**DotProduct Budget Tracker Application - Project Documentation**

**Overview**

The Budget Tracker is a full-stack web application built to help users manage their personal finances efficiently. It enables users to log in securely, record income and expenses, visualize financial summaries using interactive charts, and monitor monthly budgets. The goal of the project is to provide a simple yet powerful tool for personal financial management.

**Tech Stack**

**Frontend:**

* React (with Hooks)
* React Router
* Html and CSS
* Javascript
* Axios for HTTP requests
* D3.js for data visualizations

**Backend:**

* Django
* Django REST Framework (DRF)
* JWT Authentication

**Database:**

* SQLite (for development)

**Tooling & Deployment:**

* VS Code (Development)
* Postman (API Testing)

**Features**

**1. Authentication**

* JWT-based secure login/logout
* Persistent auth via localStorage

**2. Transactions Management**

* Add/Edit/Delete transactions
* Filter by income/expense/amount/date/description
* Search by amount/date/description
* Pagination support

**3. Dashboard**

* Financial summary
* D3.js charts for monthly trends
* Budget vs Actual comparison

**4. Budget Management**

* Set monthly budget
* Track remaining budget dynamically

**Folder Structure**

BudgetTracker/

|--- backend/ # Django Project Root

│ ├ budget/ # Django App

│ └──manage.py # Django Project Entrypoint

|--- frontend/ # React App

│ ├ components/ # Reusable components

│ ├ pages/ # Route pages (Login, Dashboard, etc.)

│ ├ api/ # Axios instance

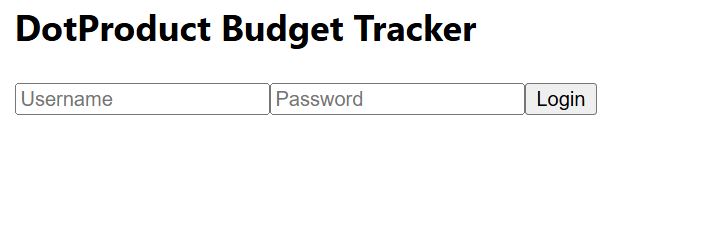
│ └── App.js

**Backend API Endpoints**

|  |  |  |
| --- | --- | --- |
| **Endpoint** | **Method** | **Description** |
| <http://localhost:8000/api/token/> | POST | Obtain JWT token |
| http://localhost:8000/api/token/refresh/ | POST | Refresh access token |
| http://localhost:8000/transactions/ | GET/POST | List or create transaction |
| http://localhost:8000/transactions/<id>/ | PATCH/DELETE | Update or delete transaction |
| http://localhost:8000/budget/ | GET/POST | Get or set monthly budget |
| http://localhost:8000/financial-summary/ | GET | Returns summarized financial data |

