



Thiết kế và xây dựng phần mềm

Sinh viên: Vũ Việt Anh - 20215261

Lê Hoàng Long - 20215279

Trần Thành Nam - 20215285

Trần Nhật Minh - 20215284

Giáo viên hướng dẫn: TS. Nguyễn Thị Thu Trang

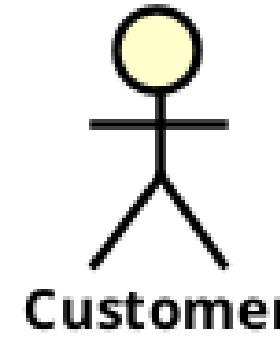
ONE LOVE. ONE FUTURE.

NỘI DUNG

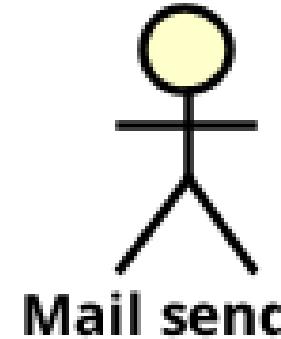
1. Tổng quan hệ thống
2. Thiết kế kiến trúc
3. Đánh giá source code
4. Đề xuất giải pháp
5. Mô hình hóa dữ liệu
6. Thiết kế lớp

1. TỔNG QUAN HỆ THỐNG

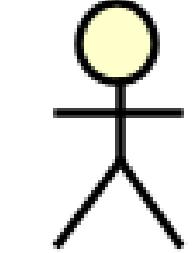
1.1 Các tác nhân



Customer



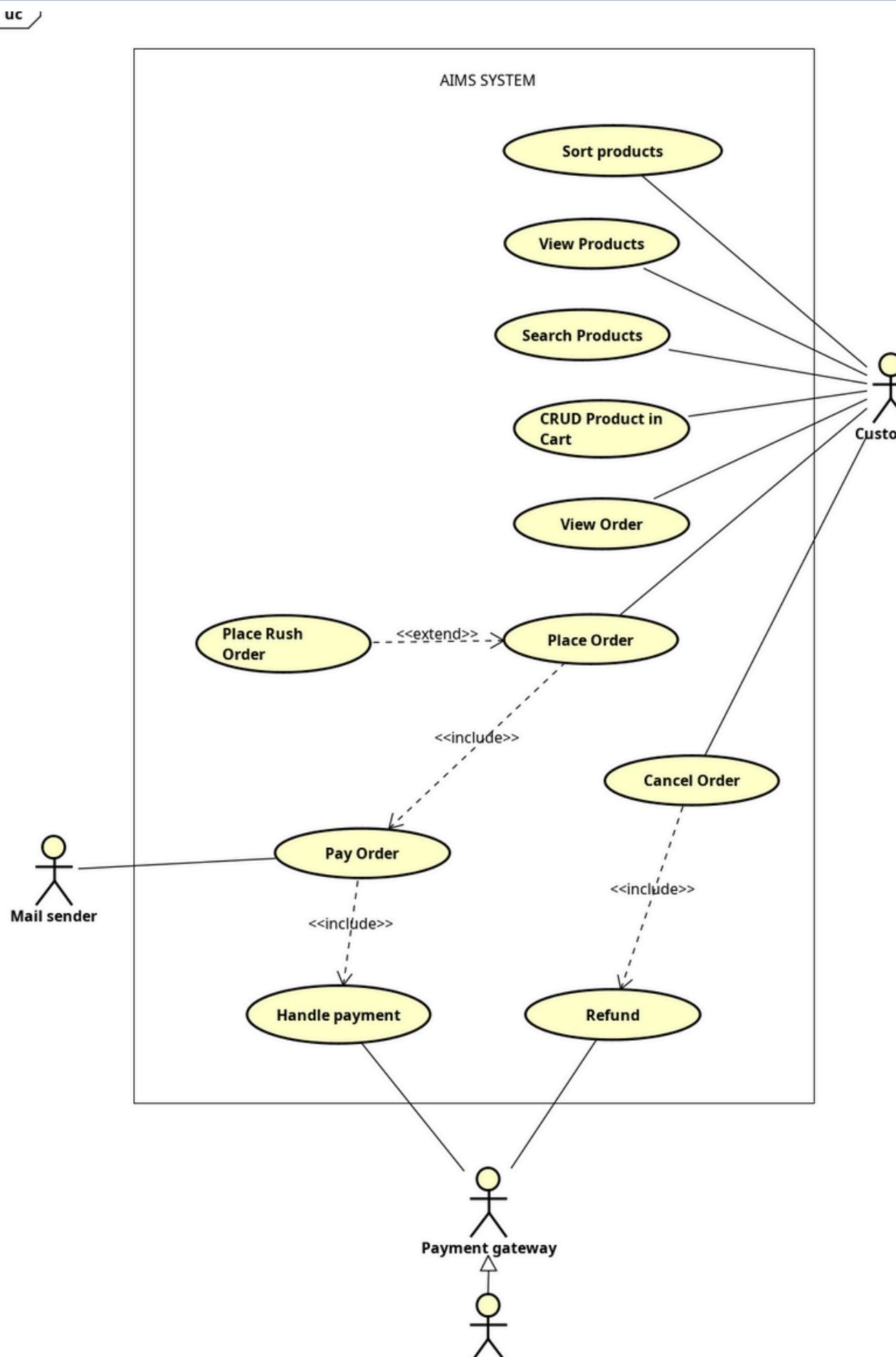
Mail sender



Payment gateway

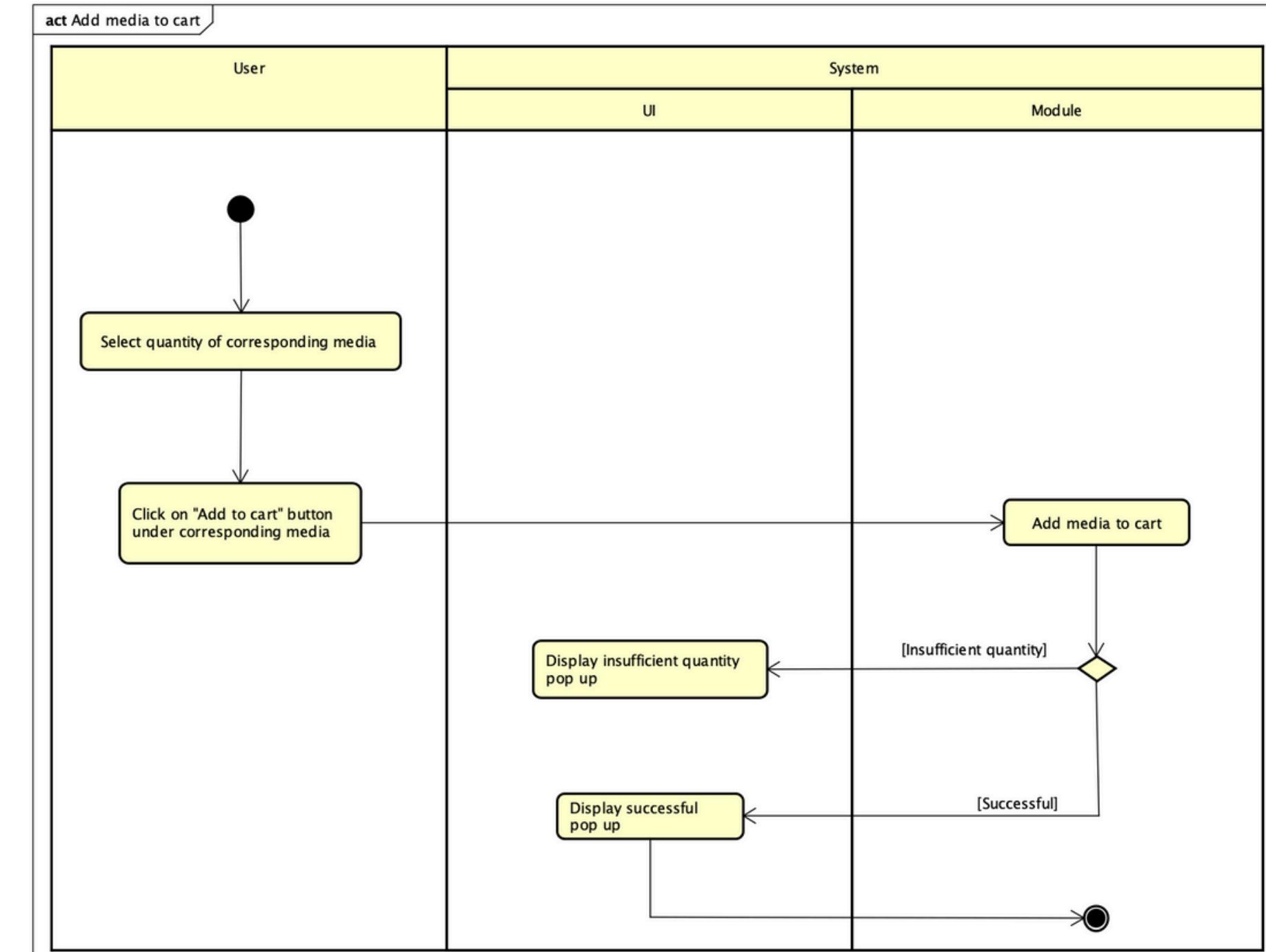
1. TỔNG QUAN HỆ THỐNG

1.2 Biểu đồ usecase



1. TỔNG QUAN HỆ THỐNG

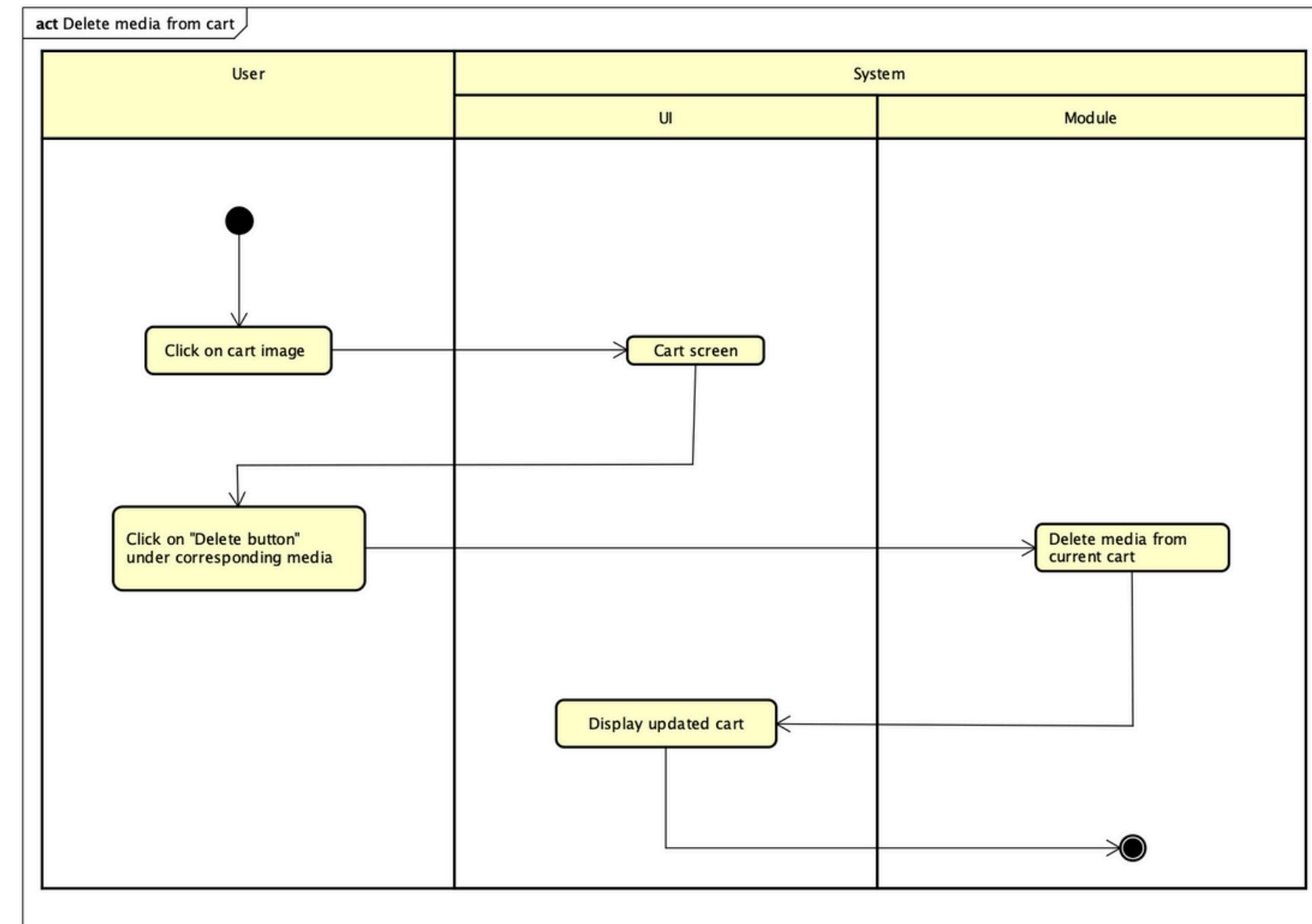
1.3 Activity Diagram



Add media to cart

