🚀 Deep Summary of the Full-Stack Expense Tracker Project

This **Expense Tracker** is a **full-stack** web application designed to **manage expenses, track budgets, view transactions, and analyze financial data**. It features **user authentication, role-based access control, CRUD operations for transactions, budgets, and categories, and a calendar view for expense tracking**.

🛠️ Project Structure

nginx Copy Edit

```
src
 ├── App.jsx                # Main entry component
 ├── main.jsx               # React application entry point
 ├── routes.jsx             # Handles route navigation
 ├── components/            # Reusable UI components
 │    ├── Common/           # Header, Footer, SideMenu
 │    ├── Transactions/     # Transaction-related components
 │    ├── Budgets/          # Budget-related components
 │    ├── Users/            # User management components
 │    ├── Categories/       # Category management components
 │    ├── Expenses/         # Expense form and filters
 │    ├── ProtectedRoute.jsx # Restricts access to certain pages
 ├── pages/                 # Main pages (Dashboard, Login, Transactions, etc.)
 ├── services/              # API calls and authentication logic
 ├── styles/                # Global and component-specific CSS
```

📌 Core Features & Their Flow

1 Authentication & Authorization

- **Login Page (** `Login.jsx` **)** allows users to authenticate.
- Uses **JWT-based authentication** (`sessionStorage` stores the token).
- **Role-based access control (** `ProtectedRoute.jsx` **)** ensures only admins can access sensitive pages (e.g., Users & Categories).

💡 **Flow:**

1. User enters credentials and logs in.
2. **Backend returns a JWT token**, which is stored in `sessionStorage` .
3. The token is attached to all API requests for authentication.
4. Pages like `Users.jsx` and `Categories.jsx` are **restricted** based on roles.

---

2 Navigation & Routing

- `App.jsx` → The main component that loads the **Header, Footer, and Routes**.
- `routes.jsx` → Defines application routes and restricts access to protected pages.
- `SideMenu.jsx` → Controls navigation based on **user role** (Admins see more options).

💡 **Flow:**

1. `App.jsx` initializes the app and checks if the user is logged in.
2. `SideMenu.jsx` displays menu items based on **user role**.
3. `routes.jsx` ensures users can only access allowed pages.

---

3 Managing Transactions

- `Transactions.jsx` → Displays all transactions with filters, pagination, and actions (edit/delete).
- `ExpenseForm.jsx` → Allows users to add/edit transactions.
- `TransactionFilter.jsx` → Filters transactions based on **category, date, and type (income/expense)**.

💡 **Flow:**

1. `Transactions.jsx` fetches **all transactions** from `/expenses/get/all` .
2. Users **filter** transactions (`TransactionFilter.jsx`).

3. Clicking "Add Transaction" opens `ExpenseForm.jsx`, which submits a **POST** request.
4. Clicking "Edit" pre-fills `ExpenseForm.jsx` and sends a **PUT** request on submission.
5. Clicking "Delete" removes the transaction via **DELETE** request.

4️⃣ Budget Management

- `Budgets.jsx` → Displays all budgets in a paginated list.
- `BudgetForm.jsx` → Allows users to **add/edit budgets** with a **category, amount, and time period**.
- `BudgetList.jsx` → Lists all budgets and provides edit/delete options.

💡 **Flow:**

1. `Budgets.jsx` fetches **all budgets** from `/budget/get/all`.
2. Users can **add/edit budgets** using `BudgetForm.jsx` (`POST` or `PUT` request).
3. Users can **delete budgets** (`DELETE /budget/delete/:id`).

5️⃣ Category Management

- `Categories.jsx` → Displays all categories with pagination.
- `CategoryForm.jsx` → Allows users to **add/edit categories**.
- `CategoryList.jsx` → Lists categories and enables editing/deletion.
- `CategoryPieChart.jsx` → Visualizes **income & expense distribution** across categories.

💡 **Flow:**

1. `Categories.jsx` fetches **all categories** from `/category/get/all`.
2. Users can **add/edit categories** (`CategoryForm.jsx`, `POST` or `PUT`).
3. Users can **delete categories** (`DELETE /category/delete/:id`).
4. The **pie chart (`CategoryPieChart.jsx`) updates** to reflect category-based spending trends.

6️⃣ User Management

- `Users.jsx` → Displays all users with pagination.
- `UserForm.jsx` → Allows admins to **add/edit users**.
- `UserList.jsx` → Lists users and provides actions for editing/deleting them.

💡 **Flow:**

1. `Users.jsx` fetches **all users** from `/users/get/all`.
2. Admins **add/edit users** (`POST` or `PUT` request).
3. Admins can **delete users** (`DELETE /users/delete/:id`).

7️⃣ Calendar View (Expense Tracking)

- `Calendar.jsx` → Displays **daily income & expenses** inside a **calendar view**.

💡 **Flow:**

1. Fetches **all expenses** from `/expenses/get/all`.
2. **For each day**, calculates total income & expenses.
3. Shows income in **green** and expenses in **red** inside the calendar.

📊 API Requests & Backend Communication

1️⃣ API Configuration (`api.js`)

- **Handles authentication** by adding `Authorization: Bearer <token>` to all API requests.
- Uses **Axios interceptors** to simplify requests.

2️⃣ API Endpoints Used

Feature	Method	Endpoint	Description
Auth	`POST`	`/auth/login`	Logs in user, returns token
Transactions	`GET`	`/expenses/get/all`	Fetches all transactions

Feature	Method	Endpoint	Description
	POST	/expenses/add	Adds a new transaction
	PUT	/expenses/update/:id	Updates an existing transaction
	DELETE	/expenses/delete/:id	Deletes a transaction
Budgets	GET	/budget/get/all	Fetches all budgets
	POST	/budget/set	Adds a new budget
	PUT	/budget/update/:id	Updates a budget
	DELETE	/budget/delete/:id	Deletes a budget
Categories	GET	/category/get/all	Fetches all categories
	POST	/category/add	Adds a new category
	PUT	/category/update/:id	Updates a category
	DELETE	/category/delete/:id	Deletes a category
Users	GET	/users/get/all	Fetches all users
	POST	/users/add	Adds a new user
	PUT	/users/update/:id	Updates a user
	DELETE	/users/delete/:id	Deletes a user

✅ Conclusion

This project provides a **full financial management system** with:

- **Authentication & role-based access control**
- **CRUD operations for transactions, budgets, users, and categories**
- **A calendar view for financial tracking**
- **Data visualization (Pie Charts for spending trends)**