Personal Finance Tracker Application Project

Twenty-twoBCE11364 Divyansh Joshi

Twenty-twoBCE11413 Shambhavi Jha

Twenty-twoBCE10793 Raj Aryan Sharma

22BCE10239 Yash

Introduction

The Personal Finance Tracker offers a comprehensive solution for users to manage their finances efficiently. With a user-friendly interface, users can easily record their expenses and income, categorize transactions, set budget limits, and generate detailed reports. The application provides visual representations of financial data through charts, helping users analyze their spending patterns and make informed financial decisions. Built using the MERN stack (MongoDB, Express.js, React.js, Node.js), the application prioritizes security and privacy, ensuring users' financial data remains protected. With robust functionalities including user authentication and data management, the Personal Finance Tracker offers a seamless experience for users to take control of their finances.

Application Overview

With a clean interface and intuitive design, our Personal Finance Tracker allows users to effortlessly record transactions, categorize them, and analyze spending patterns. Users can:

- Set budgets for different expense categories
- Track spending against those budgets with visual progress indicators
- View financial data through interactive charts and graphs
- Generate comprehensive reports for better financial insights
- Record and categorize both income and expenses
- View recent transactions on a personalized dashboard

The application ensures security and privacy with JWT-based authentication, enabling users to manage their finances with confidence.

Scenario-Based Case Studies

Scenario 1: Personal Finance Management

User Profile: Ananya - Software Engineer
Scenario Overview: Ananya uses the application to take control of her finances by tracking expenses and income, setting budgets, and generating reports to analyze spending patterns.

Key Actions:

1. Registration and login.
2. Tracking expenses and income.
3. Budget management.
4. Generating financial analysis reports.

Scenario 2: Freelance Income Management

User Profile: Raj - Freelance Web Developer
Scenario Overview: Raj manages variable income and business expenses efficiently using the application.

Key Actions:

1. Tracking payments from clients.
2. Managing business expenses.
3. Tax planning.
4. Reviewing cash flow.

Scenario 3: Student Budget Management

User Profile: Priya - Graduate Student
Scenario Overview: Priya uses the application to budget her stipend, monitor expenses, and analyze spending for academic needs.

Key Actions:

1. Creating budgets.
2. Monitoring expenses.
3. Financial planning.
4. Planning semester-specific expenses.

Scenario 4: Family Budget Planning

User Profile: Vikram and Neha - Parents
Scenario Overview: Vikram and Neha organize family finances and teach financial responsibility to their children.

Key Actions:

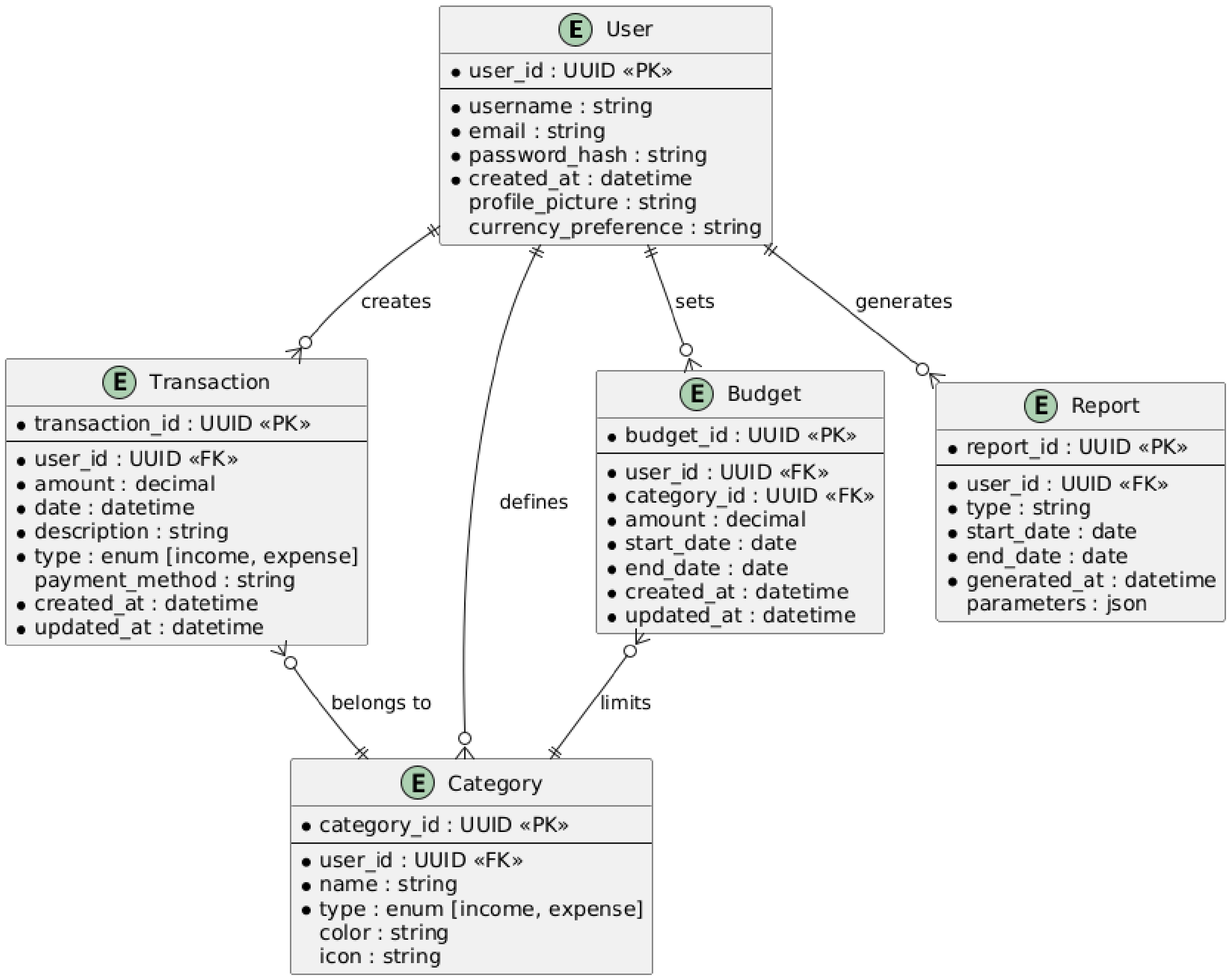
1. Setting up family finances.
2. Tracking expenses.
3. Reviewing budgets.
4. Financial education for children.