**Web Flow of the Tracker**

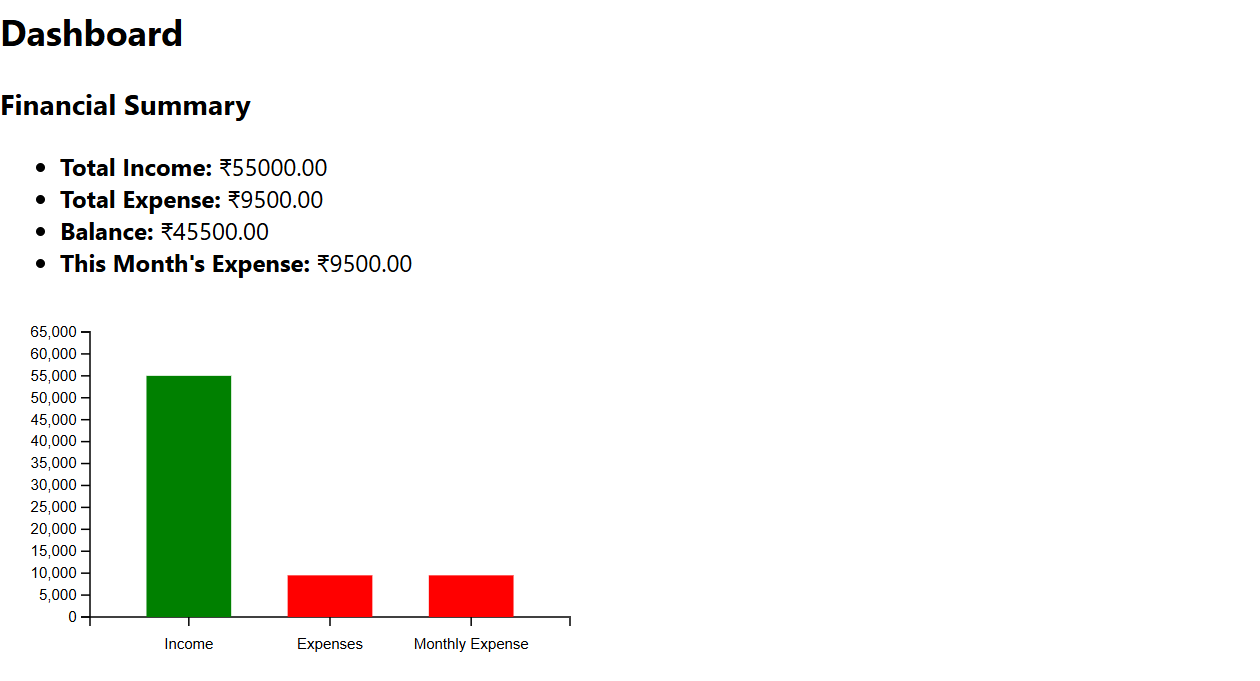
1. **Login Page**: <http://localhost:3000/login>
   * User enters credentials.
   * Username: **Priya**
   * Password: **qwerty@01**
   * JWT tokens are saved in localStorage on success.
   * Redirects to Dashboard.
   * Login Page



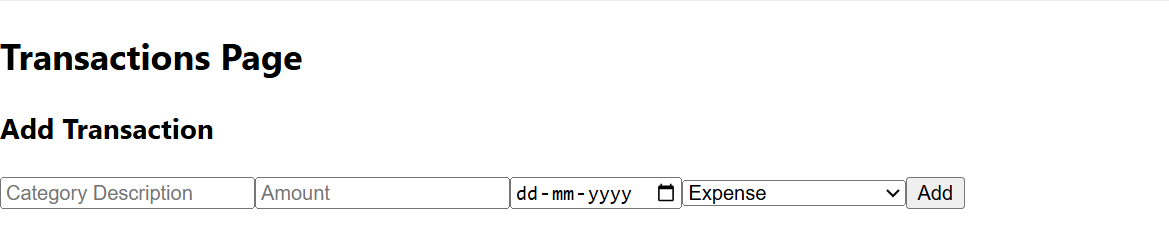
1. **Navigation Bar**



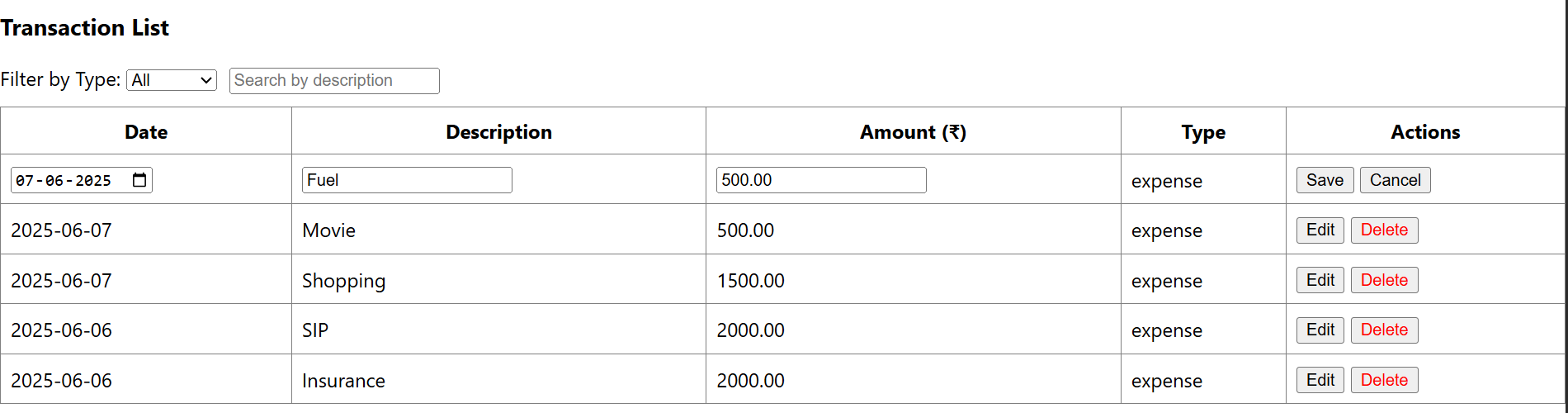
1. **Dashboard:** http://localhost:3000/dashboard
   * Shows income/expense summary.
   * Displays D3.js charts.
   * Shows monthly budget and remaining amount.
   * Dashboard Page
     1. Financial Summary

