Assignment 3 — ShopEase Multi-Page & Advanced Hooks

Objective:

Transform your existing **ShopEase E-commerce Application** into a **multi-page application** using **React Router** and implement **advanced React Hooks** including **useMemo**, **useCallback**, **useTransition**, and a **Custom Hook (useFetch)** for API integration.

Tasks (requirements)

1. **Set up React Router**
   - Install and configure **React Router DOM (v6)**.
   - Create routes for the following pages:
     - **Home**
     - **Products**
     - **Cart**
     - **About**
   - Each page should be a separate component under a pages/ directory.
2. **Product Details Page (Dynamic Routing)**
   - Create a dynamic route /products/:id to display individual product details.
   - Clicking a product on the Products page should navigate to its detail page.
   - Use useParams() to extract the product ID and display corresponding details.
3. **Custom Hook — useFetch (Data Fetching)**
   - Create a **Custom Hook** named useFetch to fetch data from an API.
   - Use the mock API: **https://fakestoreapi.com/products**.
   - The hook should handle:
     - Loading state
     - Error state
     - Fetched data state
4. **Product Filtering with useMemo / useCallback**
   - Implement category-based filtering on the Products page.
   - Use **useMemo** to optimize filtered product lists.
   - Use **useCallback** to memoize event handlers for filtering to prevent unnecessary re-renders.

5. **Loading State with useTransition**
   - Use **useTransition** to display a “Loading…” indicator or shimmer effect when switching between categories.
   - Ensure smooth UX by avoiding UI freezes during data changes.

Implementation notes (recommended)

• Use **React Router DOM (v6)** with <Routes> and <Route> components.
• Keep the **Navbar** visible across all pages (use <Outlet> or wrap routes).
• Organize code into folders like pages/, components/, hooks/, and context/.
• Use functional components with hooks (no class components).
• Keep API logic modular inside useFetch.
• You may reuse existing ProductCard from previous assignments for product display.
• Use basic CSS or Tailwind for styling; focus on layout and clarity.

Deliverables (what to submit)

1. **README.md** file containing:
   o Setup and run instructions (npm install, npm run dev).
   o Description of your routing setup and custom hook structure.
   o Notes on how optimization hooks (useMemo, useCallback, useTransition) are used.
2. **ZIP File of the Project** uploaded before the deadline.

Grading Rubric (suggested)

Functionality (60%)

- React Router setup with multiple pages *(15%)*
- Dynamic Product Details page works correctly *(15%)*
- Custom Hook (useFetch) implemented properly *(15%)*
- Filtering and optimization with hooks (useMemo/useCallback/useTransition) *(15%)*

Code Quality & Structure (25%)

- Organized folder structure and reusable components
- Proper hook implementation and naming conventions
- Clean, readable, and modular code

Styling & Presentation (15%)

- Clear navigation between pages
- Loading and filter transitions are visually smooth
- Responsive and consistent design

Submission Instructions

• Upload the **ZIP file** of your completed project before the deadline.
• Include a **README.md** file with setup instructions and implementation notes.