

ĐẠI HỌC BÁCH KHOA HÀ NỘI

TÀI LIỆU THIẾT KẾ VÀ XÂY DỤNG PHẦN MỀM

Tài liệu thiết kế phần mềm AIMS

Ngành Công nghệ thông tin và truyền thông

Giảng viên hướng dẫn: TS. Nguyễn Thị Thu Trang

Chữ ký GVHD

Nhóm: 01

Sinh viên thực hiện:
Vũ Thành An 20205197
Đỗ Duy Anh 20205198
Nguyễn Ngọc Ánh 20205228

Khoa: Công nghệ thông tin Việt-Pháp

Trường: Công nghệ Thông tin và Truyền thông

HÀ NỘI, 12/2023

MỤC LỤC

CHƯƠNG 1. GIỚI THIỆU	1
1.1 Mục tiêu	1
1.2 Phạm vi đề tài	1
1.3 Bảng chú giải	1
1.4 References	1
CHƯƠNG 2. KHẢO SÁT VÀ PHÂN TÍCH YÊU CẦU.....	2
2.1 Survey	2
2.2 Overall requirements.....	2
2.2.1 Biểu đồ use case tổng quát	2
2.2.2 Biểu đồ hoạt động	3
CHƯƠNG 3. PHÂN TÍCH YÊU CẦU.....	8
3.1 Đặc tả ca sử dụng	8
3.1.1 Use case “Quản lý sản phẩm”.....	8
3.1.2 Use case “Thanh toán”	10
3.1.3 Use case “RU Giỏ hàng”	12
3.1.4 Use case “Đặt hàng”	13
3.1.5 Use case “Giao hàng nhanh”	16
CHƯƠNG 4. KIẾN TRÚC VÀ THIẾT KẾ.....	18
4.1 Architectural Patterns.....	18
4.2 Interaction Diagrams.....	19
4.2.1 Use case "Quản lý sản phẩm"	19
4.2.2 Use case "Tìm kiếm sản phẩm"	20
4.2.3 Use case "Giỏ hàng"	21
4.2.4 Use case "Đặt hàng "	22

4.2.5 Use case "Giao hàng nhanh"	23
4.2.6 Use case "Thanh toán".....	24
4.3 Analysis Class Diagrams	27
4.3.1 General Class Diagrams	27
4.3.2 Use case "CRUD Sản phẩm"	27
4.3.3 Use case "Giỏ hàng "	28
4.3.4 Use case "Đặt hàng "	29
4.3.5 Use case "Giao hàng nhanh "	29
4.3.6 Use case "Thanh toán".....	30
4.3.7 Use case "Hoàn tiền"	30
CHƯƠNG 5. THIẾT KẾ CHI TIẾT	32
5.1 Data Modeling	32
5.1.1 Conceptual Data Model	32
5.1.2 Logical Data Model	33
5.1.3 Physical Data Model	33
5.2 Interface Design	38
5.2.1 User Interface Design.....	38
5.2.2 System Interface Design	44
CHƯƠNG 6. LUU Ý THIẾT KẾ	48
6.1 Architectural Strategies	48
6.2 Goals and Guidelines	48
6.3 Coupling and Cohesion	48
6.3.1 App.ts	48
6.3.2 express.server.ts	49
6.3.3 mongoose.connection.ts:.....	51

6.4 Design Principles.....	52
6.4.1 Single Responsibility.....	52
6.4.2 Open Closed.....	52
6.4.3 Liskov substitution	55
6.4.4 Interface segregation	56
6.4.5 Dependency Inversion	58
6.5 Design Patterns	59
6.5.1 Singleton Pattern.....	59
6.5.2 Factory Pattern	59
CHƯƠNG 7. ĐÁNH GIÁ CÔNG VIỆC	64

DANH MỤC HÌNH VẼ

Hình 2.1	Biểu đồ use case tổng quan hệ thống	2
Hình 2.2	Biểu đồ hoạt động nghiệp vụ CRUD sản phẩm	3
Hình 2.3	Biểu đồ hoạt động nghiệp vụ CRUD người dùng	4
Hình 2.4	Biểu đồ hoạt động nghiệp vụ Tìm kiếm sản phẩm	5
Hình 2.5	Biểu đồ hoạt động nghiệp vụ Quản lý đơn hàng	6
Hình 2.6	Biểu đồ hoạt động nghiệp vụ đặt hàng	7
Hình 4.1	Sơ đồ kiến trúc hình lục giác	18
Hình 4.2	Biểu đồ tuần tự nghiệp vụ thêm xem sửa xóa sản phẩm	19
Hình 4.3	Biểu đồ tuần tự nghiệp vụ tìm kiếm sản phẩm	20
Hình 4.4	Biểu đồ tuần tự nghiệp vụ đọc hoặc cập nhật giỏ hàng	21
Hình 4.5	Biểu đồ tuần tự nghiệp vụ đặt hàng	22
Hình 4.6	Biểu đồ tuần tự nghiệp vụ giao hàng nhanh	23
Hình 4.7	Biểu đồ tuần tự nghiệp vụ thanh toán đơn hàng	24
Hình 4.8	Biểu đồ tuần tự nghiệp vụ hoàn tiền đơn hàng	25
Hình 4.9	Biểu đồ tuần tự nghiệp vụ người quản lý quản lý đơn hàng	26
Hình 4.10	Biểu đồ tuần tự nghiệp vụ người khách hàng quản lý đơn hàng	26
Hình 4.11	General Class Diagrams	27
Hình 4.12	Biểu đồ lớp nghiệp vụ thêm xem sửa xóa Sản phẩm	27
Hình 4.13	Biểu đồ lớp nghiệp vụ xem và cập nhật giỏ hàng	28
Hình 4.14	Biểu đồ lớp nghiệp vụ xem và cập nhật giỏ hàng	29
Hình 4.15	Biểu đồ lớp nghiệp vụ xem và cập nhật giỏ hàng	29
Hình 4.16	Biểu đồ lớp nghiệp vụ thanh toán đơn hàng	30
Hình 4.17	Biểu đồ lớp nghiệp vụ hoàn tiền đơn hàng	30
Hình 4.18	Biểu đồ lớp nghiệp vụ khách hàng quản lý đơn hàng	31
Hình 4.19	Biểu đồ lớp nghiệp vụ khách hàng quản lý quản lý đơn hàng	31
Hình 5.1	Biểu đồ ERD	32
Hình 5.2	Logical Data Model	33
Hình 5.3	Màn hình trang chủ	38
Hình 5.4	Màn hình chi tiết sản phẩm	39
Hình 5.5	Màn hình giỏ hàng	40
Hình 5.6	Màn hình địa chỉ	40
Hình 5.7	Màn hình thông tin liên hệ	41
Hình 5.8	Màn hình thanh toán	41
Hình 5.9	Màn hình xác nhận	42

Hình 5.10 Màn hình email	42
Hình 5.11 Màn hình thông tin đơn hàng	43
Hình 5.12 Biểu đồ gói payment subsystem	44
Hình 5.13 Biểu đồ tuần tự nghiệp vụ thanh toán	44
Hình 5.14 Biểu đồ tuần tự nghiệp vụ hoàn tiền	45
Hình 5.15 Biểu đồ gói notification subsystem	45
Hình 5.16 Biểu đồ gói Storage Service Subsystem	46
Hình 5.17 Biểu đồ tuần tự quá trình upload	47
 Hình 6.1 Product Controller	52
Hình 6.2 File Interface	53
Hình 6.3 Storage Manage Interfaces	53
Hình 6.4 Firebase Storage Manager	54
Hình 6.5 Remote Storage Manage	54
Hình 6.6 Class PaymentGatewayFactory	55
Hình 6.7 Product Dao Factory	56
Hình 6.8 Class PaymentService	57
Hình 6.9 OrderDaoOpen	57
Hình 6.10 OrderDaoController	57
Hình 6.11 Class PayPalService	58
Hình 6.12 Class OrderMongooseDaoDIP	58
Hình 6.13 Class OrderControllerDIP	59
Hình 6.14 Class TransactionModel	60
Hình 6.15 Product Interfaces	60
Hình 6.16 Product Factory	61
Hình 6.17 Product Factory Usage	61
Hình 6.18 Class PaymentGatewayFactory	62
Hình 6.19 Decorator PaymentProvider	63
Hình 6.20 Class PaypalService sử dụng decorator PaymentProvider để tự động register với PaymentGatewayFactory	63

CHƯƠNG 1. GIỚI THIỆU

1.1 Mục tiêu

Mục đích của tài liệu thiết kế phần mềm (AIMS) này là để cung cấp một cái nhìn tổng quan chi tiết về cách thiết kế ứng dụng AIMS. Tài liệu sẽ hỗ trợ như một nguồn thông tin chi tiết giúp đội ngũ phát triển thực hiện việc xây dựng phần mềm, và giúp đội kiểm thử đảm bảo rằng phần mềm hoạt động theo đúng yêu cầu. Ngoài ra, AIMS cũng sẽ cung cấp thông tin cần thiết cho quản lý dự án và các bên liên quan khác để hiểu rõ về phạm vi dự án, cấu trúc thiết kế và các vấn đề rủi ro có thể xảy ra trong quá trình phát triển phần mềm.

1.2 Phạm vi đề tài

Phần mềm AIMS Project mang lại cơ hội mua sắm trực tuyến cho sản phẩm phương tiện truyền thông. Với khả năng hoạt động 24/7 phần mềm này đảm bảo trải nghiệm mượt mà và ổn định. Người quản lý sản phẩm có thể thêm, sửa, và xóa sản phẩm một cách linh hoạt, trong khi khách hàng có thể dễ dàng tìm kiếm, xem thông tin và đặt mua sản phẩm yêu thích.

Mục tiêu của AIMS Project là cung cấp một nền tảng thương mại điện tử hiệu quả và dễ sử dụng cho việc mua bán sản phẩm truyền thông. Phần mềm được thiết kế để hỗ trợ nhu cầu của cả người quản lý và khách hàng, đồng thời nâng cao khả năng tiếp cận sản phẩm văn hóa và giải trí. AIMS Project không chỉ giúp người quản lý quản lý kho hàng hiệu quả mà còn giúp khách hàng tận hưởng trải nghiệm mua sắm tiện lợi và nhanh chóng.

Tóm lại, AIMS Project là một giải pháp toàn diện cho việc mua sắm trực tuyến, cung cấp nền tảng kỹ thuật số cho người quản lý sản phẩm để quản lý hiệu quả và cho phép khách hàng tìm kiếm, so sánh và mua sắm sản phẩm dễ dàng. Mục tiêu chính của phần mềm là thúc đẩy thị trường mua bán sản phẩm truyền thông và cung cấp trải nghiệm mua sắm tiện ích, an toàn cho người dùng.

1.3 Bảng chú giải

NONE

1.4 References

NONE

CHƯƠNG 2. KHẢO SÁT VÀ PHÂN TÍCH YÊU CẦU

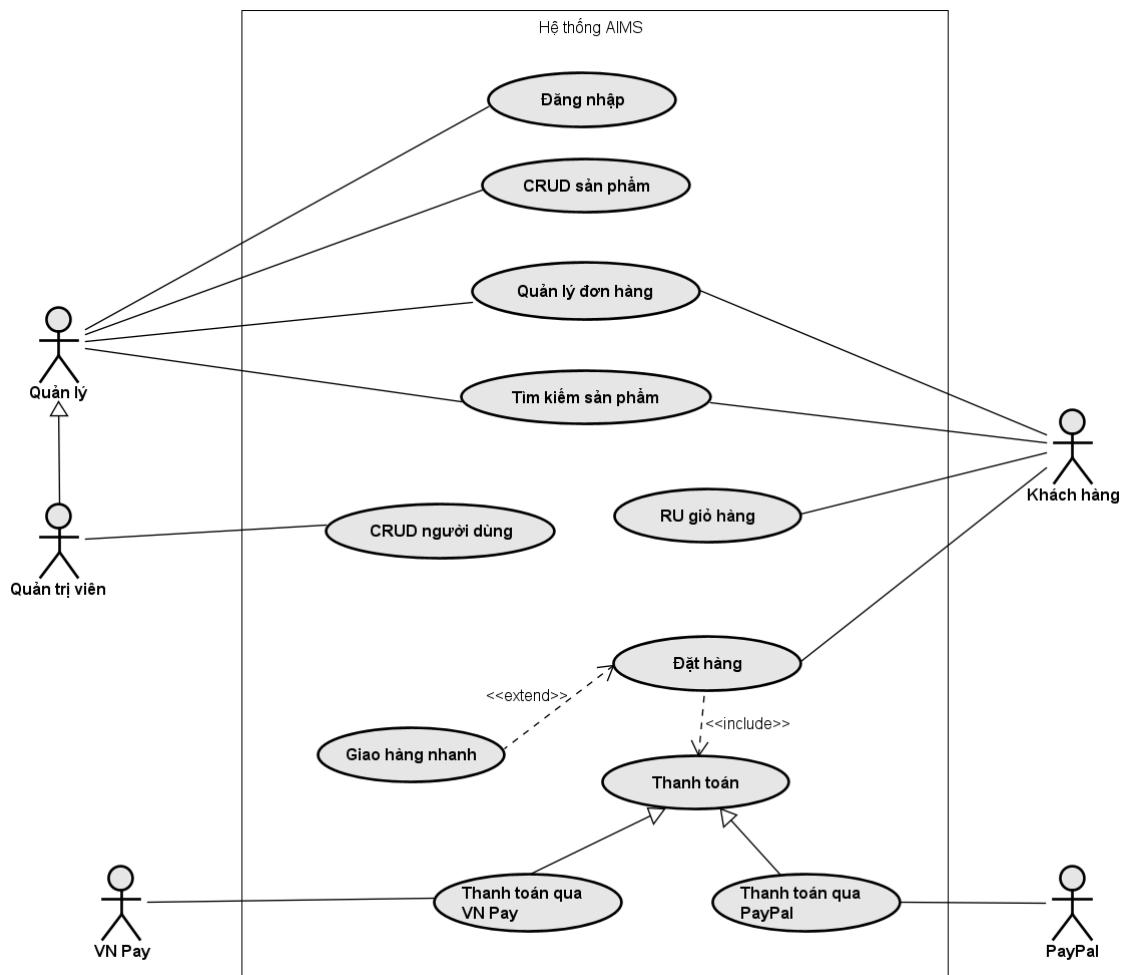
2.1 Survey

Danh sách các tác nhân và mô tả

- Khách hàng: là đối tượng sử dụng chính của hệ thống. Họ sẽ lên trang web tìm kiếm các sản phẩm phù hợp và đặt mua.
- Quản trị viên: là tác nhân sẽ quản lý các tài khoản người dùng.
- Quản lý: là tác nhân quản lý sản phẩm và đơn hàng.
- VN Pay & Pay Pal: là bên thứ 3 cung cấp dịch vụ thanh toán trực tuyến. Các tác nhân này sẽ giúp cho việc thanh toán trực tuyến được thực hiện một cách nhanh chóng và an toàn.