Technical Architecture

Architecture Components

1. **Frontend:** React.js, responsive design, Chart.js for visualization.
2. **Backend:** Express.js, Node.js, JWT authentication, RESTful APIs.
3. **Database:** MongoDB, Mongoose for data modeling.

Entity Descriptions



User Entity

- **Attributes:**
  - `user_id` (Primary Key): A unique identifier for each user.
  - `username`: The username selected by the user.
  - `email`: The user's email address.
  - `password_hash`: A hashed representation of the user's password for secure storage.
  - `created_at`: The date and time when the user account was created.
  - `profile_picture`: A URL or path to the user's profile picture.
  - `currency_preference`: The preferred currency for transactions.
- **Purpose:** Represents individual users of the system and their preferences.

Transaction Entity

- **Attributes:**
  - `transaction_id` (Primary Key): A unique identifier for each transaction.
  - `user_id` (Foreign Key): A reference to the associated user.
  - `amount`: The monetary value of the transaction.
  - `type`: Enum indicating whether the transaction is `income` or `expense`.
  - `description`: A brief description of the transaction.
  - `date`: The date of the transaction.
  - `payment_method`: The method used for the transaction (e.g., credit card, cash).
  - `created_at`: The timestamp of when the transaction was created.
  - `updated_at`: The timestamp of the last update to the transaction.
- **Purpose:** Tracks all financial transactions (both income and expenses) for users.

Budget Entity

- **Attributes:**
  - `budget_id` (Primary Key): A unique identifier for each budget.
  - `user_id` (Foreign Key): A reference to the associated user.
  - `category_id` (Foreign Key): The category this budget applies to.
  - `amount`: The total budgeted amount.
  - `start_date`: The start date of the budget period.
  - `end_date`: The end date of the budget period.
  - `created_at`: The timestamp of when the budget was created.
  - `updated_at`: The timestamp of the last update to the budget.
- **Purpose:** Defines financial plans for specific categories and time periods.

Category Entity

- **Attributes:**
  - `category_id` (Primary Key): A unique identifier for each category.
  - `user_id` (Foreign Key): A reference to the user who created the category.
  - `name`: The name of the category (e.g., "Food", "Travel").
  - `type`: Enum indicating whether the category is for `income` or `expense`.
  - `color`: A color code to visually represent the category.
  - `icon`: An icon associated with the category.
- **Purpose:** Categorizes transactions to help users organize their finances.

Report Entity

- **Attributes:**
  - `report_id` (Primary Key): A unique identifier for each report.
  - `user_id` (Foreign Key): A reference to the associated user.
  - `type`: The type of report (e.g., monthly summary).
  - `start_date`: The starting date for the report period.
  - `end_date`: The ending date for the report period.
  - `generated_at`: The timestamp of when the report was created.
  - `parameters`: Additional metadata or configuration for the report.
- **Purpose:** Provides summarized insights for users based on their transactions.

Relationships

1. **User and Transaction:**
   - Each user can have multiple transactions.
   - Represented by `user_id` in the `Transaction` entity referencing `user_id` in the `User` entity.
2. **User and Budget:**
   - A user can define multiple budgets.
   - Represented by `user_id` in the `Budget` entity referencing `user_id` in the `User` entity.
3. **User and Category:**
   - Each user can create multiple categories.
   - Represented by `user_id` in the `Category` entity referencing `user_id` in the `User` entity.
4. **User and Report:**
   - Each user can generate multiple reports.
   - Represented by `user_id` in the `Report` entity referencing `user_id` in the `User`

entity.

5. **Budget and Category:**
   - A budget can be specific to a category.
   - Represented by `category_id` in the `Budget` entity referencing `category_id` in the `Category` entity.
6. **Category and Transaction:**
   - Transactions belong to specific categories.
   - Represented implicitly by the `type` and `category` attributes in both entities

Key Technical Features

- Responsive dashboard with financial summaries and charts.
- Budget management with visual progress bars.
- Transaction management with filters and editing capabilities.
- Reporting system with customizable date ranges and visualizations.
- User profile management with currency preferences and account settings

Pre-requisites for Personal Finance Tracker Application

To develop a full-stack personal finance tracker application using React.js, Node.js, and MongoDB, there are several prerequisites. Below is a detailed guide:

Development Environment Setup

Node.js and npm

Install Node.js, which includes npm (Node Package Manager), on your development machine to run JavaScript on the server side.

- Download Node.js
- Installation Instructions

MongoDB

Set up a MongoDB database to store user data, transactions, and budget information. Install MongoDB locally or use a cloud-based MongoDB service like MongoDB Atlas.

- Download MongoDB
- Installation Instructions

Express.js

Express.js is a web application framework for Node.js. Install it to handle server-side routing, middleware, and API development.

```
npm install express
```

Frontend Setup

React

React is a JavaScript library for building user interfaces. Use Create React App to set up your project.

Install Create React App

```
npm install -g create-react-app
```

Create a New React Project

```
npx create-react-app personal-finance-tracker-frontend
cd personal-finance-tracker-frontend
```

Start the Development Server

```
npm start
```

Additional Frontend Libraries

- **Chart.js and react-chartjs-2** (For data visualization):

```
npm install chart.js react-chartjs-2
```

- **React Router** (For navigation):

```
npm install react-router-dom
```

Backend Setup

Additional Backend Libraries

- **Mongoose** (ODM for MongoDB):

```
npm install mongoose
```

- **JSON Web Token (JWT)** (Authentication):

```
npm install jsonwebtoken
```

- **bcrypt** (Password hashing):

```
npm install bcrypt
```

- **dotenv** (Environment variables):

```
npm install dotenv
```

- **cors** (Cross-Origin Resource Sharing):

```
npm install cors
```

Database Connectivity

1. Install Mongoose:

```
npm install mongoose
```

2. Create `db.js`:

```
const mongoose = require('mongoose');
require('dotenv').config();

const connectDB = async () => {
  try {
    await mongoose.connect(process.env.MONGO_URI, {
      useNewUrlParser: true,
      useUnifiedTopology: true,
    });
    console.log('MongoDB connected successfully');
  } catch (error) {
    console.error('MongoDB connection error:', error);
    process.exit(1);
  }
};

module.exports = connectDB;
```

Version Control

Use Git to manage version control.