1. TỔNG QUAN HỆ THỐNG

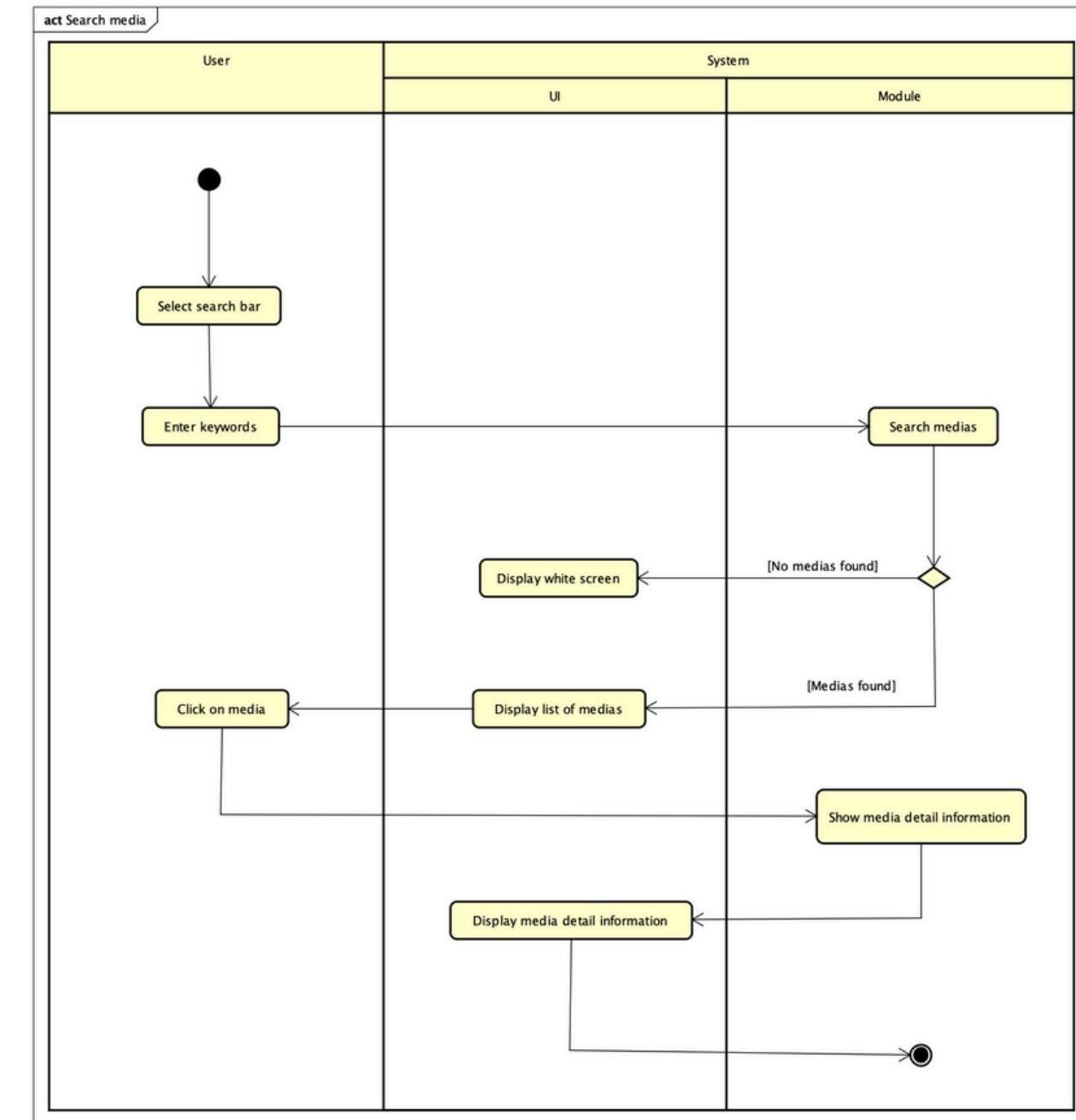
1.3 Activity Diagram



Delete media from cart

1. TỔNG QUAN HỆ THỐNG

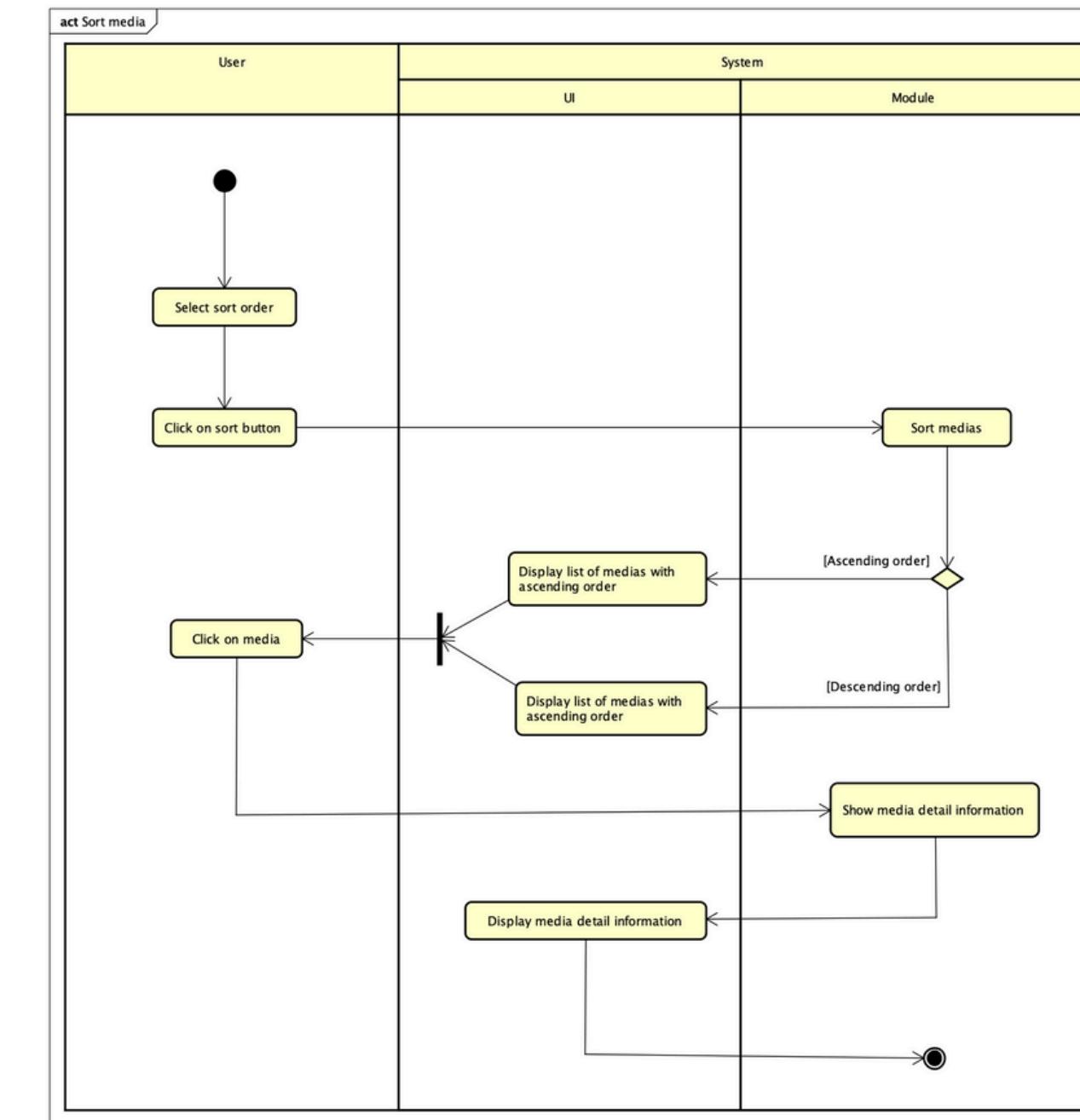
1.3 Activity Diagram



Search Media

1. TỔNG QUAN HỆ THỐNG

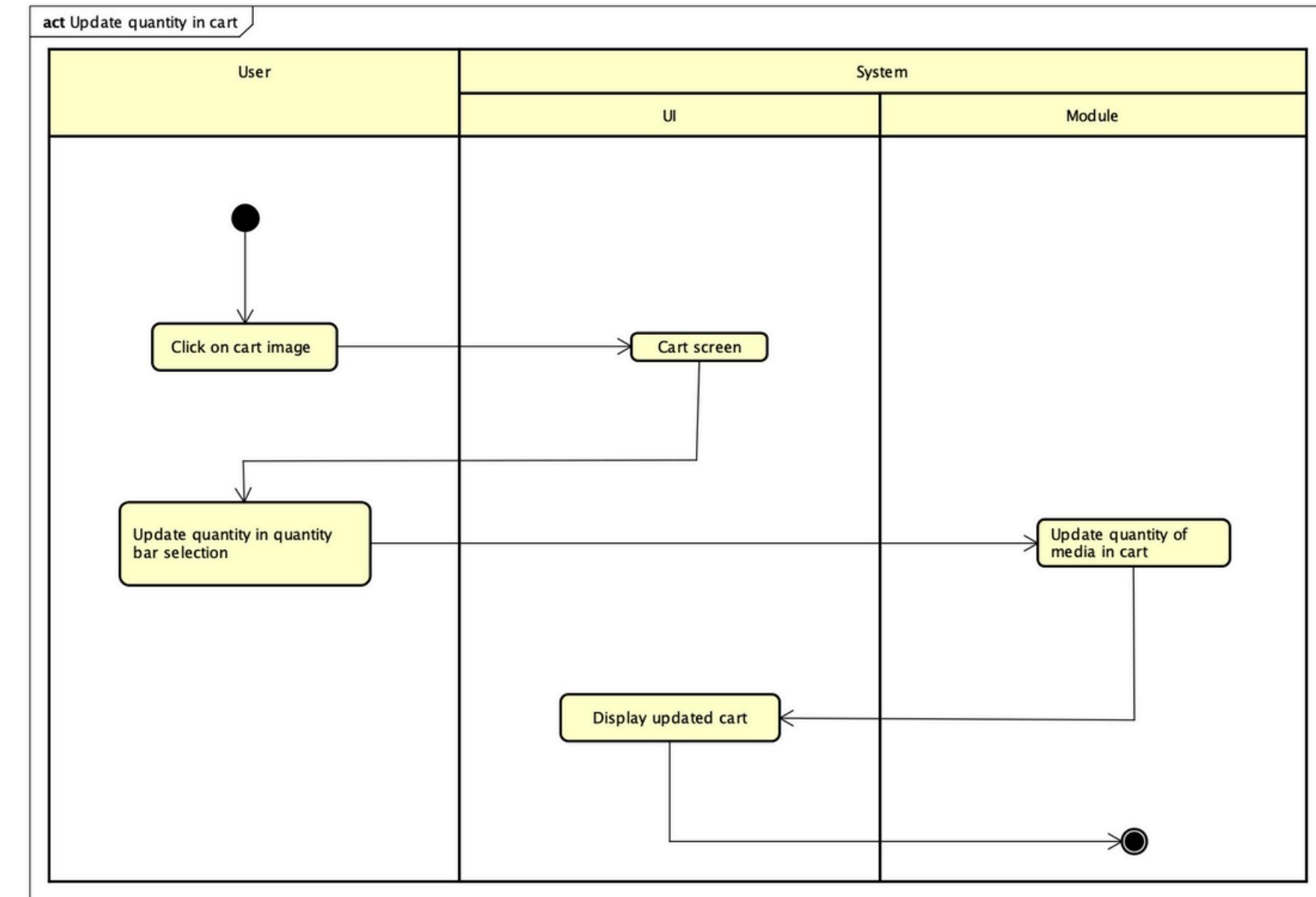
1.3 Activity Diagram



Sort Media

1. TỔNG QUAN HỆ THỐNG

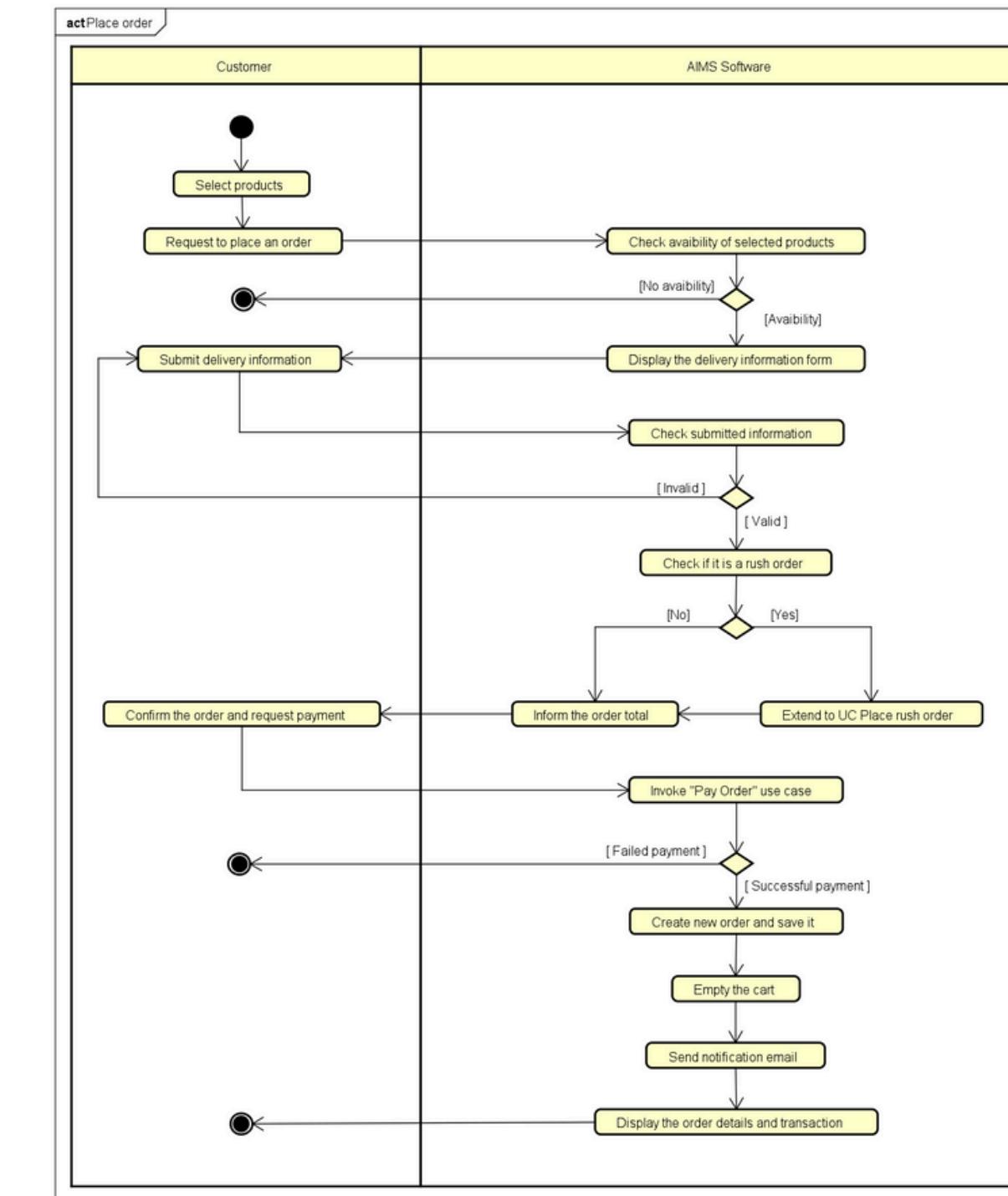
1.3 Activity Diagram



Update Quantity in Cart

1. TỔNG QUAN HỆ THỐNG

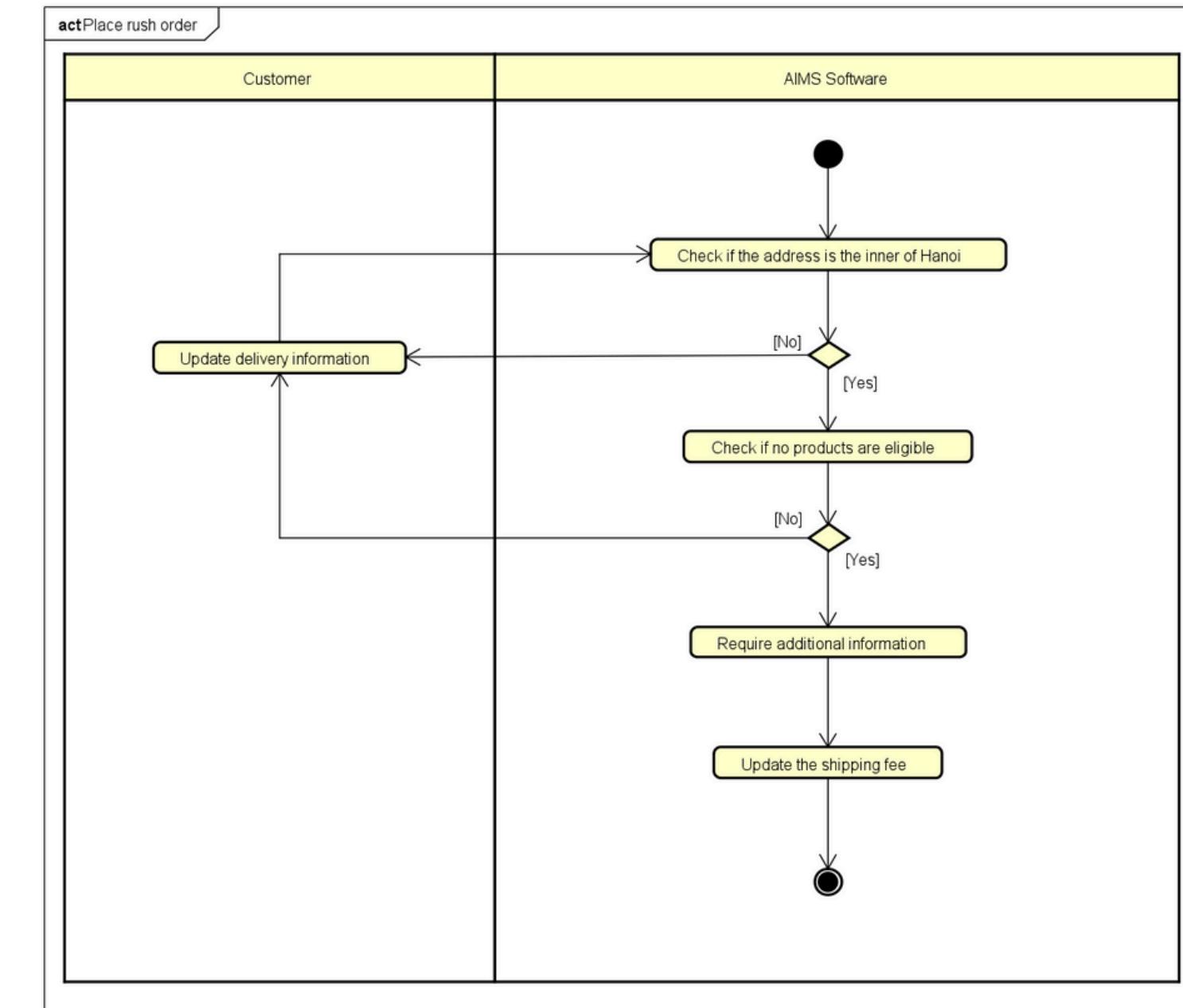
1.3 Activity Diagram



Place Order

1. TỔNG QUAN HỆ THỐNG

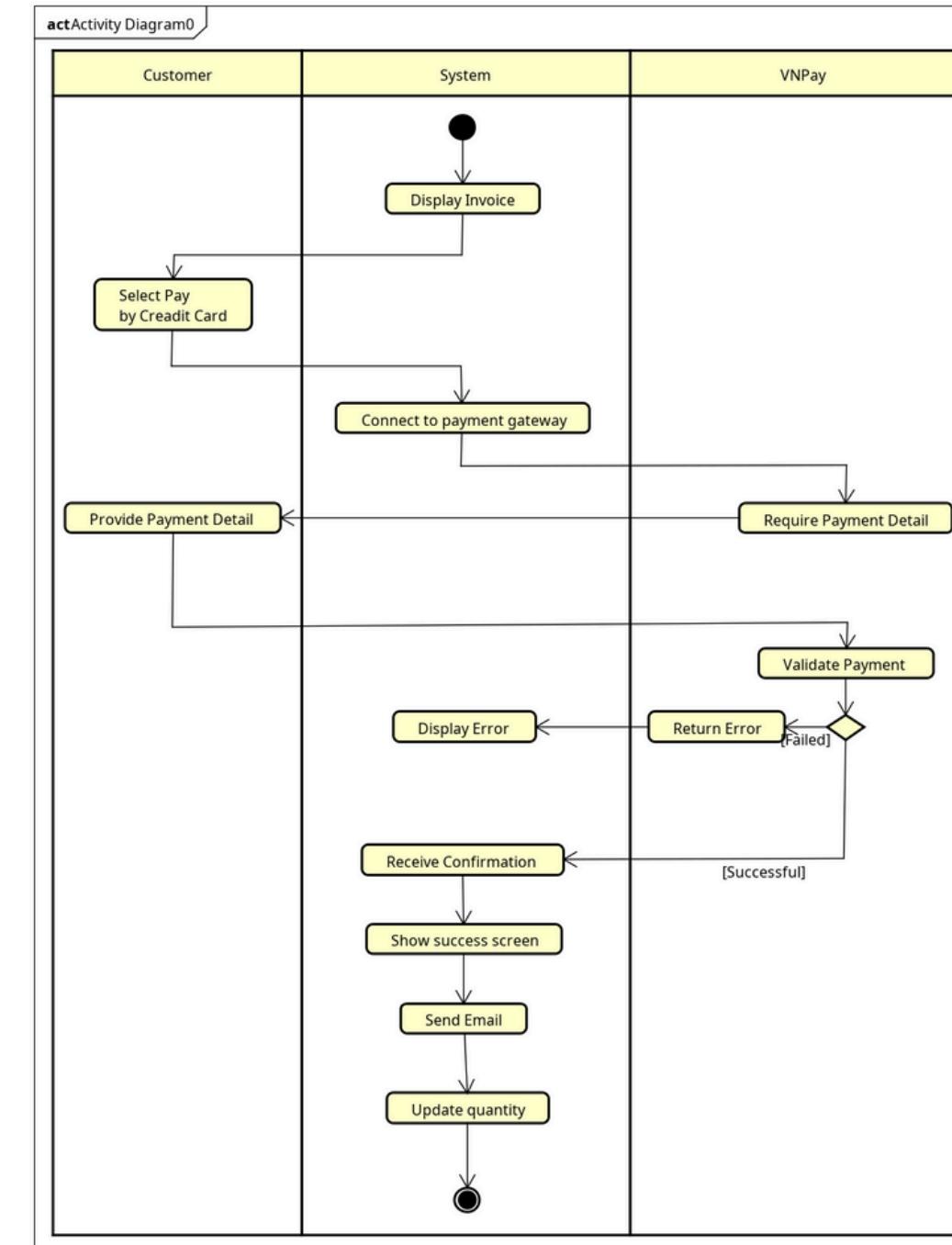
1.3 Activity Diagram



Place Rush Order

1. TỔNG QUAN HỆ THỐNG

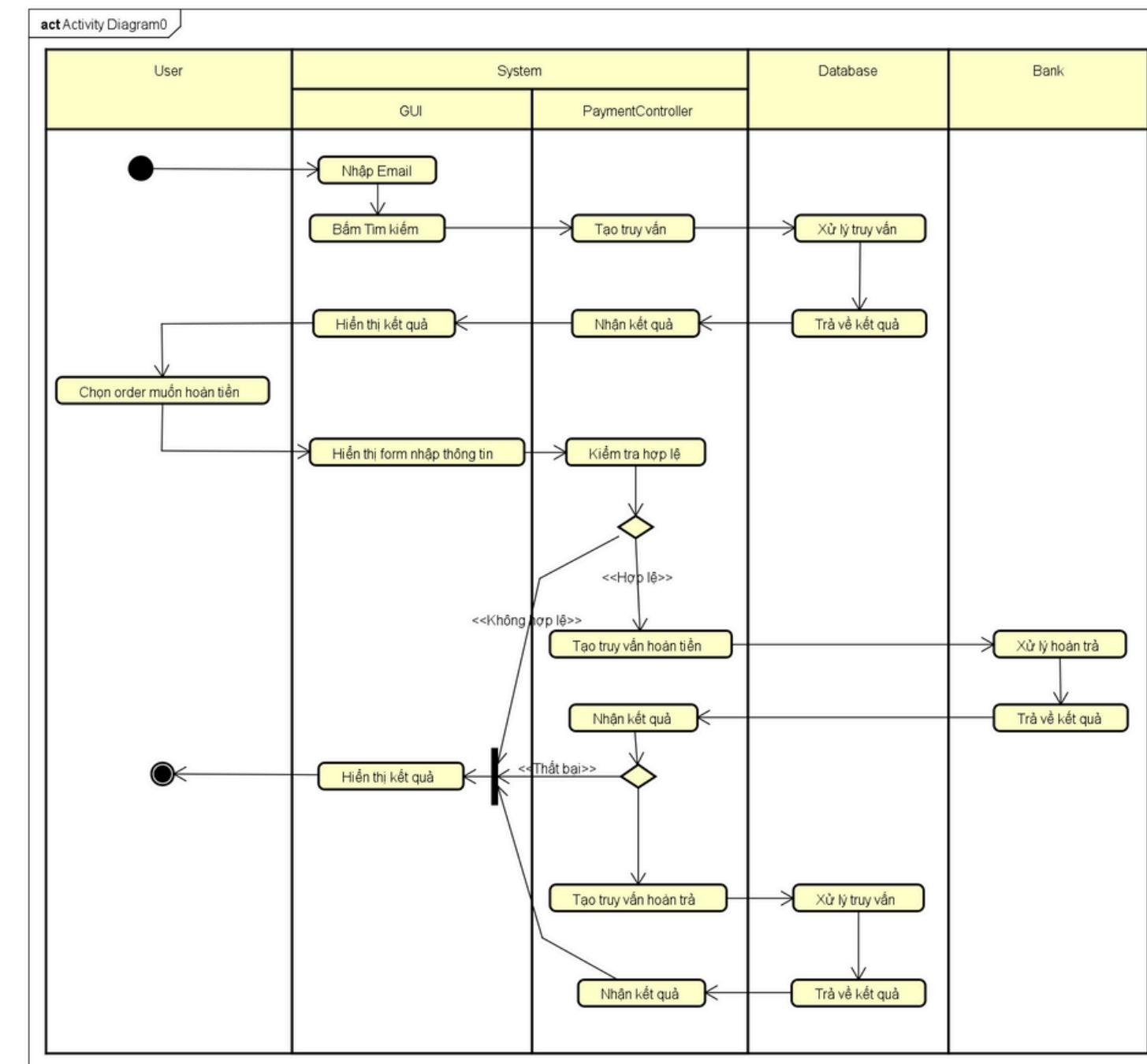
1.3 Activity Diagram



Payorder

1. TỔNG QUAN HỆ THỐNG

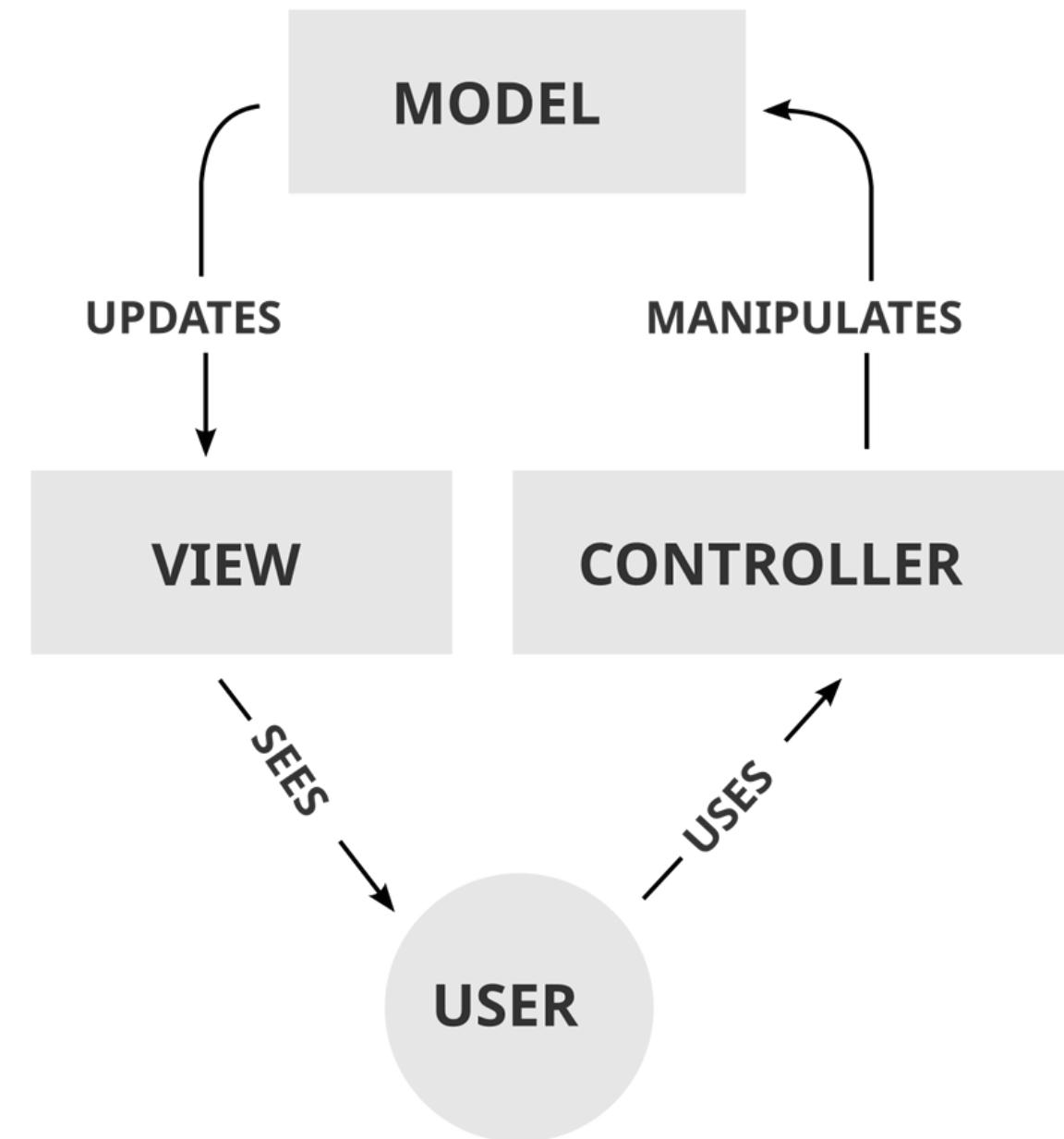
