Project: E-Commerce Platform

1. Introduction

This document outlines the Low-Level Design (LLD) for an **E-Commerce Platform** that allows users to browse products, manage shopping carts, make payments, and track orders. The platform includes an admin interface to manage products, orders, users, and analytics.

This design supports both **Java (Spring Boot)** and **.NET (ASP.NET Core)** frameworks for backend development.

2. Module Overview

2.1 Product Management Module

- Manage product listings, including creation, updates, and categorization.

2.2 Shopping Cart Module

- Users can add, remove, and manage items in their shopping cart.

2.3 Order Management Module

- Process customer orders, track order status, and manage payments.

2.4 User Authentication and Profile Management Module

- User registration, login, profile updates, and password management.

2.5 Admin Dashboard Module

- Admins can manage users, products, and view analytics on sales and user activity.

3. Architecture Overview

3.1 Architectural Style

- Frontend: Angular or React
- Backend: REST API-based architecture
- Database: Relational Database (MySQL/PostgreSQL/SQL Server)

3.2 Component Interaction

- The frontend communicates with the backend via REST APIs for product browsing, cart management, order placement, and user account management.
- The backend manages the CRUD operations for products, users, and orders.

4. Module-Wise Design

4.1 Product Management Module

Features:

- Create, update, delete, and categorize products.
- Display product details, including name, description, price, and image.

Data Flow:

- Admin adds/edit product details via the frontend.
- Backend validates the input and stores product data in the database.
- The product is then listed on the user-facing site for browsing.

Entities:

- **Product**:
  - ProductID
  - Name
  - Description
  - Price
  - Category
  - ImageURL

4.2 Shopping Cart Module

Features:

- Add/remove products to/from the shopping cart.
- View items and total price.

Data Flow:

- User adds/removes items from the cart via the frontend.
- Backend updates the cart data in the database.
- The updated cart information is shown to the user.

Entities:

- **CartItem**:
  - CartItemID
  - ProductID
  - Quantity
  - TotalPrice

4.3 Order Management Module

Features:

- Place orders with shipping details.
- Track order status (pending, shipped, delivered).
- Payment processing.

Data Flow:

- User submits order details via the frontend.
- Backend processes the order, calculates total price, and initiates payment.
- The order is confirmed, and payment status is updated in the database.

Entities:

- **Order**:
  - OrderID
  - UserID
  - TotalPrice
  - ShippingAddress
  - OrderStatus
  - PaymentStatus

4.4 User Authentication and Profile Management Module

Features:

- User registration and login functionality.
- Profile updates (name, address, payment methods).

Data Flow:

- User registers or logs in via the frontend.
- Backend authenticates the user and retrieves profile data.
- User can update their profile, and the backend stores the updated information.

Entities:

- **User**:
  - UserID
  - Name
  - Email
  - Password
  - ShippingAddress
  - PaymentDetails

4.5 Admin Dashboard Module

Features:

- Manage products, view order statuses, and customer details.
- Generate sales analytics and reports.

Data Flow:

- Admin interacts with the frontend to manage products and orders.
- Backend processes data, returns relevant information, and generates reports.

Entities:

- **Admin**:
  - AdminID
  - Name
  - Role
  - Permissions

5. Deployment Strategy

5.1 Local Deployment

- Frontend: Served using local servers (e.g., ng serve for Angular or equivalent for React).
- Backend: Deployed locally using Spring Boot or ASP.NET Core.
- Database: Local instance of the relational database for testing.

6. Database Design

6.1 Tables and Relationships

1. **Product**
   - Primary Key: ProductID
2. **User**
   - Primary Key: UserID
3. **CartItem**
   - Primary Key: CartItemID
   - Foreign Key: ProductID
4. **Order**
   - Primary Key: OrderID
   - Foreign Key: UserID
5. **Admin**
   - Primary Key: AdminID

7. User Interface Design

7.1 Wireframes:

- Product Listing Page
- Shopping Cart Page
- Order Confirmation Page
- User Profile Management Page
- Admin Dashboard

8. Non-Functional Requirements

8.1 Performance

- The platform should handle up to 1000 concurrent users.

8.2 Scalability

- The system is designed to scale vertically and horizontally as demand increases.

8.3 Security

- Implement secure authentication, authorization, and encryption protocols.

8.4 Usability

- The user interface must be responsive, intuitive, and accessible across devices.