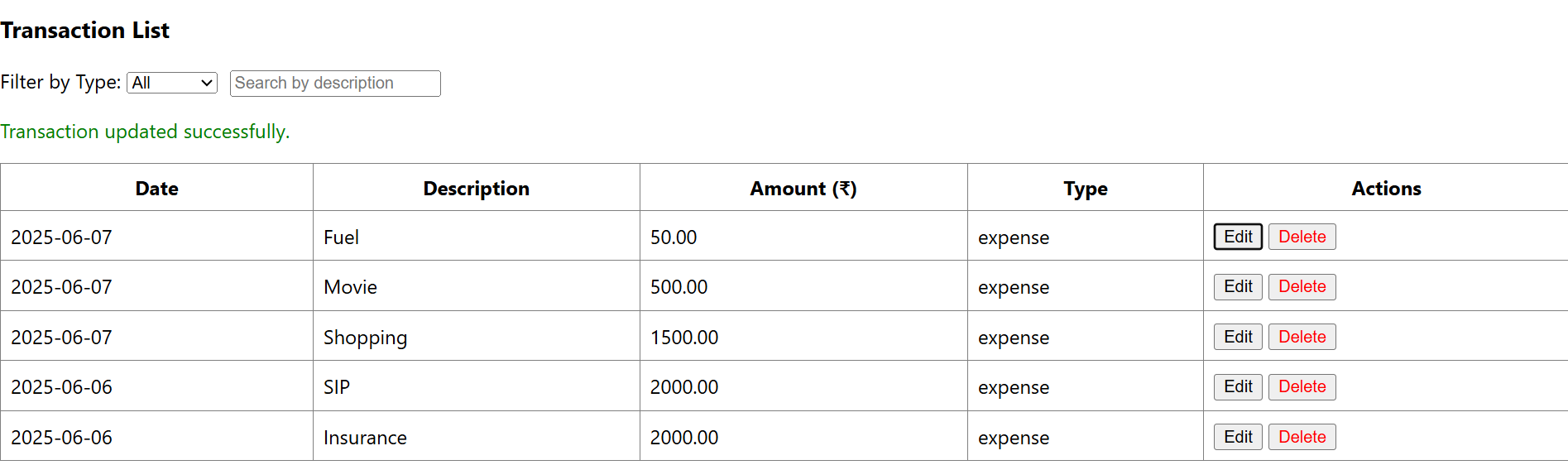


1. **Transactions**: http://localhost:3000/transactions
   * User can add/edit/delete transactions.
     1. New Transaction