1.3 Activity Diagram



Refund

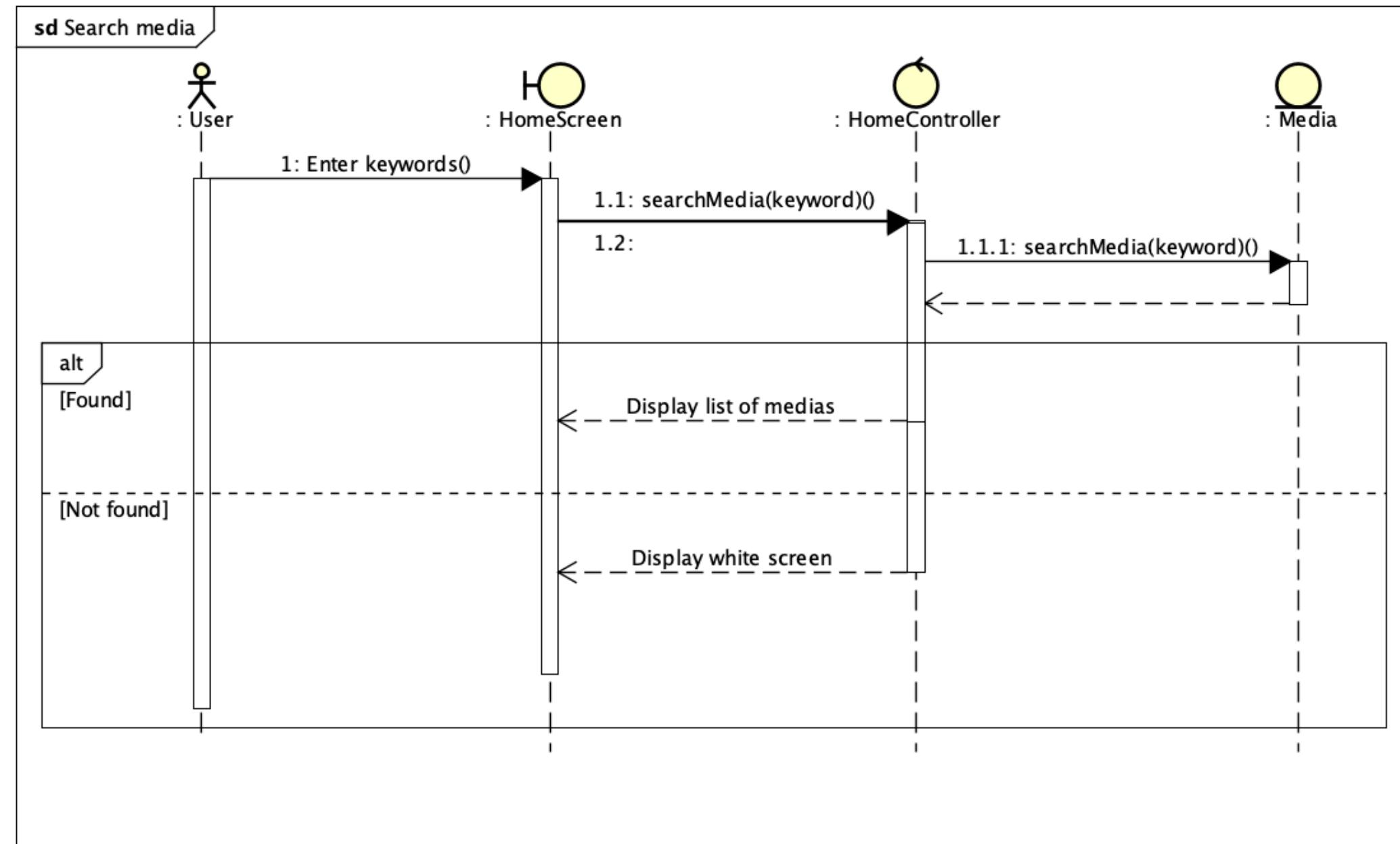
2. THIẾT KẾ KIẾN TRÚC

2.1 MVC



2. THIẾT KẾ KIẾN TRÚC

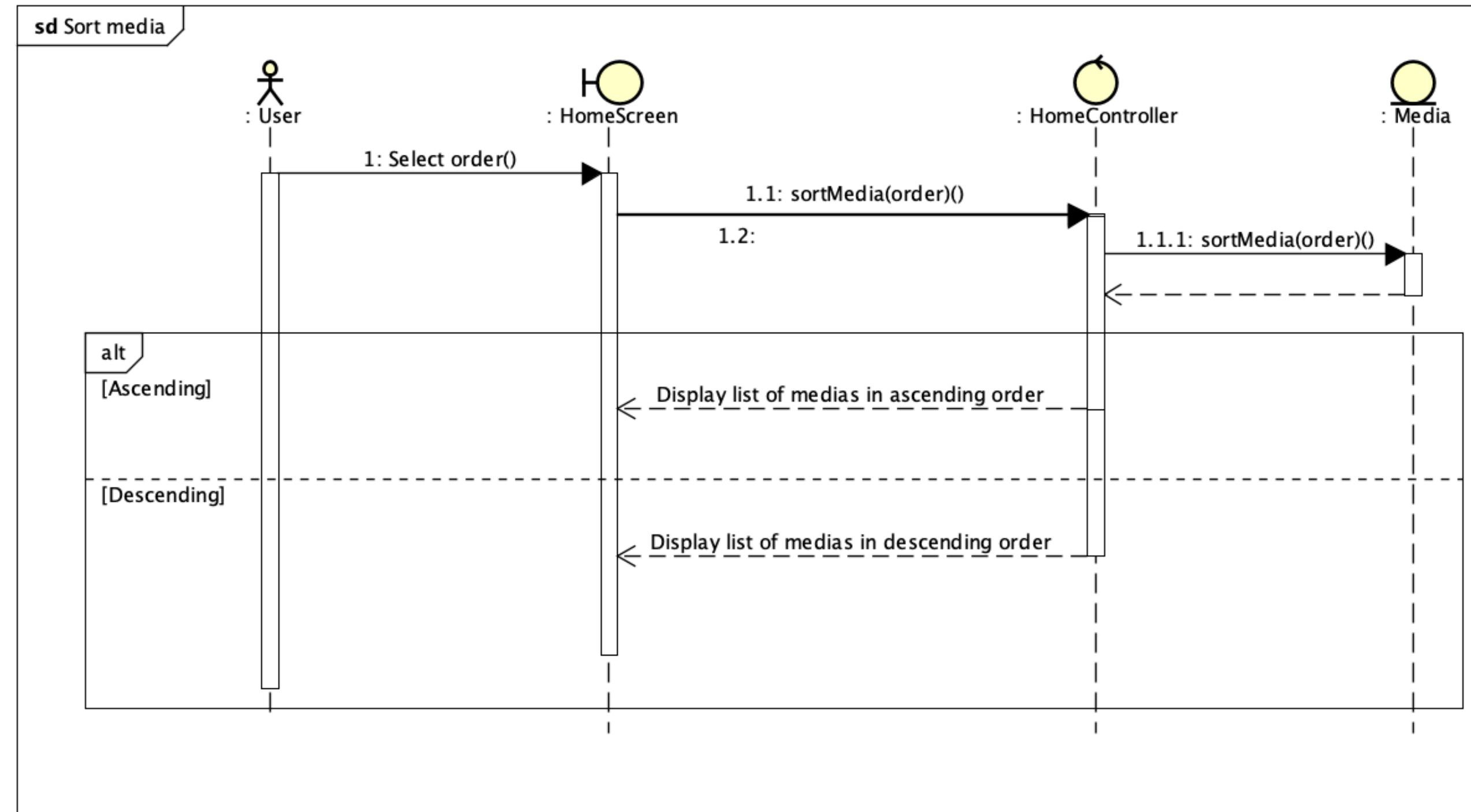
2.2 Sequence Diagram



Search media

2. THIẾT KẾ KIẾN TRÚC

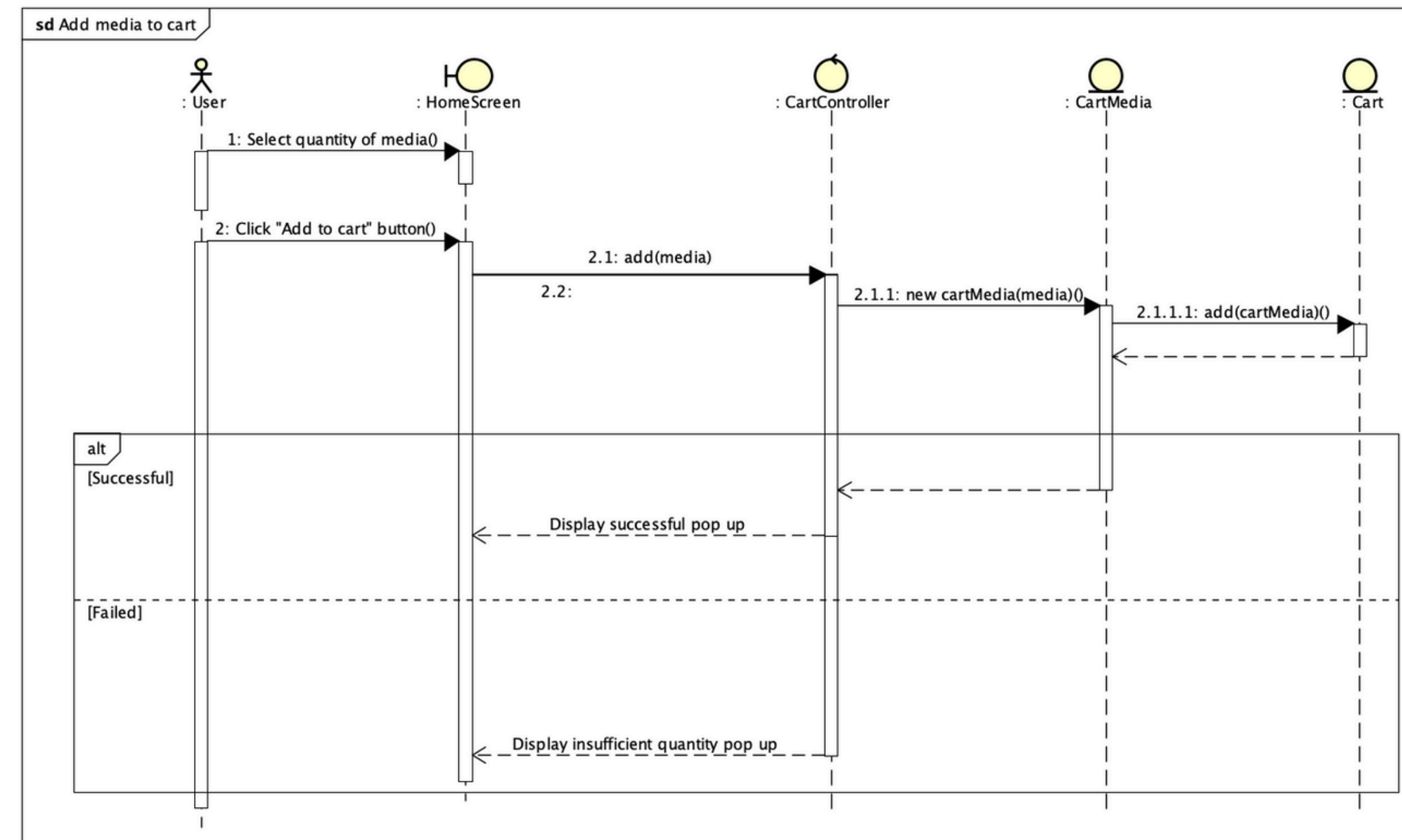
2.2 Sequence Diagram



Sort media

2. THIẾT KẾ KIẾN TRÚC

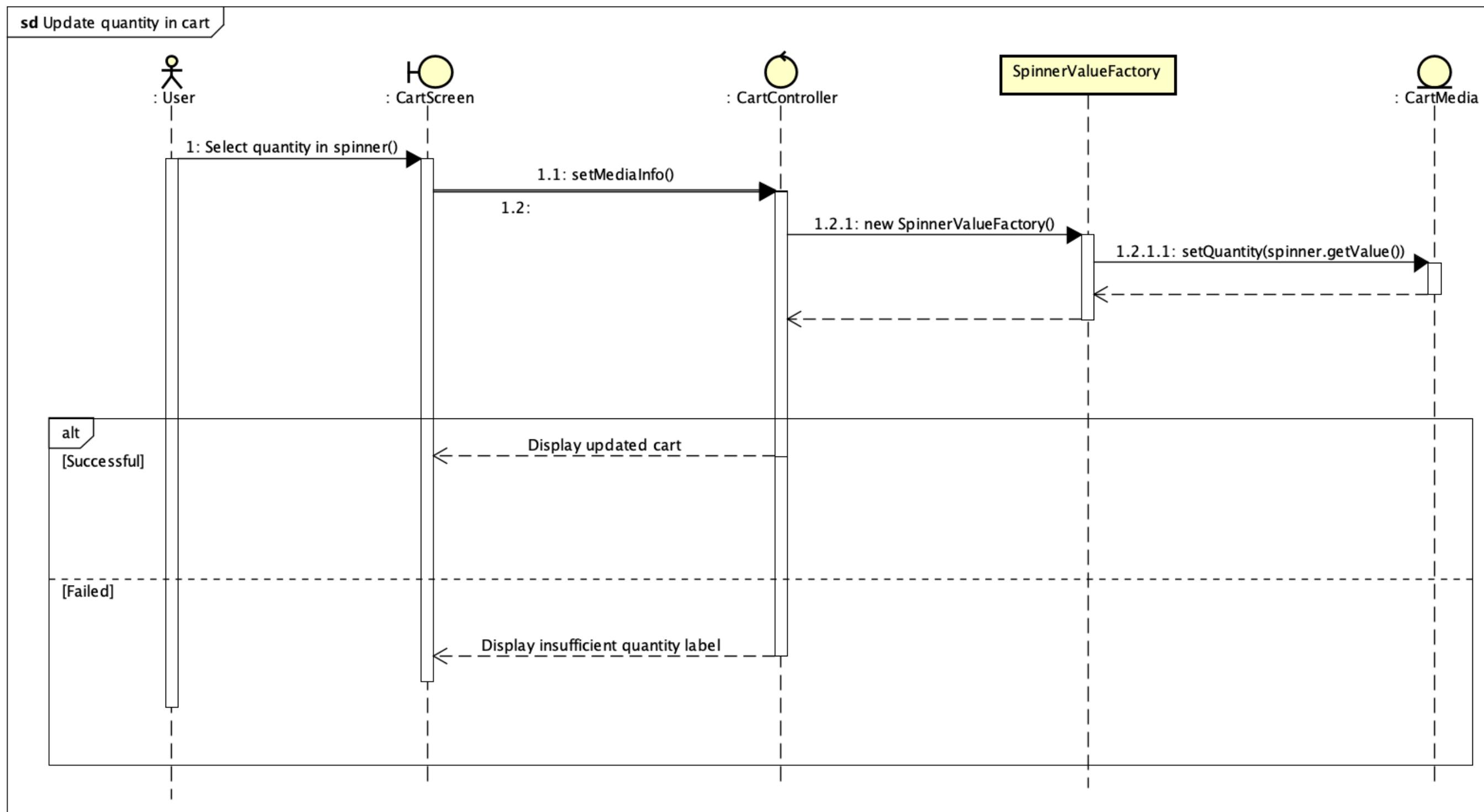
2.2 Sequence Diagram



Add media to cart

2. THIẾT KẾ KIẾN TRÚC

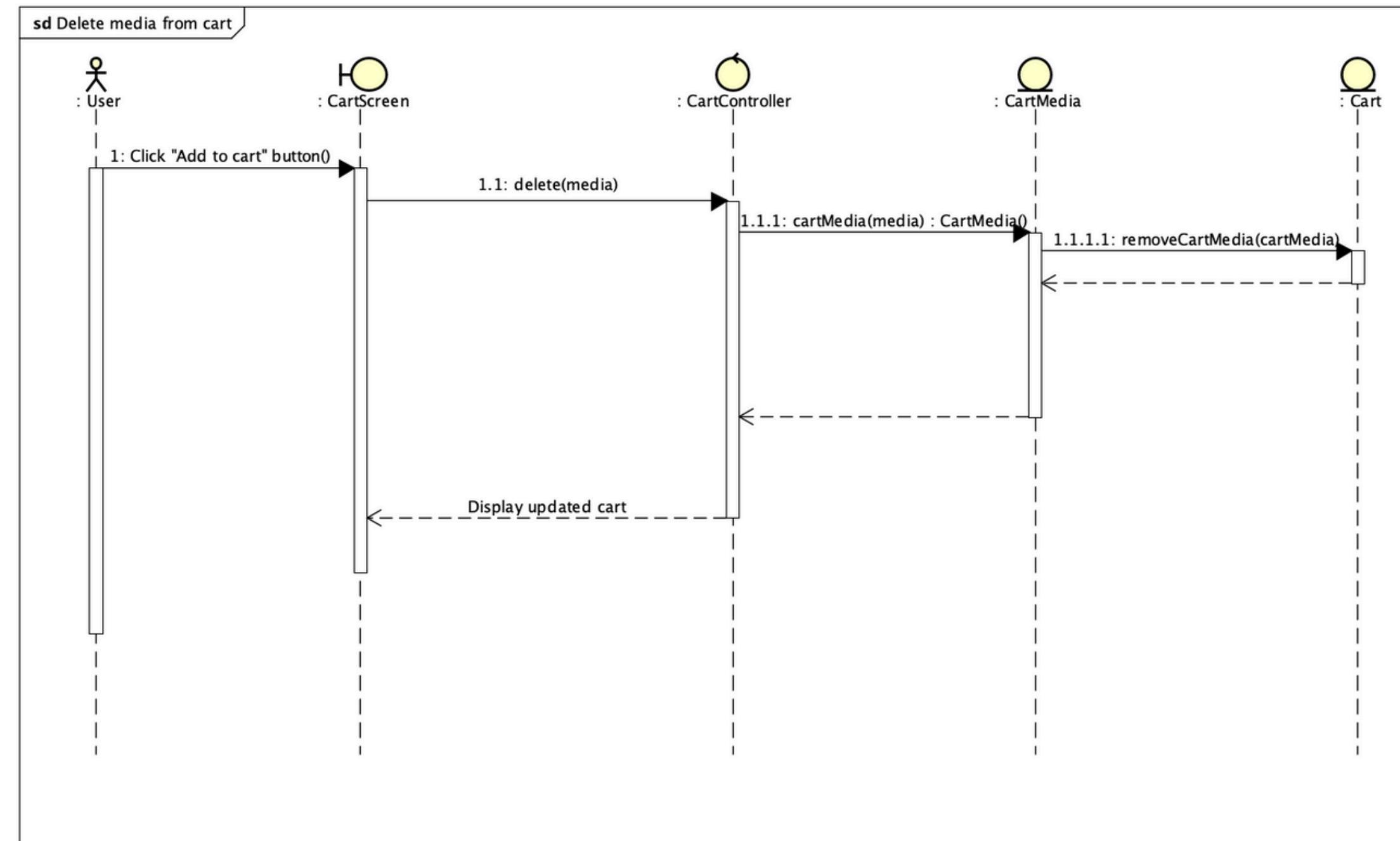
2.2 Sequence Diagram



Update quantity in cart

2. THIẾT KẾ KIẾN TRÚC

2.2 Sequence Diagram

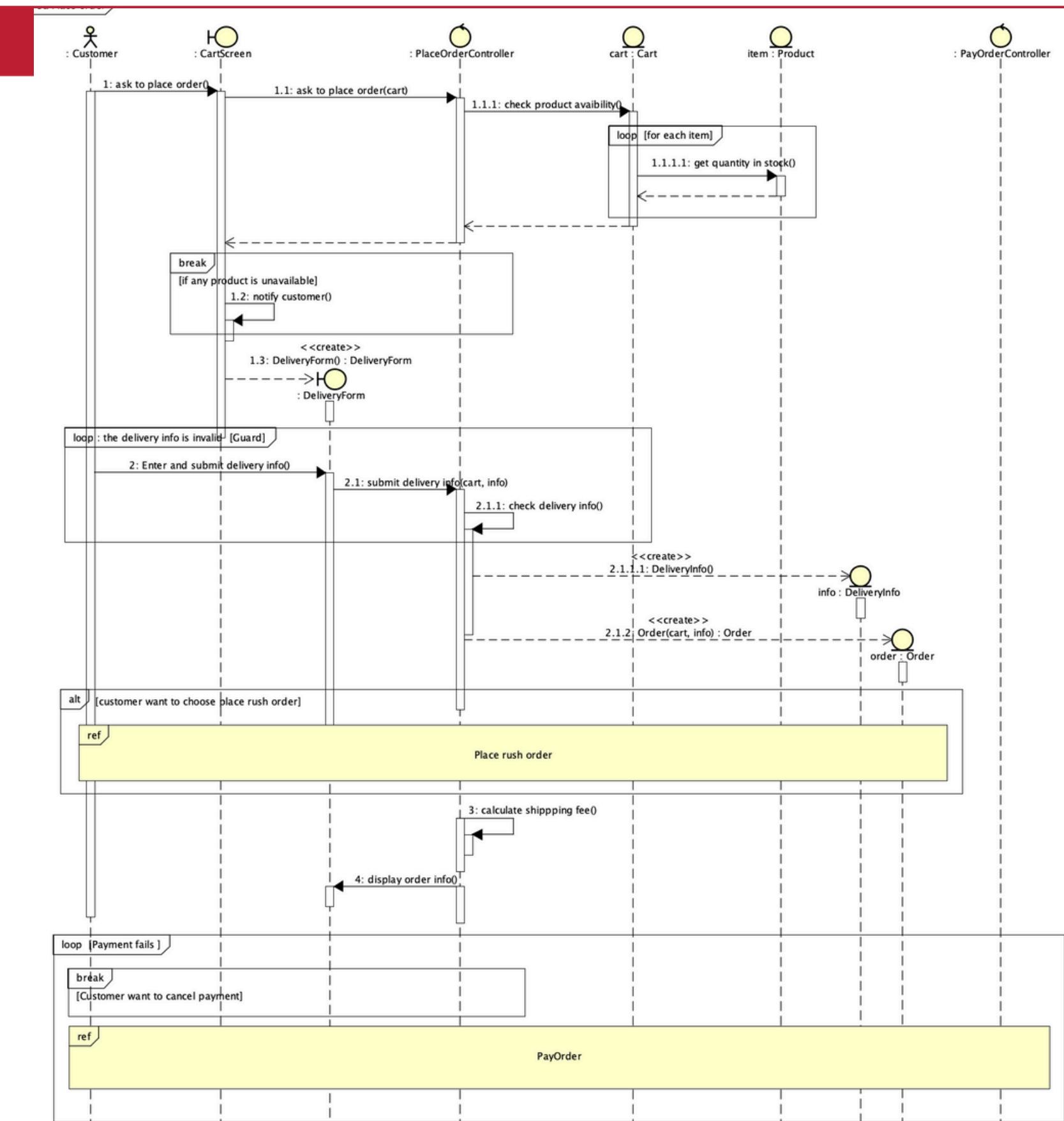


Delete media from cart

2. THIẾT KẾ KIẾN TRÚC

2.2 Sequence Diagram

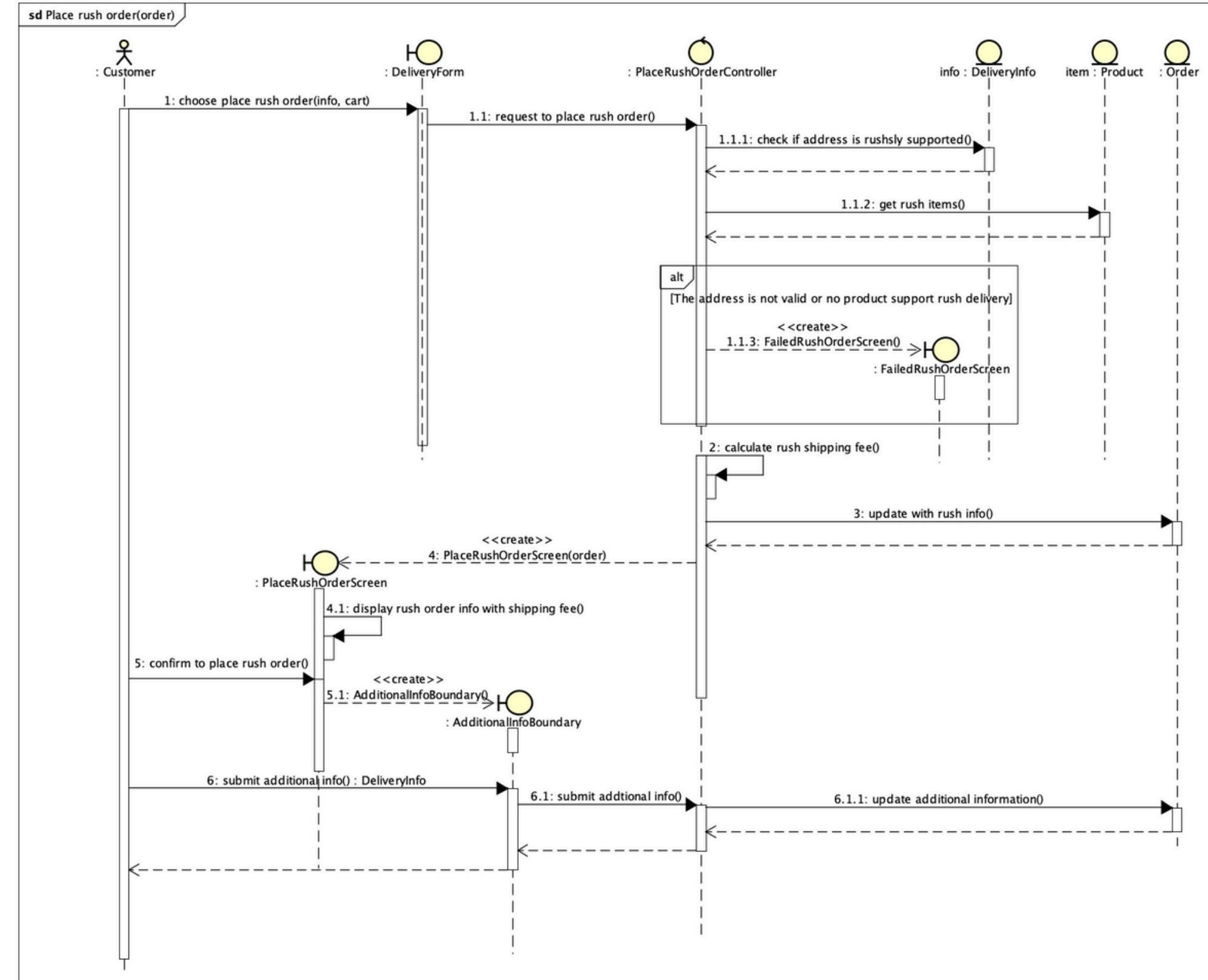
Place order



2. THIẾT KẾ KIẾN TRÚC

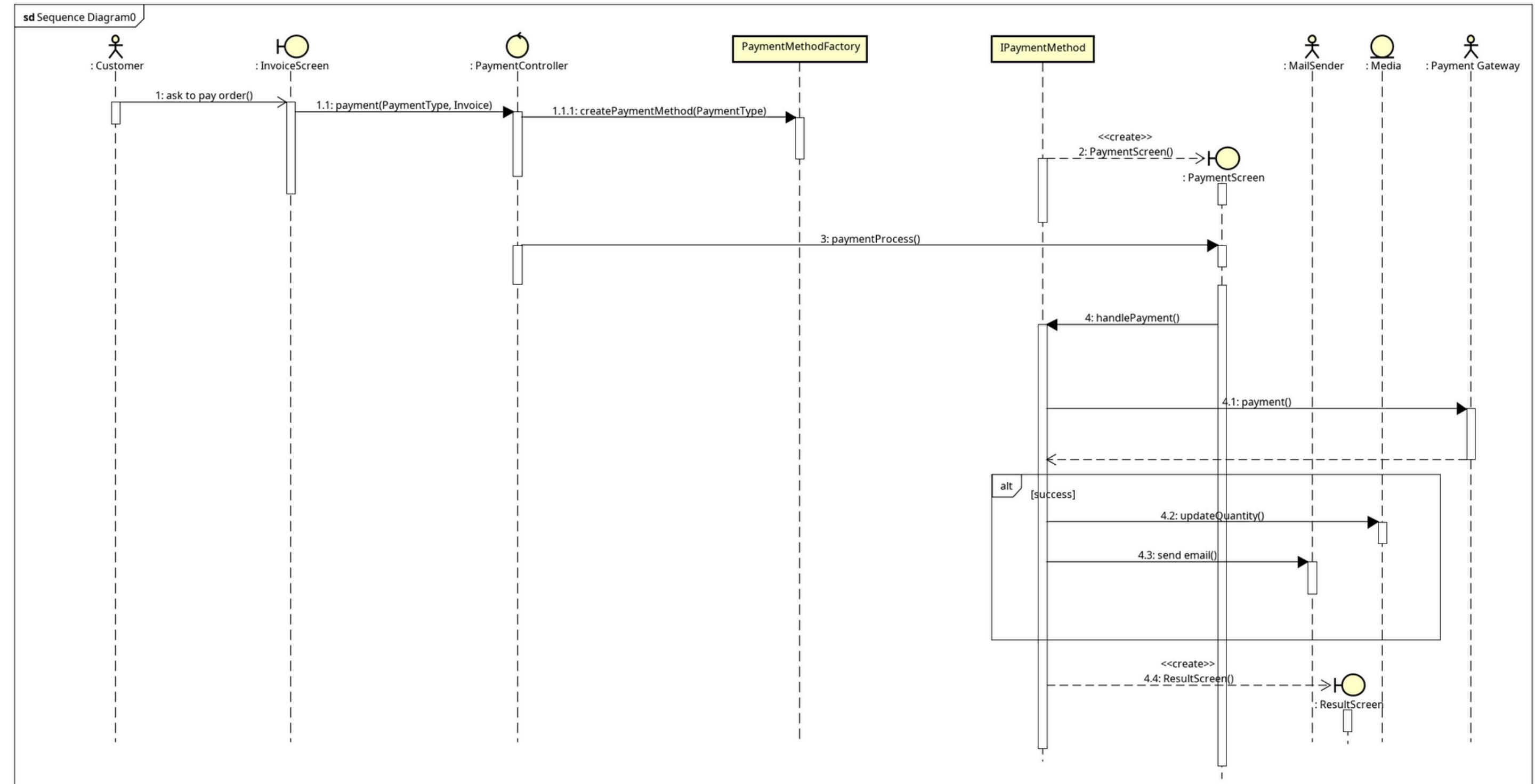
