# 🛒 CommerceOS: E-Commerce Platform (Frontend Simulation)

This project provides a complete, feature-rich frontend simulation for a modern e-commerce platform, demonstrating key functionalities like product browsing, shopping cart management, a multi-step checkout process, and role-based access control for inventory management.

## 🌟 Project Status & Scope

Due to the single-file environment constraint, this application is a highly interactive React frontend simulation. All data persistence (products, cart, user roles) is handled through React state and context. It includes detailed comments showing where real API calls to a backend (Node.js/MongoDB) and payment processing (Stripe API) would be integrated.

## ✨ Key Simulated Features

|  |  |  |
| --- | --- | --- |
| Feature | Description | Implementation Details |
| Product Listings & Filters | Displays a list of available products with images, price, and stock status. | Products are rendered from a static MOCK\_PRODUCTS\_DB. |
| Search & Filtering | Users can search products and filter by categories like Electronics, Apparel, and Home Goods. | Implemented using useMemo for optimized filtering and sorting. |
| Shopping Cart | Add, view, update, or remove cart items. | Managed with ShopContext and maintains a running subtotal. |
| Multi-Step Checkout | Two-step checkout: Shipping → Payment → Success. | Step-based UI that simulates form flow. |
| User Authentication | Sign-in and sign-out determine user roles (guest, user, admin). | Uses mock credentials and state-based role assignment. |
| Admin Panel (Inventory) | Accessible only by admin users for CRUD operations on product list. | Implemented using AdminDashboard and AdminProductForm components. |

## 🛠 Tech Stack

* Frontend
* React: Functional components and Hooks (useState, useEffect, useContext).
* Styling: Tailwind CSS for responsive UI.
* Icons: Lucide React for interface icons.
* State Management: React Context API.
* Full-Stack Integration (To Be Added)
* Designed to integrate with:
* Backend: Node.js / Express
* Database: MongoDB or MySQL
* Payments: Stripe API

## 🔑 Mock Credentials for Testing

|  |  |  |  |
| --- | --- | --- | --- |
| Role | Email | Password | Access |
| Standard User | user@shop.com | user | Shopping, Cart, Checkout |
| Admin User | admin@shop.com | admin | Shopping, Cart, Checkout, Admin Panel |

## 🚀 Full-Stack Deployment Steps

1. 1. Setup Backend: Create a Node.js/Express server and connect to MongoDB or MySQL.
2. 2. API Endpoints: Implement secure REST APIs for authentication, products, and orders.
3. 3. Payment Integration (Stripe): Use Stripe API to handle secure payment transactions.
4. 4. Replace Mock Logic: Update ShopProvider CRUD functions to interact with backend APIs using fetch or axios.

Example APIs:  
/api/auth/login  
/api/auth/logout  
/api/products (GET, POST, PUT, DELETE)  
/api/orders  
/api/checkout