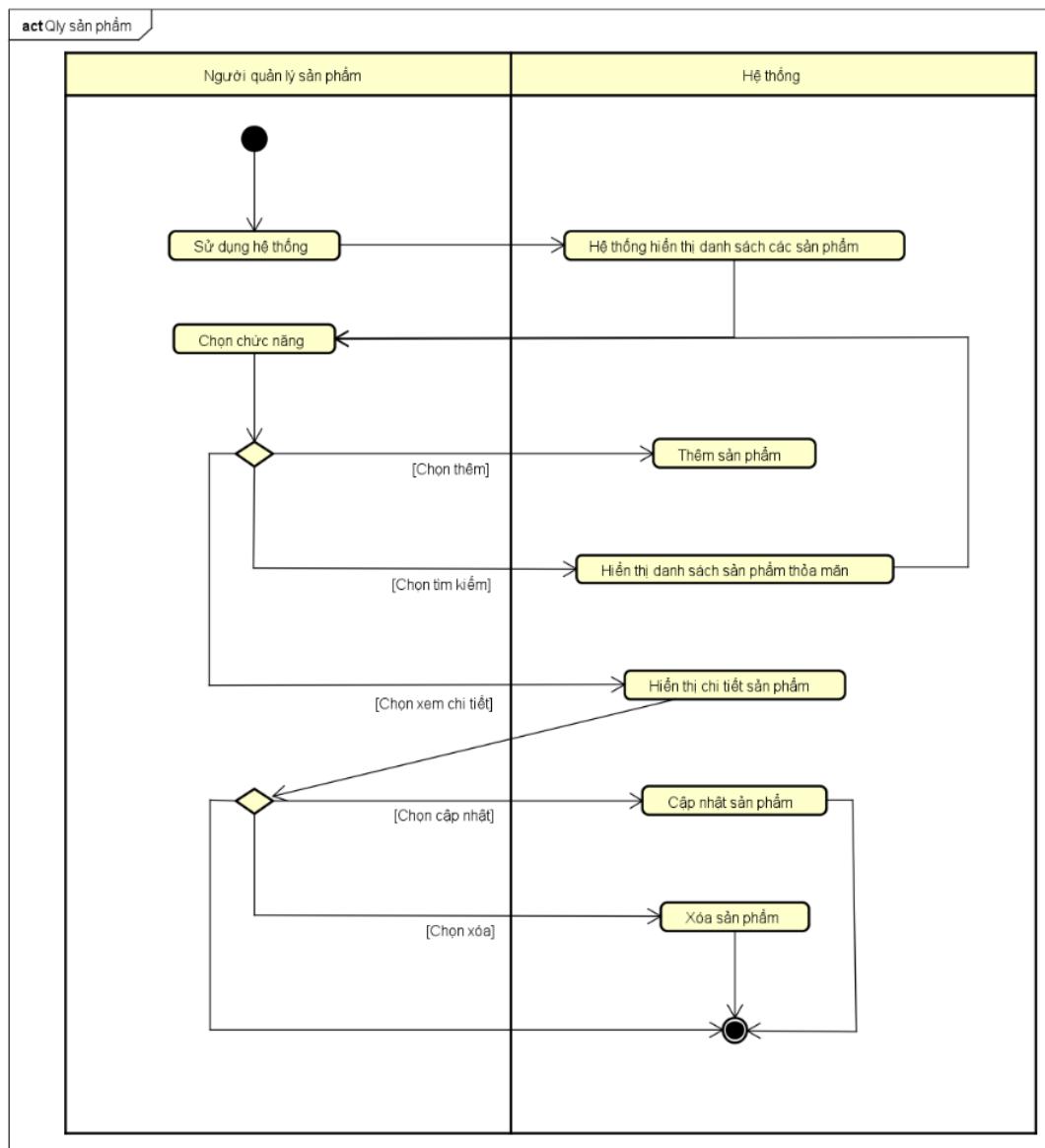
2.2 Overall requirements

2.2.1 Biểu đồ use case tổng quát

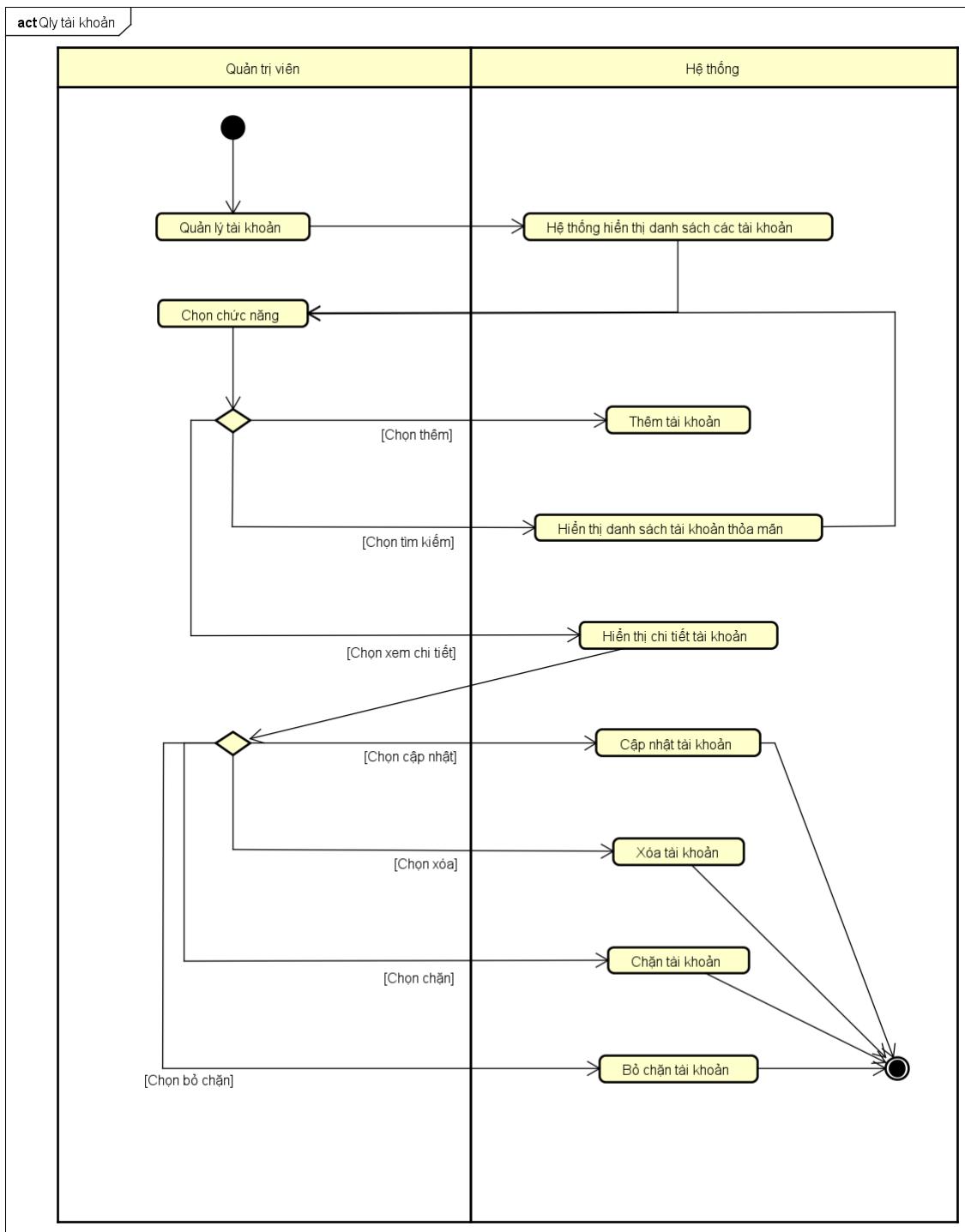


Hình 2.1: Biểu đồ use case tổng quan hệ thống

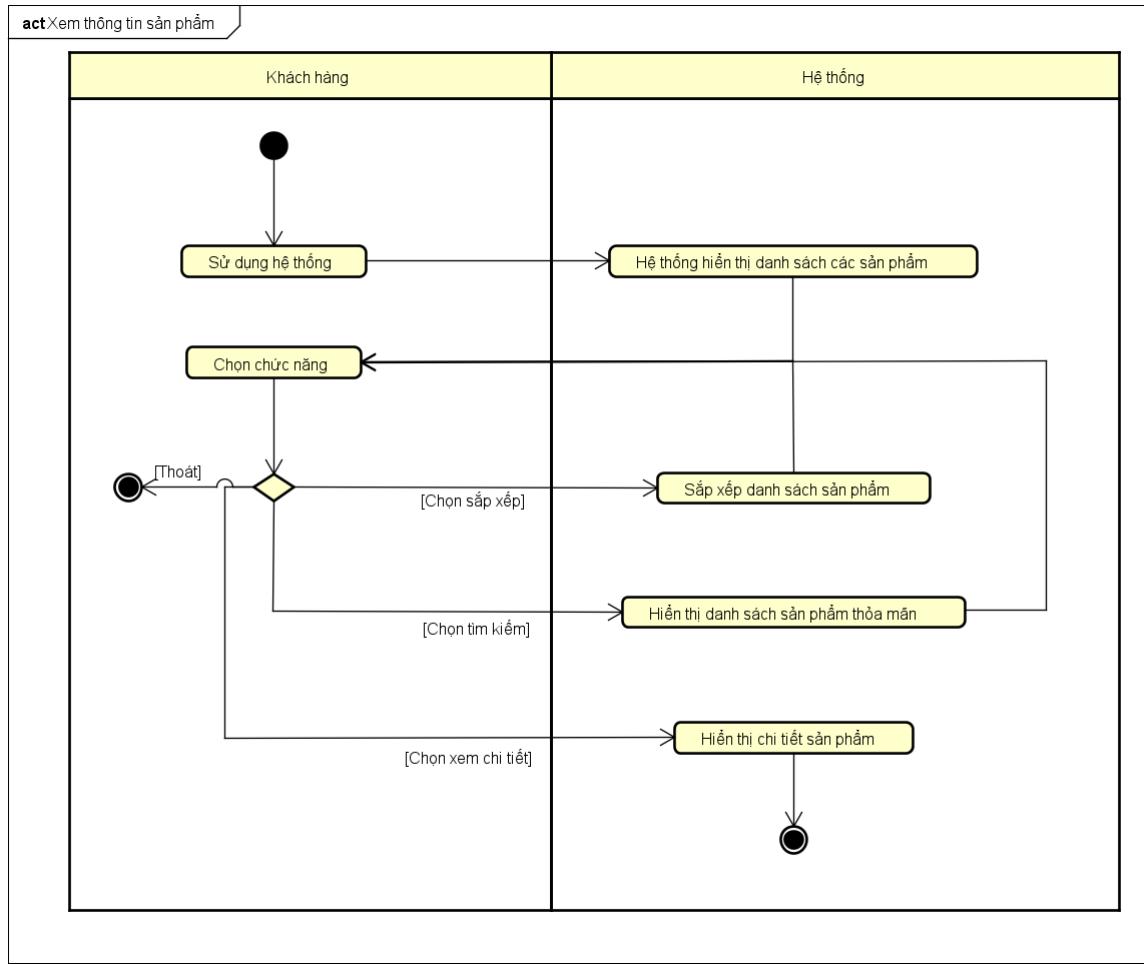
2.2.2 Biểu đồ hoạt động



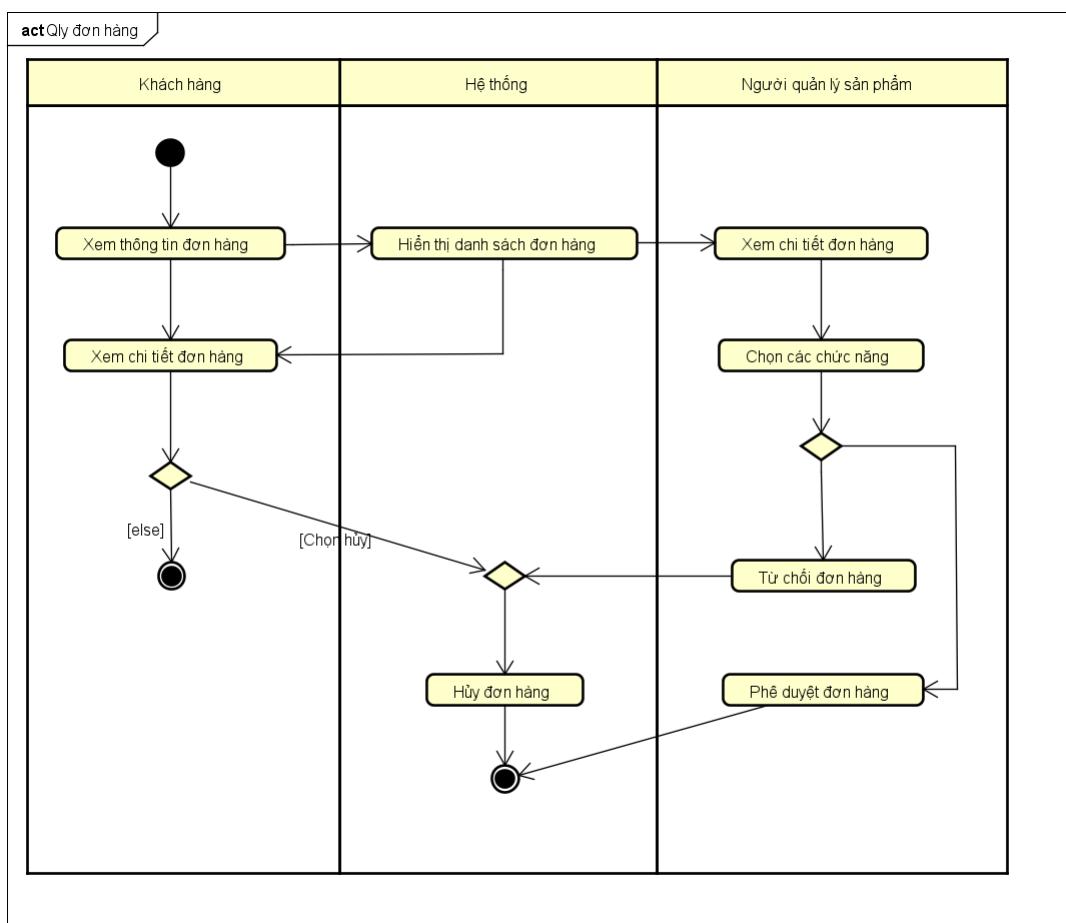
Hình 2.2: Biểu đồ hoạt động nghiệp vụ CRUD sản phẩm



