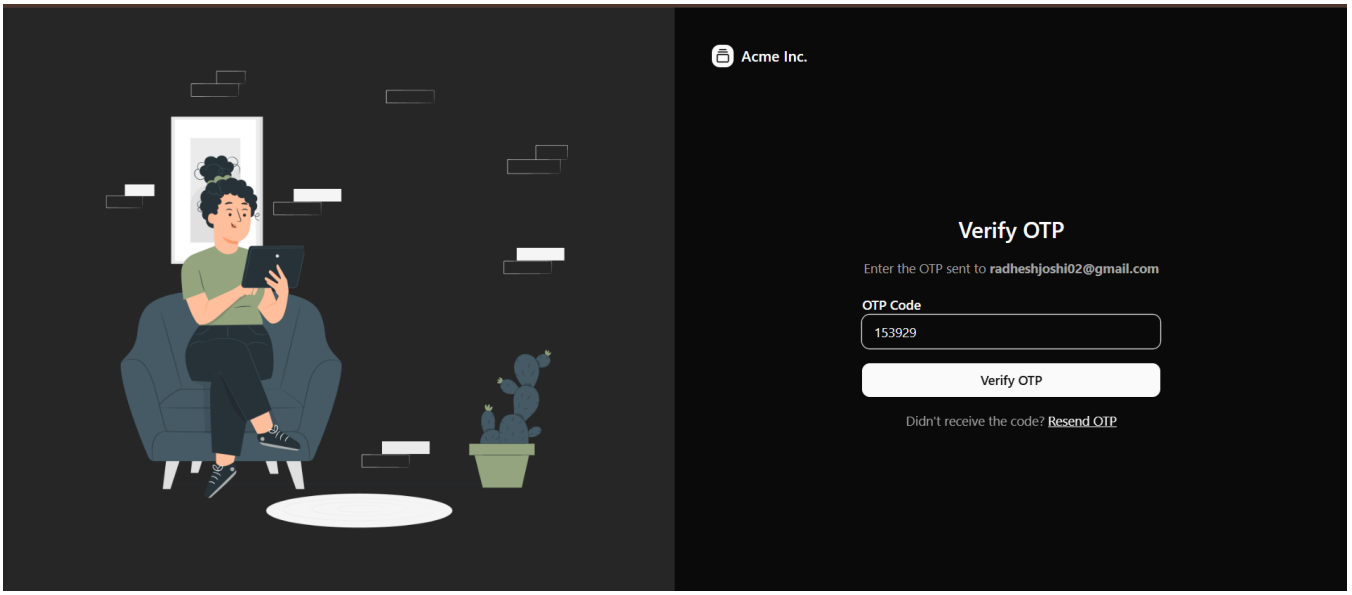





Acme Inc
Enterprise
Dashboard
Income
Expenses
Add New Income
Enter the income details below
Search
Frequently Used
What's Your Mood?
Income Source*
Description
Amount*
dd-mm-yyyy
Add Income
Monthly Income Ov
Income Chart
Monthly income trend
Recent Incomes
1 income records found
salary
this month salary
2025-09-17
Radhesh Joshi
radheshjoshi02@gmail.com
Add Income
Page 1 of 1
$100000.00


Acme Inc
Enterprise
Dashboard
Income
Expenses
Add New Expense
Enter the expense details below
Search
Frequently Used
What's Your Mood?
Expense expensecategory*
Description
Amount*
dd-mm-yyyy
Add Expense
Monthly Expense Ov
Expense Chart
Monthly expense trend
Recent Expenses
1 expense records found
drink
at canteen
2025-09-17
Radhesh Joshi
radheshjoshi02@gmail.com
Add Expense
Page 1 of 1
$500.00


Acme Inc.
Forgot Password
Enter your email to receive an OTP
Email Address
radheshjoshi02@gmail.com
Send OTP
Remembered your password? Login



- **User Manual: Step-by-Step Instructions**

1. User Registration

1. Open the Expense Tracker application.
2. Click on Sign Up.
3. Enter your details:
    - Name
    - Email
    - Password
    - Confirm Password
4. Verify your email using the OTP sent to your registered email ID.
5. Once verified, your account will be created successfully.

---

2. User Login

1. Open the application and click on Login.
2. Enter your registered email and password.
3. If OTP verification is enabled, enter the OTP received in your email.
4. Click Login to access the dashboard.

---

3. Dashboard Overview

1. After login, the dashboard will display:
    - Total Income
    - Total Expenses

- Remaining Budget
- Graphs/Charts summarizing financial data.

2. Use the navigation menu to move between Income, Expenses, and Reports.

---

4. Adding an Expense

1. Go to the Expenses section.
2. Click on Add Expense.
3. Fill in the details:
   - Amount
   - Expense Category (e.g., Food, Travel, Shopping)
   - Description (optional)
   - Emoji (optional, default 💰)
   - Date (defaults to today)
4. Click Save Expense.
5. The expense will be stored in the database and shown in the list below.

---

5. Viewing & Managing Expenses

1. Navigate to the Expenses section.
2. View a list of all added expenses with details.
3. Use filters (by category, date) to refine results.
4. To Edit an expense:
   - Click the Edit icon beside the expense entry.
   - Update the details and click Save Changes.
5. To Delete an expense:
   - Click the Delete icon beside the expense entry.
   - Confirm deletion.

