**Assignment: E-Commerce CLI Application using Spring Boot, Data JPA, and MySQL**

**Objective:** Develop a command-line e-commerce application where users can view products, add them to a cart, and place orders through a series of text-based prompts and responses.

**Requirements:**

1. **Setup & Configuration:**
   * Setup a new Spring Boot project using the Spring Initializr.
   * Include the following dependencies: Spring Data JPA and MySQL Driver.
   * Configure the **application.properties** or **application.yml** for the MySQL database connection.
2. **Entities:**
   * Same as previously provided.
3. **Repositories:**
   * Same as previously provided.
4. **Services:**
   * **ProductService**: CRUD operations for products.
   * **CartService**: Operations like add product to cart, remove product from cart, and view cart.
   * **OrderService**: Operations like place an order and view order history.
5. **CLI Interface:**
   * On starting the application, display a menu with options such as:
     1. List all products
     2. Add a product to the cart
     3. View cart
     4. Place an order
     5. View order history
     6. Exit
   * Depending on the user's choice, perform the corresponding action, and then return to the menu.
6. **Validation:**
   * Ensure products added to the cart or orders are available in stock.
   * Prevent adding products to the cart if they are out of stock.
7. **Exception Handling:**
   * Handle potential issues in a user-friendly way, such as product not found, out of stock, etc.
8. **Bonus (Optional):**
   * Implement user authentication at the start of the application and associate carts and orders with specific users.
   * Persist the state of the application such that carts and orders survive application restarts.
   * Implement unit tests for the services.

**Deliverables:**

* Source code of the CLI e-commerce application.
* README.md explaining how to set up, run, and interact with the application.

**Evaluation Criteria:**

* Proper implementation of Spring Boot, Data JPA, and MySQL integrations.
* Code quality, organization, and readability.
* Handling edge cases and validation.
* (Optional) Extra features implemented in the bonus section.

This revised assignment focuses on backend logic and data persistence, with interaction done via a text-based interface. It provides a good challenge for implementing Spring Data JPA functionalities without the overhead of web or REST APIs.

Top of Form

Mock and sample code as followsBottom of Form

* + 1. Entity classes

@Entity

public class Product {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private String description;

private Double price;

private Integer stock;

// getters, setters, constructors, etc.

}

@Entity

public class Cart {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@ManyToMany

private List<Product> products;

// getters, setters, constructors, etc.

}

@Entity

public class Order {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@ManyToMany

private List<Product> products;

private LocalDateTime orderDate;

// getters, setters, constructors, etc.

}

2. JpaRepository interfaces

public interface ProductRepository extends JpaRepository<Product, Long> {}

public interface CartRepository extends JpaRepository<Cart, Long> {}

public interface OrderRepository extends JpaRepository<Order, Long> {}

**3. Services**

@Service

public class ProductService {

@Autowired

private ProductRepository productRepository;

public List<Product> listAllProducts() {

return productRepository.findAll();

}

// Other CRUD methods ...

}

@Service

public class CartService {

@Autowired

private CartRepository cartRepository;

public Cart addToCart(Product product) {

// logic to add product to cart

}

// Other methods...

}

@Service

public class OrderService {

@Autowired

private OrderRepository orderRepository;

public Order placeOrder(Cart cart) {

// logic to place an order

}

// Other methods...

}

4. **CLI Interface** (A simple mock):

public class CliApp {

@Autowired

private ProductService productService;

@Autowired

private CartService cartService;

@Autowired

private OrderService orderService;

public void run() {

Scanner scanner = new Scanner(System.in);

while (true) {

System.out.println("Choose an option:");

System.out.println("1. List all products");

System.out.println("2. Add a product to the cart");

// ... Other options ...

int choice = scanner.nextInt();

switch (choice) {

case 1:

List<Product> products = productService.listAllProducts();

// Display products ...

break;

case 2:

// Add product logic ...

break;

// ... Other cases ...

}

}

}

}

1. **Product Table:**

| **Column Name** | **Data Type** | **Constraints** |
| --- | --- | --- |
| id | BIGINT | PRIMARY KEY, AUTO\_INCREMENT |
| name | VARCHAR(255) | NOT NULL |
| description | TEXT |  |
| price | DECIMAL(10,2) | NOT NULL |
| stock | INT | DEFAULT 0 |

1. **Cart Table:**

| **Column Name** | **Data Type** | **Constraints** |
| --- | --- | --- |
| id | BIGINT | PRIMARY KEY, AUTO\_INCREMENT |

1. **Order Table:**

| **Column Name** | **Data Type** | **Constraints** |
| --- | --- | --- |
| id | BIGINT | PRIMARY KEY, AUTO\_INCREMENT |
| orderDate | DATETIME | NOT NULL |

1. **Many-to-Many Relationship Tables:**

For both the Cart and Order, there are many-to-many relationships with the Product. Let's define junction tables:

a. **Cart\_Product Table:**

| **Column Name** | **Data Type** | **Constraints** |
| --- | --- | --- |
| cart\_id | BIGINT | FOREIGN KEY REFERENCES Cart(id) |
| product\_id | BIGINT | FOREIGN KEY REFERENCES Product(id) |
| PRIMARY KEY | (cart\_id, product\_id) |  |

b. **Order\_Product Table:**

| **Column Name** | **Data Type** | **Constraints** |
| --- | --- | --- |
| order\_id | BIGINT | FOREIGN KEY REFERENCES Order(id) |
| product\_id | BIGINT | FOREIGN KEY REFERENCES Product(id) |
| PRIMARY KEY | (order\_id, product\_id) |  |

**SQL Statements to create the tables:**

Top of Form

CREATE TABLE Product (

id BIGINT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255) NOT NULL,

description TEXT,

price DECIMAL(10, 2) NOT NULL,

stock INT DEFAULT 0

);

CREATE TABLE Cart (

id BIGINT AUTO\_INCREMENT PRIMARY KEY

);

CREATE TABLE Cart\_Product (

cart\_id BIGINT,

product\_id BIGINT,

PRIMARY KEY (cart\_id, product\_id),

FOREIGN KEY (cart\_id) REFERENCES Cart(id),

FOREIGN KEY (product\_id) REFERENCES Product(id)

);

CREATE TABLE `Order` (

id BIGINT AUTO\_INCREMENT PRIMARY KEY,

orderDate DATETIME NOT NULL

);

CREATE TABLE Order\_Product (

order\_id BIGINT,

product\_id BIGINT,

PRIMARY KEY (order\_id, product\_id),

FOREIGN KEY (order\_id) REFERENCES `Order`(id),

FOREIGN KEY (product\_id) REFERENCES Product(id)

);Bottom of Form