2.2 Sequence Diagram

Place rush order



2. THIẾT KẾ KIẾN TRÚC

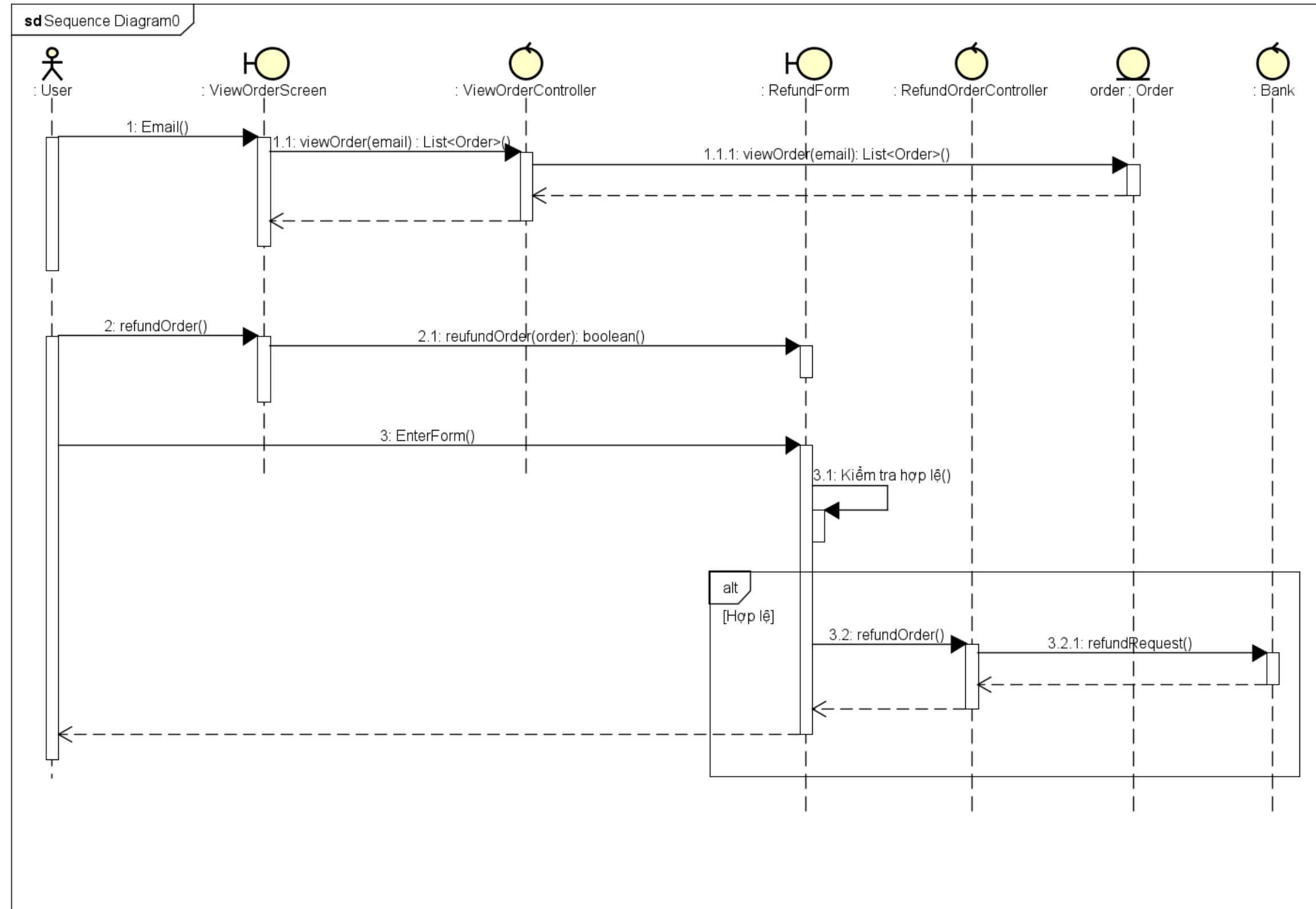
2.2 Sequence Diagram



2. THIẾT KẾ KIẾN TRÚC

2.2 Sequence Diagram

Refund



3. ĐÁNH GIÁ

3.1 Coupling

a. Content coupling

Các module liên quan	Mô tả	Hướng cải thiện
Order.setIstOrderMedia(List lstOrderMedia)	Bất kỳ module bên ngoài nào cũng có thể thay đổi hoàn toàn thuộc tính lstOrderMedia	Xóa phương thức này
Order.setShippingFees(int shippingFees)	Bất kỳ module bên ngoài nào cũng có thể thay đổi hoàn toàn thuộc tính shippingFees	Điều này không cần thiết vì shippingFees có thể được tính toán từ lstOrderMedia và deliveryInfo
Order.getDeliveryInfo()	Bất kỳ module nào có quyền truy cập deliveryInfo có thể thay đổi nó thông qua phương thức put()	Trả về đối tượng DeliveryInfo thay vì cặp hash map