- Download Git

Development Environment

Choose an editor or IDE:

- Visual Studio Code
- Sublime Text
- WebStorm

Running the Personal Finance Tracker Application

Steps to Run

1. Clone the repository:

```
git clone https://github.com/oldregime/MERN_Tracker
```

2. Install backend dependencies:

```
cd personal-finance-tracker/backend
npm install
```

3. Install frontend dependencies:

```
cd ../frontend
npm install
```

4. Start the backend server:

```
npm start
```

5. Start the frontend server:

```
npm start
```

6. Access the app at http://localhost:3000.

Features Implementation

1. **Responsive Dashboard**
   - Financial summary with visual charts.
   - Recent transactions list with filters.
2. **User Authentication**
   - Secure login/registration using JWT.
   - Password encryption.
3. **Transaction Management**
   - Add, edit, and categorize transactions.
4. **Budget Management**
   - Set budgets by category and track spending.
5. **Reports and Analysis**
   - Generate reports for time periods.
   - Export data for analysis.

Project Structure for Personal Finance Tracker

Overview

The project structure follows a modern approach with clear separation between frontend and backend components. It is organized to improve code maintainability and facilitate collaboration among developers.

Directory Structure

Frontend (React.js)

Directory Layout:

- **frontend/**
    - **public/**
        - `index.html`
        - `favicon.ico`
        - `assets/`
    - **src/**
        - **components/**
            - **common/**
                - `Navbar.js`
                - `Footer.js`
                - `Loader.js`
                - `Alert.js`
            - **auth/**
                - `Login.js`
                - `Register.js`
            - **dashboard/**
                - `FinancialSummary.js`
                - `ExpensePieChart.js`
                - `MonthlyBarChart.js`
                - `RecentTransactions.js`
            - **transactions/**
                - `ExpenseForm.js`
                - `IncomeForm.js`
                - `TransactionList.js`
                - `TransactionItem.js`
            - **budget/**
                - `BudgetForm.js`

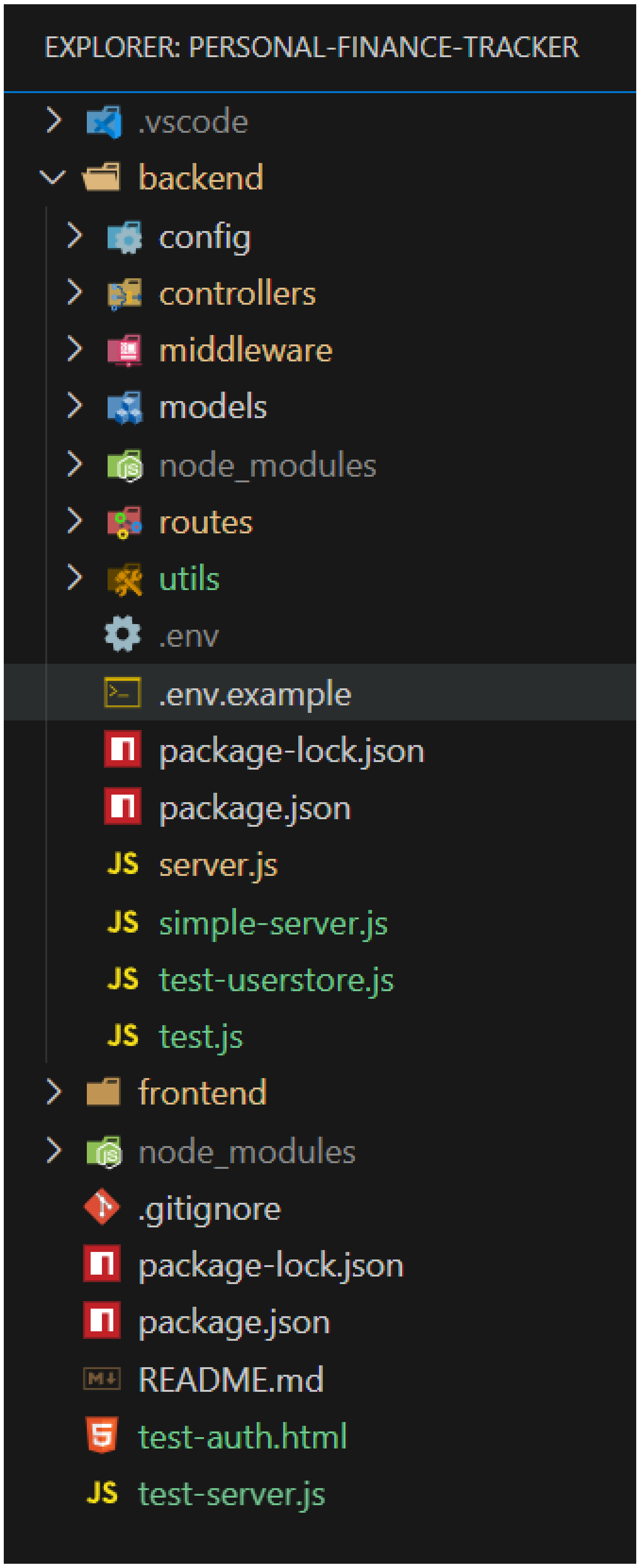
                - `BudgetList.js`
        - **pages/**
            - `Dashboard.js`
            - `Expenses.js`
            - `Income.js`
            - `Budgets.js`
            - `Reports.js`
            - `Profile.js`
            - `Login.js`
            - `Register.js`
        - **services/**
            - `authService.js`
            - `expenseService.js`
            - `incomeService.js`
            - `budgetService.js`
            - `dataService.js`
        - **hooks/**
            - `useAuth.js`
            - `useFetch.js`
        - **utils/**
            - `formatters.js`
            - `validators.js`
        - `App.js`
        - `index.js`
        - `index.css`
    - `package.json`
    - `README.md`

Backend (Node.js/Express)