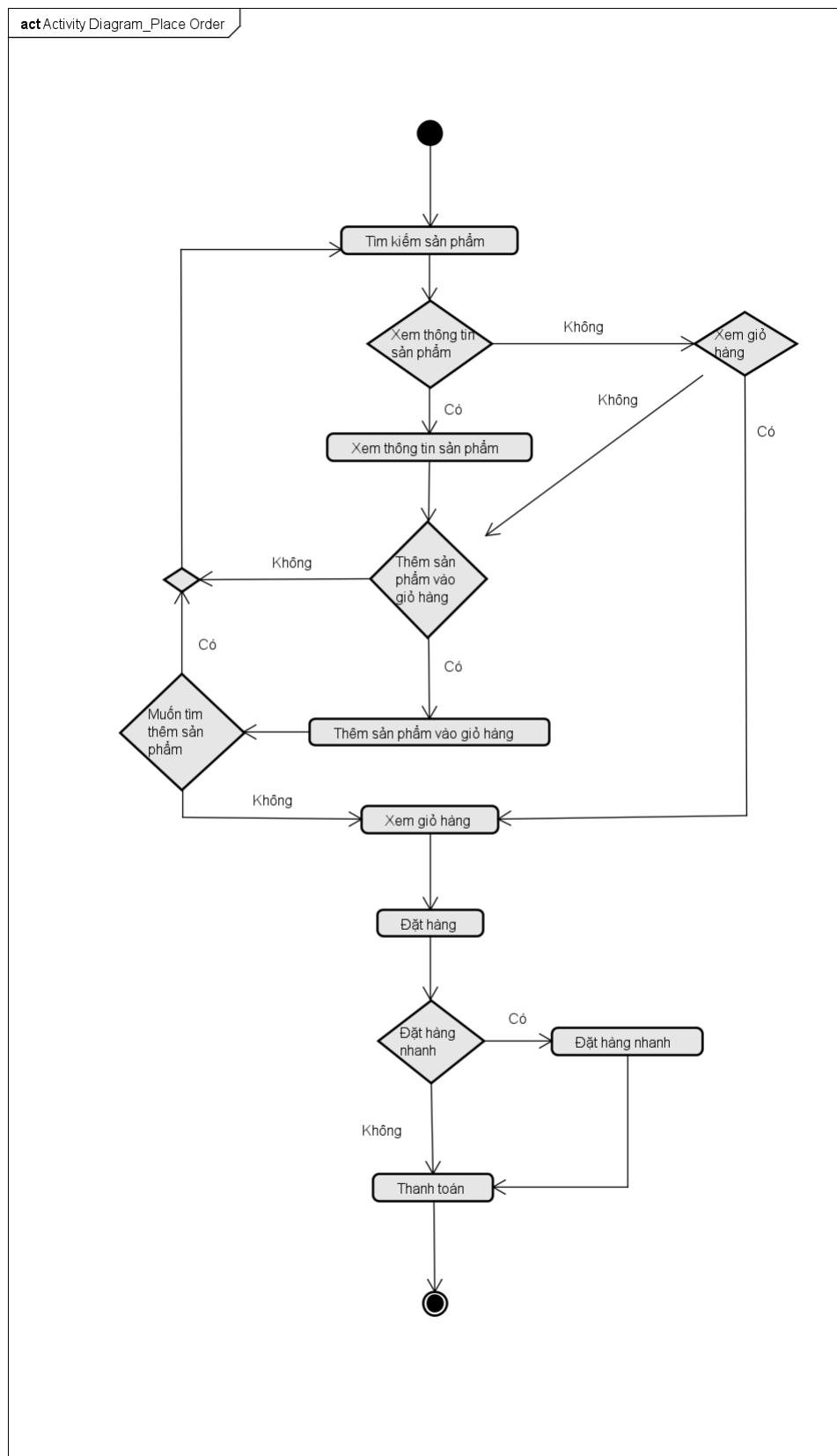
Hình 2.3: Biểu đồ hoạt động nghiệp vụ CRUD người dùng



Hình 2.4: Biểu đồ hoạt động nghiệp vụ Tìm kiếm sản phẩm



Hình 2.5: Biểu đồ hoạt động nghiệp vụ Quản lý đơn hàng



Hình 2.6: Biểu đồ hoạt động nghiệp vụ đặt hàng

CHƯƠNG 3. PHÂN TÍCH YÊU CẦU

3.1 Đặc tả ca sử dụng

3.1.1 Use case “Quản lý sản phẩm”

a, Use case code

UC001

b, Brief Description

Use case mô tả sự tương tác giữa quản trị viên và hệ thống khi quản trị viên muốn thực hiện các thao tác CRUD trên sản phẩm

c, Actors

Người quản lý

d, Preconditions

Tài khoản đăng nhập phải là tài khoản quản lý

e, Basic Flow of Events

1. View:

- (a) Hệ thống hiển thị danh sách sản phẩm
- (b) Admin yêu cầu xem chi tiết sản phẩm
- (c) Hệ thống hiển thị chi tiết sản phẩm

2. Add:

- (a) Quản trị viên lựa chọn chức năng thêm sản phẩm
- (b) Hệ thống yêu cầu quản trị viên cung cấp thông tin về sản phẩm
- (c) Quản trị viên nhập các thông tin cần thiết
- (d) Hệ thống lưu sản phẩm và lịch sử hoạt động

3. Edit:

- (a) Quản trị viên tìm kiếm sản phẩm cần chỉnh sửa từ danh sách sản phẩm
- (b) Quản trị viên lựa chọn xem chi tiết sản phẩm cần chỉnh sửa
- (c) Hệ thống hiển thị chi tiết sản phẩm
- (d) Quản trị viên lựa chọn chỉnh sửa thông tin sản phẩm
- (e) Hệ thống yêu cầu người quản trị nhập các thông tin cần chỉnh sửa
- (f) Quản trị viên cung cấp thông tin cần chỉnh sửa

CHƯƠNG 3. PHÂN TÍCH YÊU CẦU

(g) Hệ thống lưu thông tin mới cho sản phẩm

4. Delete:

- (a) Quản trị viên tìm kiếm sản phẩm cần xóa khỏi danh sách sản phẩm
- (b) Quản trị viên chọn xem chi tiết sản phẩm cần xóa
- (c) Hệ thống hiển thị chi tiết sản phẩm
- (d) Quản trị viên chọn xóa sản phẩm
- (e) Hệ thống xóa sản phẩm

f, Alternative flows

No	Pos.	Condition	Action	Cont. at
1	2c	Nếu người quản lý sản phẩm nhập sai	Thông báo lỗi hệ thống: Cần nhập các trường bắt buộc hoặc đúng định dạng	2b
2	3f	Nếu người quản lý sản phẩm nhập sai	Thông báo lỗi hệ thống: Cần nhập các trường bắt buộc hoặc đúng định dạng	3e

g, Input data

STT	Trường dữ liệu	Mô tả	Bắt buộc	Điều kiện hợp lệ	Ví dụ
1	Id của sản phẩm	Id của sản phẩm	Có	Là một chuỗi	65771b4f73453a1fe3ac9de6
2	Title	Tiêu đề của sản phẩm	Có	-	Harry Potter
3	Category	Thể loại của sản phẩm	Có	-	Fantasy
4	Price	Giá của sản phẩm	Có	-	612
5	Import Date	Ngày nhập hàng	Có	-	2023-12-11
6	Quantity	Số lượng hàng	Có	-	12
7	Description	Mô tả về sản phẩm	Có	-	Thrilling stories about HP
8	Product Dimensions	Kích thước sản phẩm	Có	-	height: 22 width: 12 length: 12 weight: 5
9	Barcode	Mã vạch của sản phẩm	Có	-	12345789
10	Support Rush	Hỗ trợ giao hàng nhanh	Có	-	yes

h, Output data

Field	Value
Message	Created Product Successfully

i, Postconditions

Return message success and status code 200

3.1.2 Use case “Thanh toán”

a, Mã use case

UC001

b, Mô tả

Use case mô tả chức năng thanh toán của khách hàng

c, Actors

Khách hàng

d, Tiền điều kiện

Khách hàng đã tạo thành công đơn hàng với đầy đủ thông tin theo yêu cầu

e, Luồng sự kiện chính

1. Hệ thống hiển thị màn hình thanh toán bao gồm thông tin đơn hàng thông tin giao hàng và các phương thức thanh toán
2. Khách hàng chọn phương thức thanh toán
3. Hệ thống điều hướng màn hình sang màn hình thanh toán của bên cung cấp dịch vụ thanh toán
4. Khách hàng nhập các thông tin theo yêu cầu
5. Khách hàng kiểm tra và xác nhận giao dịch
6. Hệ thống điều hướng màn hình về trang thông báo giao dịch thành công và gửi mail kèm đường link thông tin đơn hàng đến địa chỉ mail trên đơn hàng

f, Luồng sự kiện thay thế

No	Pos.	Condition	Action	Cont. at
1	2	Khách hàng muốn hủy đơn hàng	Hệ thống cập nhật trạng thái đơn hàng sang đã hủy	Kết thúc

g, Input data

Tên trường	Giá trị
Phương thức thanh toán	Paypal

h, Output data

Tên trường	Giá trị
Message	Paid Order Successfully

i, Hậu điều kiện

Return message success and status code 200

3.1.3 Use case “RU Giỏ hàng”

a, Use case code

UC003

b, Brief Description

Use case mô tả ngắn gọn chức năng đọc và cập nhật giỏ hàng

c, Actor

Khách hàng

d, Preconditions

Không

e, Luồng sự kiện chính

1. Xem giỏ hàng :

- (a) Khách hàng thêm sản phẩm vào giỏ hàng
- (b) Khi khách hàng bấm vào nút giỏ hàng
- (c) Hệ thống sẽ hiển thị màn hình các sản phẩm trong giỏ hàng

2. Cập nhật giỏ hàng :

- (a) Sau khi xem giỏ hàng
- (b) Khách hàng cập nhật số lượng trong giỏ hàng
- (c) Hệ thống ghi nhận số lượng mới sau khi cập nhật

f, Luồng sự kiện thay thế

No	Pos.	Condition	Action	Cont. at
1	2c	Số lượng trong kho không đủ	Hệ thống thông báo số lượng còn lại	Quay lại 2b

g, Input data

STT	Trường dữ liệu	Mô tả	Bắt buộc	Điều kiện hợp lệ	Ví dụ
1	Id của sản phẩm	Id của sản phẩm	Có	Là một chuỗi	65771b4f73453a1fe3ac9de6
2	Số lượng	Số lượng sản phẩm cần đặt	Có	Bao gồm các chữ số từ 0->9	10

h, Output data

STT	Trường dữ liệu	Mô tả	Bắt buộc	Điều kiện hợp lệ	Ví dụ
1	Tên sản phẩm	Id của sản phẩm	Có	Là một chuỗi	65771b4f73453a1fe3ac9de6
2	Số lượng sản phẩm	Số lượng sản phẩm	Có	Bao gồm các chữ số từ 0->9	10
3	Tổng tiền	Tổng tất tiền tất cả các sản phẩm	Có	Bao gồm các chữ số từ 0->9	100000
3	Tiền thuế	Thuế cho tất cả sản phẩm	Có	Bao gồm các chữ số từ 0->9	100

i, Postconditions

Không có

3.1.4 Use case “Đặt hàng”

a, Use case code

UC004

b, Brief Description

Use case mô tả tương tác giữa khách hàng và hệ thống khi quản trị viên muốn đặt hàng

c, Actor

Khách hàng

d, Preconditions

Không

e, Luồng sự kiện chính

1. Khách hàng yêu cầu đặt hàng vào giỏ hàng
2. Phần mềm AIMS kiểm tra tình trạng còn hàng của sản phẩm trong giỏ hàng
3. Phần mềm AIMS hiển thị dạng thông tin giao hàng
4. . Phần mềm AIMS tính phí vận chuyển
5. Khách hàng xác nhận đặt hàng

6. Phần mềm AIMS gọi UC là “Pay order”
7. Phần mềm AIMS tạo đơn hàng mới
8. Phần mềm AIMS làm trống giỏ hàng
9. Phần mềm AIMS gửi mail cho người dùng đặt hàng thành công

f, Luồng sự kiện thay thế

No	Pos.	Condition	Action	Cont. at
1	3	Nếu số lượng hàng tồn kho không đủ cho một hoặc nhiều sản phẩm	<ul style="list-style-type: none"> ▷ Hệ thống thông báo về số lượng sản phẩm thiếu và yêu cầu khách hàng cập nhật giỏ hàng ▷ Khách hàng cập nhật giỏ hàng với số lượng sản phẩm phù hợp 	Quay lại 2
2	5	Nếu thông tin giao hàng sai sai	<ul style="list-style-type: none"> ▷ Thông báo thông tin giao hàng không đúng 	Quay lại 3
3	5	Nếu chọn giao hàng nhanh	<ul style="list-style-type: none"> ▷ Hệ thống sẽ thêm use case giao hàng nhanh 	Quay lại 5

g, Input data

STT	Trường dữ liệu	Mô tả	Bắt buộc	Điều kiện hợp lệ	Ví dụ
1	Id của sản phẩm	Id của sản phẩm	Có	Là một chuỗi	65771b4f73453a1fe3ac9de6
2	Số lượng	Số lượng sản phẩm đặt	Có	Bao gồm các chữ số từ 0->9	10
3	Tên	Tên người đặt hàng	Có	Là chuỗi	Nguyen Van An
4	Email	Email của người đặt hàng	Có	Thỏa mãn điều kiện của 1 email	abcd@email.com
5	Số điện thoại	Số điện thoại người đặt hàng	Có	Bao gồm các chữ số từ 0->9	0988212834
6	Địa chỉ	Địa chỉ nhận hàng	Có	Là một chuỗi	4 ngõ 17 tạ quang bửu
7	Thành phố	Thành phố nơi bạn muốn nhận hàng	Có	Là chuỗi	Thành phố hà nội
8	Quận huyện	Quận huyện nơi muốn nhận hàng	Có	Là một chuỗi	Sóc sơn
9	Thông tin	Thông tin muốn giao hàng	không	Là chuỗi	Giao giờ hành chính
10	Phương thức giao hàng	Kiểu giao hàng	Có	Là chuỗi	Normal

h, Output data

STT	Trường dữ liệu	Mô tả	Bắt buộc	Điều kiện hợp lệ	Ví dụ
1	Danh sách sản phẩm	Danh sách sản phẩm đã đặt	Có	Là danh sách	Game of Thrones ,Fantasy ", 153.99,.....
2	Tổng tiền	Tổng tiền các sản phẩm với số lượng tương ứng	Có	Bao gồm các chữ số từ 0->9	100000
3	Tổng tiền thuế	Tổng thuế tất tiền tất cả các sản phẩm	Có	Bao gồm các chữ số từ 0->9	100000
4	Tiền ship	Tiền ship tính theo cân nặng	Có	Bao gồm các chữ số từ 0->9	100
5	Tổng tất cả	Tổng tiền cần thanh toán	Có	Bao gồm các chữ số từ 0->9	10000000

i, Postconditions

Không có

3.1.5 Use case “Giao hàng nhanh”

a, Use case code

UC005

b, Brief Description

Ca sử dụng này mô tả quá trình người dùng đặt giao hàng nhanh

c, Actor

Khách hàng

d, Preconditions

Thông tin đơn hàng đã được tạo và đã có địa chỉ giao hàng phù hợp

e, Luồng sự kiện chính

1. Khách hàng lựa chọn "Tùy chọn Giao hàng nhanh"
2. Hệ thống thực hiện kiểm tra để xác định xem có sản phẩm nào và địa chỉ nhận hàng nào hỗ trợ dịch vụ giao hàng nhanh