Order.setDeliveryInfo(HashMap deliveryInfo)	Bất kỳ module bên ngoài nào cũng có thể thay đổi hoàn toàn thuộc tính deliveryInfo	Xóa phương thức này tạo và tạo một lớp DeliveryInfo
Order.setId(Integer Id)	Bất kỳ module bên ngoài nào cũng có thể thay đổi hoàn toàn thuộc tính Id	Xóa phương thức này

3. ĐÁNH GIÁ

3.1 Coupling

b. Control coupling

Các module liên quan	Mô tả	Hướng cải thiện
PaymentController	Hiện chưa hỗ trợ để có thể mở rộng thêm các phương thức thanh toán khác	Sử dụng PaymentMethodFactory để quét các package kế thừa từ IPaymentMethod để tránh if-else nhiều lần khi sử dụng các phương thức thanh toán, và dễ dàng cho tích hợp

3. ĐÁNH GIÁ

3.2 Cohesion

b. Logical Cohesion

Các module liên quan	Mô tả	Hướng cải thiện
MyMap	Tất cả phương thức liên quan đến xử lý dữ liệu JSON, nhưng thực hiện các tác vụ khác nhau như chuyển đổi, phân tích cú pháp và ánh xạ	Sử dụng một lớp riêng cho mỗi phương thức

3. ĐÁNH GIÁ

3.2 Cohesion

c. Procedural Cohesion

Các module liên quan	Mô tả	Hướng cải thiện
validateName(string Name), validatePhoneNumber(string phoneNumber), validateAddress(string address) trong lớp PlaceOrderController	Tất cả phương thức là các bước để xác thực thông tin giao hàng	Sử dụng một template cho việc đánh giá thông tin (Validator)

3. ĐÁNH GIÁ

3.2 Cohesion

d. Communicational Cohesion

Các module liên quan	Mô tả	Hướng cải thiện
createInvoice(Order order), calculateShippingFee(Order order) trong lớp PlaceOrderController	Tất cả phương thức chia sẻ cùng một đầu vào	Nên di chuyển phương thức createInvoice về lớp Invoice