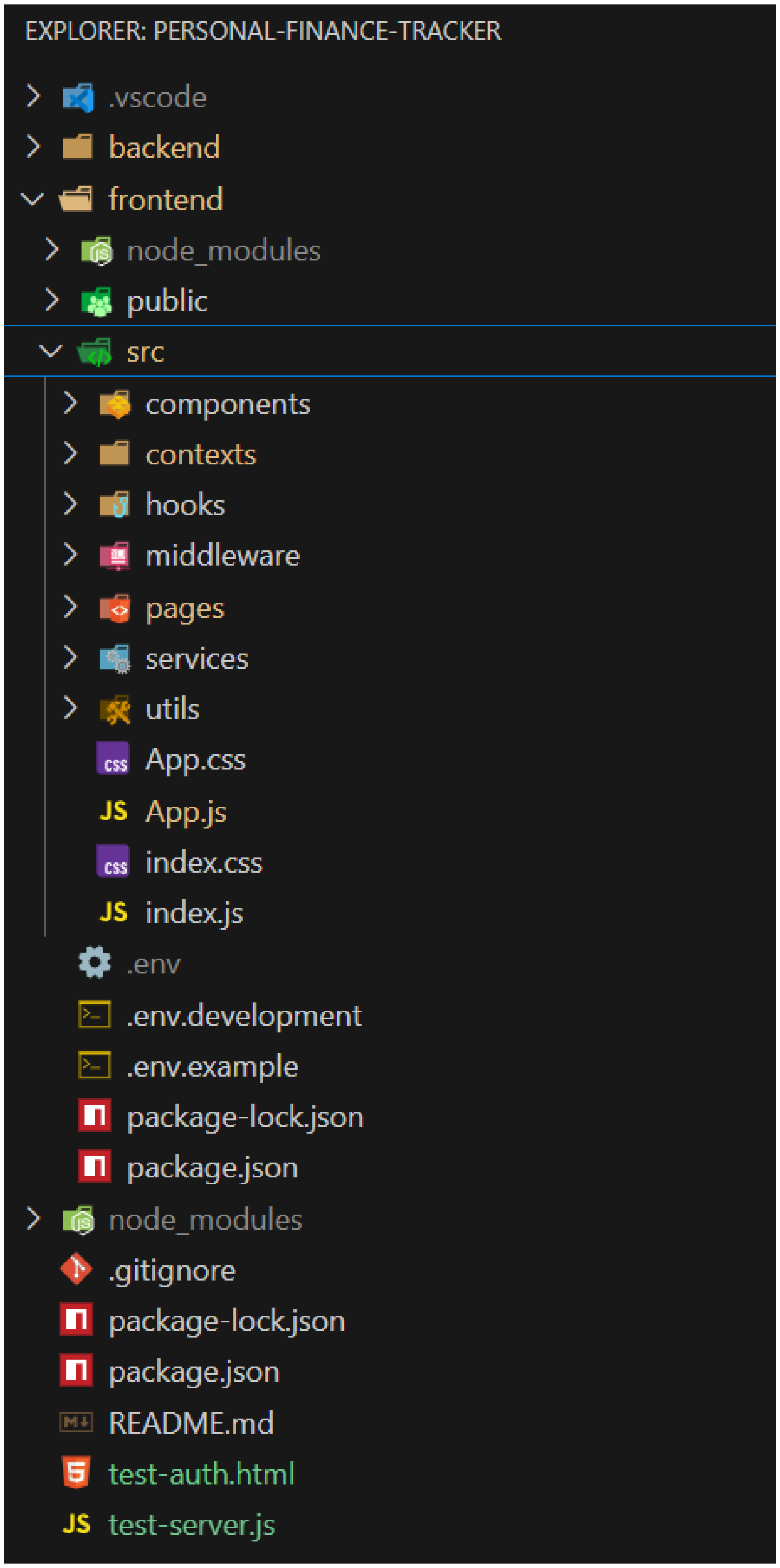
Directory Layout:

- **backend/**
    - **config/**
        - `db.js`
        - `default.json`
    - **controllers/**
        - `authController.js`
        - `expenseController.js`
        - `incomeController.js`
        - `budgetController.js`

  - `reportController.js`
- **middleware/**
  - `auth.js`
  - `errorHandler.js`
- **models/**
  - `User.js`
  - `Expense.js`
  - `Income.js`
  - `Budget.js`
  - `Category.js`
- **routes/**
  - `auth.js`
  - `expenses.js`
  - `income.js`
  - `budgets.js`
  - `reports.js`
- **utils/**
  - `generateToken.js`
- `server.js`
- `package.json`
- `README.md`



```
EXPLORER: PERSONAL-FINANCE-TRACKER
> .vscode
v backend
  > config
  > controllers
  > middleware
  > models
  > node_modules
  > routes
  > utils
    .env
    .env.example
    package-lock.json
    package.json
    server.js
    simple-server.js
    test-userstore.js
    test.js
> frontend
> node_modules
  .gitignore
  package-lock.json
  package.json
  README.md
  test-auth.html
  test-server.js
```



```
EXPLORER: PERSONAL-FINANCE-TRACKER
> .vscode
> backend
v frontend
  > node_modules
  > public
  v src
    > components
    > contexts
    > hooks
    > middleware
    > pages
    > services
    > utils
      App.css
      App.js
      index.css
      index.js
    .env
    .env.development
    .env.example
    package-lock.json
    package.json
> node_modules
  .gitignore
  package-lock.json
  package.json
  README.md
  test-auth.html
  test-server.js
```

Application Flow

1. Account Creation and Setup

- **Registration**: Users register by providing username, email, and password through the Register component.
- **Authentication**: Upon successful registration, JWT tokens are generated for secure authentication.
- **Profile Setup**: Users can set up and manage personal account details including name, contact information, and preferred currency through the Profile page.

2. Dashboard Experience

- Users are directed to the **Dashboard** (`Dashboard.js`) after login.
- Key components:
    - Financial summary statistics with visual indicators.
    - Expense category breakdown using pie charts.
    - Monthly income vs expenses comparison with bar charts.
    - Recent transactions with filtering options.

3. Expense and Income Tracking

- **Adding Expenses**:
    - Navigate to the Expenses page or use the quick-add form.
    - Enter details like amount, category, and date.
    - Saved to the database linked to the user's account.
- **Adding Income**:
    - Similar process as expenses.
    - Categorized (e.g., salary, freelance).
- **Managing Transactions**:
    - View, edit, delete, or sort transactions.

4. Budgeting and Goal Setting

- **Budget Creation**:
    - Set monthly limits for expense categories.
    - Recurring budgets supported.
- **Monitoring**:
    - Visual indicators track budget usage.

5. Reporting and Analysis

- **Report Generation**:
    - Accessible reports for any date range.
- **Data Visualization**:
    - Pie charts for spending, bar charts for trends.
- **Export Functionality**:
    - CSV or PDF format export options.

6. Data Security and Privacy

- **Authentication**:
    - JWT-based for secure access.
- **Data Protection**:
    - User-specific data isolation.

Project Flow
Let us start with the project development with the help of the given activities.