CHƯƠNG 3. PHÂN TÍCH YÊU CẦU

3. Hệ thống hiển thị một biểu mẫu cho khách hàng để họ có thể cập nhật thông tin về dịch vụ giao hàng nhanh.
4. Khách hàng cung cấp và cập nhật thông tin liên quan đến dịch vụ giao hàng nhanh
5. Hệ thống tự động cập nhật việc tính phí cho dịch vụ giao hàng ưu tiên dựa trên thông tin khách hàng đã cung cấp

f, Luồng sự kiện thay thế

Không

g, Input data

STT	Trường dữ liệu	Mô tả	Bắt buộc	Điều kiện hợp lệ	Ví dụ
1	Thời gian nhận	Thời gian muốn nhận hàng	Có	Kiểu Date	22/12/2022

h, Output data

Không

i, Postconditions

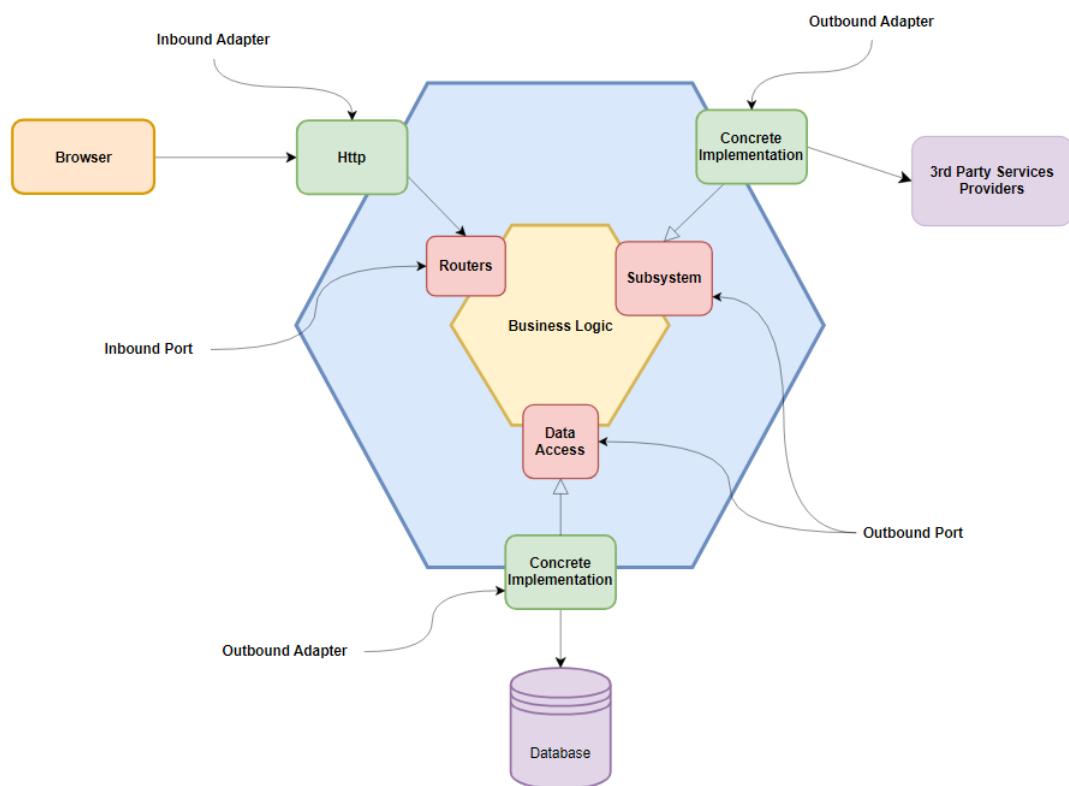
Không có

CHƯƠNG 4. KIẾN TRÚC VÀ THIẾT KẾ

4.1 Architectural Patterns

Hệ thống được xây dựng dựa trên kiến trúc hình lục giác (Hexagonal Architecture). Kiến trúc này là biến thể của kiến trúc phân tầng, kiến trúc lấp thành phần xử lý logic làm trung tâm. Thay vì tầng trình diễn, hệ thống có thể có 1 hay nhiều "inbound adapter" để xử lý các yêu cầu từ bên ngoài bằng cách gọi đến tầng xử lý logic. Tương tự hệ thống có thể có 1 hay nhiều "outbound adapter" được sử dụng bởi tầng xử lý logic và gọi đến các dịch vụ bên ngoài. Đặc thù cũng như ưu điểm của kiến trúc này là tầng xử lý logic sẽ không phải phụ thuộc vào các thành phần bên ngoài hệ thống, thay vào đó chúng phải phụ thuộc vào tầng xử lý logic.

Tầng xử lý logic có thể có 1 hay nhiều "port". Một "port" sẽ định nghĩa tập các phương thức mà tầng xử lý logic cho phép các thành phần bên ngoài tương tác với nó.



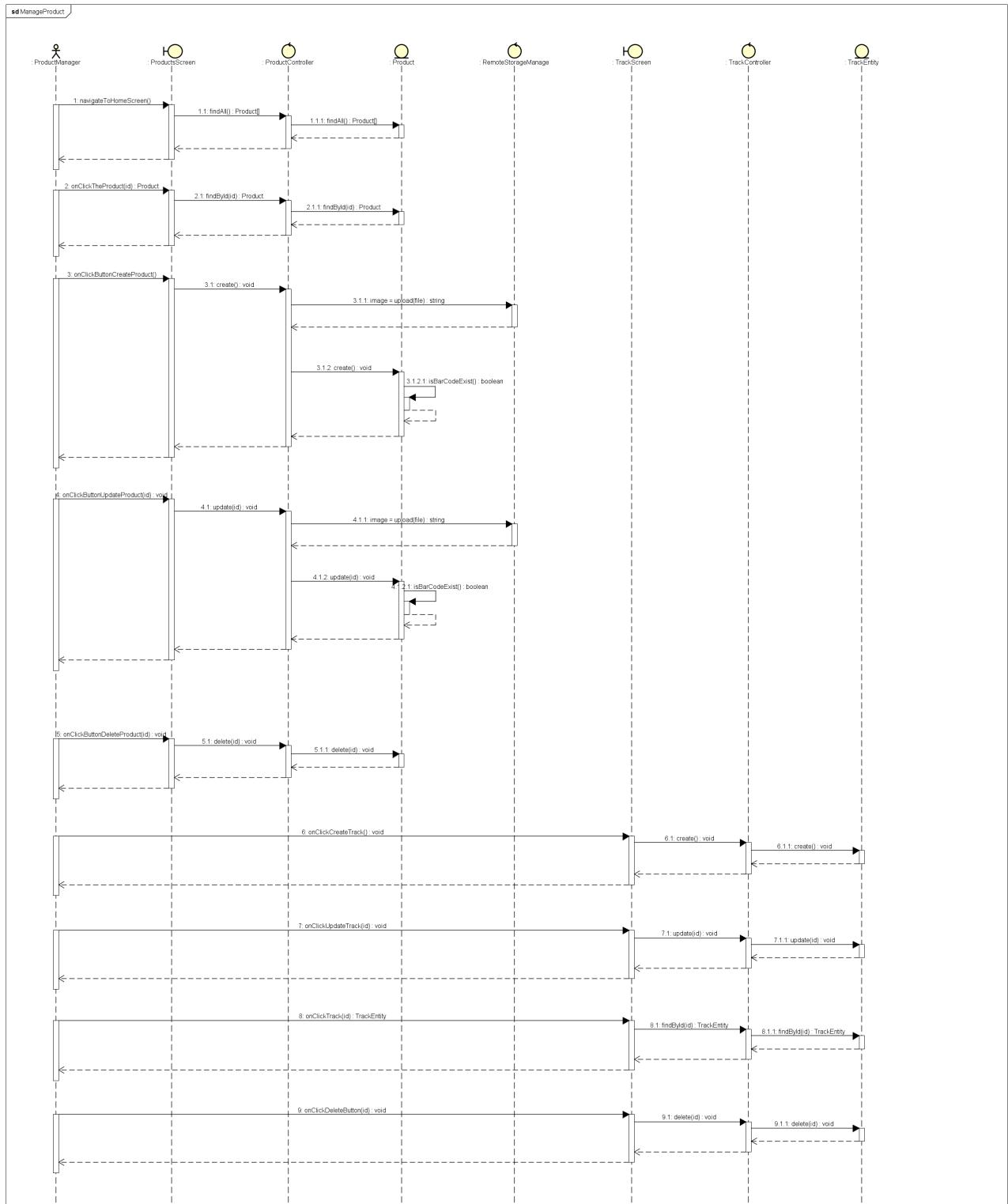
Hình 4.1: Sơ đồ kiến trúc hình lục giác

1. Routers: là nơi định nghĩa các endpoints, nhận các yêu cầu của clients
2. Subsystem: là các hệ thống con được đóng gói độc lập, giao tiếp với các bên cung cấp dịch vụ như thanh toán (paypal, vnpay), lưu trữ (firebase), gửi mail (gmail), ...

3. Data Access: là nơi giao tiếp với hệ quản trị cơ sở dữ liệu

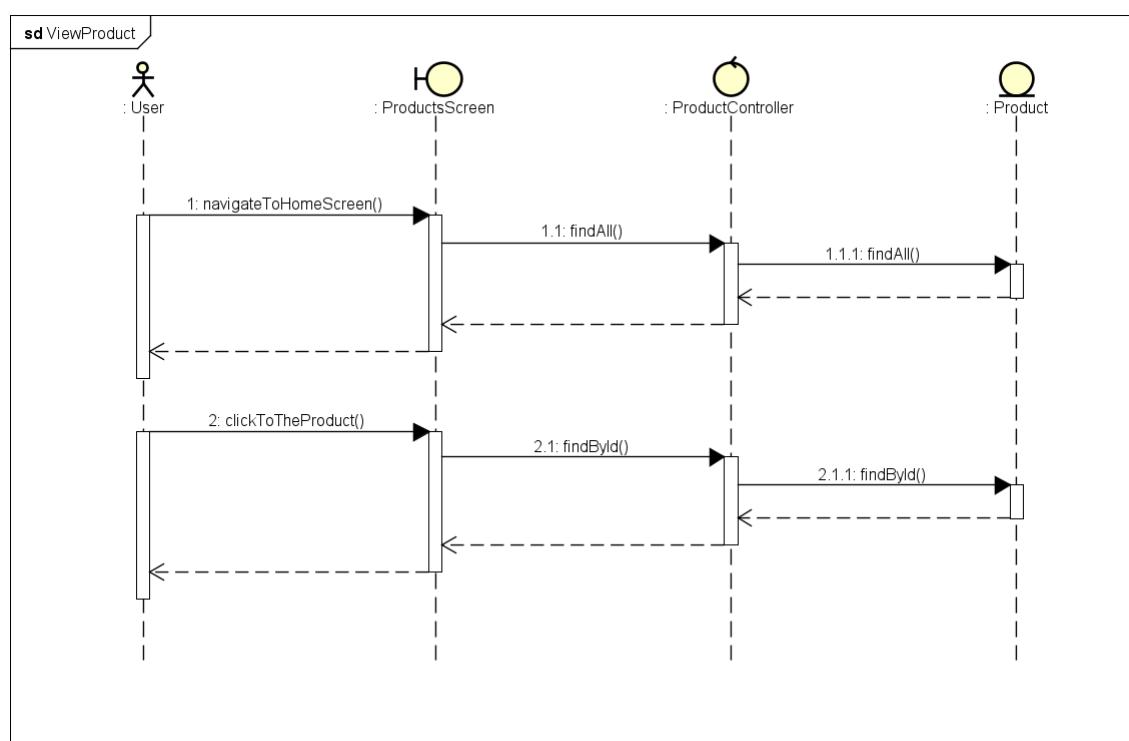
4.2 Interaction Diagrams

4.2.1 Use case "Quản lý sản phẩm"



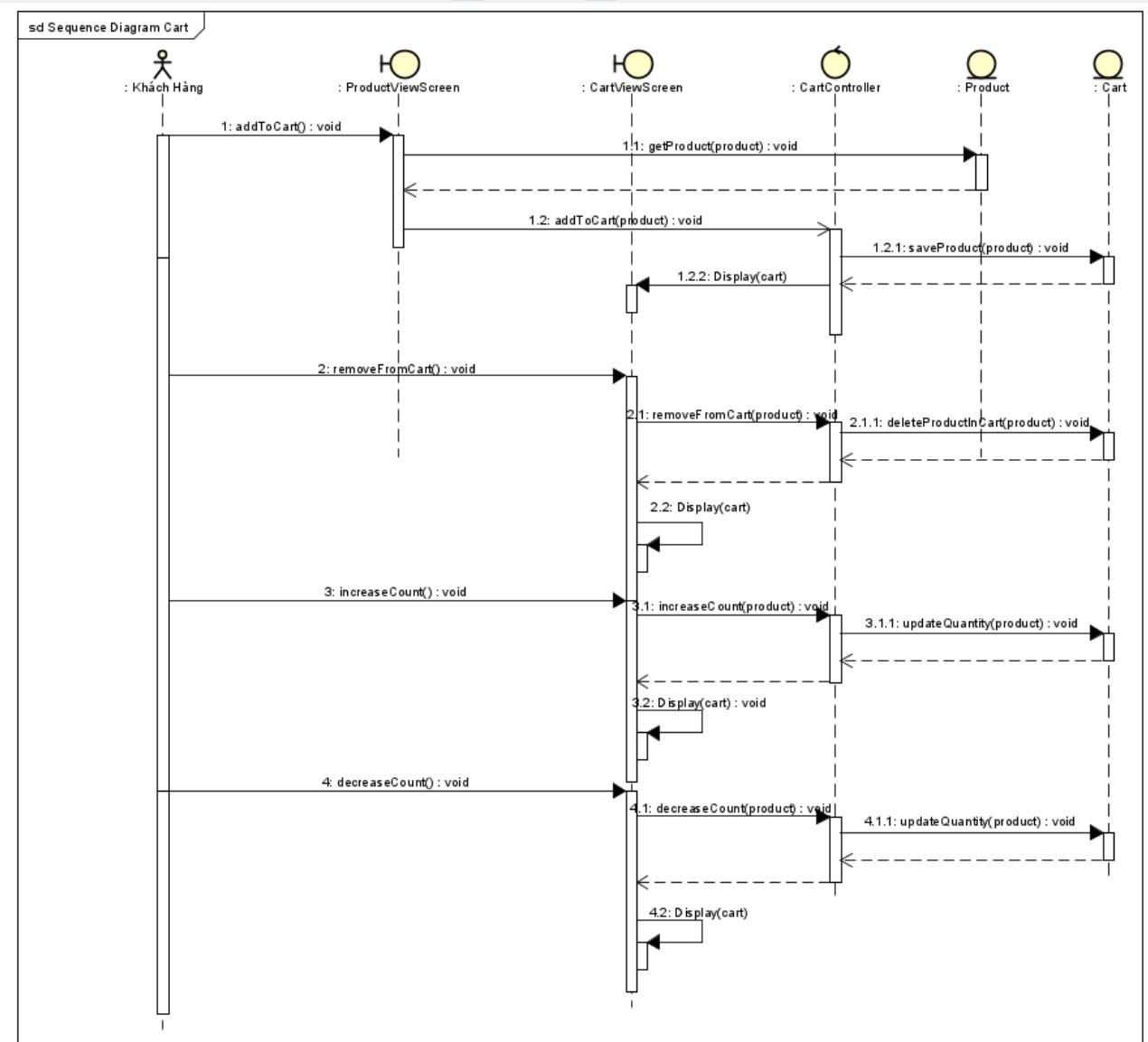
Hình 4.2: Biểu đồ tuần tự nghiệp vụ thêm xem sửa xóa sản phẩm

