1. Single Responsibility Principle

Related modules	Description	Improvement Direction
Class Cart	Lớp này hiện tại quản lý cả instance và các item. Các phương thức liên quan đến quản lý giỏ hàng và instance không có sự liên kết cần thiết.	Chia thành hai lớp riêng biệt: CartManager (quản lý singleton) và Cart (quản lý giỏ hàng).
Class BaseController	Lớp này không hoàn toàn đáp ứng vai trò là lớp cha cơ bản, vì nó chứa logic đặc thù của giỏ hàng, điều này gây không phù hợp với các controller không liên quan đến giỏ hàng.	Di chuyển các phương thức liên quan đến giỏ hàng (ví dụ: checkMediaInCart, getListCartMedia) vào một lớp tiện ích như CartService. Giữ BaseController thuần túy để các controller khác có thể kế thừa nếu cần chức năng chung như ghi log hoặc xử lý yêu cầu cơ bản.
Class PlaceOrderController	Lớp này thực hiện quá nhiều chức năng, vi phạm nguyên lý SRP khi đảm nhận việc kiểm tra tính khả dụng sản phẩm, tạo đơn hàng, hóa đơn, xử lý thông tin giao hàng và tính toán phí vận chuyển.	Tạo các lớp hoặc dịch vụ riêng biệt như OrderService (quản lý logic đặt hàng), InvoiceService (quản lý hóa đơn), ShippingService (tính phí vận chuyển), và DeliveryInfoValidator (xác thực thông tin giao hàng). Controller chỉ điều phối luồng xử lý và gọi các dịch vụ.
Class PaymentController	Lớp này xử lý nhiều trách nhiệm như điều phối thanh toán, lưu kết quả giao dịch vào cơ sở dữ liệu, và làm trống giỏ hàng. Đồng thời, việc tạo đối tượng VnPaySubsystemController trực tiếp cũng không cần thiết.	Tạo các dịch vụ cụ thể như PaymentService, CartService, và TransactionService để xử lý thanh toán, giỏ hàng và lưu giao dịch. Tiêm đối tượng VnPaySubsystemController hoặc IPayment qua constructor để giảm sự phụ thuộc.

2. Open-Closed Principle

Related modules	Description	Improvement Direction
BaseController	Lớp này hiện phụ thuộc trực tiếp vào lớp Cart. Nếu muốn thay đổi hệ thống giỏ hàng, sẽ	Tách biệt Cart ra khỏi BaseController thông

	cần sửa đổi logic trong lớp BaseController.	qua việc sử dụng interface.
HomeController	getAllMedia gọi trực tiếp các phương thức của Media. Điều này sẽ cần sửa đổi khi có yêu cầu lọc hoặc xử lý đặc thù cho từng loại media.	Tạo một service hoặc DAO để tách biệt việc truy vấn dữ liệu khỏi lớp HomeController.
ViewCartController	Lớp này phụ thuộc vào lớp Cart qua các phương thức tĩnh.	Tách biệt Cart thông qua interface.
DBConnection	Lớp này chỉ hỗ trợ SQLite với cấu hình cố định. Nếu muốn hỗ trợ các cơ sở dữ liệu khác như MySQL hoặc PostgreSQL, phải sửa lại lớp này.	Trừu tượng hóa việc kết nối cơ sở dữ liệu thông qua một interface.
Media, Book, CD, DVD	Các phương thức như getMediaById chứa logic SQL trực tiếp, gây khó khăn trong việc thay đổi hoặc mở rộng cách truy vấn.	Tách logic truy vấn ra thành một lớp DAO riêng.
Order	Phương thức getAmount có logic tính toán trực tiếp trong lớp. Nếu cần thay đổi cách tính (ví dụ: thêm chiết khấu, phí dịch vụ), phải sửa đổi mã trong lớp này. Dữ liệu giao hàng (deliveryInfo) sử dụng HashMap, khiến việc mở rộng thêm các trường thông tin cụ thể (vd: số điện thoại, mã bưu chính) khó khăn.	Tách logic tính toán tổng số tiền (getAmount) sang một lớp service riêng để dễ mở rộng Thay thế HashMap trong deliveryInfo bằng một lớp cụ thể
PaymentTransaction	Logic truy vấn SQL nằm trong lớp, gây khó khăn khi muốn thay đổi cách lưu trữ dữ liệu.	Tách logic truy vấn dữ liệu thành một DAO riêng.
Request và VnPaySubsystemController	Có sự trùng lặp trong logic giữa hai lớp này, điều này khiến việc thay đổi trở nên phức tạp.	Sử dụng một builder để xử lý các yêu cầu và phản hồi độc lập.
API	Các phương thức get và post hiện chứa logic HTTP giống nhau, không hoàn toàn tuân thủ OCP.	Tạo một phương thức chung để thiết lập và gửi yêu cầu HTTP.