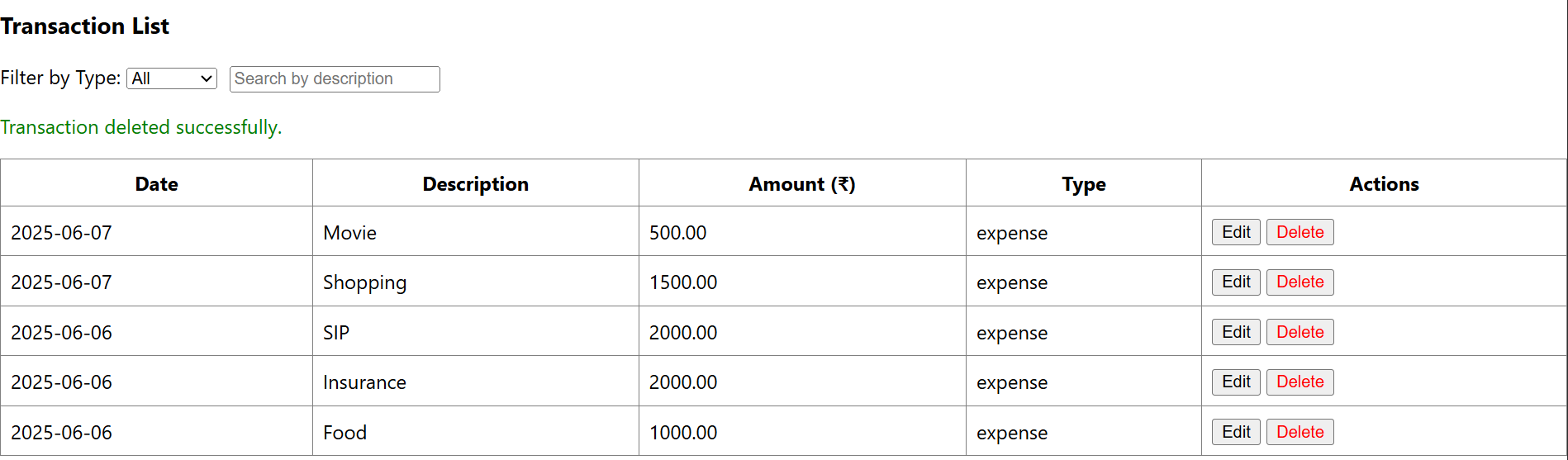


* + 1. Update Transaction

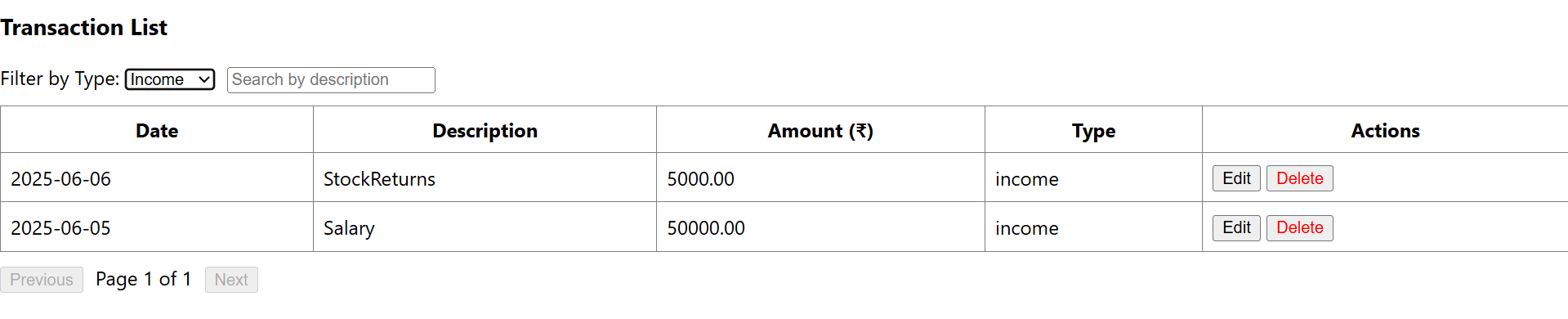




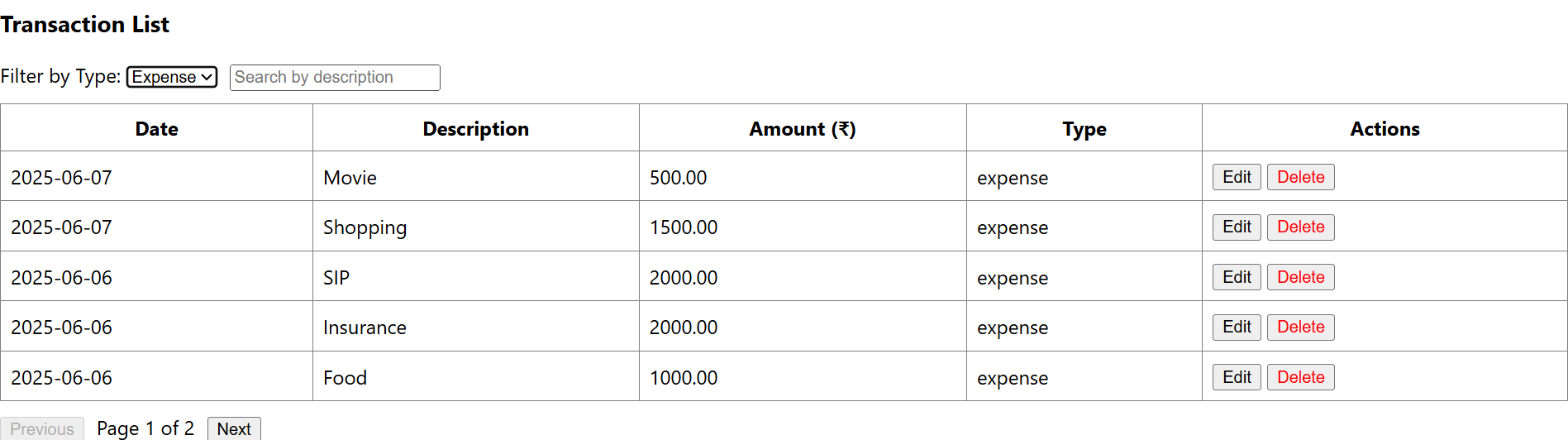
* + 1. Delete Transaction



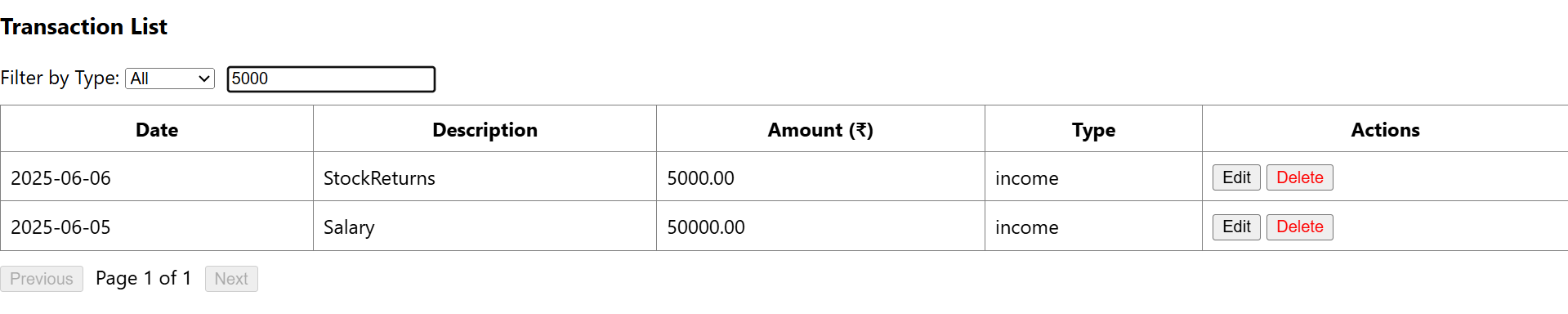
* + 1. Filter transactions by Income



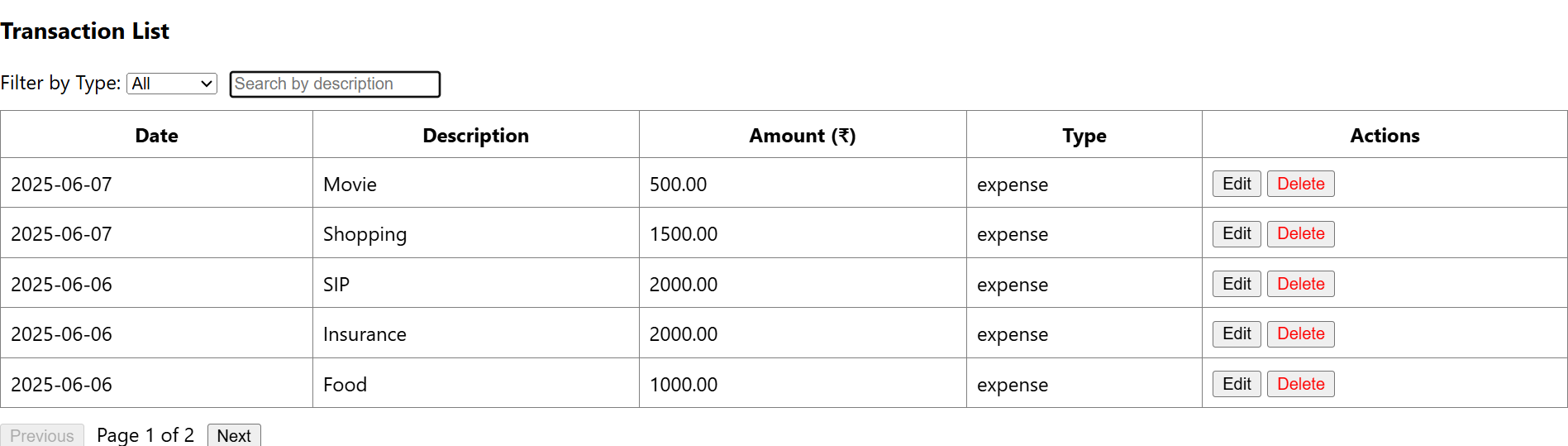
* + 1. Filter transactions by Expenses



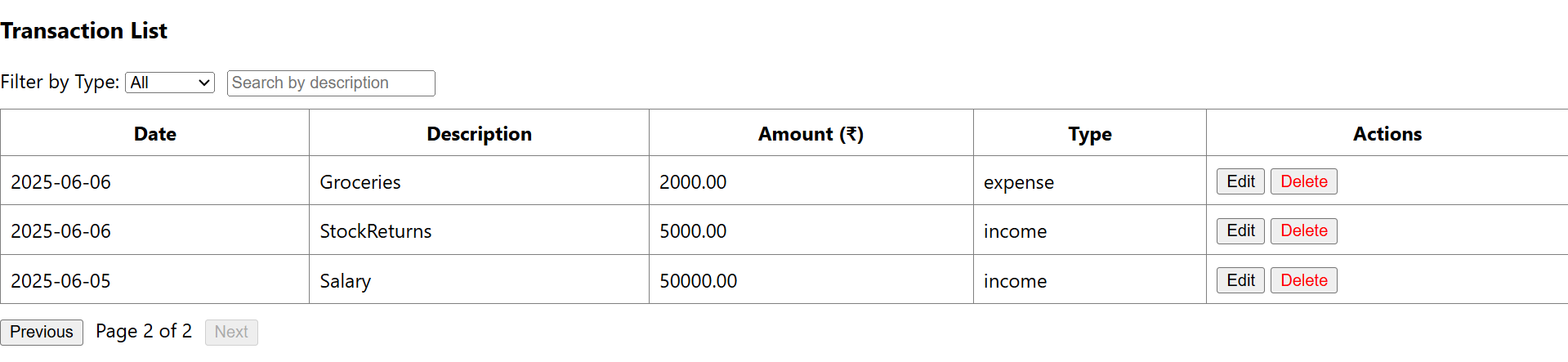
* + 1. Search transaction by Amount



1. **Pagination for better navigation**
   * 1. Current page



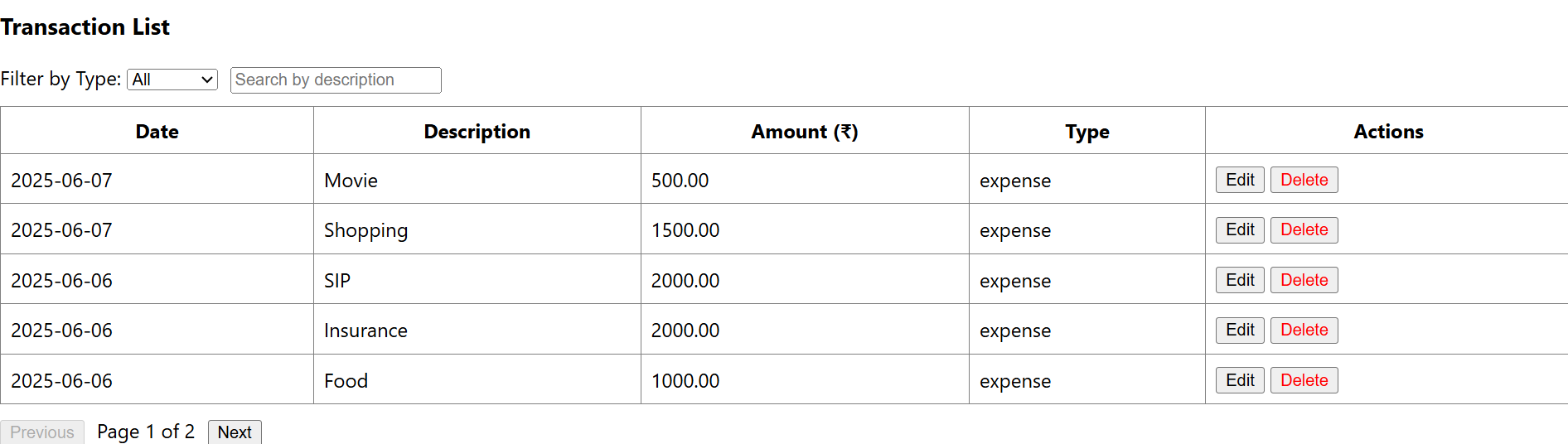
* + 1. Next page



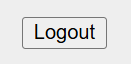
1. **Budget Setup** (part of Dashboard)