4.2.2 Use case "Tìm kiếm sản phẩm"



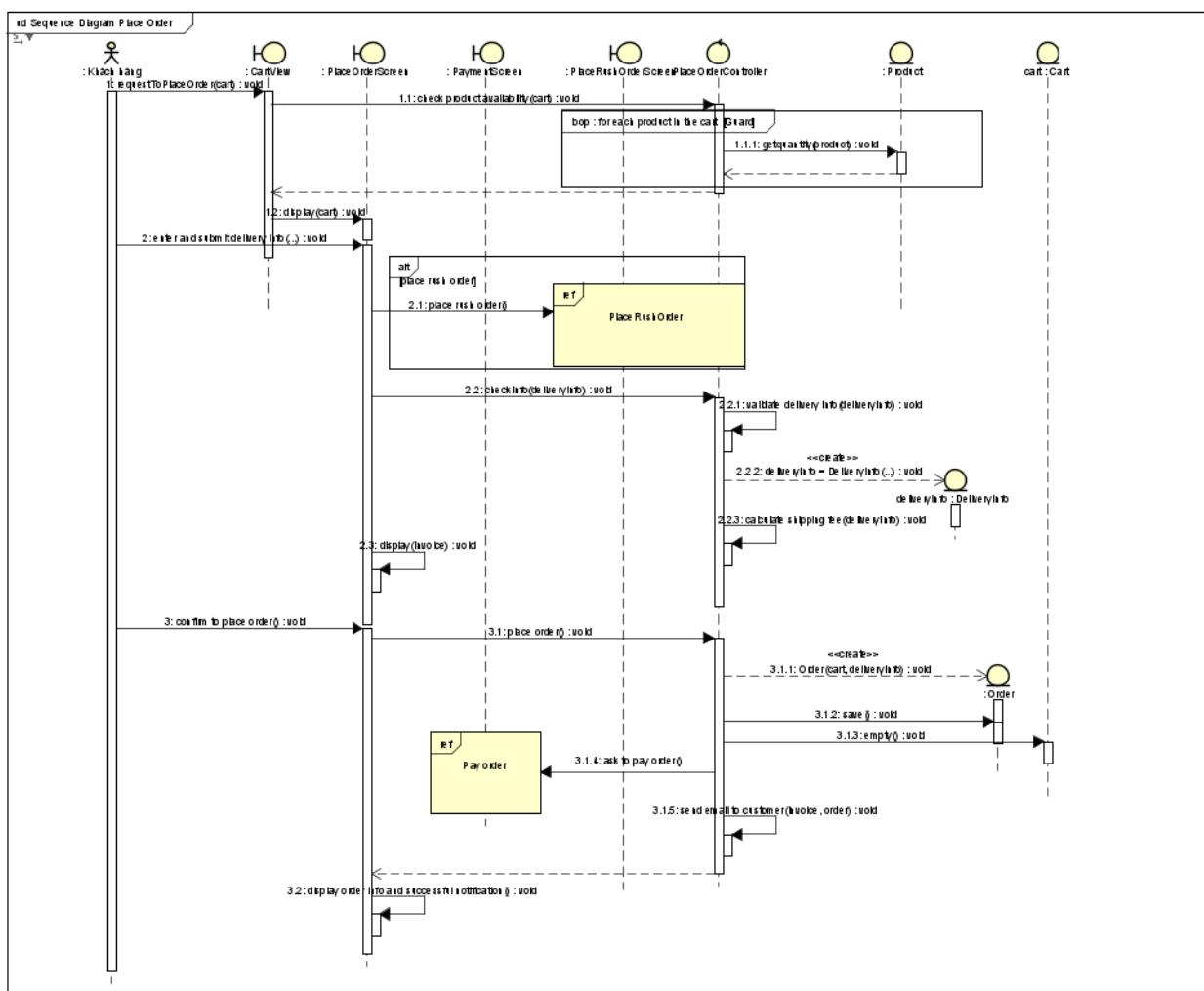
Hình 4.3: Biểu đồ tuần tự nghiệp vụ tìm kiếm sản phẩm

4.2.3 Use case "Giỏ hàng"



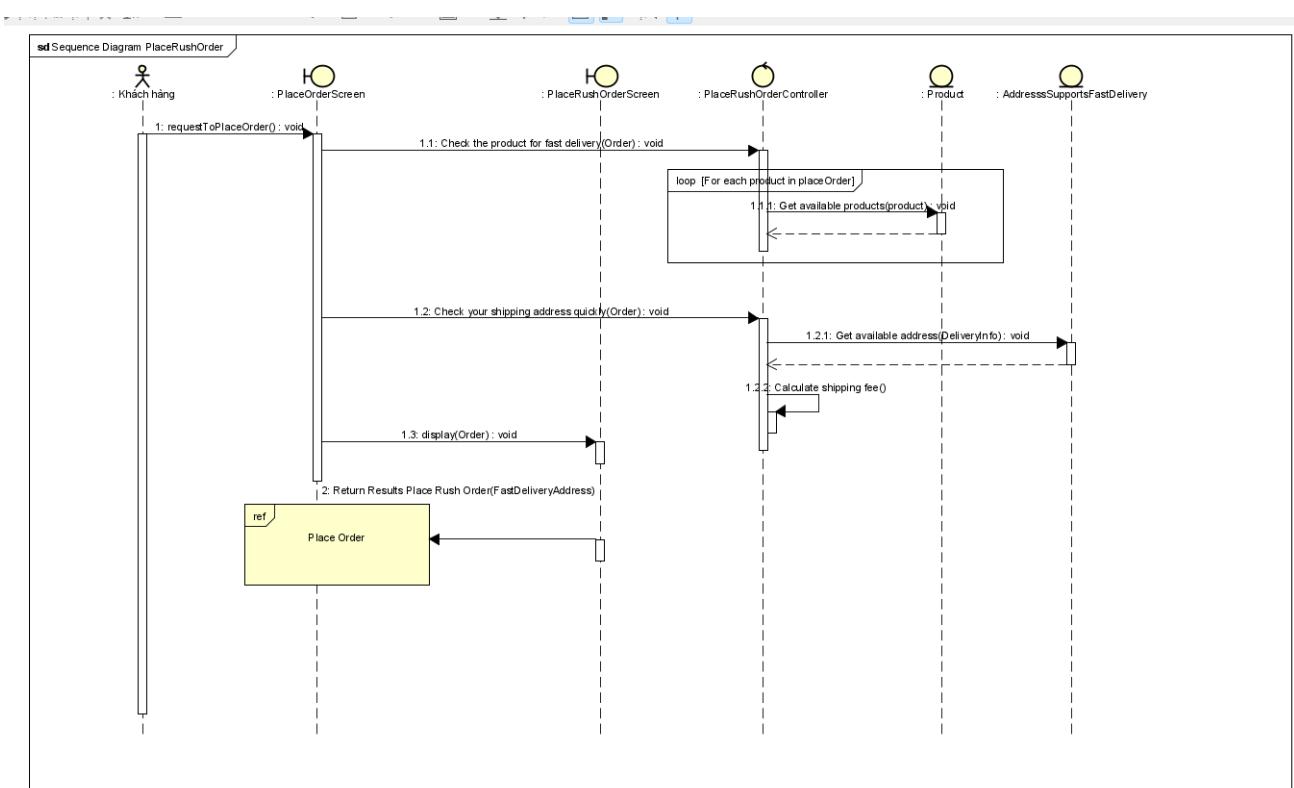
Hình 4.4: Biểu đồ tuần tự nghiệp vụ đọc hoặc cập nhật giỏ hàng

4.2.4 Use case "Đặt hàng"



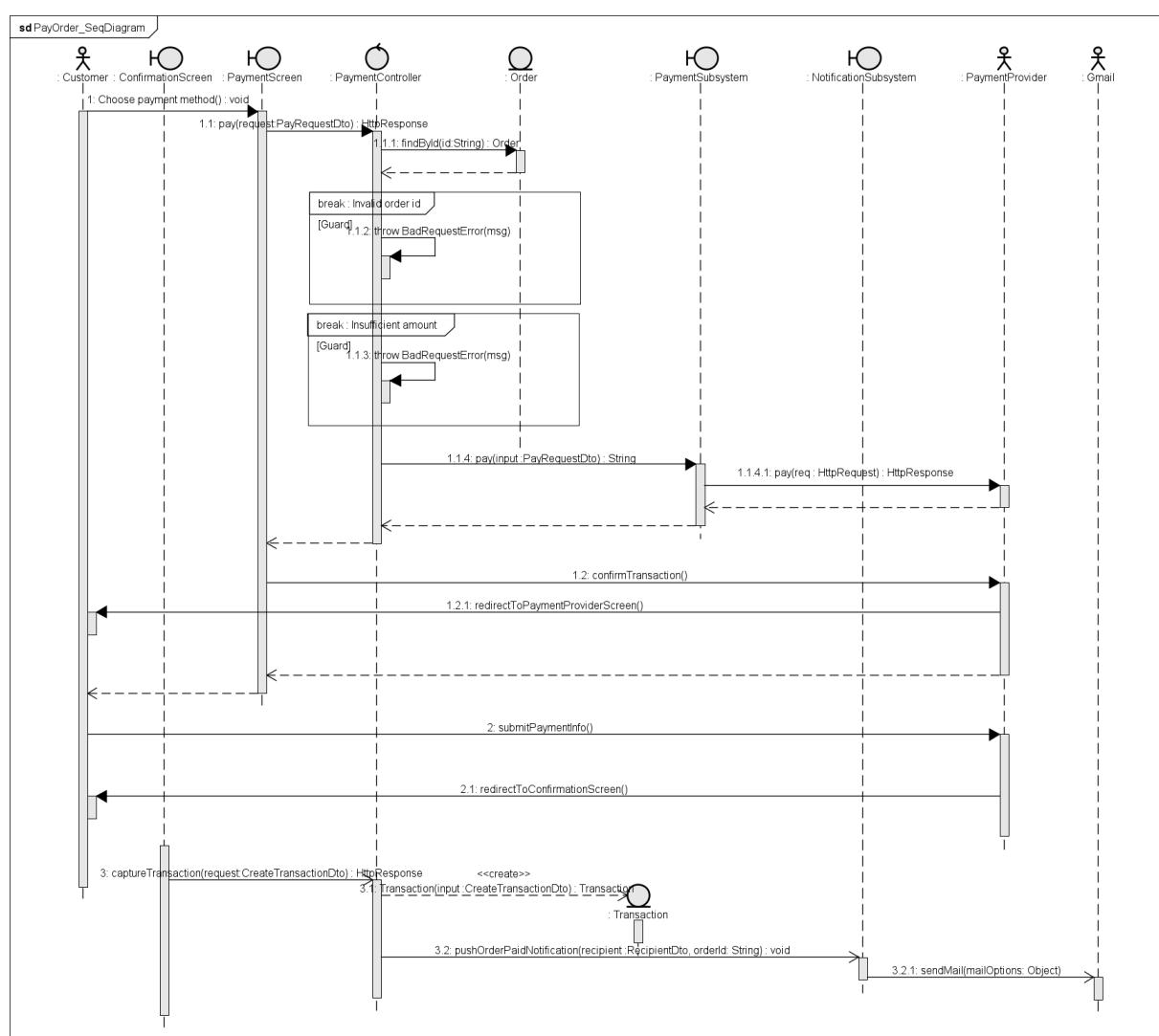
Hình 4.5: Biểu đồ tuần tự nghiệp vụ đặt hàng

4.2.5 Use case "Giao hàng nhanh"



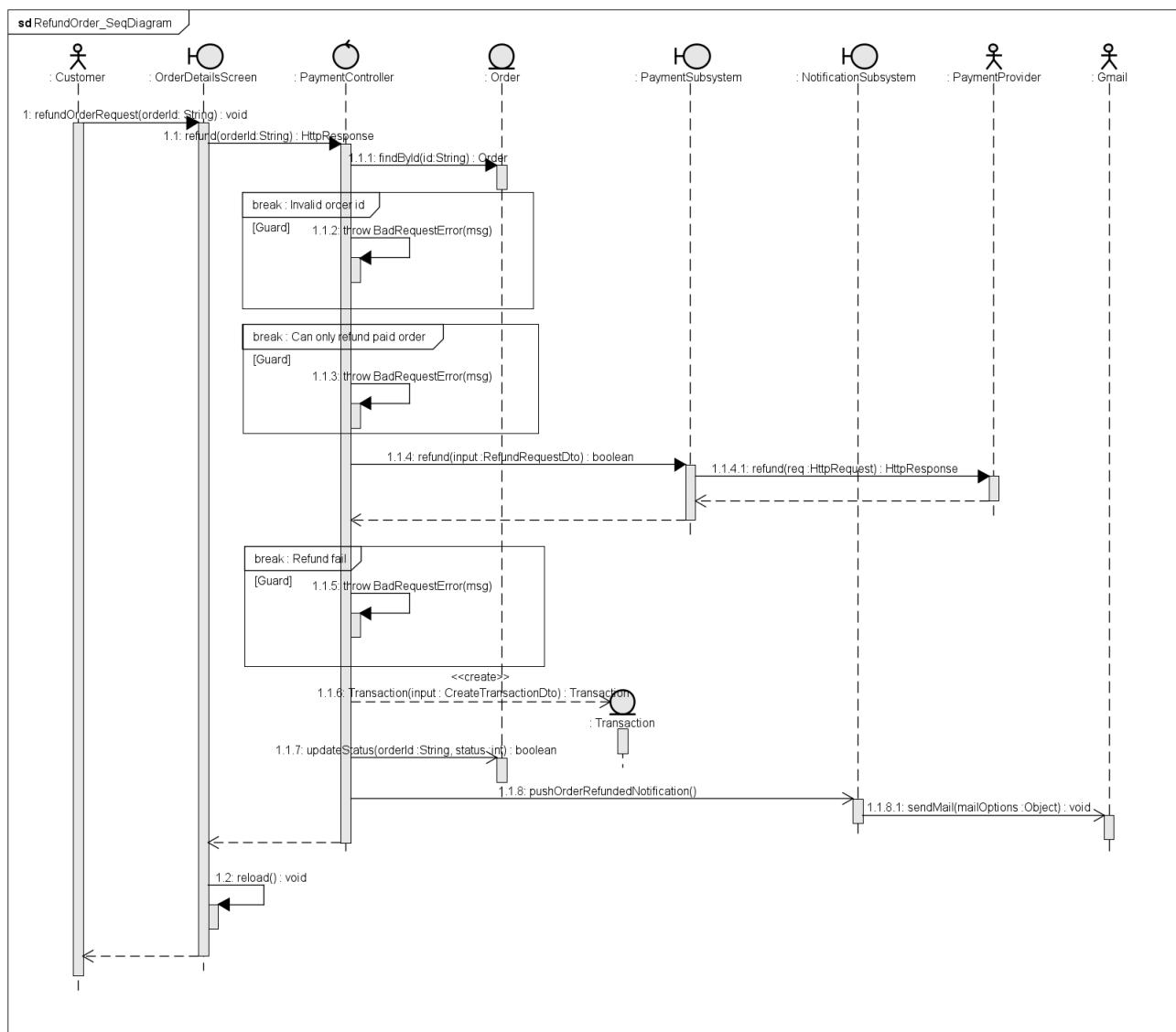
Hình 4.6: Biểu đồ tuần tự nghiệp vụ giao hàng nhanh

