E-Commerce System Case Study

1. Introduction

This document outlines the design of an object-oriented e-commerce system. The system aims to provide a seamless online shopping experience for customers, enabling them to browse products, manage their shopping carts, place orders, and track order statuses. It also provides administrative functionalities for managing products, users, inventory, and payments.

2. Problem Statement

Design an object-oriented e-commerce system that enables customers to browse products, add them to a shopping cart, place orders, and track order status. The system should accommodate different product categories, manage inventory, process payments, and generate invoices.

3. System Requirements

3.1 User Management:

- Store user information (name, email, billing/shipping addresses).
- Support different user roles (Customer, Admin, Guest).
- Track user purchase history.

3.2 Product Catalog:

- Organize products in categories and subcategories.
- Support different product types (physical, digital, subscription).
- Track product inventory and availability.
- Handle product attributes (size, color, weight, etc.).

3.3 Shopping Cart & Order Processing:

- Allow adding/removing products from cart.
- Apply discounts and coupons.
- Process payments through various methods.
- Generate order confirmations and invoices.

3.4 Inventory Management:

- Track stock levels for products.
- Support restocking and inventory adjustments.

- Notify when products fall below a threshold.

3.5 Payment Processing:

- Support multiple payment methods.
- Handle payment status (pending, completed, failed).
- Process refunds when necessary.

4. Class Identification and Relationships

4.1 User-related Classes:

- **User:**
    - Attributes: userId, name, email, password, billingAddress, shippingAddress, role.
    - Methods: updateProfile(), changePassword().
- **Customer (Inherits from User):**
    - Attributes: purchaseHistory.
    - Methods: placeOrder(), viewOrderHistory().
- **Admin (Inherits from User):**
    - Methods: manageProducts(), manageUsers(), processRefund(), updateInventory().
- **Address:**
    - Attributes: street, city, state, zip, country.

4.2 Product-related Classes:

- **Product:**
    - Attributes: productId, name, description, price, category, stockQuantity.
    - Methods: updateStock(), displayDetails().
- **PhysicalProduct (Inherits from Product):**
    - Attributes: weight, dimensions.
- **DigitalProduct (Inherits from Product):**
    - Attributes: downloadLink, fileSize.
- **SubscriptionProduct (Inherits from Product):**
    - Attributes: subscriptionDuration, renewalDate.

- **ProductCategory:**
    - Attributes: categoryId, categoryName, parentCategory.
    - Methods: addProduct(), removeProduct().
- **ProductAttribute:**
    - Attributes: attributeId, attributeName, attributeValue.

4.3 Order-related Classes:

- **Order:**
    - Attributes: orderId, customerId, orderDate, orderStatus, totalAmount, shippingAddress, billingAddress, paymentId, discountCode.
    - Methods: calculateTotal(), updateStatus(), generateInvoice().
- **OrderItem:**
    - Attributes: orderItemId, productId, quantity, price.
- **ShoppingCart:**
    - Attributes: cartId, customerId, items (List of OrderItem).
    - Methods: addItem(), removeItem(), calculateTotal().
- **Payment:**
    - Attributes: paymentId, paymentMethod, paymentDate, amount, status.
    - Methods: processPayment(), refundPayment().
- **Invoice:**
    - Attributes: invoiceId, orderId, invoiceDate, items, totalAmount, paymentDetails.
    - Methods: displayInvoice().

5. Key OOP Concepts

- **Inheritance:**
    - Customer and Admin inherit from User.
    - PhysicalProduct, DigitalProduct, and SubscriptionProduct inherit from Product.
- **Encapsulation:**

    - Classes encapsulate data and behavior, with private attributes accessed through getter and setter methods.

- **Polymorphism:**
    - Different product types can be handled uniformly through the Product base class.
- **Abstraction:**
    - The Product class provides an abstract interface for all product types.
- **Composition:**
    - Order contains OrderItem objects.
    - User contains Address objects.
- **Aggregation:**
    - ProductCategory contains Product objects.
    - ShoppingCart contains OrderItem objects.

6. Implementation Notes

- Database Selection: A relational database (e.g., PostgreSQL, MySQL) is recommended for structured data storage.
- Payment Gateway Integration: Integration with a reliable payment gateway (e.g., Stripe, PayPal) is crucial.
- Security: Implement robust security measures, including HTTPS, data encryption, and input validation.
- Scalability: Design the system to handle increasing traffic and data volume.
- Error Handeling: Implement robust error handeling.
- Logging: Implement logging of important events.
- Testing: Unit, integration, and end-to-end tests are vital for ensuring system reliability.

7. Conclusion

This case study provides a comprehensive design for an object-oriented e-commerce system. By adhering to the principles of OOP and addressing the specified requirements, the system can offer a user-friendly and efficient online shopping experience. This design can be used as the foundation for the software development of the E-commerce system.