3. ĐÁNH GIÁ

3.2 Cohesion

e. Sequential Cohesion

Các module liên quan	Mô tả	Hướng cải thiện
createOrder(), createInvoice(Order order) trong lớp PlaceOrderController	createOrder trả về một đơn hàng, là đầu vào cho createInvoice	Chuyển phương thức này sang lớp Order, Invoice tương ứng để hạn chế sự phụ thuộc giữa các lớp

3. ĐÁNH GIÁ

3.2 Cohesion

f. Functional Cohesion

Các module liên quan	Mô tả	Hướng cải thiện
DBConnection	Tất cả thuộc tính và phương thức giúp quản lý kết nối tới cơ sở dữ liệu	Kết hợp sử dụng Design Pattern DAO

3. ĐÁNH GIÁ

3.2 SOLID

a. Single Responsibility Principle

Vấn đề	Mô tả
Các lớp thực thể (entities) vừa đảm nhiệm việc lưu trữ dữ liệu và kết nối với Database.	Lớp Media vừa chứa các thuộc tính của Media, vừa đảm nhiệm việc kết nối với database
Các lớp controller đảm nhiệm nhiều trách nhiệm	Lớp PlaceOrderController vừa đảm nhiệm việc kiểm tra giỏ hàng, tính phí, validate đơn hàng, tạo order và tạo invoice

3. ĐÁNH GIÁ

3.2 SOLID

b. Open-Closed Principle

Vấn đề	Mô tả
Lớp DBConnection chỉ hỗ trợ SQLite	Nếu cần mở rộng sang các hệ quản trị cơ sở dữ liệu khác, cần phải viết lớp này lại từ đầu
Các lớp thực thể trực tiếp đảm nhiệm việc kết nối với database	Nếu cần mở rộng sang các hệ quản trị cơ sở dữ liệu khác, cần phải thay đổi toàn bộ các kết nối database trong các lớp thực thể
Request và VnPaySubsystemController	Một số logic trùng lặp giữa Request và VnPaySubsystemController (vd: xây dựng URL thanh toán). Điều này khiến việc sửa đổi logic khó khăn vì cần thay đổi ở nhiều nơi.
Lớp PlaceOrderController đảm nhiệm nhiều nhiệm vụ	Nếu muốn mở rộng cách thức tính phí giao hàng hoặc validate đơn hàng, cần phải thay đổi trực tiếp lớp này

3. ĐÁNH GIÁ

3.2 SOLID

c. Liskov Substitution Principle

Vấn đề	Mô tả
Lớp PaymentTransaction khó mở rộng	Nếu trong tương lai, PaymentTransaction được mở rộng (ví dụ: thêm các loại giao dịch khác nhau như RefundTransaction), cần đảm bảo các lớp con giữ nguyên hành vi của các phương thức như isSuccess.

3. ĐÁNH GIÁ

3.2 SOLID

d. Interface Segregation Principle

Vấn đề	Mô tả
Interface IPayment	Thiếu các method cho việc hoàn tiền và xử lý sau khi thanh toán thành công hoặc hoàn tiền thành công. Khó mở rộng ra nhiều phương thức thanh toán khác

3. ĐÁNH GIÁ

3.2 SOLID

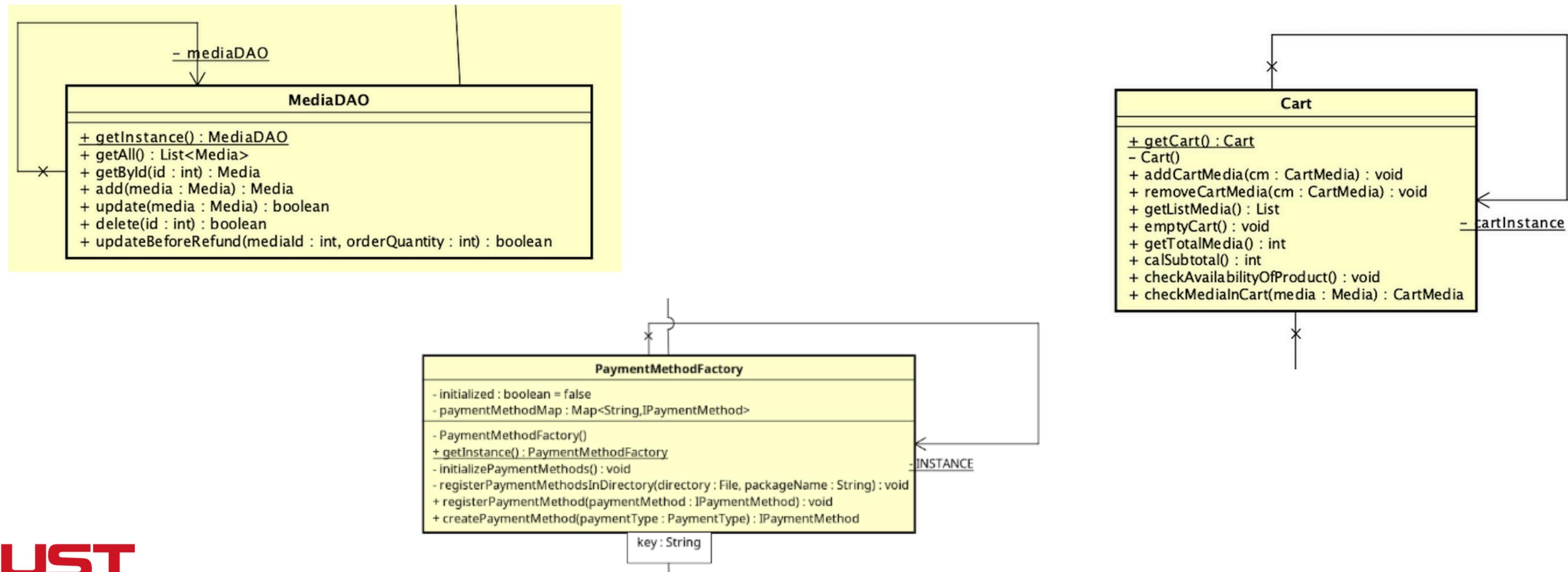
e. Dependency Inversion Principle

PaymentController	Phương thức payOrder gọi đến phương thức payOrder của lớp VnPaySubsystemController thay vì phương thức payOrder của interface IPayment. Trong tương lai nếu ta muốn sử dụng phương thức thanh toán khác, ta sẽ phải chỉnh sửa code
Media	Constructor Media gọi đến phương thức getConnection của lớp DBConnection. Trong tương lai nếu ta muốn sử dụng cơ sở dữ liệu khác, ta sẽ phải chỉnh sửa code
Book	Phương thức getMediaById gọi đến phương thức getConnection của lớp DBConnection. Trong tương lai nếu ta muốn sử dụng cơ sở dữ liệu khác, ta sẽ phải chỉnh sửa code
PlaceOrderController	Trong lớp này xây dựng thêm 2 phương thức createInvoice và createOrder, điều này là không nên vì việc tạo đối tượng Invoice hay Order nên được thực thi ở lớp Invoice hoặc Order tương ứng

4. ĐỀ XUẤT GIẢI PHÁP

4.1 Singleton design pattern

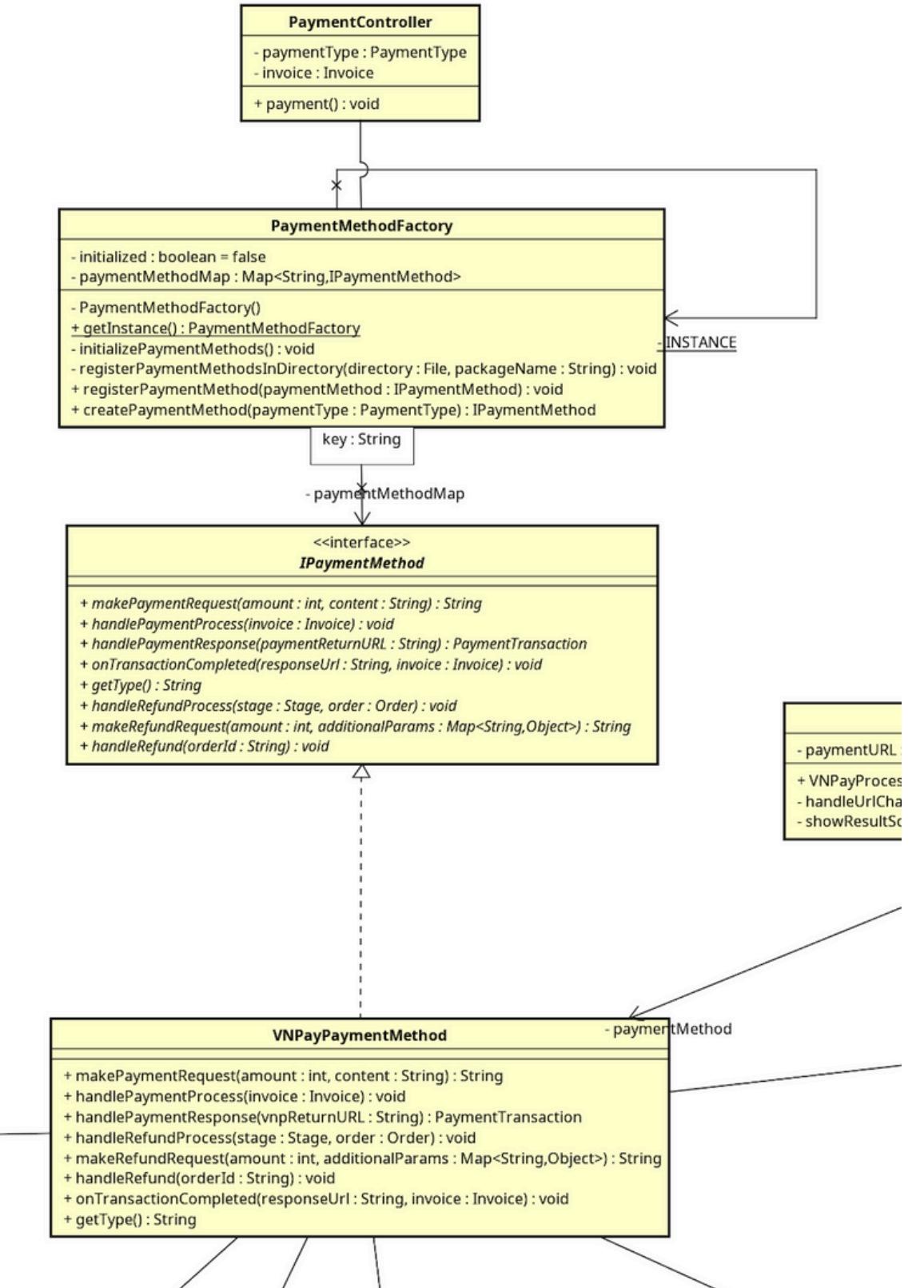
Áp dụng cho các Lớp mà xuyên suốt quá trình hoạt động chỉ cần sử dụng duy nhất một đối tượng như Cart, DBConnection, Các lớp Data Access Object



4. ĐỀ XUẤT GIẢI PHÁP

4.2 Factory Design Pattern

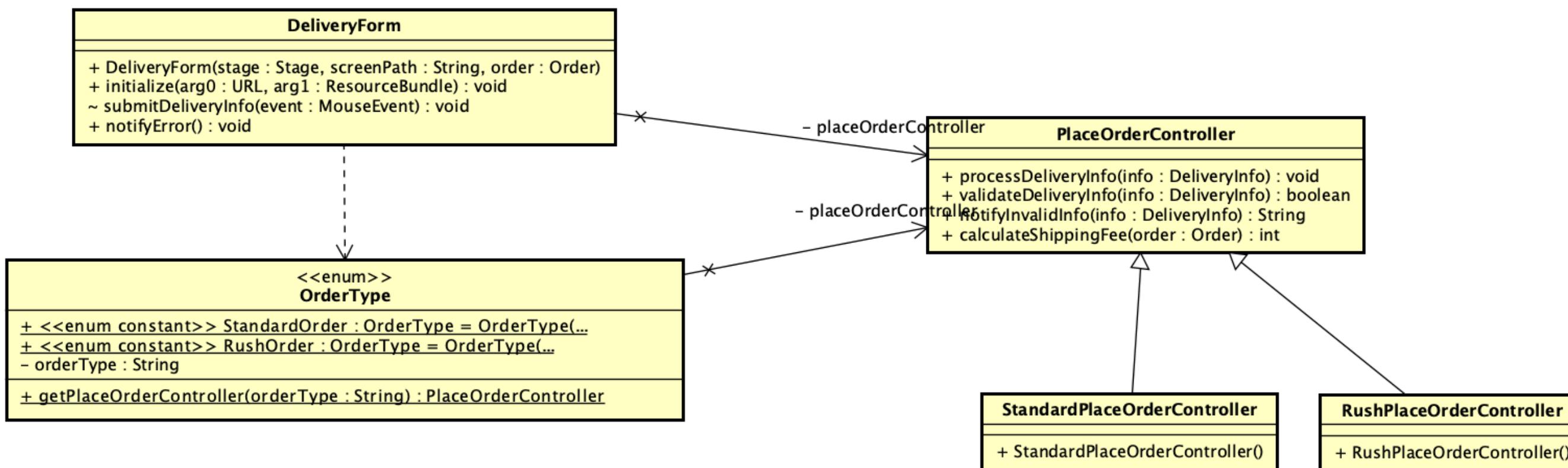
Áp dụng cho việc tạo các lớp phương thức thanh toán, tạo điều kiện để tích hợp các phương thức thanh toán khác trong tương lai.