   * Allows users to set or view monthly budget.



1. **Transaction List**



1. **Logout**
   * Clears tokens and redirects to login.



**Component Responsibilities**

**React Components**

* **LoginPage:** Handles user login
* **Dashboard:** Displays charts & summary
* **TransactionForm:** Add/Edit transaction
* **TransactionList:** Lists transactions with edit/delete
* **BudgetInput:** Allows user to set monthly budget
* **Navbar:** Navigation and logout functionality

**Django Views**

* Class-based views for Category, Transaction, and MonthlyBudget
* Function-based views for summary calculations

**Involvement of LLM**

**LLM Assisted Code:**

* Axios instance setup and usage pattern
* JWT authentication integration in React
* Django DRF view optimizations and serializer refinement

**Human-Written Code:**

* UI styling decisions
* Custom state management patterns (especially around form handling)
* Pagination and filtering logic pattern suggestions
* D3.js chart setup and logic for Budget vs Actual graph and monthly summary
* Form validations and user experience flows
* End-to-end integration of frontend and backend
* Testing, debugging, and error state management

**Development Notes**

* All development was done using VS Code.
* Frequent commits and testing were done using Git.
* Authentication and authorization were implemented after backend APIs were tested.
* D3.js integration was the most challenging and also the most insightful.

**Future Enhancements**

* UI enhancements for web page
* Enhanced Graphs/dashboards
* Mobile-responsive UI with TailwindCSS or MUI
* Export transactions as CSV/excel/pdf
* Use Redux for larger state management
* Recurring Transactions (e.g., monthly rent, salary)
* Category-wise Budgeting with budget limits per category
* Currency Support and real-time exchange rate conversion
* Notifications & Reminders for overspending or due dates
* Automated Testing (Jest, Pytest)

**Mark this as end of document**