4.2.6 Use case "Thanh toán"



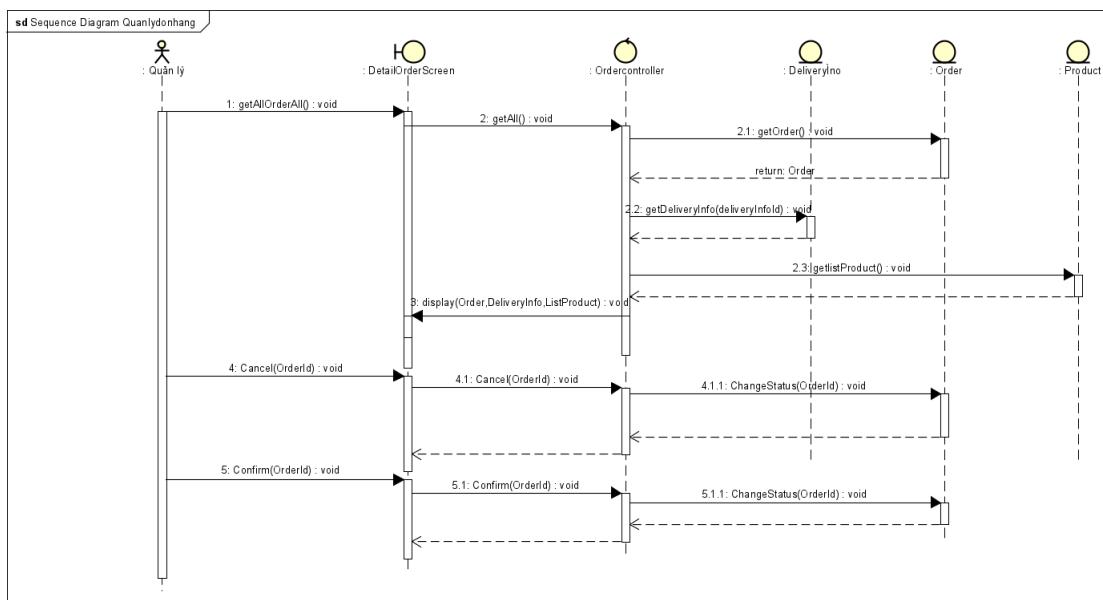
Hình 4.7: Biểu đồ tuần tự nghiệp vụ thanh toán đơn hàng

