Practical Assignment: Full-Stack E-Commerce Website

Objective

Build an e-commerce platform where users can:

- Authenticate & manage accounts

- Browse, search, and filter products

- Manage shopping carts

- Place orders & track them

- Use Redis caching for performance

- Upload images to AWS S3

Backend (NestJS + TypeScript)

Tech Stack

- NestJS (TypeScript)

- PostgreSQL (TypeORM)

- Redis (Caching & rate-limiting)

- JWT authentication (Access & refresh tokens)

- AWS S3 (For product image storage)

- Swagger Documentation

- Docker (For easy deployment)

Database Schema (PostgreSQL)

Tables:

1. users – Stores user details (Admin & Customers)

2. products – Stores product details (name, price, stock, images, description)

3. orders – Stores orders (linked to users & products)

4. order_items – Stores order items (linked to orders)

5. cart_items – Stores items in the cart (linked to users & products)

API Endpoints

Authentication APIs:

POST /auth/signup – Register a new user (Customer role by default)

POST /auth/login – Login with JWT (returns access & refresh tokens)

POST /auth/refresh – Refresh JWT token

GET /auth/profile – Get logged-in user details

Product APIs:

POST /products – Create a product (Admin only)

GET /products – Get all products (Cache in Redis)

GET /products/:id – Get product details (Cache in Redis)

PUT /products/:id – Update product details (Admin only)

DELETE /products/:id – Delete a product (Admin only)

Cart APIs:

POST /cart – Add item to cart

GET /cart – Get user's cart items

PUT /cart/:id – Update cart item quantity

DELETE /cart/:id – Remove item from cart

Order APIs:

POST /orders – Create an order (Checkout)

GET /orders – Get all orders (Admin only)

GET /orders/user – Get orders of the logged-in user

GET /orders/:id – Get order details

PUT /orders/:id/cancel – Cancel an order (Only before shipping)

Frontend (React + Redux + TypeScript)

Tech Stack

- React.js (TypeScript)

- Redux Toolkit (For state management)

- React Router (For navigation)

- Material-UI / TailwindCSS (For UI components)

- React Query (For API calls)

- React Hook Form + Yup (For form validation)

Pages to Develop

Authentication:

- Login Page

- Signup Page

Product Pages:

- Home Page (Product listing with filters)

- Product Details Page (With add-to-cart button)

- Admin Dashboard (Create/update/delete products)

Cart & Checkout:

- Cart Page (List cart items, update/remove items)

- Checkout Page (Enter shipping details & confirm order)

Order Management:

- Order History Page (View past orders)

- Admin Order Management (View & update order status)

Bonus Features:

- Implement Wishlist Functionality

- Add Reviews & Ratings for products

Evaluation Criteria

1. Code Quality & Structure

2. Database Design

3. Error Handling & Security

4. Performance Optimization

5. Best Practices

Submission Guidelines

1. GitHub Repository – Should include:

- Backend (NestJS + TypeScript)

- Frontend (React + Redux + TypeScript)

- README with setup instructions

- Postman Collection for API testing

- Swagger Documentation