Shopping Cart Mini Project Documentation

Overview

This is a React-based shopping cart application demonstrating key React concepts, including functional components, hooks, Redux for state management, React Router for navigation, and Axios for API calls. The app includes product listing, cart management, authentication, and routing. Redux is implemented using only traditional reducers and configureStore from Redux Toolkit.

Features

- **Product Listing**: Displays products fetched from a mock API (JSON Server).
- **Cart Management**: Add, remove, and update quantities of items in the cart.
- **Authentication**: Simple login/logout with mock JWT token storage in localStorage.
- **Routing**: Navigate between home, cart, and login pages with protected routes.
- **State Management**: Redux with traditional reducers and configureStore.
- **API Integration**: Axios handles API calls with an interceptor for auth tokens.

Tech Stack

- **React**: Frontend library for building UI components.
- **React Router**: For client-side routing.
- **Redux Toolkit**: For global state management (using configureStore only).
- **Redux**: Traditional reducers for state logic.
- **Axios**: For making HTTP requests to APIs.
- **Tailwind CSS**: Recommended for styling. You can use any other CSS libraries as per your choice.
- **Mock Backend**: JSON Server for products; local simulation for auth.

Project Structure

```
src/
├── components/          # Reusable React components
|   ├── Header.js        # Navigation bar
|   ├── ProductList.js   # Lists all products
|   ├── ProductItem.js   # Single product card
|   ├── Cart.js          # Cart view with totals
|   ├── CartItem.js      # Single cart item
```

```
|  └── Login.js          # Login form
├── redux/               # Redux store and reducers
|  ├── store.js          # Redux store configuration
|  ├── authReducer.js    # Authentication reducer
|  └── cartReducer.js    # Cart reducer
├── services/            # API service
|  └── api.js            # Axios instance with interceptors
├── App.js               # Main app with routing
├── index.js/main.js             # Entry point with Redux Provider
└── index.css            # Basic styles (extend as needed)
```

Setup Instructions

1. **Create new react project using vite**:
2. **Install dependencies**:
    - npm install react-router-dom redux react-redux @reduxjs/toolkit axios
3. **JSON Server as Backend**:
    - Install JSON Server globally: npm install -g json-server@0.17.4
    - Create a db.json file with mock data (e.g., products, users).
    - Run: json-server --watch db.json --port 3001.

Redux Reducers

authReducer.js

- **State**: { user: null, token: string, isLoading: boolean }.
- **Actions**:
    - LOGIN_REQUEST: Sets isLoading to true.
    - LOGIN_SUCCESS: Sets user, token, and clears isLoading.
    - LOGOUT: Clears user and token, removes token from localStorage.
- **Implementation**:
    - Handles login with mock API response.
    - Persists token in localStorage.

cartReducer.js

- **State**: { items: array }.
- **Actions**:
    - ADD_TO_CART: Adds item or increments quantity.
    - REMOVE_FROM_CART: Removes item by ID.
    - UPDATE_QUANTITY: Updates item quantity.
- **Implementation**:
    - Manages cart items array with immutable updates.

API Service (api.js)

- **Purpose**: Configures Axios with a base URL and auth interceptor.
- **Features**:
    - Adds Authorization header with token from localStorage.
    - Base URL set to http://localhost:3001 (mock backend).
- **Usage**: Imported in components or thunks for API calls.

Routing

- **Library**: react-router-dom.
- **Routes**:
    - /: ProductList (home page).
    - /cart: Cart (protected, redirects to login if not authenticated).
    - /login: Login form.
- **Protected Routes**: Uses useSelector to check auth state in App.js.

Key React Concepts Covered

- **Functional Components**: All components use function syntax.
- **Hooks**:
    - useState: Manages form inputs and local state (e.g., quantity in CartItem).
    - useEffect: Fetches products on mount in ProductList.
    - useSelector/useDispatch: Accesses Redux state and dispatches actions.

- useNavigate: Handles navigation after login/logout.

- **Props**: Passes data (e.g., product to ProductItem, item to CartItem).
- **Conditional Rendering**: Shows loading states, auth status, and empty cart message.
- **Lists and Keys**: Maps products and cart items with unique keys.
- **Event Handling**: Buttons and form submissions (e.g., add to cart, login).
- **State Management**: Redux with traditional reducers.
- **Routing**: React Router for navigation.
- **API Integration**: Axios with interceptors for auth.

Styling

- Basic inline styles are used for simplicity.
- **Recommendation**: Use Tailwind / Bootstrap CSS for responsive styling.
  - Install: npm install tailwindcss.
  - Configure: Add Tailwind to index.css and update tailwind.config.js.
  - Apply classes like flex, grid, p-4, bg-gray-100 for layouts.

Mock Backend Setup

- **JSON Server**:
  - Create db.json:

```
{
  "products": [
    { "id": 1, "name": "Product 1", "price": 20, "image": "https://via.placeholder.com/150" },
    ...
  ],
  "users": [
        {"id": 1, "name": "Admin", password:  "Admin123", token :  "MockToken" },
        {"id": 2, "name": "Smith", password:  "Smith123", token :  "MockToken" }
   ]
}
```

  - Run: json-server --watch db.json --port 3001.

http://localhost:3001/products

http://localhost:3001/users

Final Execution

1. Start the app: npm run dev
2. Visit http://localhost:5173
    - Browse products on the home page.
    - Log in to access the cart (use localStorage to organize the cart)
    - Add/remove items in the cart and update quantities.
    - Log out to clear session.