Project Setup and Configuration

Duration: 1 Hour
Skill Tags:

1. **Create project folders and files:**
   - Create folders for frontend and backend to organize code and install essential libraries.
   - **Folders:**
     - Backend folder
     - Frontend folder
2. **Install required tools and software:**
   - For the backend, install libraries such as:
     - Node.js
     - MongoDB
     - Bcrypt
     - Body-parser
   - For the frontend, use libraries such as:
     - React.js
     - Material UI
     - Bootstrap
     - Axios

After installation, the `package.json` files for both frontend and backend look like this:
(Insert `package.json` structure here)

Backend Development

Duration: 1 Hour
Skill Tags:

1. **Set Up Project Structure:**
   - Create a new directory and set up a `package.json` file using the `npm init` command.
   - Install dependencies such as Express.js, Mongoose, etc.
2. **Create Express.js Server:**
   - Set up an Express.js server to handle HTTP requests and API endpoints.
   - Configure middleware like `body-parser` and `cors`.
3. **Define API Routes:**
   - Create route files for functionalities like authentication, transactions, etc.
   - Implement route handlers using Express.js.
4. **Implement Data Models:**
   - Define Mongoose schemas (e.g., User, Transaction).
   - Create CRUD operations for each model.
5. **User Authentication:**
   - Implement authentication using JSON Web Tokens (JWT).
   - Create routes for registration, login, and logout.
6. **Handle Transactions:**
   - Enable users to make transactions with real-time updates.
7. **Admin Functionality:**
   - Add admin-specific routes and controllers to fetch data.
8. **Error Handling:**
   - Implement middleware to handle API request errors.

Database Development

Duration: 1 Hour
Skill Tags:

1. **Setup MongoDB:**
   - Install MongoDB locally or use MongoDB Atlas.
2. **Define Schemas:**
   - Create schemas for User and Expense using Mongoose.
3. **Create Models:**
   - Define Mongoose models for interacting with the database.
4. **Connect to MongoDB:**
   - Establish a connection using Mongoose in the app's entry point.
5. **Test Connection:**
   - Perform CRUD operations to test database connectivity.

Reference Video for Database Connection:

Frontend Development

Duration: 1 Hour
Skill Tags:

1. **Setup React Application:**
   - Install libraries like React, Material UI, and Bootstrap.
   - Create directories for components, styles, and assets.
2. **Design UI Components:**
   - Build reusable components for forms, listings, buttons, and notifications.
   - Ensure responsive design and consistent styling.
3. **Implement Frontend Logic:**
   - Connect frontend components with backend APIs.
   - Implement features like adding, editing, and deleting expenses.

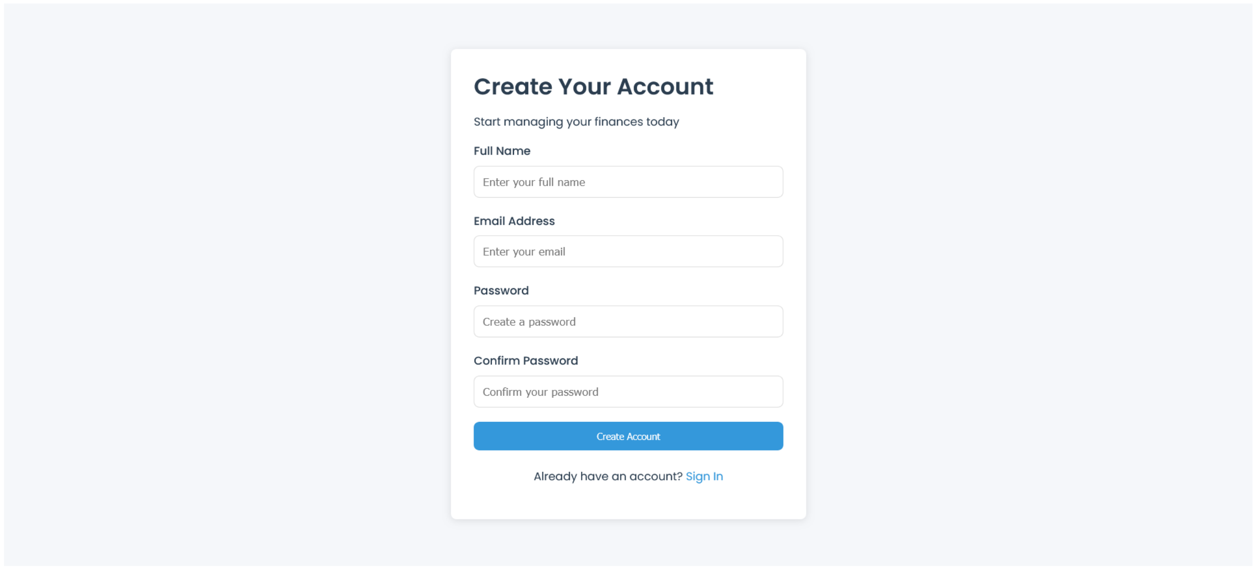
Project Implementation

Duration: 1 Hour
Skill Tags:

- Verify functionalities by running the application and testing for bugs.

Screenshots:

