✅ Project Description (Professional)

- **Goal**
  - Build a React web app that lets users:
    - Browse products from a real API (Fake Store API)
    - Filter by category
    - Search by product title
    - View product details
    - Add/remove items in a cart (with quantity + totals)
- **Why it's a strong foundation project**
  - You'll practice:
    - API fetching patterns (list + categories + single item)
    - State management using hooks + lifting state
    - Component design with props (clean data flow)
    - Custom hooks (useProducts, useCart)
    - Clean separation of concerns (API layer vs UI)

🧱 Target Architecture (How it will look as a "real app")

Folder structure (professional)

- src/
  - api/
    - productsApi.js *(all fetch calls here)*
  - hooks/
    - useProducts.js *(fetch + filter + search + loading/error)*
    - useCart.js *(cart logic + localStorage optional)*
  - components/
    - CategoryFilter.jsx
    - SearchBar.jsx
    - ProductGrid.jsx
    - ProductCard.jsx
    - ProductDetails.jsx *(modal or page section)*
    - CartDrawer.jsx

        - CartItem.jsx
        - StatusMessage.jsx
    - utils/
        - debounce.js *(optional)*
        - formatCurrency.js
    - App.jsx

🔌 API Endpoints We'll Use (Fake Store API)

- **All products**
    - GET /products
- **All categories**
    - GET /products/categories
- **Products by category**
    - GET /products/category/:category
- **Single product details**
    - GET /products/:id

(We'll keep all of this inside productsApi.js.)

✅ Implementation Phases (Software Engineering Process)

Phase 0 — Requirements & Scope (Planning)

- **Deliverables**
    - Written feature list + UI sketch (simple)
    - Data model definition:
        - Product: id, title, price, description, category, image, rating
        - CartItem: id, title, price, image, quantity
- **Acceptance criteria**
    - User can browse products
    - Cart works correctly (add/remove/update)
    - Search and filter affect product list correctly

Phase 1 — Project Setup & Baseline UI

- **Tasks**
  - Create Vite React project
  - Setup basic layout:
    - Header (logo + cart icon)
    - Left/Top controls area (search + category filter)
    - Main grid (products)
    - Side drawer/modal (cart)
- **Deliverables**
  - Skeleton UI renders without API
  - Components created with placeholder data
- **Quality checkpoints**
  - ESLint optional
  - Clean component naming + folder structure

Phase 2 — API Layer (productsApi.js)

- **Purpose**
  - Keep fetch logic outside UI (professional separation)
- **Tasks**
  - Implement:
    - getAllProducts()
    - getCategories()
    - getProductsByCategory(category)
    - getProductById(id)
  - Add:
    - try/catch
    - response.ok checks
- **Deliverables**
  - You can import functions and test them with console logs
- **Acceptance criteria**
  - API functions return parsed JSON reliably
  - Errors are thrown with readable messages

Phase 3 — Products Hook (useProducts)

- **Purpose**
  - Central “products logic” (fetch + filter + search + UI states)
- **State inside the hook**
  - products, loading, error
  - categories
  - selectedCategory
  - searchText
- **Logic**
  - Fetch categories once on mount
  - Fetch products whenever category changes
  - Search runs on already-fetched products (client-side filtering)
- **Deliverables**
  - useProducts() returns:
    - productsToShow
    - categories
    - setters: setSearchText, setSelectedCategory
    - loading, error
- **Acceptance criteria**
  - Filtering + searching works without breaking pagination/fetch
  - Proper loading and error UI states

Phase 4 — Product List UI (Props & Components)

- **Components**
  - SearchBar (props: value, onChange, onClear)
  - CategoryFilter (props: categories, selected, onSelect)
  - ProductGrid (props: products, onSelectProduct, onAddToCart)
  - ProductCard (props: product, onSelect, onAddToCart)
- **Deliverables**
  - Full product browsing experience

- **Acceptance criteria**
    - Clicking product opens details
    - Add to cart works from card
    - UI does not crash when data is null (safe rendering)

Phase 5 — Product Details (Second API Call)

- **Purpose**
    - Teach you “fetch by ID” pattern (very real-world)
- **Design**
    - When user clicks a product:
        - Store selectedProductId
        - Fetch product details via getProductById(id)
- **Components**
    - ProductDetails (props: productId, onClose, onAddToCart)
- **Deliverables**
    - Details modal/panel with loading/error
- **Acceptance criteria**
    - Handles slow API correctly
    - Shows correct product every time (no stale data)

Phase 6 — Cart Hook (useCart) + Cart Drawer

- **Purpose**
    - Real state management + business logic
- **Cart behavior**
    - Add:
        - if exists → increase quantity
        - else → add new item quantity 1
    - Remove:
        - remove item completely
    - Quantity buttons:
        - + increases

        - - decreases (if hits 0 → remove)
    - Totals:
        - subtotal = sum(price * qty)
        - item count
- **Components**
    - CartDrawer (props: open/close)
    - CartItem
- **Deliverables**
    - Fully working cart
- **Acceptance criteria**
    - Cart totals correct
    - No duplicate lines for same item (only quantity updates)

Phase 7 — UX Improvements (Professional polish)

- **Add-ons**
    - Debounced search (so typing feels smooth)
    - Persist cart in localStorage (optional but professional)
    - Empty states (no products, empty cart)
    - Disable buttons during loading
    - Toast / status message on add-to-cart
- **Acceptance criteria**
    - App feels stable and “real”

Phase 8 — Testing & Validation (Lightweight but real)

- **Manual test checklist**
    - Products load
    - Categories load
    - Filter works
    - Search works
    - Open details works
    - Add/remove/update cart works

    - Refresh page (if localStorage enabled) keeps cart
- **Optional automated tests later**
    - Pure functions: cart reducers / helpers

Phase 9 — Deployment & Documentation

- **Deploy**
    - Vercel / Netlify / Cloudflare Pages
- **README**
    - Features
    - Tech stack
    - Folder structure
    - How to run
    - API used

✅ **Deliverables Summary (What you'll end with)**

- A clean React app with:
    - API layer
    - Custom hooks
    - Component-driven UI using props
    - Proper loading/error/empty handling
    - Cart functionality (real business logic)
    - Product details with a second fetch
    - Professional folder structure + README