# Personal Expense Tracker Documentation

## Project Overview

The **Personal Expense Tracker** is a user-friendly application designed to help individuals track their daily expenses and categorize them efficiently. It includes user authentication, expense management, and basic reporting to provide insights into spending habits.

## Functional Requirements

### 1. ****User Features****

* **User Authentication:**
  + User signup and login with JWT-based authentication.
  + Secure password storage using hashing (e.g., bcrypt).
* **Expense Management:**
  + Add a new expense with details like date, amount, category, and description.
  + Edit or delete existing expenses.
  + View a list of expenses sorted by date.
* **Expense Categories:**
  + Use predefined categories such as Food, Travel, Shopping, etc.
  + Allow users to create custom categories.
* **Reports:**
  + View a summary of expenses categorized by type.
  + Generate total expenses for a selected date range.

### 2. ****Non-Functional Requirements****

* **Responsive Design:**
  + Mobile-friendly user interface.
* **Performance:**
  + Optimize database queries for faster responses.
* **Security:**
  + Protect sensitive user data and prevent unauthorized access using JWT.

## Tech Stack

### 1. ****Frontend:****

* Angular (UI Development)
* ng-bootstrap or Angular Material (UI components and styling)
* ngx-charts (for data visualization in reports)

### 2. ****Backend:****

* Node.js with Express.js (Server-Side Logic)
* MongoDB (Database with Mongoose for object modeling)
* bcrypt (Password hashing)
* JSON Web Token (Authentication)

### 3. ****Tools and Libraries:****

* Postman (API Testing)
* Multer (File uploads for receipt images, optional)

## Database Schema

### 1. ****User Collection****

{

"\_id": "ObjectId",

"name": "String",

"email": "String",

"password": "String (hashed)",

"createdAt": "Date"

}

### 2. ****Expense Collection****

{

"\_id": "ObjectId",

"userId": "ObjectId",

"amount": "Number",

"category": "String",

"description": "String",

"date": "Date",

"createdAt": "Date"

}

### 3. ****Category Collection**** (optional, for custom categories)

{

"\_id": "ObjectId",

"name": "String",

"userId": "ObjectId" (null for default categories),

"createdAt": "Date"

}

## API Endpoints

### 1. ****Authentication APIs:****

* POST /api/auth/signup
  + Register a new user.
* POST /api/auth/login
  + Authenticate and log in a user.

### 2. ****Expense APIs:****

* GET /api/expenses
  + Fetch all expenses for the logged-in user.
* POST /api/expenses
  + Add a new expense.
* PUT /api/expenses/:id
  + Edit an existing expense.
* DELETE /api/expenses/:id
  + Delete an expense.

### 3. ****Report APIs:****

* GET /api/reports/summary?startDate&endDate
  + Fetch an expense summary categorized by type.

### 4. ****Category APIs:****

* GET /api/categories
  + Fetch predefined and custom categories.
* POST /api/categories
  + Add a custom category.

## Angular Components

### 1. ****Authentication Components:****

* LoginComponent: Login form.
* SignupComponent: Signup form.

### 2. ****Dashboard Components:****

* DashboardComponent: Displays the list of expenses and options to add/edit/delete expenses.
* AddExpenseComponent: Form to add or edit expenses.
* ExpenseListComponent: Displays a sortable and searchable list of expenses.

### 3. ****Report Components:****

* ReportsComponent: Shows charts and summaries of expenses.
* DateRangePickerComponent: Allows users to select a date range for reports.

### 4. ****Category Components:****

* CategoryListComponent: Displays a list of categories.
* AddCategoryComponent: Form to add custom categories.

## Angular Services

### 1. ****AuthService:****

* Handles user authentication, JWT storage, and logout functionality.

### 2. ****ExpenseService:****

* Manages CRUD operations for expenses.

### 3. ****CategoryService:****

* Handles fetching predefined and custom categories.

### 4. ****ReportService:****

* Fetches summary data for reports.

## Application Flow

### 1. ****Authentication Flow:****

* Users register or log in via the Angular frontend.
* The backend validates credentials and issues a JWT.
* The JWT is stored in localStorage and used for authenticated API requests.

### 2. ****Expense Management Flow:****

* Users add, edit, or delete expenses via forms in the Angular app.
* The changes are sent to the backend, where they are stored in MongoDB.

### 3. ****Report Generation Flow:****

* Users select a date range via the UI.
* The frontend sends a request to the backend to fetch summary data.
* The summary is displayed using charts and tables in the Angular app.

## Bonus Features

* **Receipt Uploads:**
  + Allow users to upload receipts as image files.
  + Use Multer for file handling in the backend and display receipts in the expense details.
* **Monthly Summary Emails:**
  + Send users a monthly expense summary using Node.js and Nodemailer.
* **Charts and Graphs:**
  + Integrate libraries like Chart.js or ngx-charts to visualize expense data.

## Testing

### 1. ****Frontend Testing:****

* Use Jasmine and Karma for unit testing Angular components and services.

### 2. ****Backend Testing:****

* Use Jest or Mocha for testing API endpoints.
* Test with valid, invalid, and edge-case data.

### 3. ****End-to-End Testing:****

* Use Protractor or Cypress to test the entire user flow.

## Deployment

### 1. ****Frontend:****

* Deploy the Angular application on a platform like Vercel, Netlify, or AWS S3.

### 2. ****Backend:****

* Host the Node.js server on platforms like AWS EC2, Heroku, or DigitalOcean.
* Use MongoDB Atlas for database hosting.

### 3. ****Environment Variables:****

* Store sensitive data like JWT secrets and database connection strings securely.

This documentation provides a comprehensive guide to developing the **Personal Expense Tracker** application. Let me know if you need further elaboration or additional features!