- Login Page



Create Your Account

Start managing your finances today

Full Name

Enter your full name

Email Address

Enter your email

Password

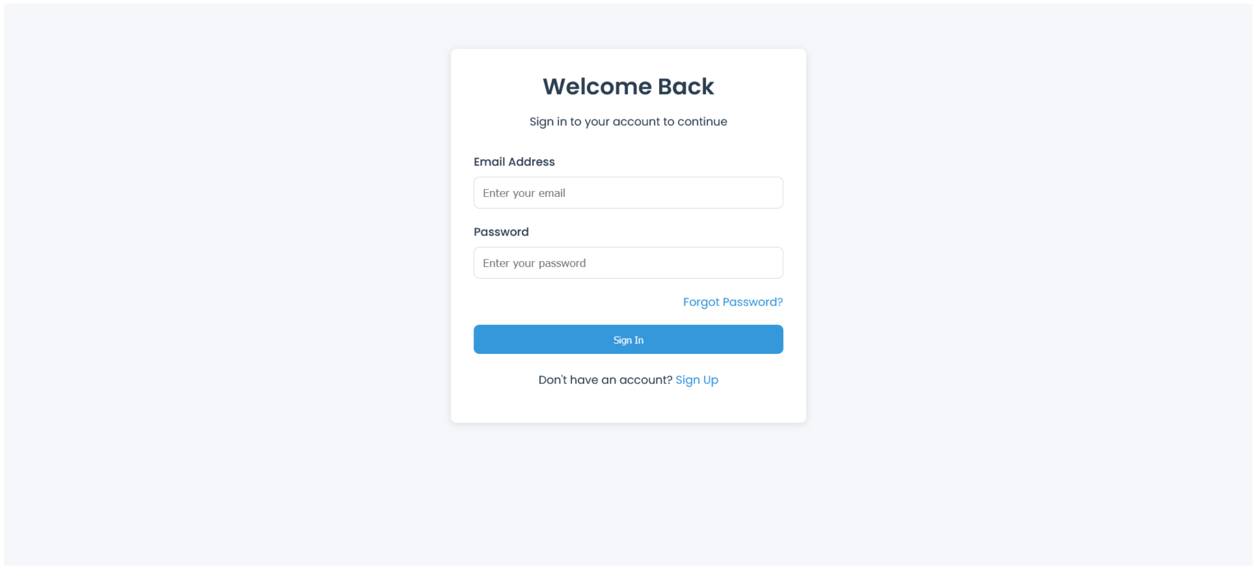
Create a password

Confirm Password

Confirm your password

Already have an account? Sign In

- Registration Page



Welcome Back

Sign in to your account to continue

Email Address

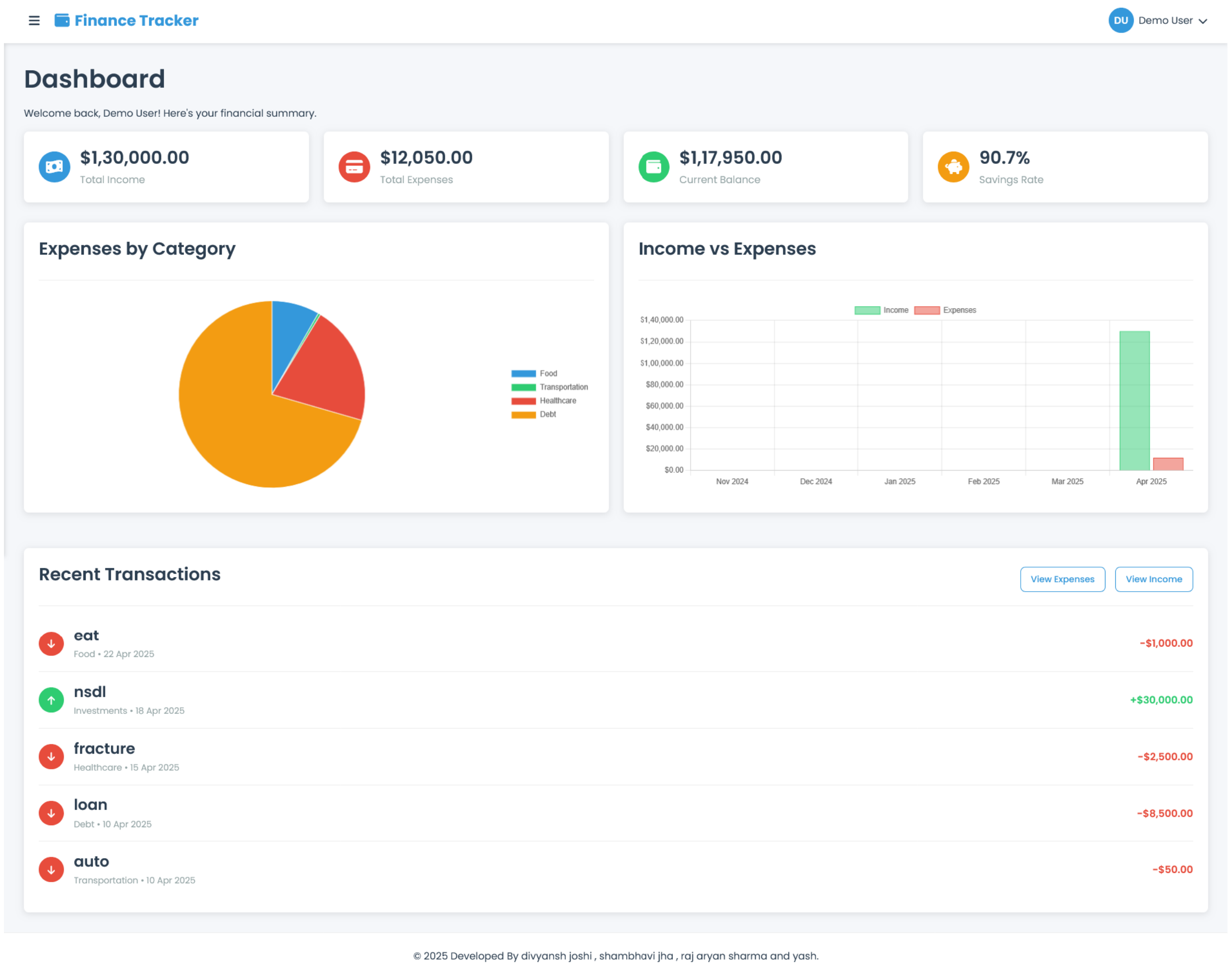
Enter your email

Password

Enter your password

Don't have an account? Sign Up

- Landing Page



Dashboard

Welcome back, Demo User! Here's your financial summary.

$1,30,000.00	$12,050.00	$1,17,950.00	90.7%
Total Income	Total Expenses	Current Balance	Savings Rate

