

■ Prompt: Build a Full-Stack Stationery E-commerce Website (PERN Stack)

Goal:

Develop a fully functional Stationery E-commerce platform using the **PERN stack** with **user authentication, role-based access, and robust product, cart, and order management, including returns.**

🔧 Technical Requirements

1. Authentication and Authorization

- Implement **user authentication** (Sign-Up, Login) using **JWT**:
 - **Access Token** (short-lived)
 - **Refresh Token** (long-lived)
- Store refresh token securely (e.g., HTTPOnly cookie or secure local storage).
- Middleware to **protect routes** and verify roles.
- Two roles:
 - **Admin** – Full access to product/order management.
 - **User** – Can view, purchase, and return products.

2. Role-Based Access Control

- Only **Admins** can:
 - Add, update, or delete products.
 - View all orders.
 - Update order status.
- **Users** can:
 - View products
 - Add to cart
 - Place orders
 - View own orders
 - Request returns

3. Product Management

- Admin panel with full **CRED** operations:
 - **Create , Read , Edit , Delete** products
 - Manage **stock status**: **in-stock , sold , out-of-stock**
- Product Schema:

```
{ "title": "Notebook", "description": "120 pages, A4 size", "price": 100, "stock": 20, "status": "in-stock" }
```

4. Product Detail Page

- Single product page with:
 - Image
 - Title
 - Price
 - Description
 - "Add to Cart" button
 - Stock availability

5. Cart Page

- Authenticated users can:
 - Add products to cart
 - Increase/decrease quantity
 - Remove items
 - View total price
 - Proceed to checkout

6. Order Management

- Users can place orders from cart.
- Store order details including:
 - Product info
 - Quantity
 - Total price
 - Order status (pending , in-transit , delivered , cancelled , returned)
- Admins can:
 - View all orders
 - Change order status
- Users can:
 - View their orders
 - Track order status

7. Return Management

- Users can request return if order is marked delivered

- Admin can:
 - Approve or reject return
 - Update order status to `returned` if approved
 - Track return lifecycle in the order model
-

Tech Stack

Frontend (React)

- React + React Router
- Redux or Context API for state
- Axios for API requests
- JWT token handling (Access & Refresh)
- Role-based route protection
- Admin Dashboard
- Product List, Detail, Cart, Orders

Backend (Node.js + Express)

- Express REST API
- JWT-based Auth
- Middleware for auth & roles
- Routes:
 - `/auth/register` , `/auth/login` , `/auth/refresh` , `/auth/logout`
 - `/products` (GET, POST, PUT, DELETE)
 - `/cart` (user-specific)
 - `/orders` (user/admin)
 - `/returns`

Database (PostgreSQL)

- Tables:
 - `users` (id, name, email, passwordHash, role)
 - `products` (id, title, description, price, stock, status)
 - `cart_items` (user_id, product_id, quantity)
 - `orders` (id, user_id, total, status, created_at)
 - `order_items` (order_id, product_id, quantity, price)
 - `returns` (id, order_id, user_id, reason, status)
-

Optional Enhancements

- File/image upload with cloud storage (Cloudinary/S3)
- Email notifications for orders/returns
- Pagination & filtering on products

- Responsive design for mobile
- Analytics dashboard for admin