Project Documentation: Expense Tracker Application

Project Overview

The Expense Tracker Application is a web-based platform designed to help users efficiently manage their personal finances by tracking expenses and incomes. The application allows users to categorize expenses, set monthly budgets, and visualize their spending patterns through interactive charts. The platform focuses on simplicity, ease of use, and data security.

Key Features

- User Registration and Authentication: Users can sign up, log in, and manage their personal profiles securely.
- Expense Management: Allows users to add, update, and delete expenses categorized by type (e.g., groceries, travel, rent).
- Income Tracking: Users can record and manage their incomes, assigning them to relevant categories.
- Budgeting: Users can set budgets for specific categories and receive alerts when they exceed their limits.
- Visual Reports: Graphical representation of expenses and incomes for better financial insights.
- Export Data: Users can export their financial data in CSV format for further analysis.

Technologies Used

- Backend: Node.js (Express)
- Database: MongoDB (for storing user data, expenses, and income records)
- Frontend: React.js
- Authentication: JWT for user authentication
- Validation: Express-validator for input validation

Setup Instructions

Prerequisites:

1. Node.js (v16 or later) installed on your local machine.

2. MongoDB (ensure it is running locally or use a cloud-based service like MongoDB Atlas).
3. IDE: Visual Studio Code or any text editor.

Steps to Set Up:

1. Clone the repository:

```
git clone https://github.com/yashfaujdar/expense-tracker.git
```

2. Configure Database:

Create a MongoDB database (e.g., expense_tracker_db). Update the connection settings in .env:

```
MONGO_URI=mongodb://localhost:27017/expense_tracker_db

JWT_SECRET=your_jwt_secret_key

PORT=5000
```

3. Install Dependencies:

```
npm install
```

4. Run the Application:

```
npm start
```

5. Access the Application:

Open your browser and go to http://localhost:5000 to access the platform.

API Documentation

User APIs

- POST /api/users/register - Register a new user
- POST /api/users/login - User login
- GET /api/users/profile - Get user profile

Expense APIs

- POST /api/expenses - Add a new expense
- GET /api/expenses - Get all expenses
- PUT /api/expenses/{id} - Update an expense

• DELETE /api/expenses/{id} - Delete an expense

Income APIs

• POST /api/incomes - Add a new income

• GET /api/incomes - Get all incomes

Budget APIs

• POST /api/budgets - Set a new budget

• GET /api/budgets - Retrieve budget information

Unit Testing

Unit tests have been implemented for the service layer using Jest. Test cases validate expense tracking, budget enforcement, and user authentication.

• To run unit tests:

```
npm test
```

Architecture and Design

The application follows the Model-View-Controller (MVC) architecture:

• Controller: Manages API requests and responses.

• Service: Contains business logic for handling expenses, incomes, and budgets.

• Model: Defines the data schema for expenses, users, and incomes.

• Repository: Interacts with MongoDB for CRUD operations.

Future Improvements

• Notifications: Implement email notifications for budget overages.

• Multi-Currency Support: Allow users to track expenses in multiple currencies.

• Mobile App: Develop a mobile application for on-the-go expense tracking.

• AI Insights: Use AI to suggest savings and investment opportunities based on spending patterns.

• Data Visualization: Enhance visual reporting with more detailed charts and filters.