CHƯƠNG 4. KIẾN TRÚC VÀ THIẾT KẾ

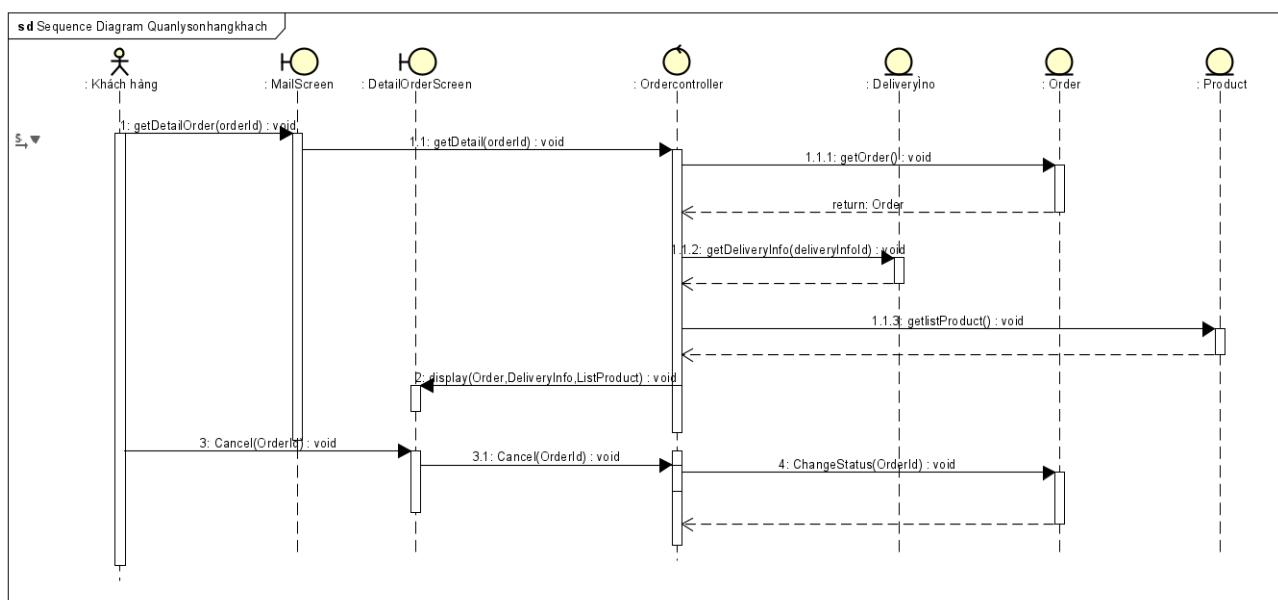


Hình 4.8: Biểu đồ tuần tự nghiệp vụ hoàn tiền đơn hàng

CHƯƠNG 4. KIẾN TRÚC VÀ THIẾT KẾ

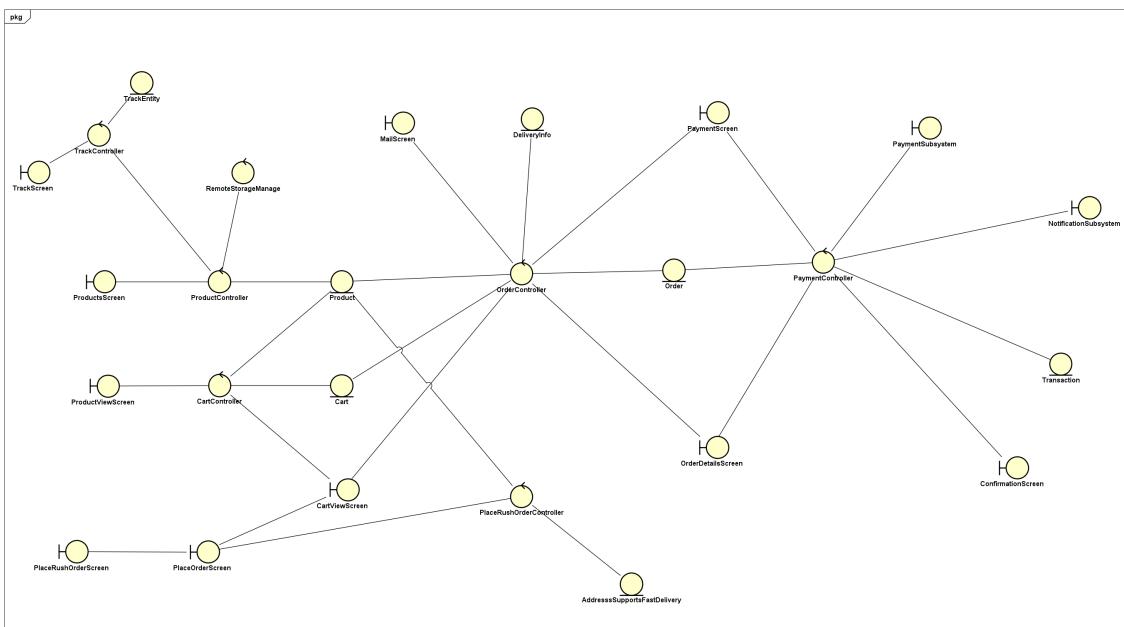


Hình 4.9: Biểu đồ tuần tự nghiệp vụ người quản lý quản lý đơn hàng



Hình 4.10: Biểu đồ tuần tự nghiệp vụ người khách hàng quản lý đơn hàng

CHƯƠNG 4. KIẾN TRÚC VÀ THIẾT KẾ

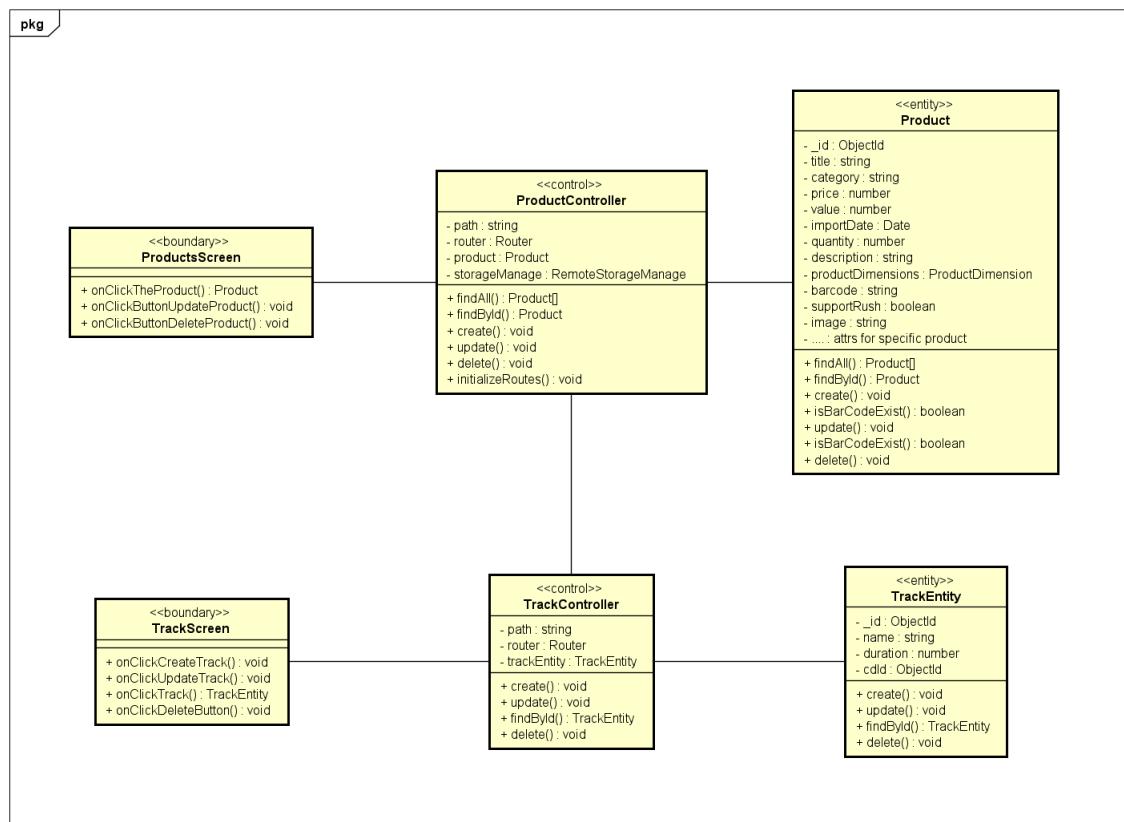


Hình 4.11: General Class Diagrams

4.3 Analysis Class Diagrams

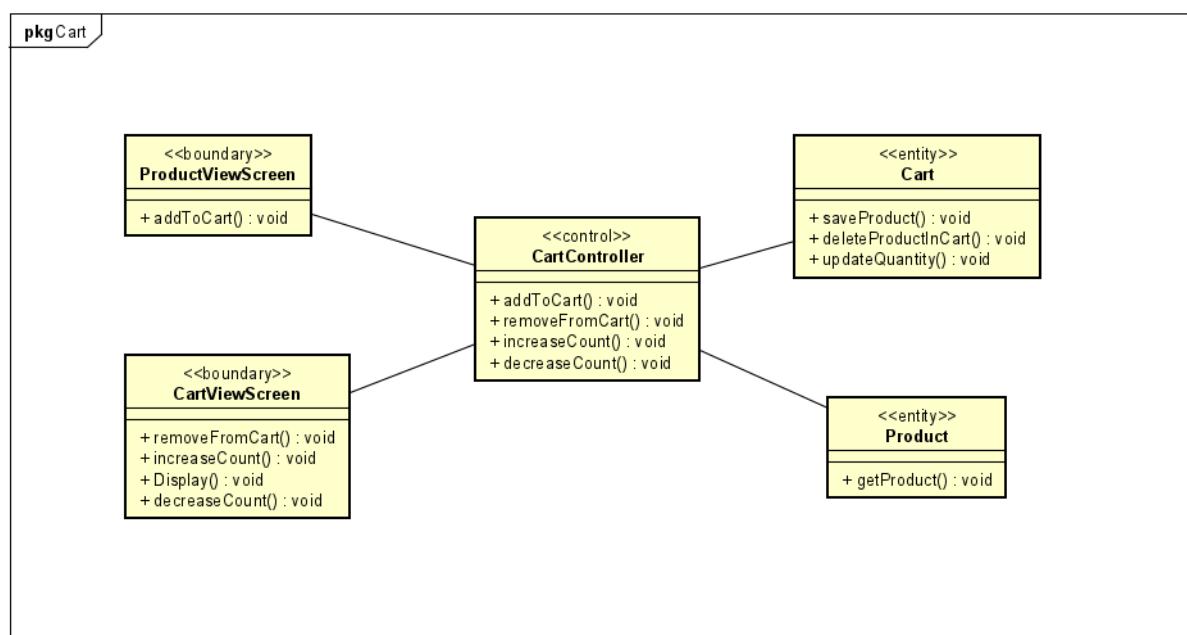
4.3.1 General Class Diagrams

4.3.2 Use case "CRUD Sản phẩm"



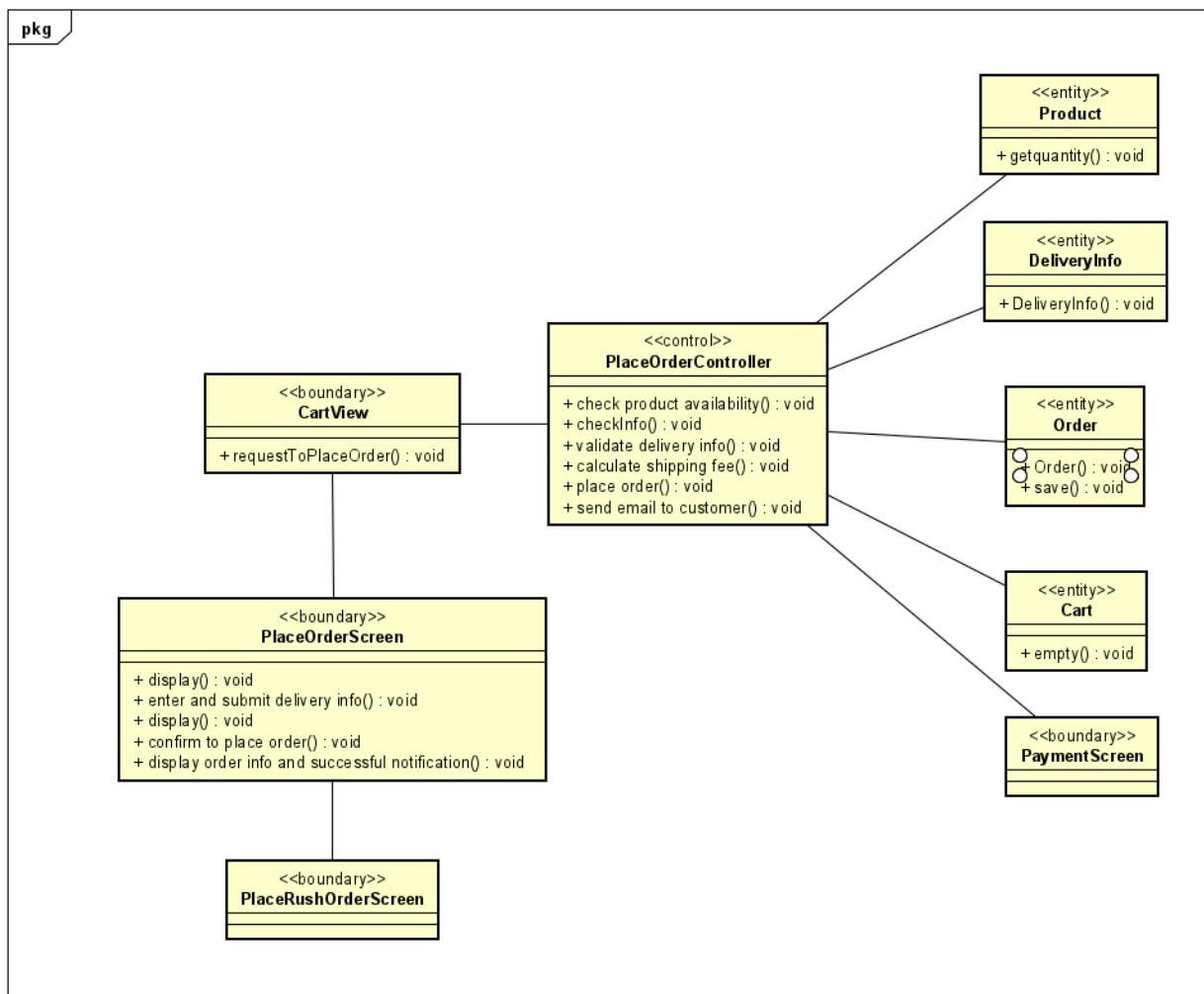
Hình 4.12: Biểu đồ lớp nghiệp vụ thêm xem sửa xóa Sản phẩm

4.3.3 Use case "Giỏ hàng "



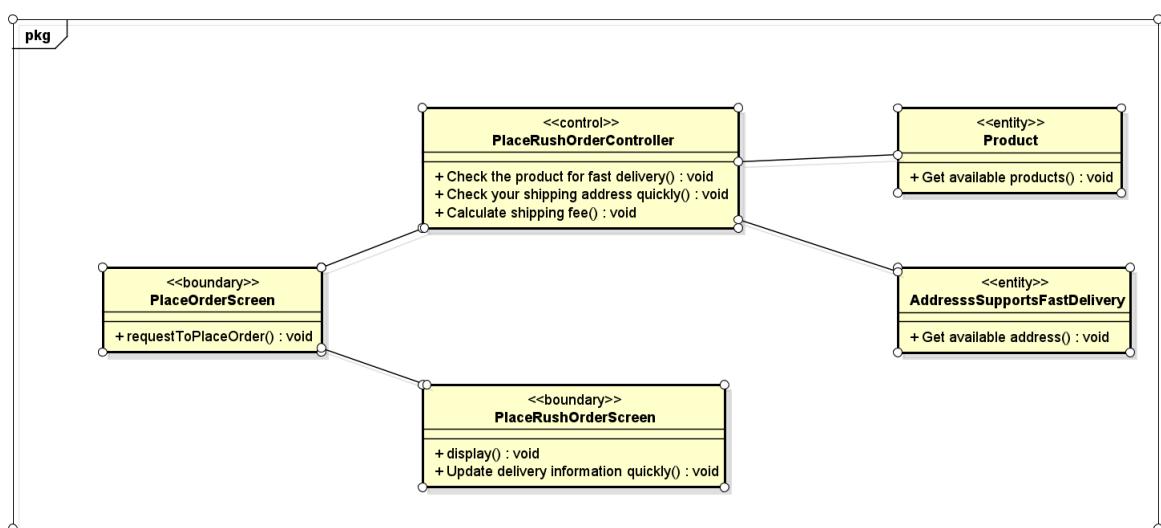
Hình 4.13: Biểu đồ lớp nghiệp vụ xem và cập nhật giỏ hàng

4.3.4 Use case "Đặt hàng "



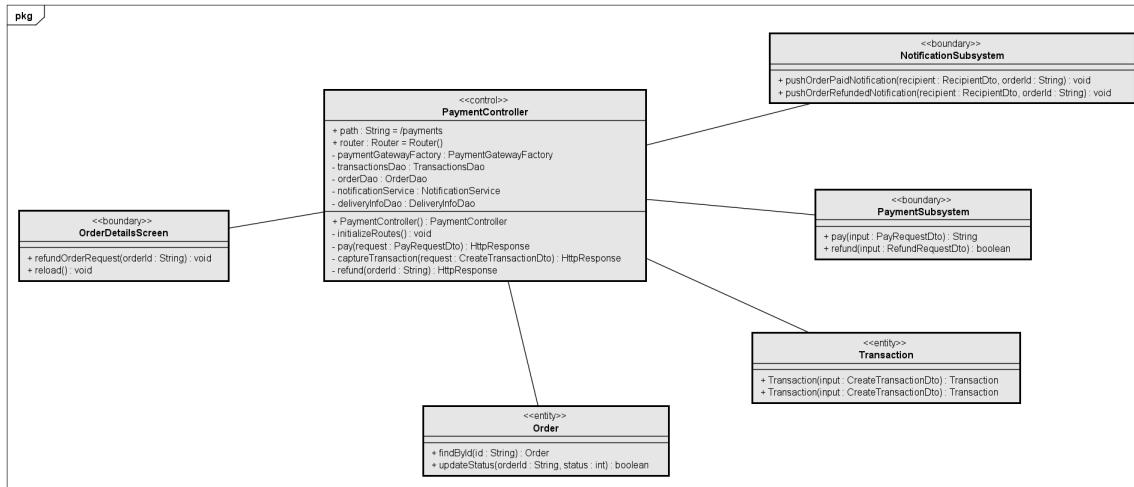
Hình 4.14: Biểu đồ lớp nghiệp vụ xem và cập nhật giỏ hàng

4.3.5 Use case "Giao hàng nhanh "



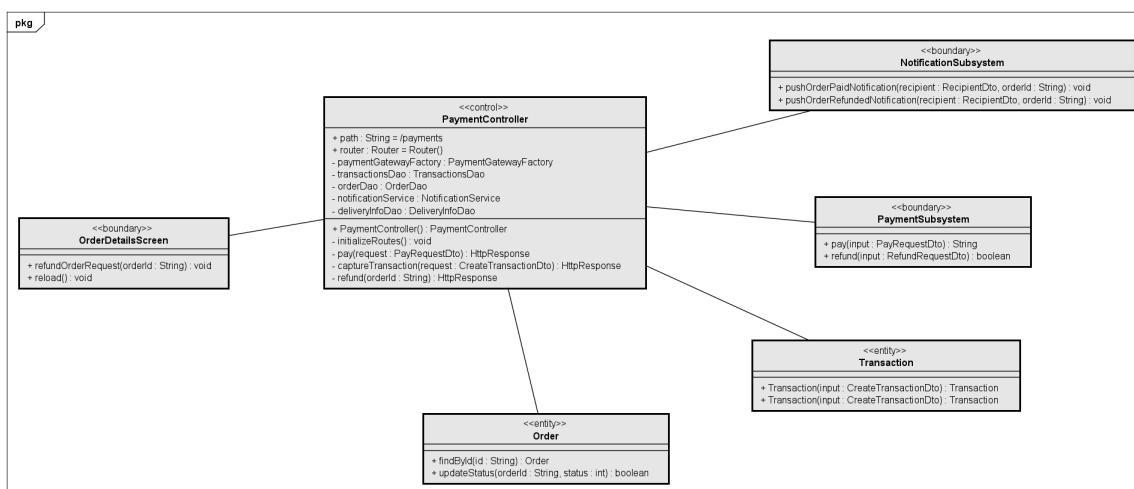
Hình 4.15: Biểu đồ lớp nghiệp vụ xem và cập nhật giỏ hàng

4.3.6 Use case "Thanh toán"



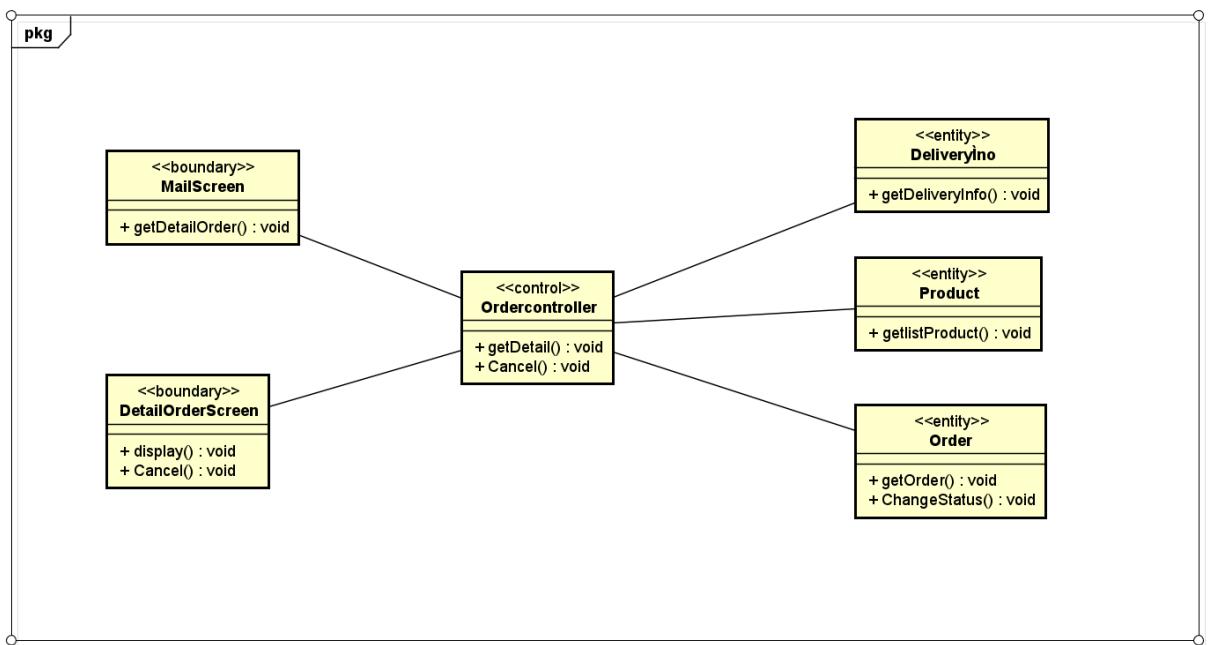
Hình 4.16: Biểu đồ lớp nghiệp vụ thanh toán đơn hàng