Configi	Lớp Configs không tách biệt rõ các loại cấu hình.	Chia lớp Configs thành các nhóm cấu hình riêng biệt.
Utils	Lớp này chứa các phương thức tiện ích hỗn hợp, gây khó khăn khi mở rộng	: Tách các tiện ích thành các lớp nhỏ và chuyên biệt hơn.
SplashForm	: Đường dẫn đến hình ảnh logo được mã hóa cứng trong phương thức initialize, gây khó khăn khi muốn thay đổi hoặc thêm hình ảnh.	Đưa đường dẫn hình ảnh thành một tham số cấu hình

3. Liskov Substitution Principle

Related modules	Description	Improvement Direction
PaymentTransaction	Nếu lớp PaymentTransaction được mở rộng trong tương lai (ví dụ thêm loại giao dịch như RefundTransaction), cần đảm bảo các lớp con duy trì hành vi của các phương thức như isSuccess.	Sử dụng abstraction để định nghĩa các hành vi chung, bảo đảm tính kế thừa hợp lý.

4. Interface Segregation Principle

Related modules	Description	Improvement Direction
Interface IPayment	Một hoạt động thanh toán bao gồm ba quy trình chính: thanh toán, hoàn tiền và xử lý kết quả. Những quy trình này khác nhau giữa các cổng thanh toán, vì vậy giao diện cần được tách biệt.	Tách giao diện thành ba interface riêng biệt như sau: 1. **IPaymentProcessor**: Xử lý các hoạt động liên quan đến thanh toán. 2. **IRefundProcessor**: Xử lý các hoạt động hoàn tiền. 3. **ITransactionResult**: Xử lý kết quả giao dịch.
Thêm interface cho quản lý media trong cart	Có 3 logic cần xử lý trong phần giỏ hàng liên quan đến media: quản lý media trong giỏ hàng, tính tổng phí, tổng số lượng media, và xác thực dữ liệu.	Chia thành 3 Interface với các method 1. ICartMediaManager (add, remove, getlist, empty), 2. ICartCalculator (getTotalMedia, calSubtotal)

		3. ICartValidator (checkAvailabilityOfProduct , checkMediaInCart)
Thêm interface cho quản lý order	Một đơn hàng có 3 quy trình: xử lý thông tin giao hàng, tạo đơn hàng và tính toán phí vận chuyể	Chia thành 3 Interface với các menthod: 1. IDeliveryInfoProcessor (validateDeliveryInfo, processDeliveryInfo), 2. IOrderCreator (createOrder, createInvoice), 3. IShippingCalculator (calculateShippingFee)
Triển khai Factory design pattern cho việc lựa chọn cổng thanh toán	Trong tương lai, chúng ta có thể sẽ phải tích hợp nhiều phương thức thanh toán khác nhau.	1. IPaymentMethodFactory (Nơi chọn phương thức thanh toán), 2. IPaymentMethod (Khi tích hợp phương thức thanh toán mới, chỉ cần triển khai interface này và thực hiện logic thanh toán bằng cách ghi đè phương thức)

5. Dependency Inversion Principle

Related modules	Description	Improvement Direction
PaymentController	Phương thức payOrder gọi trực tiếp phương thức của lớp VnPaySubsystemController thay vì gọi từ interface IPayment.	Gọi phương thức từ interface IPayment và tiêm cổng thanh toán qua constructor.
Media	Constructor Media gọi trực tiếp phương thức getConnection từ lớp DBConnection. Điều này gây khó khăn khi muốn thay đổi cơ sở dữ liệu.	Tạo interface IDatabase và cho lớp DBConnection implement nó, sử dụng interface này trong constructor của Media.
Book	Phương thức getMediaById cũng gọi phương thức getConnection từ DBConnection, gây khó khăn khi thay đổi cơ sở dữ liệu.	Tạo interface IDatabase và cho DBConnection implement nó, sử dụng trong phương thức getMediaById.