perplexity

eCommerce Platform Project Plan

1. Overview

A robust, modular eCommerce platform designed using Object-Oriented Programming (OOP) and SOLID principles. This document outlines all models with attributes, service interfaces, and feature specifications to guide development, testing, and future enhancements.

2. Domain Models

2.1 User

Represents a registered platform user.

- **id**: Long
- **name**: String
- **email**: String
- **password**: String
- **role**: UserRole (Enum: ADMIN, CUSTOMER)
- **createdAt**: LocalDateTime
- **updatedAt**: LocalDateTime

2.2 Product

Describes an item available for purchase.

- **id**: Long
- **name**: String
- **description**: String
- **price**: BigDecimal
- **stock**: Integer
- **category**: Category (Enum: ELECTRONICS, FASHION, GROCERY, BOOKS, TOYS)
- **rating**: Double
- **createdAt**: LocalDateTime
- **updatedAt**: LocalDateTime

2.3 Cart

Holds items a Customer intends to purchase.

- **id**: Long
- **user**: User
- **cartItems**: List<CartItem>
- **totalAmount**: BigDecimal (calculated)

2.4 CartItem

Line-item within a Cart.

- **id**: Long
- **product**: Product
- **quantity**: Integer
- **price**: BigDecimal (quantity × product.price)

2.5 Order

Finalized purchase record.

- **id**: Long
- **user**: User
- **orderItems**: List<OrderItem>
- **orderDate**: LocalDateTime
- **status**: OrderStatus (Enum: PENDING, PLACED, SHIPPED, DELIVERED, CANCELLED)
- **totalAmount**: BigDecimal

2.6 OrderItem

Line-item within an Order.

- **id**: Long
- **product**: Product
- **quantity**: Integer
- **price**: BigDecimal

2.7 Payment

Tracks payment transactions.

- **id**: Long
- **order**: Order

- **paymentMethod**: PaymentType (Enum: CARD, UPI, COD)
- **amount**: BigDecimal
- **status**: PaymentStatus (Enum: PENDING, SUCCESS, FAILED)
- **paymentDate**: LocalDateTime

2.8 Review

Customer feedback on Products.

- **id**: Long
- **user**: User
- **product**: Product
- **rating**: Integer (1–5)
- **comment**: String
- **createdAt**: LocalDateTime

3. Service Interfaces

Each interface adheres to single-responsibility and dependency-inversion principles, facilitating testability and future extension.

3.1 UserService

- **registerUser(UserDto)**
- **loginUser(LoginDto)**
- **getUserProfile(Long userId)**

3.2 ProductService

- **getAllProducts()**
- **getProductById(Long productId)**
- **filterProductsByCategory(Category category)**
- **searchProductsByName(String query)**
- **addReview(Long productId, ReviewDto)**

3.3 CartService

- **addToCart(Long userId, CartItemDto)**
- **removeFromCart(Long cartItemId)**
- **updateQuantity(Long cartItemId, int quantity)**
- **getCartItems(Long userId)**
- **clearCart(Long userId)**

3.4 OrderService

- **placeOrder(Long userId)**
- **getOrderHistory(Long userId)**
- **getOrderById(Long orderId)**
- **cancelOrder(Long orderId)**

3.5 PaymentService

- **makePayment(PaymentRequestDto)**
- **validatePayment(Long paymentId)**
- **getPaymentDetails(Long paymentId)**

4. Feature Specifications

4.1 Authentication & Authorization

- **User Registration**: Validate input, hash passwords, assign default CUSTOMER role.
- **User Login**: Issue JWT or session token.
- **Role-Based Access**: ADMIN endpoints secured via Spring Security annotations.

4.2 Product Module

- **Catalog Browsing**: Paginated product listing.
- **Search**: Full-text search on name and description.
- **Filtering**: By category and price range.
- **Product Details**: Full description, stock availability, average rating.
- **Reviews & Ratings**: Submit and view reviews; recalculate average rating.

4.3 Shopping Cart

- **Add/Remove Items**: CRUD operations on CartItem.
- **Quantity Adjustment**: Recalculate line price and cart total dynamically.
- **Cart Summary**: Display totalAmount and itemized list.

4.4 Order Processing

- **Checkout Flow**: Validate cart, reduce stock, create Order and OrderItems.
- **Order Tracking**: View status updates—PENDING → PLACED → SHIPPED → DELIVERED.
- **Order History**: List past orders with status and totals.
- **Cancellation**: Allow cancellation when status is PENDING.

4.5 Payment Handling

- **Multiple Payment Types**: Implement PaymentType strategy (CARD, UPI, COD).
- **Transaction Simulation**: Interface with mock payment gateway.
- **Validation & Retry**: Handle failed transactions with retry logic.
- **Audit Trail**: Store PaymentStatus history for each Order.

5. Technical Considerations

- **Framework**: Spring Boot, Spring Security, Spring Data JPA.
- **Database**: PostgreSQL with proper indexing on name, category, and user_id.
- **Messaging**: Asynchronous email notifications via RabbitMQ or AWS SQS.
- **Testing**:
  - Unit tests with JUnit and Mockito.
  - Integration tests using Testcontainers for PostgreSQL.
- **CI/CD**: GitHub Actions for build, test, and Docker image publishing.
- **Documentation**: OpenAPI (Swagger) for REST endpoints.

6. Next Steps

1. Finalize DTO definitions and mapping strategies.
2. Create database migration scripts (Flyway/Liquibase).
3. Implement core services, starting with Authentication and Product modules.
4. Set up continuous integration and containerized development environment.

This professional project plan provides a clear, extensible blueprint for developing a maintainable, SOLID-compliant eCommerce platform.