Project Title: E-Commerce Platform

1. Overview

The **E-Commerce Platform** is a comprehensive web-based solution for managing online store operations. The platform facilitates a seamless shopping experience for customers and efficient order management for store administrators. This system follows the **MVC architecture**, ensuring compatibility with both **Java (Spring MVC)** and **.NET (ASP.NET Core MVC)** frameworks.

The core modules are:

1. **Product Management** – Handles product information and categorization.
2. **Shopping Cart** – Manages customer shopping carts.
3. **Order Processing** – Manages the order lifecycle from creation to delivery.
4. **Payment Gateway Integration** – Facilitates payment processing.
5. **User Management** – Manages customer authentication and authorization.

2. Assumptions

1. The application will be deployed locally during development using a relational database (e.g., MySQL or SQL Server).
2. Role-based authentication will secure sensitive information for customers and administrators.
3. ORM frameworks (Hibernate for Java or Entity Framework for .NET) will handle database interactions.
4. The application will support both Java and .NET environments for seamless development.
5. The payment gateway will use a mock implementation for local testing.

3. Module-Level Design

3.1 Product Management Module

Purpose: Manages product listings, categories, and inventory.

- **Controller**:
  - ProductController
    - addProduct(productData)
    - updateProduct(productId, productData)
    - getProductDetails(productId)
    - deleteProduct(productId)

        - getAllProducts()
- **Service**:
    - ProductService
        - Handles CRUD operations for products.
- **Model**:
    - **Entity**: Product
        - Attributes:
            - productId (PK)
            - name (VARCHAR)
            - description (TEXT)
            - price (DECIMAL)
            - categoryId (FK)
            - stockQuantity (INT)

3.2 Shopping Cart Module

Purpose: Handles adding/removing products to/from the cart and displays the current cart status.

- **Controller**:
    - CartController
        - addToCart(cartData)
        - removeFromCart(cartId)
        - getCartDetails(userId)
- **Service**:
    - CartService
        - Manages cart operations.
- **Model**:
    - **Entity**: Cart
        - Attributes:
            - cartId (PK)
            - userId (FK)
            - productId (FK)

      - quantity (INT)

3.3 Order Processing Module

Purpose: Handles order creation, order status, and delivery processing.

- **Controller**:
  - OrderController
    - createOrder(orderData)
    - getOrderDetails(orderId)
    - updateOrderStatus(orderId, status)
- **Service**:
  - OrderService
    - Handles order creation and status updates.
- **Model**:
  - **Entity**: Order
    - Attributes:
      - orderId (PK)
      - userId (FK)
      - totalAmount (DECIMAL)
      - orderDate (DATE)
      - status (ENUM: PENDING, SHIPPED, DELIVERED, CANCELLED)

3.4 Payment Gateway Integration Module

Purpose: Integrates with a payment gateway for processing payments.

- **Controller**:
  - PaymentController
    - processPayment(paymentData)
    - getPaymentStatus(paymentId)
- **Service**:
  - PaymentService
    - Processes payment requests and verifies status.
- **Model**:

- **Entity**: Payment
  - Attributes:
    - paymentId (PK)
    - orderId (FK)
    - amount (DECIMAL)
    - paymentStatus (ENUM: PENDING, COMPLETED, FAILED)
    - paymentDate (DATE)

3.5 User Management Module

Purpose: Manages user registration, login, and authentication.

- **Controller**:
  - UserController
    - registerUser(userData)
    - loginUser(username, password)
    - getUserProfile(userId)
    - updateUserProfile(userId, userData)
- **Service**:
  - UserService
    - Handles user authentication and profile management.
- **Model**:
  - **Entity**: User
    - Attributes:
      - userId (PK)
      - username (VARCHAR)
      - password (VARCHAR, Encrypted)
      - role (ENUM: CUSTOMER, ADMIN)
      - email (VARCHAR)

4. Database Schema

4.1 Table Definitions

1. **Product Table**

```
CREATE TABLE Product (
   productId INT PRIMARY KEY AUTO_INCREMENT,
   name VARCHAR(100),
   description TEXT,
   price DECIMAL(10, 2),
   categoryId INT,
   stockQuantity INT,
   FOREIGN KEY (categoryId) REFERENCES Category(categoryId)
);
```

2. **Category Table**

```
CREATE TABLE Category (
   categoryId INT PRIMARY KEY AUTO_INCREMENT,
   name VARCHAR(100)
);
```

3. **Cart Table**

```
CREATE TABLE Cart (
   cartId INT PRIMARY KEY AUTO_INCREMENT,
   userId INT,
   productId INT,
   quantity INT,
   FOREIGN KEY (userId) REFERENCES User(userId),
   FOREIGN KEY (productId) REFERENCES Product(productId)
);
```

4. **Order Table**

```
CREATE TABLE Order (
   orderId INT PRIMARY KEY AUTO_INCREMENT,
   userId INT,
   totalAmount DECIMAL(10, 2),
   orderDate DATE,
   status ENUM('PENDING', 'SHIPPED', 'DELIVERED', 'CANCELLED'),
   FOREIGN KEY (userId) REFERENCES User(userId)
);
```

5. **Payment Table**

```
CREATE TABLE Payment (
   paymentId INT PRIMARY KEY AUTO_INCREMENT,
   orderId INT,
   amount DECIMAL(10, 2),
   paymentStatus ENUM('PENDING', 'COMPLETED', 'FAILED'),
   paymentDate DATE,
   FOREIGN KEY (orderId) REFERENCES Order(orderId)
);
```

6. **User Table**

```
CREATE TABLE User (
   userId INT PRIMARY KEY AUTO_INCREMENT,
   username VARCHAR(50) UNIQUE,
   password VARCHAR(255),
   role ENUM('CUSTOMER', 'ADMIN'),
   email VARCHAR(100)
);
```

5. Local Deployment Details

1. **Environment Setup:**
   - Install JDK 17 or .NET SDK 7.0.
   - Install MySQL or SQL Server.
   - Use an application server (Tomcat for Java, Kestrel for .NET).
2. **Deployment Steps:**
   - Clone the repository.
   - Configure the database connection string in application.properties (Java) or appsettings.json (.NET).
   - Run the provided SQL scripts to initialize the database schema.
   - Build and start the application locally.

6. Conclusion

This document outlines the low-level design for the **E-Commerce Platform**, ensuring modularity, scalability, and compatibility with **Spring MVC** and **ASP.NET Core MVC** frameworks. The system handles key operations such as product management, shopping carts, order processing, payments, and user management in an intuitive, secure manner.