Intern Task – E-Commerce Product & Order Tracker

Overview

Build a small but complete e-commerce prototype with two roles—Admin and Customer. Customers can register, log in, browse products, and place orders. Admins can manage products and update order statuses. Focus on clean code, working functionality, and clear documentation.

User Roles

- Customer: Register, log in, view/filter products, add to cart, place orders, view order history.
- Admin: Log in, add/update/delete products, view all orders, update order status.

Backend Requirements

1. User Authentication (JWT)

- Register with role (admin or customer), email, and password (hashed).
- Login returns a JWT access token (set a reasonable expiry).
- Secure endpoints by role; use guards/middleware in NestJS.

2. Products

- Admin: Create, update, delete products.
- Customer: List products; filter by category.
- Suggested fields: id, name, description, price, category, imageUrl (optional), stock.

3. Orders

- Customer: Place an order by selecting products and quantities.
- Admin: Update order status: pending → shipped → delivered.
- Suggested fields: id, userId, items[{ productId, quantity, priceSnapshot }], total, status, createdAt.

Minimum API Endpoints (sketch)

Method	Path	Auth	Description
POST	/auth/register	Public	Create account with role (admin/customer).
POST	/auth/login	Public	Return JWT on valid credentials.
GET	/products	Public	List products; support ?category= and ?q= (optional).

Method	Path	Auth	Description
POST	/products	Admin	Create product.
PATCH	/products/:id	Admin	Update product.
DELETE	/products/:id	Admin	Delete product.
POST	/orders	Customer	Place order from cart items.
GET	/orders/me	Customer	List current user's orders.
GET	/orders	Admin	List all orders.
PATCH	/orders/:id/status	Admin	Update order status.

Frontend Requirements

- Authentication: Login & registration pages; store JWT securely.
- Products: List products from backend; filter by category; Add to cart (customers).
- Orders:
  - Customer: View order history & statuses.
  - Admin: View all orders & update statuses.
- UI: Responsive layout, minimal but clean styling.

Non-Functional Expectations (recommended)

- Input validation and basic error handling on both client and server.
- Password hashing, role-based guards, and CORS configured appropriately.
- Seed data for at least one admin and a few products.
- Environment variables for secrets and DB connection (document in README).

Deliverables

- Loom Demo Video (5–10 min): Show features for customer and admin. Include a quick tour of API calls via Postman/Swagger.
- GitHub Repository: Clear folder structure (backend/, frontend/).
- README: Setup instructions, .env sample, API docs, and project overview.

Bonus / Creativity Space

- Search bar for products.
- Pagination.
- Dashboard with order counts.
- Unit & integration tests.
- Dockerize the project.

Tech Requirements

- Backend: NestJS
- Frontend: React or Next.js
- Database: PostgreSQL, MySQL, or MongoDB
- Auth: JWT
- Styling: Any framework or custom

Evaluation Criteria

- Completeness of required features for both roles.
- Code quality and project structure (modular NestJS modules, clear React/Next.js components).
- Security & robustness basics (hashing, guards, validation, error handling).
- UX clarity and responsiveness.
- Documentation quality (README and API notes).
- Bonus items implemented well (if any).

Tip: Keep scope tight and working end-to-end. A few polished features beat many half-finished ones.