**Design Principles ver2.0**

Written by: GROUP 3 – 1 January 2025

Contents

[1. SOLID 2](#_Toc186665656)

[1.1. Single Responsibility Principle 2](#_Toc186665657)

[1.2. Open/Closed Principle 2](#_Toc186665658)

[1.3. Liskov Substitution Principle 3](#_Toc186665659)

[1.4. Interface Segregation Principle 3](#_Toc186665660)

[1.5. Dependency Inversion Principle 4](#_Toc186665661)

# 1. SOLID

## 1.1. Single Responsibility Principle

|  |  |  |  |
| --- | --- | --- | --- |
| # | Related modules | Description | Improvement |
| 1.1.1 | PaymentController | Chịu trách nhiệm cho 2 nhiệm vụ: thanh toán và gửi email | Tách phần gửi email sang một class khác là EmailService |
| 1.1.2. | VnPayConfig | Lớp này chịu trách nhiệm quá nhiều về các chức năng như mã hóa, tạo URL và lấy địa chỉ IP | Tách ra các lớp riêng biệt cho mỗi nhiệm vụ, ví dụ HashUtil, UrlBuilder, IpAddressRetriever |
| 1.1.3. | PlaceOrderController | Chịu trách nhiệm cho nhiều nhiệm vụ: điều khiển luồng dữ liệu, validate dữ liệu, tính toán phí giao hàng | Tách ra thành các class con |
| 1.1.4. | Media | Lớp có nhiều trách nhiệm khác nhau, như quản lý thông tin về media, tương tác với cơ sở dữ liệu, và cung cấp các phương thức tìm kiếm, cập nhật thông tin | Tách các chức năng xử lý cơ sở dữ liệu ra ngoài lớp Media Tạo thêm một lớp như MediaDAO (Data Access Object) để thực hiện các thao tác với cơ sở dữ liệu |
| 1.1.5. | VnPaySubsystemController | Lớp đang nhận quá nhiều trách nhiệm: xử lý thanh toán, tạo giao diện hiển thị, xử lý hoàn tiền | Cần chia thành các lớp nhỏ hơn. |

## 1.2. Open/Closed Principle

|  |  |  |  |
| --- | --- | --- | --- |
| # | Related modules | Description | Improvement |
| 1.2.1. | PaymentController | Lớp đang dùng trực tiếp phương thức thanh toán bằng VNPay, khi muốn thêm phương thức thanh toán sẽ phải sửa lớp PaymentController | Tạo Giao Diện cho các Phương Thức Thanh Toán, đảm bảo rằng tất cả các dịch vụ thanh toán (như VNPay, PayPal, v.v.) đều triển khai giao diện này.  Lớp PaymentController sẽ ủy quyền quá trình thanh toán cho một thể hiện của IPayment, và nó có thể là bất kỳ lớp nào triển khai IPayment |
| 1.2.2. | Media | Lớp Media không hoàn toàn tuân thủ OCP vì bất kỳ sự thay đổi nào trong cách xử lý cơ sở dữ liệu có thể yêu cầu thay đổi trong lớp Media. | Áp dụng mô hình Repository hoặc DAO để tách biệt lớp Media khỏi các chi tiết của việc truy cập cơ sở dữ liệu |
| 1.2.3. | Request, Response | Lớp đang được thiết kế cứng cho phương thức thanh toán bằng VNPay, cần thay đổi để có thể dễ dàng tích hợp các kiểu thanh toán khác. | Sử dụng một interface hoặc abstract class để định nghĩa các hành vi chung (PaymentRequest, PaymentResponse).  Tách logic đặc thù của từng cổng thanh toán vào các lớp riêng biệt.  Đảm bảo lớp Request chỉ chịu trách nhiệm quản lý dữ liệu chung (nếu cần), không liên quan đến logic đặc thù của từng phương thức thanh toán. |

## 1.3. Liskov Substitution Principle

|  |  |  |  |
| --- | --- | --- | --- |
| # | Related modules | Description | Improvement |
|  |  |  |  |

## 1.4. Interface Segregation Principle

|  |  |  |  |
| --- | --- | --- | --- |
| # | Related modules | Description | Improvement |
|  |  |  |  |

## 1.5. Dependency Inversion Principle

|  |  |  |  |
| --- | --- | --- | --- |
| # | Related modules | Description | Improvement |
| 1.5.1. | Cart | Cart phụ thuộc trực tiếp vào CartMedia và Media. | Tạo abstraction để dễ dàng thay đổi mà không phụ thuộc vào các lớp cụ thể |
| 1.5.2. | Media | Lớp Media trực tiếp phụ thuộc vào Statement và DBConnection, điều này tạo ra sự phụ thuộc chặt chẽ vào các chi tiết triển khai của cơ sở dữ liệu. | Sử dụng các lớp DAO làm cho Media không còn phụ thuộc vào chi tiết cài đặt của việc truy vấn cơ sở dữ liệu. |