4. ĐỀ XUẤT GIẢI PHÁP

4.3 Reflection

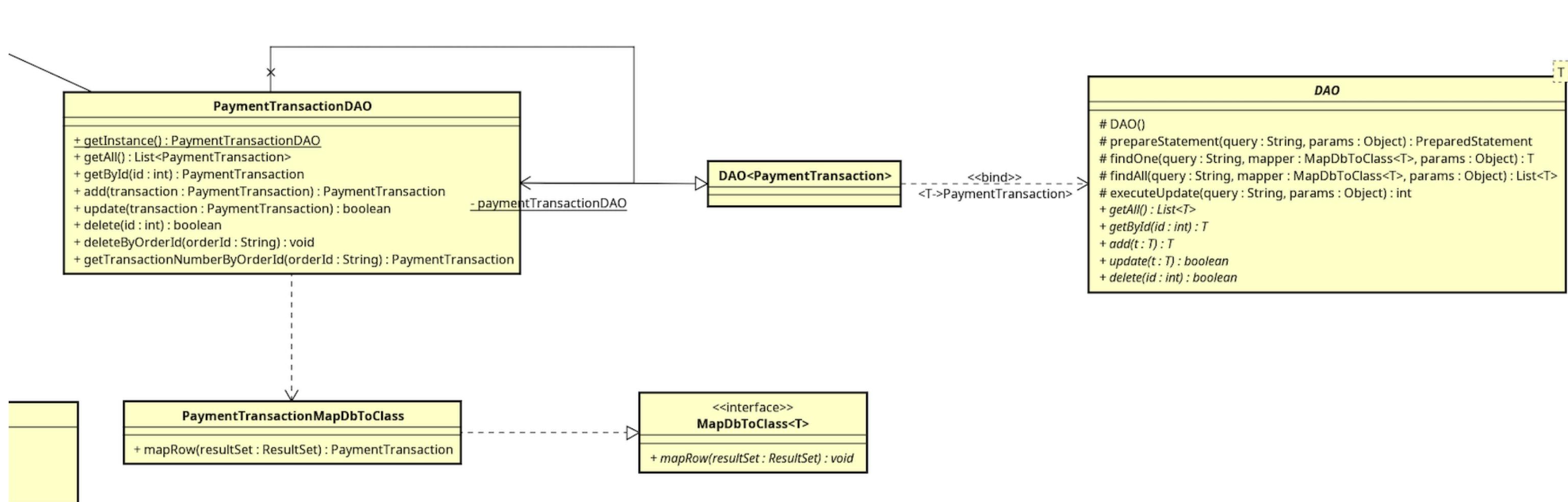
Tạo các đối tượng linh hoạt được ánh xạ trực tiếp từ thao tác lựa chọn của người dùng thông qua một lớp cấu hình



4. ĐỀ XUẤT GIẢI PHÁP

4.4 DAO Design Pattern

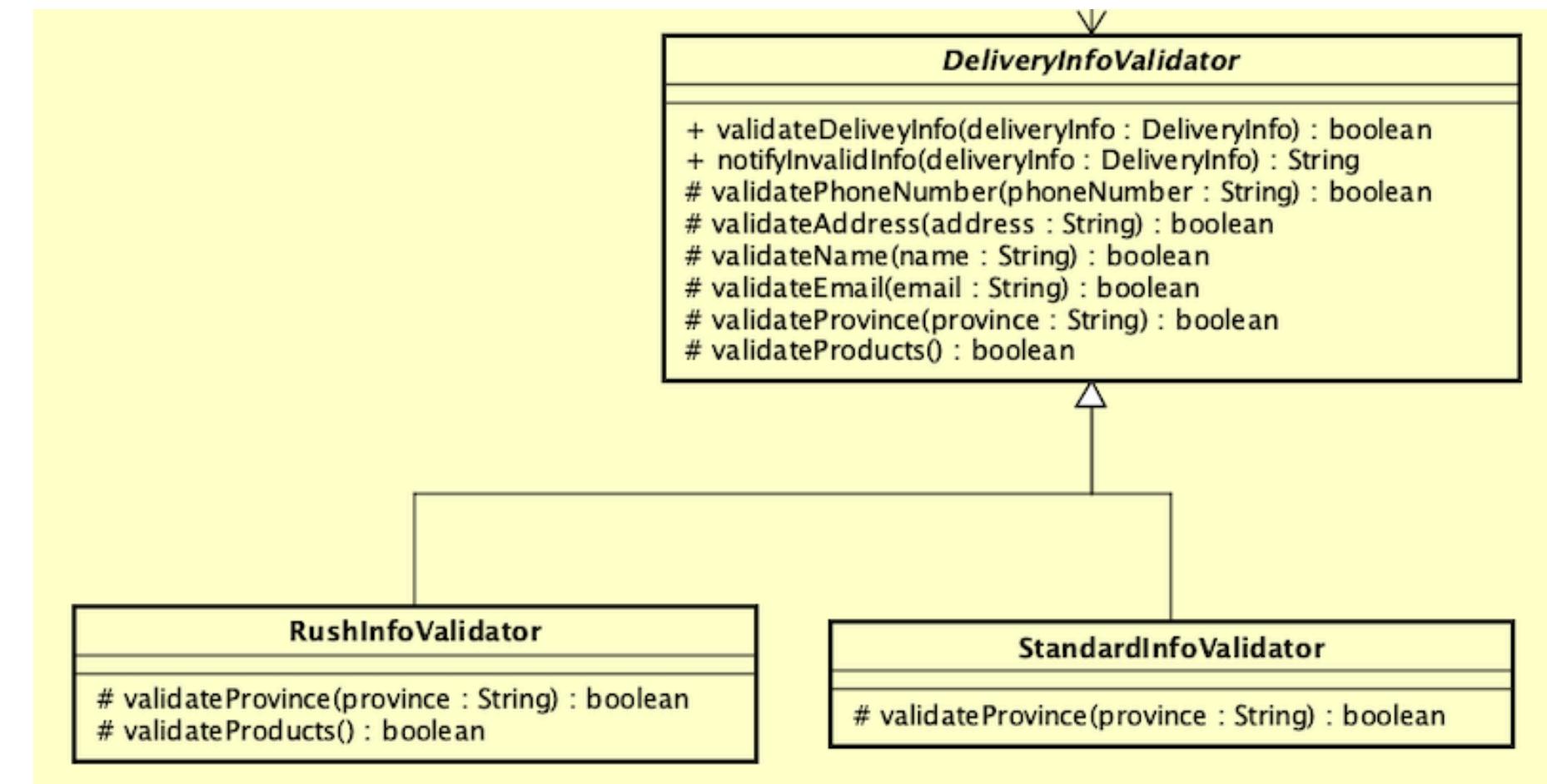
Tách biệt hoàn toàn nhiệm vụ lưu trữ thông tin và truy vấn cơ sở dữ liệu của các lớp thực thể. Mọi yêu cầu truy vấn cơ sở dữ liệu sẽ được thông qua các lớp DAO



4. ĐỀ XUẤT GIẢI PHÁP

4.5. Template

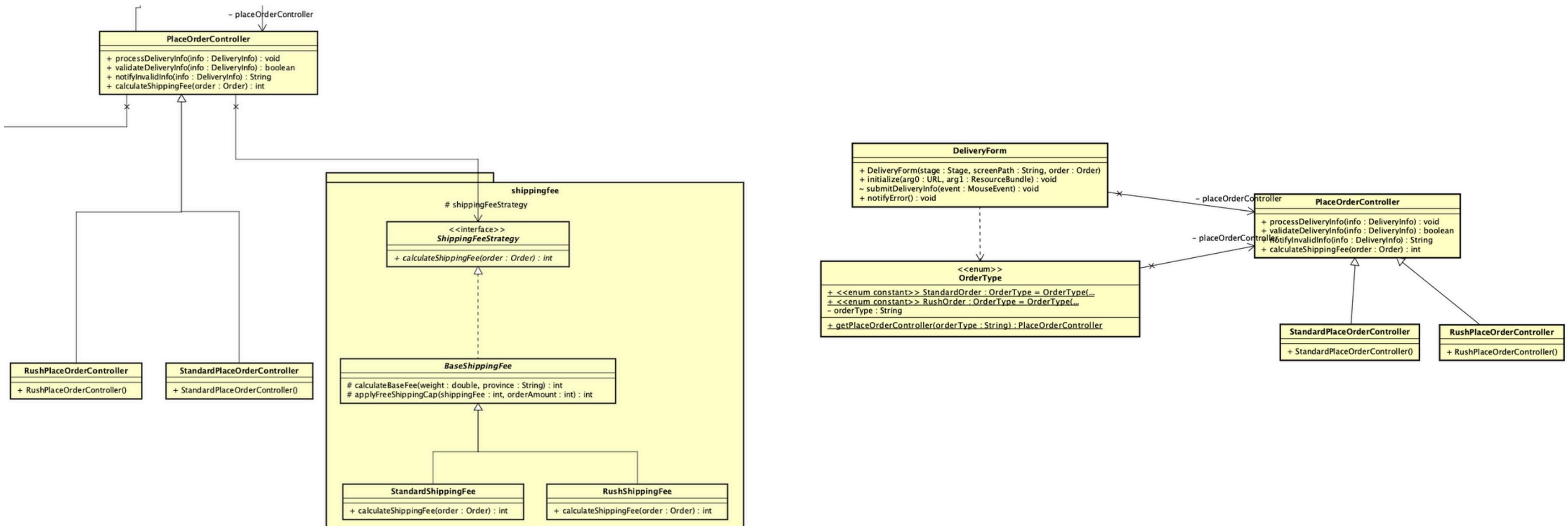
Áp dụng cho các thuật toán có cùng một khung, dễ dàng cho các lớp con có thể triển khai và mở rộng



4. ĐỀ XUẤT GIẢI PHÁP

4.5. Strategy

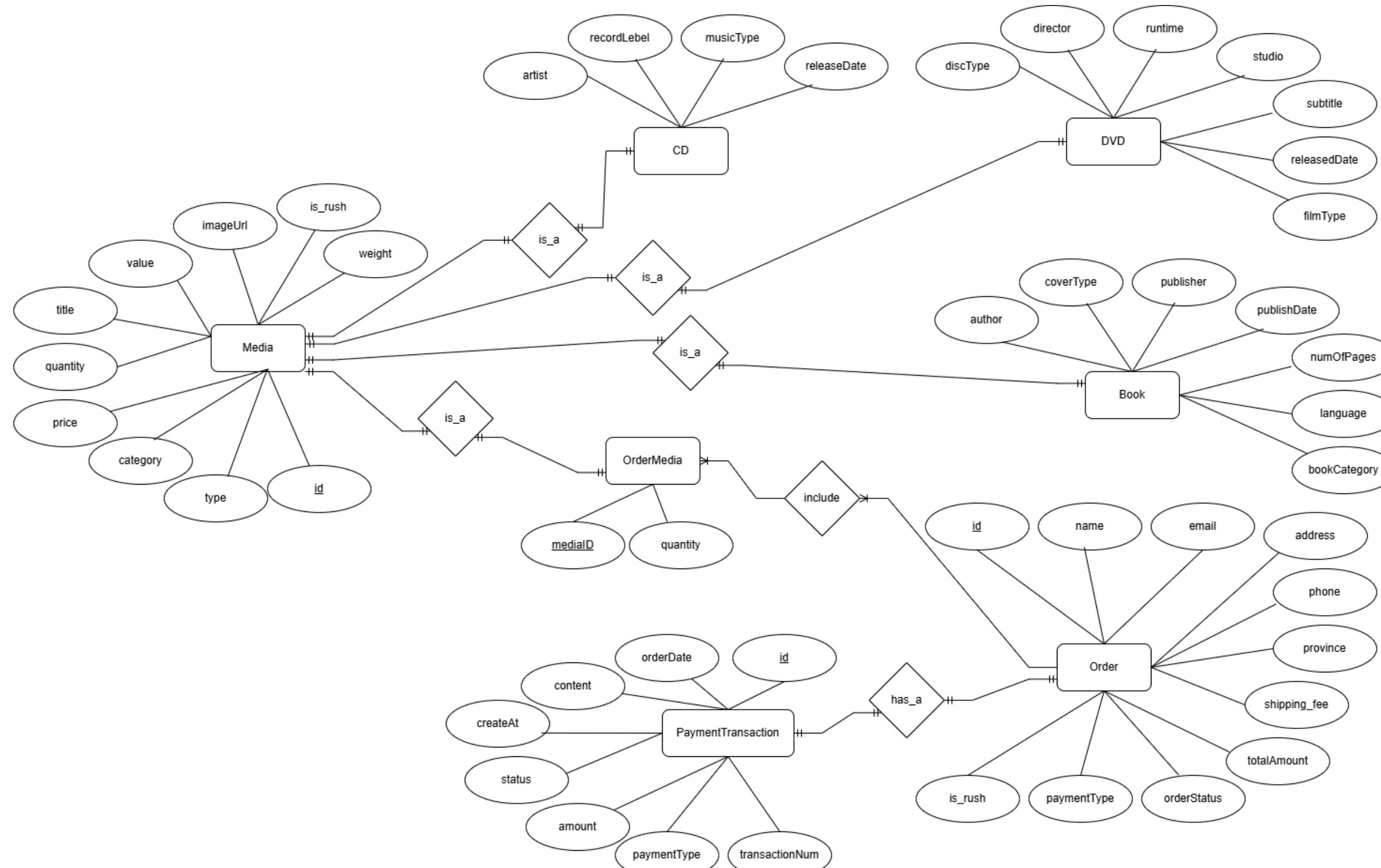
Dùng để định nghĩa một họ các thuật toán, đặt chúng trong các lớp riêng biệt, và khiến chúng có thể thay thế nhau một cách linh hoạt, cũng như dễ mở rộng trong tương lai



5. THIẾT KẾ LỚP

6. MÔ HÌNH HÓA DỮ LIỆU

6.1. Conceptual Data Model



6. MÔ HÌNH HÓA DỮ LIỆU

6.2. Logical Data Model

