Backend Development Plan for E-commerce Project

# 1. Set Up the Development Environment (Day 1) (DONE)

Install Required Tools:

* Install Java, Spring Boot, and Maven.
* Set up PostgreSQL as the database.
* Use IntelliJ IDEA or your preferred IDE for development.

Create a New Spring Boot Project:

* Generate a new Spring Boot project using Spring Initializr.
* Include dependencies: Spring Web, Spring Data JPA, PostgreSQL, Spring Security, and Lombok.

Set Up Version Control:

* Initialize a Git repository.
* Push the initial code to a remote repository (e.g., GitHub).

# 2. Design the Database Schema (Day 1-2) (DONE)

Define Entities:

* User: id, username, email, password, role, etc.
* Product: id, name, description, price, stock, category, etc.
* Order: id, orderDate, totalAmount, status, user, etc.
* OrderItem: id, quantity, price, product, order, etc.
* Cart: id, user, items, totalAmount, etc.

Create ERD:

* Use a tool like Lucidchart to design the Entity-Relationship Diagram (ERD).
* Ensure all relationships (One-to-Many, Many-to-Many) are properly mapped.

# 3. Implement User Authentication and Authorization (Day 2)

Set Up Security Configuration:

* Use Spring Security to configure JWT-based authentication.(DONE)
* Create a UserDetailsService to load user-specific data. (DONE)

Implement Role-Based Access Control:

* Define roles (e.g., USER, ADMIN). (DONE)
* Restrict access to certain endpoints based on roles. (DONE)

# 4. Develop Core Business Logic (Day 3-4)

Create Repositories:

* Use Spring Data JPA to create repositories for each entity (UserRepository, ProductRepository, OrderRepository, etc.). (DONE)

Implement Services:

* UserService: Handle user registration, authentication, profile management. (DONE)
* ProductService: Manage product CRUD operations.
* OrderService: Handle order placement, status updates, and history.
* CartService: Implement add-to-cart, update cart, and remove from cart functionalities.

Create Controllers:

* UserController: Expose endpoints for user operations (register, login, profile).
* ProductController: Expose endpoints for product operations (list products, get product details).
* OrderController: Expose endpoints for order management (place order, view order history).
* CartController: Expose endpoints for cart management (add to cart, view cart, checkout).

# 5. Implement Payment Integration (Day 4-5)

Choose a Payment Gateway:

* Select a payment gateway (e.g., Stripe, PayPal) based on your requirements.

Integrate Payment API:

* Implement the payment process in your application.
* Ensure that payment data is securely handled and stored.

# 6. Test and Validate APIs (Day 5-6)

Write Unit Tests:

* Use JUnit and Mockito to write unit tests for service layers.

Perform Integration Testing:

* Test the interaction between different components of your application.
* Ensure that all endpoints work as expected.

Document APIs:

* Use Swagger or OpenAPI to document your RESTful APIs.
* Ensure that each endpoint is well-documented with request/response details.

# 7. Deploy the Backend (Day 7)

Set Up a Production Database:

* Configure PostgreSQL for production use.

Deploy the Application:

* Use a platform like Heroku, AWS, or DigitalOcean for deployment.
* Set up environment variables for sensitive information (e.g., database credentials, JWT secret).

Monitor and Maintain:

* Implement logging and monitoring using tools like Logback and Spring Boot Actuator.
* Prepare for post-deployment bug fixes and updates.