Expenses by Category

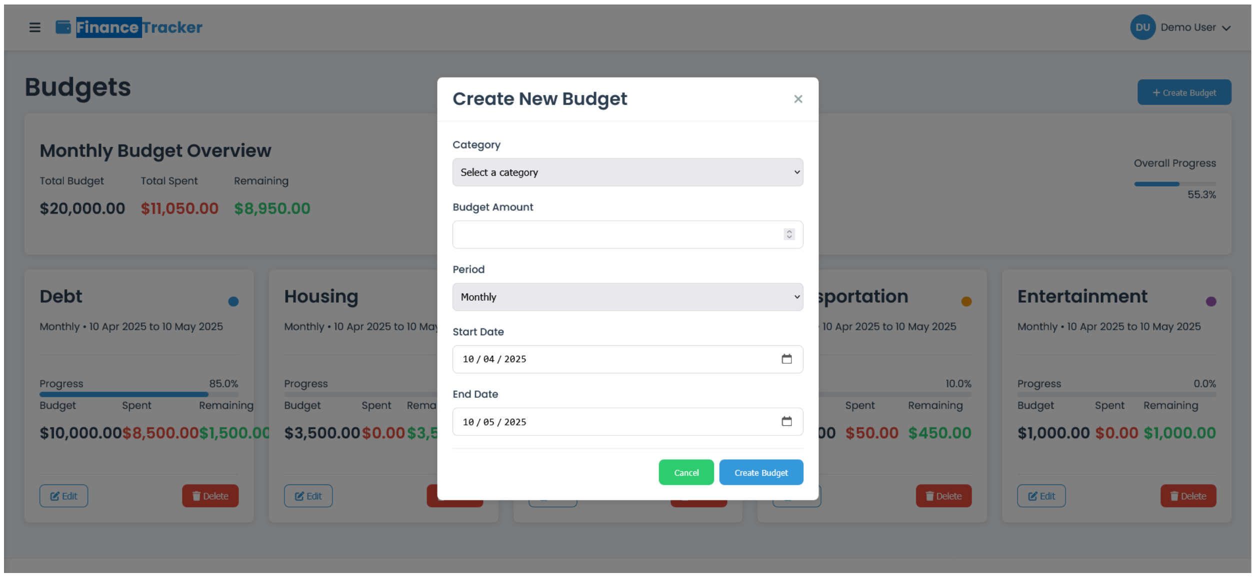
Income vs Expenses

Recent Transactions

eat Food • 22 Apr 2025	-$1,000.00
nsdl Investments • 18 Apr 2025	+$30,000.00
fracture Healthcare • 15 Apr 2025	-$2,500.00
loan Debt • 10 Apr 2025	-$8,500.00
auto Transportation • 10 Apr 2025	-$50.00

© 2025 Developed By divyansh joshi , shambhavi jha , raj aryan sharma and yash.

- User Page



Budgets

Monthly Budget Overview

Total Budget	Total Spent	Remaining
$20,000.00	$11,050.00	$8,950.00

Overall Progress 55.3%

Debt
Monthly • 10 Apr 2025 to 10 May 2025
Progress 85.0%

Budget	Spent	Remaining
$10,000.00	$8,500.00	$1,500.00

Housing
Monthly • 10 Apr 2025 to 10 May
Progress

Budget	Spent	Rema
$3,500.00	$0.00	$3,5

sportation
10 Apr 2025 to 10 May 2025
10.0%

Spent	Remaining
$50.00	$450.00

Entertainment
Monthly • 10 Apr 2025 to 10 May 2025
Progress 0.0%

Budget	Spent	Remaining
$1,000.00	$0.00	$1,000.00

Create New Budget ×

Category: Select a category

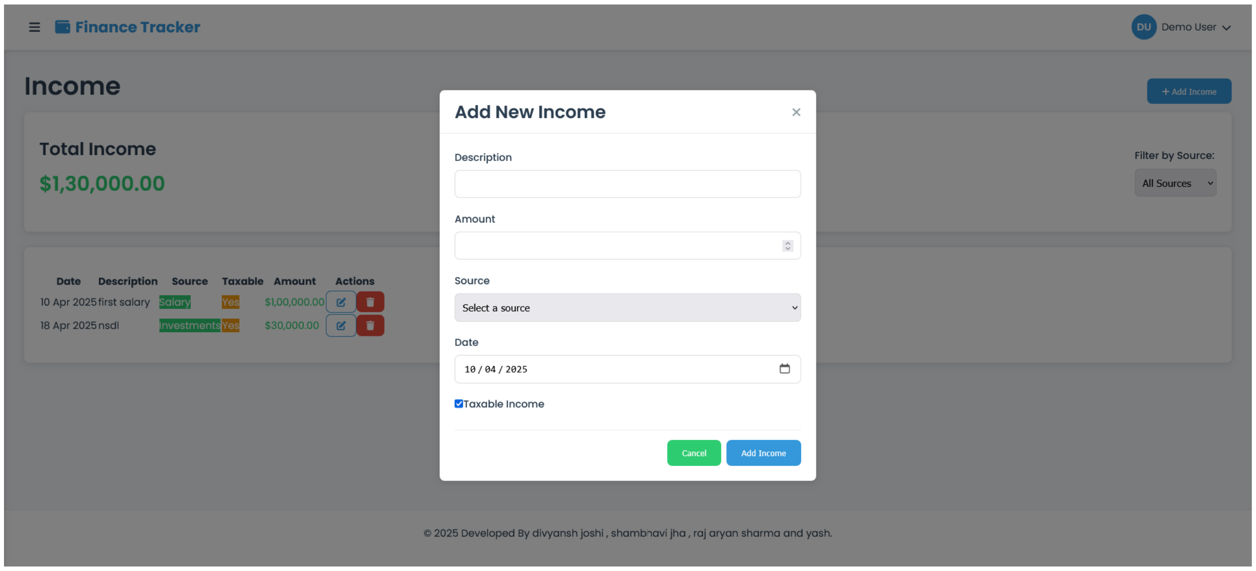
Budget Amount:

Period: Monthly

Start Date: 10 / 04 / 2025

End Date: 10 / 05 / 2025

- Add income



☰ Finance Tracker — DU Demo User

Income

+ Add Income

Total Income
$1,30,000.00

Filter by Source: All Sources

Date	Description	Source	Taxable	Amount	Actions
10 Apr 2025	first salary	Salary	Yes	$1,00,000.00	
18 Apr 2025	nsdl	Investments	Yes	$30,000.00	

Add New Income ×

Description:

Amount:

