Capstone Project: Building an E-Commerce Application with ASP.NET Core Web API and Angular Using Clean Architecture

1. Overview

The eCommerce system should allow users to browse products, add them to their cart, place orders, and manage payments. Admins can manage products, categories, and user roles. The application follows a modular design to separate concerns and ensure high testability.

2. Technology Stack

- **Backend:** ASP.NET Core Web API

- **Frontend:** Angular

- **Database:** SQL Server

- **ORM:** Entity Framework Core

- **Authentication:** JWT Authentication

- **Exception Handling:** Global exception handling middleware

- **Architecture:** Clean Architecture

3. Clean Architecture Implementation

a) Domain Layer (Core Business Logic)

- Defines entities such as Product, Order, User, etc.

- Contains business rules and validation logic.

Models:

- Product (Id, Name, Description, Price, Stock, CategoryId)

- Category (Id, Name, Description)

- Order (Id, UserId, OrderDate, Status, TotalAmount)

- OrderItem (Id, OrderId, ProductId, Quantity)

- CartItem (Id, UserId, ProductId, Quantity)

- User (Id, Username, Email, Password, Role)

b) Application Layer (Use Cases and Services)

- Implements interfaces for business logic.

- Contains services such as ProductService, OrderService, UserService, and CartService.

Interfaces:

- IProductRepository

- IOrderRepository

- IUserRepository

- ICartRepository

c) Infrastructure Layer (Database and External Services)

- Implements repositories for data access using Entity Framework Core.

- Integrates third-party services like payment gateways.

d) Presentation Layer (API Controllers)

- Exposes endpoints for the Angular frontend.

- Uses MediatR for CQRS (Command Query Responsibility Segregation).

Controllers:

- ProductsController

- OrdersController

- UsersController

- CartController

4. Backend Implementation (ASP.NET Core Web API)

- **Authentication & Authorization:** Implement using JWT tokens.

- **Controllers & Endpoints:** RESTful API with endpoints for product management, order processing, and user authentication.

- **Dependency Injection:** Ensure loose coupling.

5. Exception Handling in Web API

- **Custom Exceptions:** Defines exceptions such as ProductNotFoundException, OrderProcessingException.

6. Frontend Implementation (Angular)

- **Angular Modules:** Organized into feature-based modules such as ProductModule, CartModule, OrderModule.

- **Services:** HTTP services for API communication.

- **Error Handling:** Implement interceptors to handle HTTP errors globally.

Angular Services:

- ProductService

- OrderService

- AuthService

- CartService

7. Database Design

- **Tables:** Users, Products, Categories, Orders, OrderItems, Payments, CartItems.

- **Relationships:**

  - One-to-many between Orders and OrderItems

  - Many-to-many between Users and Orders

  - One-to-many between Users and CartItems