---

6. Adding an Income

1. Go to the Income section.
2. Click on Add Income.
3. Fill in the details:
   - Amount
   - Income Source (e.g., Salary, Freelance, Business)

- Description (optional)
- Emoji (optional, default 💵)
- Date (defaults to today)

4. Click Save Income.
5. The income entry will be added to the database.

---

7. Viewing & Managing Incomes

1. Navigate to the Income section.
2. View a list of all incomes with details.
3. To Edit an income:
   - Click the Edit icon beside the entry.
   - Update the required fields and save.
4. To Delete an income:
   - Click the Delete icon beside the entry.
   - Confirm deletion.

---

8. Generating Reports

1. Go to the Reports section from the navigation menu.
2. Choose the time period (daily, weekly, monthly, yearly).
3. View graphs/charts comparing income vs expenses.
4. See category-wise breakdowns (e.g., how much spent on food, travel, etc.).
5. Optionally, download or export the report (if feature enabled).

---

9. Budget Management

1. Navigate to the Profile/Settings section.
2. Set your monthly budget.
3. The dashboard will track your spending against the set budget.
4. Notifications/alerts will be shown if spending exceeds the budget.

---

10. Logging Out

1. Click on the Profile/Logout option.
2. Confirm logout.

3. You will be redirected back to the login page.

CHAPTER 9: LIMITATIONS AND FUTURE ENHANCEMENTS

9.1 Limitations

1. **Works only with Internet Connectivity**
    - Since the Expense Tracker is a **web-based application** connected to a cloud database (MongoDB), it requires a stable internet connection.
    - Users cannot access or update their expenses/incomes in offline mode.
    - This becomes a limitation for users in areas with **poor network coverage** or when they want to manage expenses while traveling without internet.
2. **Limited Analytics (Basic Charts Only)**
    - The current system provides **basic bar charts and pie charts** for visualizing income and expenses.
    - Advanced analytics such as **spending trends over time, predictions, or category-based comparisons across multiple months/years** are not included.
    - This limits the decision-making power for users who want **deep financial insights**.
3. **OTP Valid for Single Device Use Only**
    - The implemented OTP system is designed for **email verification** during login and account recovery.
    - OTPs expire after a short duration (10 minutes), and they are tied to a **single session/device**.
    - If the user tries to log in from multiple devices at once, the system may reject it, causing inconvenience.
4. **No Multi-Currency or Localization Support**
    - At present, the system assumes a **single currency format** (e.g., INR, USD).
    - Users in different regions cannot customize currency or language, which reduces global usability.
5. **Basic Budget Management**
    - The system allows setting a **monthly budget**, but it does not provide features like **budget alerts via notifications, rollover of unused budget, or comparative analysis across months**.
    - This limits financial planning capabilities.

9.2 Future Enhancements

1. **AI-Powered Expense Prediction**
    - Implement **machine learning models** to predict future expenses based on a user's past spending habits.
    - For example: If a user usually spends more on food at the start of the month, the system can alert them with a **spending forecast**.
    - AI can also suggest **cost-saving tips**, detect unusual spending patterns, and help with **financial planning**.
2. **Multi-Currency and Localization Support**
    - Add support for **multiple currencies** (USD, EUR, INR, GBP, etc.) so users across the globe can use the system.
    - Implement **real-time currency conversion** using APIs (like Forex APIs).
    - Add **language localization** so that users can operate the app in their **preferred language**.
3. **Export Reports in PDF/Excel**
    - Enable users to export their financial reports into **PDF and Excel formats**.
    - This will allow users to **print reports, share with accountants, or keep offline records**.
    - Export should include charts, summaries, and detailed expense/income tables.
4. **Mobile App Integration**
    - Develop a **cross-platform mobile application** (Android & iOS) using **React Native or Flutter**.
    - This will provide **real-time expense tracking** with push notifications.
    - Users can capture receipts via the phone camera and attach them directly to expenses.
5. **Advanced Notifications & Budget Alerts**
    - Add smart alerts for when the user is nearing or exceeding their budget.
    - Implement **push/email notifications** for due bills, savings reminders, and weekly summaries.
6. **Cloud Backup and Offline Mode**
    - Introduce **offline-first support**, where expenses/incomes added offline will be stored locally and synced to the cloud once the internet is available.

- This will improve accessibility for users in low-network regions.

7. **Integration with Bank APIs & Payment Systems**
   - Allow automatic **import of transactions** from bank accounts, digital wallets, or UPI apps.
   - Provide seamless integration with **payment gateways** for bill payments directly from the application.

CHAPTER 10: CONCLUSION

The **Expense Tracker System** successfully fulfills its primary goal of helping users **record, manage, and analyze their financial transactions** in a simple and secure manner. With features such as **user authentication, income and expense categorization, and visualization through charts**, the system provides a practical solution for individuals to keep track of their daily financial activities.

The implementation of this project has demonstrated how modern web technologies can be effectively used to design an application that combines **ease of use, security, and reliability**. The **database design** ensures structured data storage, while the **user-friendly interface** enables smooth interaction even for non-technical users.

The system currently meets the essential requirements such as:

- **Secure login and authentication** for data protection.
- **Income and expense tracking** with categories.
- **Basic reports and visual analytics** to understand spending patterns.
- **Simple and intuitive interface** suitable for all users.

However, the project also highlights areas of improvement that can enhance its usability and effectiveness in the future. Features like **AI-powered predictions, multi-currency support, mobile app integration, and advanced reporting options** can transform the system into a comprehensive financial assistant.

In conclusion, the Expense Tracker not only provides a **foundation for personal financial management** but also opens up opportunities for **scaling and enhancement** to meet the needs of a wider audience. It demonstrates the importance of technology in improving day-to-day financial decision-making and sets the stage for future innovations in digital expense management.