Source: Select a source

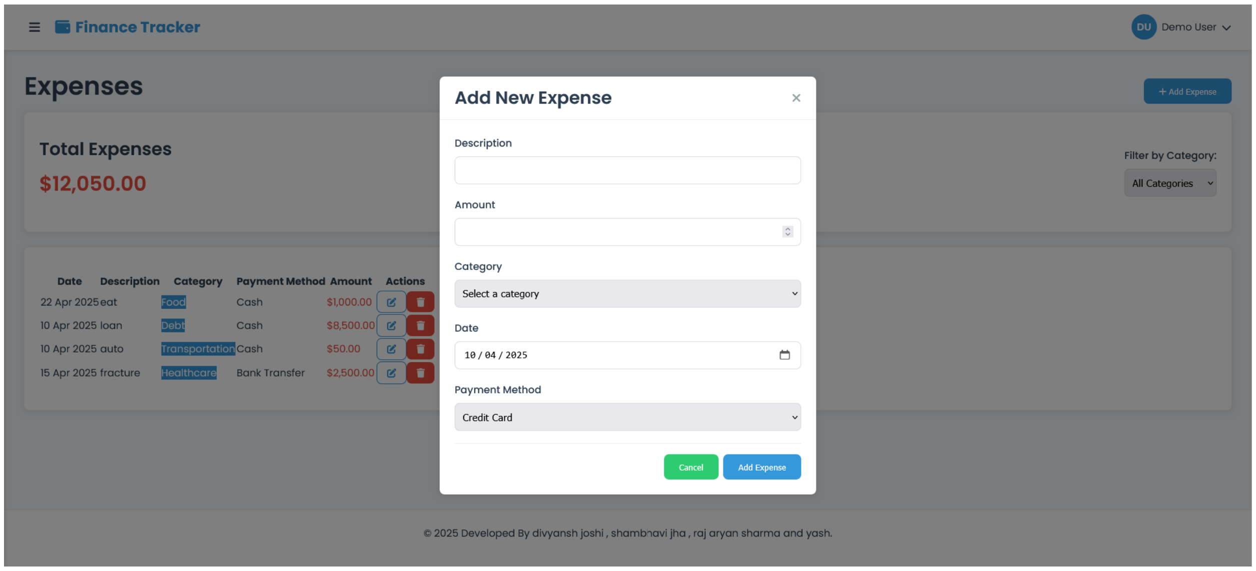
Date: 10 / 04 / 2025

☑ Taxable Income

Cancel | Add Income

© 2025 Developed By divyansh joshi , shambhavi jha , raj aryan sharma and yash.

- Expense Tracker Page



☰ Finance Tracker — DU Demo User

Expenses

+ Add Expense

Total Expenses
$12,050.00

Filter by Category: All Categories

Date	Description	Category	Payment Method	Amount	Actions
22 Apr 2025	eat	Food	Cash	$1,000.00	
10 Apr 2025	loan	Debt	Cash	$8,500.00	
10 Apr 2025	auto	Transportation	Cash	$50.00	
15 Apr 2025	fracture	Healthcare	Bank Transfer	$2,500.00	

Add New Expense ×

Description:

Amount:

Category: Select a category

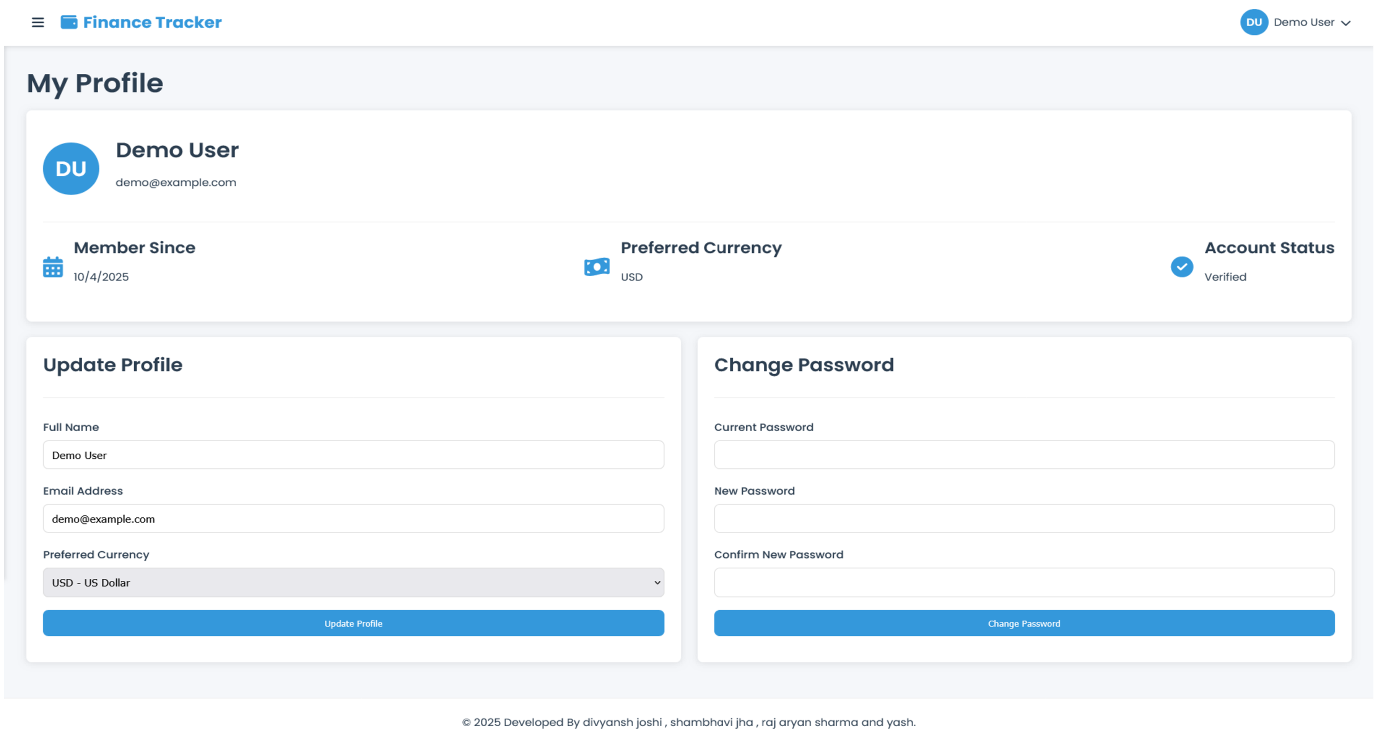
Date: 10 / 04 / 2025

Payment Method: Credit Card

Cancel | Add Expense

© 2025 Developed By divyansh joshi , shambhavi jha , raj aryan sharma and yash.

- Profile



My Profile

DU Demo User
demo@example.com

Member Since: 10/4/2025
Preferred Currency: USD
Account Status: Verified

Update Profile

Full Name: Demo User
Email Address: demo@example.com
Preferred Currency: USD - US Dollar

Change Password

Current Password:
New Password:
Confirm New Password:

© 2025 Developed By divyansh joshi , shambhavi jha , raj aryan sharma and yash.

For further assistance, refer to the following resources:

- Project Walkthrough Video

https://drive.google.com/file/d/1Q_pht41lsXsZc63zpi4-4U522w9isAab/view?usp=sharing

- Prjoect DRIVE folder

https://drive.google.com/drive/folders/1X_DNZAvoWCvj6qpT__fkKYSzUobLmpXN?usp=sharing

- Code Base

https://github.com/oldregime/MERN_Tracker

Conclusion

The Personal Finance Tracker empowers users to manage their finances effectively through intuitive tools, robust security, and comprehensive insights. Whether managing individual expenses or family budgets, it provides the necessary features to achieve financial goals and develop healthy habits.