4.3.7 Use case "Hoàn tiền"

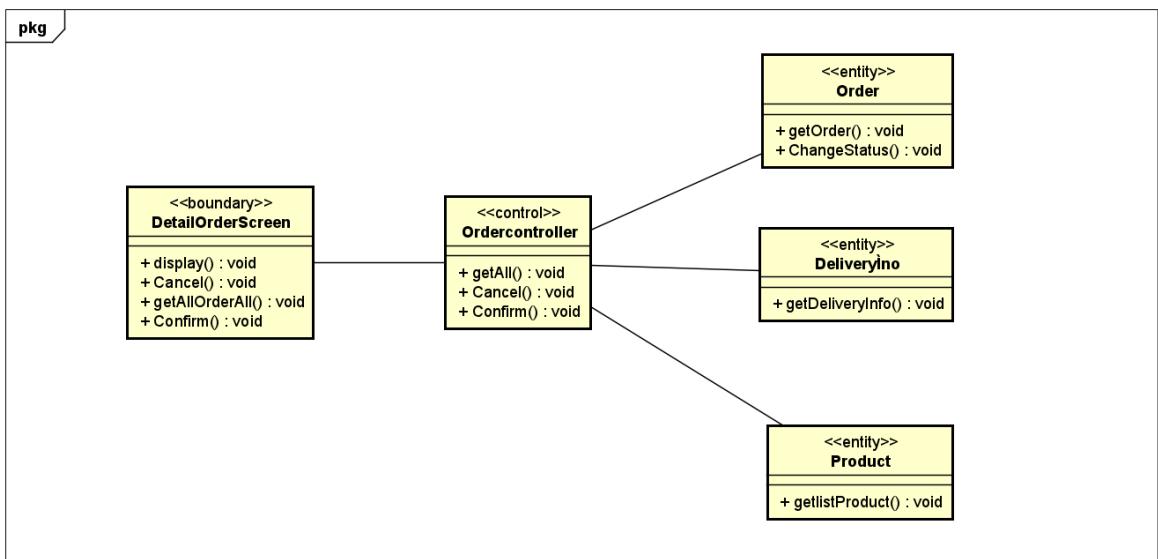


Hình 4.17: Biểu đồ lớp nghiệp vụ hoàn tiền đơn hàng

CHƯƠNG 4. KIẾN TRÚC VÀ THIẾT KẾ



Hình 4.18: Biểu đồ lớp nghiệp vụ khách hàng quản lý đơn hàng

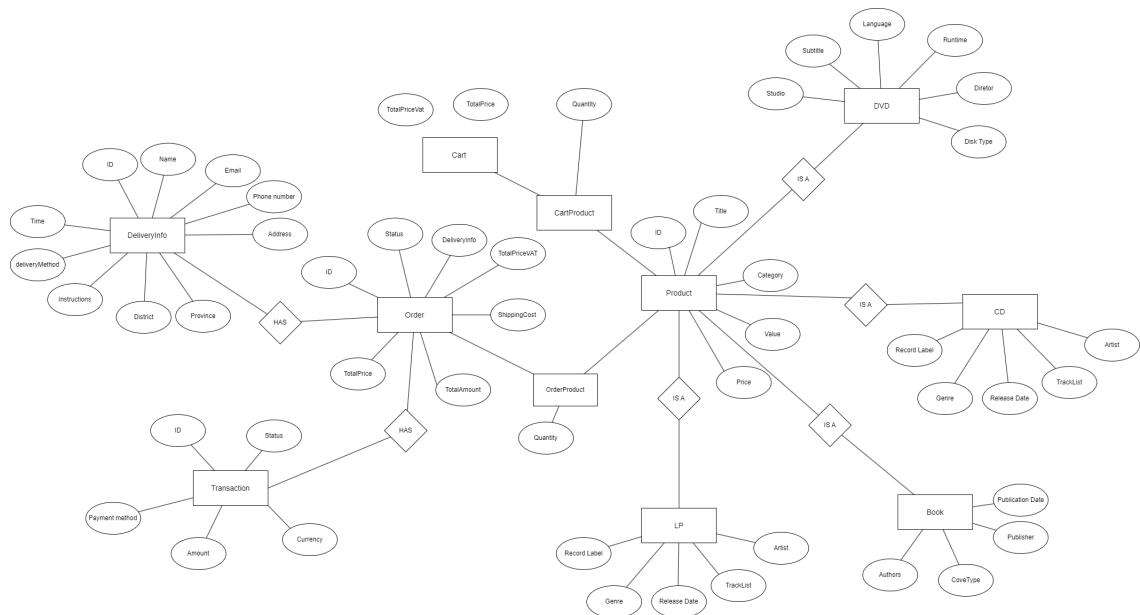


Hình 4.19: Biểu đồ lớp nghiệp vụ khách hàng quản lý đơn hàng

CHƯƠNG 5. THIẾT KẾ CHI TIẾT

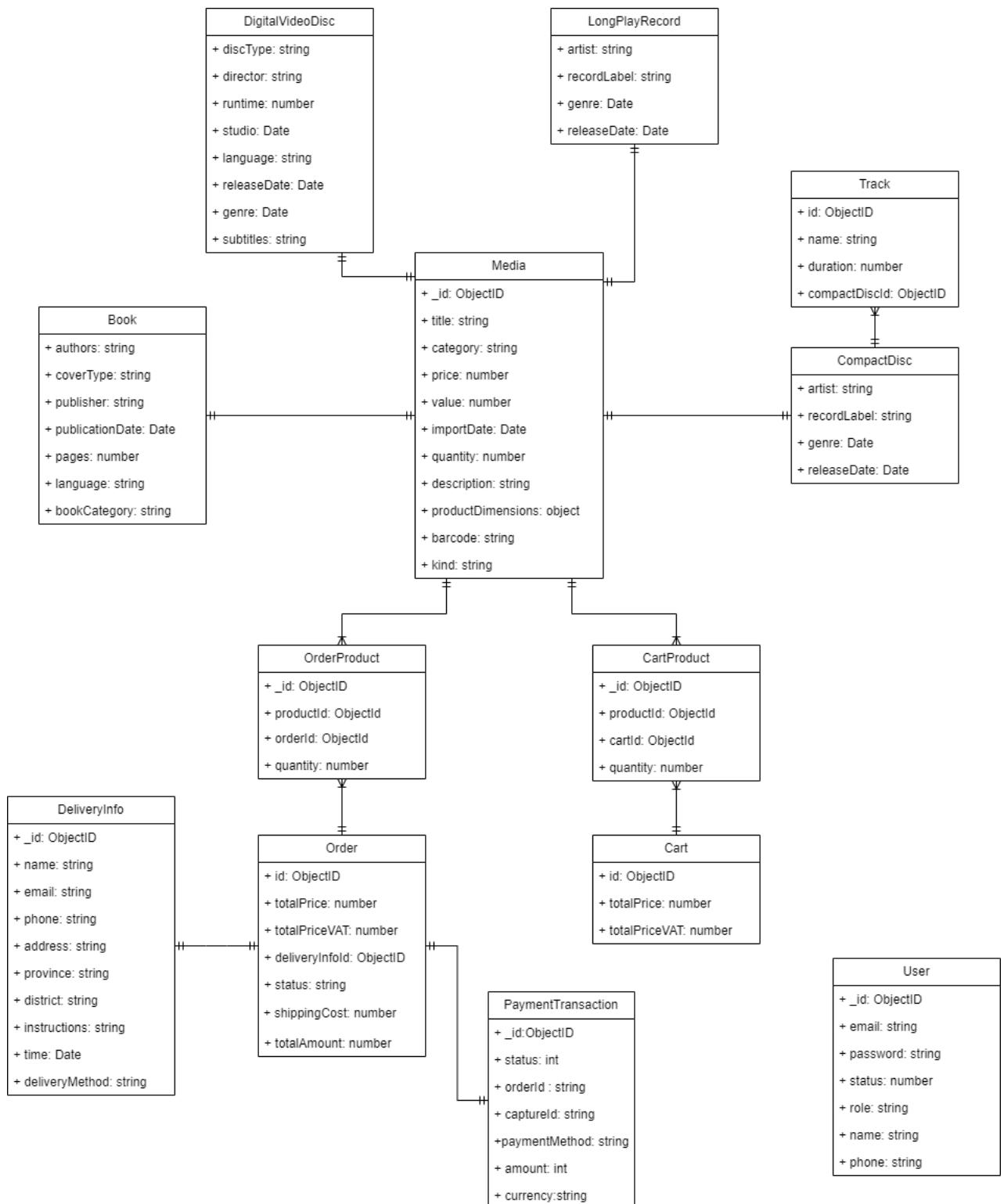
5.1 Data Modeling

5.1.1 Conceptual Data Model



Hình 5.1: Biểu đồ ERD

5.1.2 Logical Data Model



Hình 5.2: Logical Data Model

5.1.3 Physical Data Model

a, Product

CHƯƠNG 5. THIẾT KẾ CHI TIẾT

#	PK	FK	ColumnName	Data type	Mandatory	Description
1	x		_id	ObjectId	Yes	ID, auto generated by mongo
2			title	String	Yes	
3			category	String	Yes	
4			price	Number	Yes	
5			value	Number	Yes	
6			importDate	Date	Yes	
7			quantity	Number	Yes	
8			description	String	Yes	
9			productDimensions	Object	Yes	height, length, width, weight
10			barcode	String	Yes	
11			supportRush	Boolean	Yes	
12			image	String	Yes	
13			kind	String	Yes	book cd longplay dvd
14			author	String	Yes	Book Property 13-19
15			coverType	String	Yes	
16			publisher	String	Yes	
17			publicationDate	Date	Yes	
18			pages	Number	Yes	
19			language	String	Yes	
20			bookCategory	String	Yes	
21			artist	String	Yes	CD (Long Play) Property 20-23
22			recordLabel	String	Yes	
23			genre	String	Yes	
24			releaseDate	Date	Yes	
25			discType	String	Yes	DVD Property 24-31
26			director	String	Yes	
27			runtime	Number	Yes	
28			studio	String	Yes	
29			language	String	Yes	
30			releaseDate	Date	Yes	
31			subtitle	String	Yes	
32			genre	String	Yes	

b, Track

#	PK	FK	ColumnName	Data type	Mandatory	Description
1	x		_id	ObjectId	Yes	ID, auto generated by mongo
2			totalPrice	number	Yes	
3			duration	Number	Yes	
4		x	cdId	ObjectId	Yes	Reference to 'product' collection

c, Order

#	PK	FK	ColumnName	Data type	Mandatory	Description
1	x		_id	ObjectId	Yes	ID, auto generated by mongo
2			name	String	Yes	
3		x	deliveryInfoId	ObjectId	Yes	Reference to 'deliveryInfo' collection
4			totalPriceVAT	number	Yes	
5			status	String	Yes	
6			shippingCost	number	Yes	
7			totalAmount	number	Yes	

d, Order-Product

#	PK	FK	ColumnName	Data type	Mandatory	Description
1	x		_id	ObjectId	Yes	ID, auto generated by mongo
3		x	productId	ObjectId	Yes	Reference to 'Product' collection
3		x	orderId	ObjectId	Yes	Reference to 'Order' collection
4			quantity	number	Yes	

e, Cart

#	PK	FK	ColumnName	Data type	Mandatory	Description
1	x		_id	ObjectId	Yes	ID, auto generated by mongo
2		x	productId	ObjectId	Yes	Reference to 'Product' collection
3		x	cartId	ObjectId	Yes	Reference to 'Cart' collection
4			quantity	number	Yes	

f, Cartproduct

#	PK	FK	ColumnName	Data type	Mandatory	Description
1	x		_id	ObjectId	Yes	ID, auto generated by mongo
2			totalPrice	number	Yes	
3			totalPriceVAT	number	Yes	

g, DeliveryInfo

#	PK	FK	ColumnName	Data type	Mandatory	Description
1	x		_id	ObjectId	Yes	ID, auto generated by mongo
2			name	string	Yes	
3			email	string	Yes	
4			phone	string	Yes	
5			address	string	Yes	
6			province	string	Yes	
7			district	string	Yes	
8			instructions	string	Yes	
9			time	Date	No	
10			deliveryMethod	string	Yes	

h, Transaction

CHƯƠNG 5. THIẾT KẾ CHI TIẾT

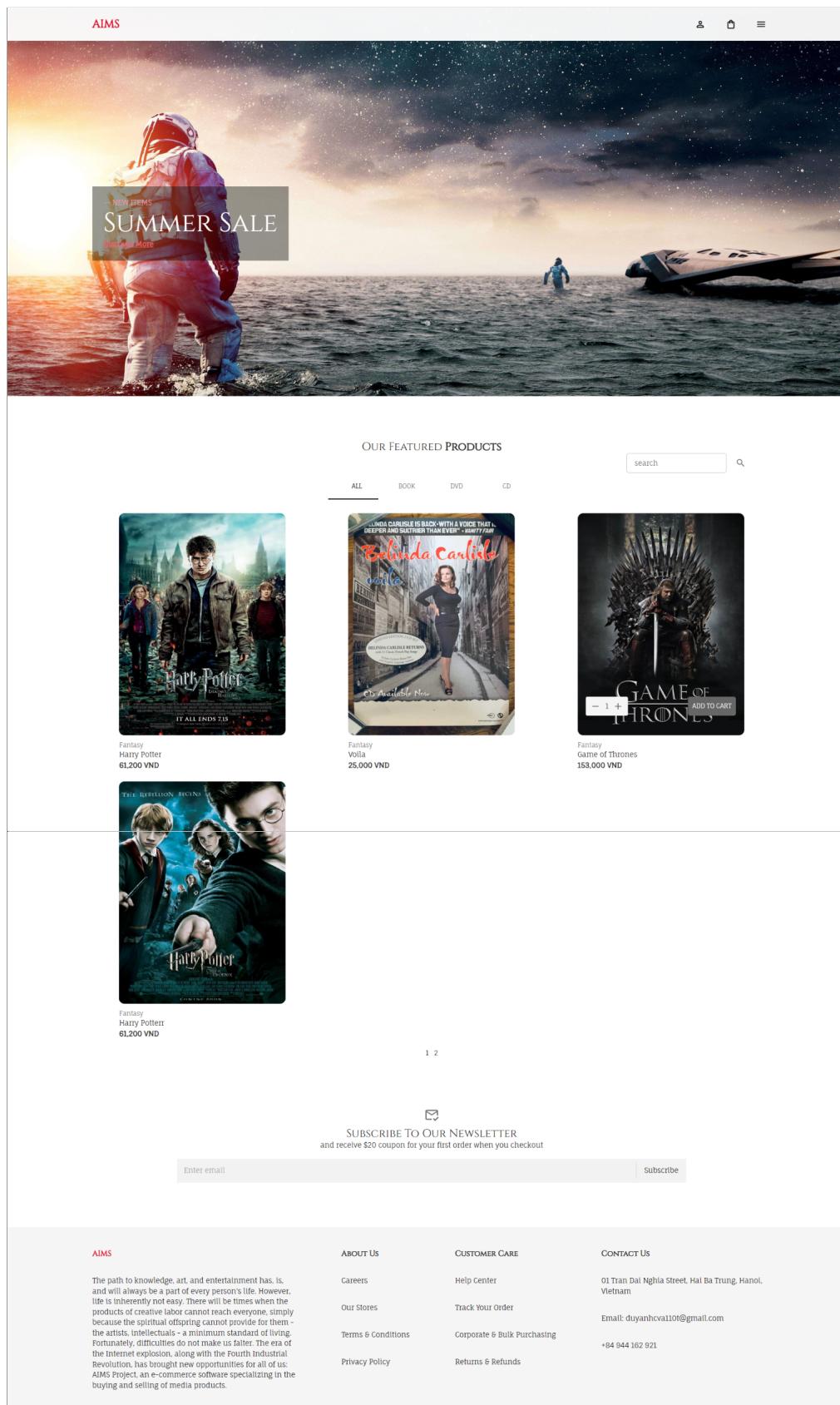
#	PK	FK	ColumnName	Data type	Mandatory	Description
1	x		_id	ObjectId	Yes	ID, auto generated by mongo
2			status	number	Yes	
3			orderId	string	Yes	
4			paymentMethod	string	Yes	
5			amount	number	Yes	
6			currency	string	Yes	
7			captureId	string	No	

i, User

#	PK	FK	ColumnName	Data type	Mandatory	Description
1	x		_id	ObjectId	Yes	ID, auto generated by mongo
2			email	string	Yes	
3			password	string	Yes	
4			status	number	Yes	
5			role	enum	Yes	

5.2 Interface Design

5.2.1 User Interface Design



Hình 5.3: Màn hình trang chủ

CHƯƠNG 5. THIẾT KẾ CHI TIẾT

AIMS

Home / Item

HARRY POTTER
\$61200
BOOK

Thrilling stories about HP
Import date: 12/11/2023, 9:18:38 PM
Barcode: 12345
Support rush delivery: yes

BOOK Information

Info	Value
Author	J.K.Rowling
CoverType	hard cover
Publisher	Bloomsbury
PublicationDate	6/26/1997, 7:00:00 AM
Language	English
Pages	2360
BookCategory	novel

- 1 + **ADD TO CART** In stock: 12

Heart ADD TO WISHLIST
CATEGORIES: Fantasy

DESCRIPTION **REVIEWS**

Thrilling stories about HP

Product Dimension

Dimension	Value
Height	22
Width	12
Length	N/A
Weight	5

RELATED PRODUCTS

Fantasy
Harry Potter
61,200 VND

Fantasy
Volta
25,000 VND

Fantasy
Game of Thrones
153,000 VND

Fantasy
Harry Potter
61,200 VND

AIMS

ABOUT US

CUSTOMER CARE

CONTACT US

The path to knowledge, art, and entertainment has, is, and will always be a part of everyone's life. However, life is not always easy, there will be times when the products of creative labor cannot reach everyone, simply because the spiritual offspring cannot provide for them - the artists, intellectuals - a minimum standard of living. Fortunately, difficulties do not make us falter. The era of the Internet explosion, along with the Fourth Industrial Revolution, has brought new opportunities for all of us.

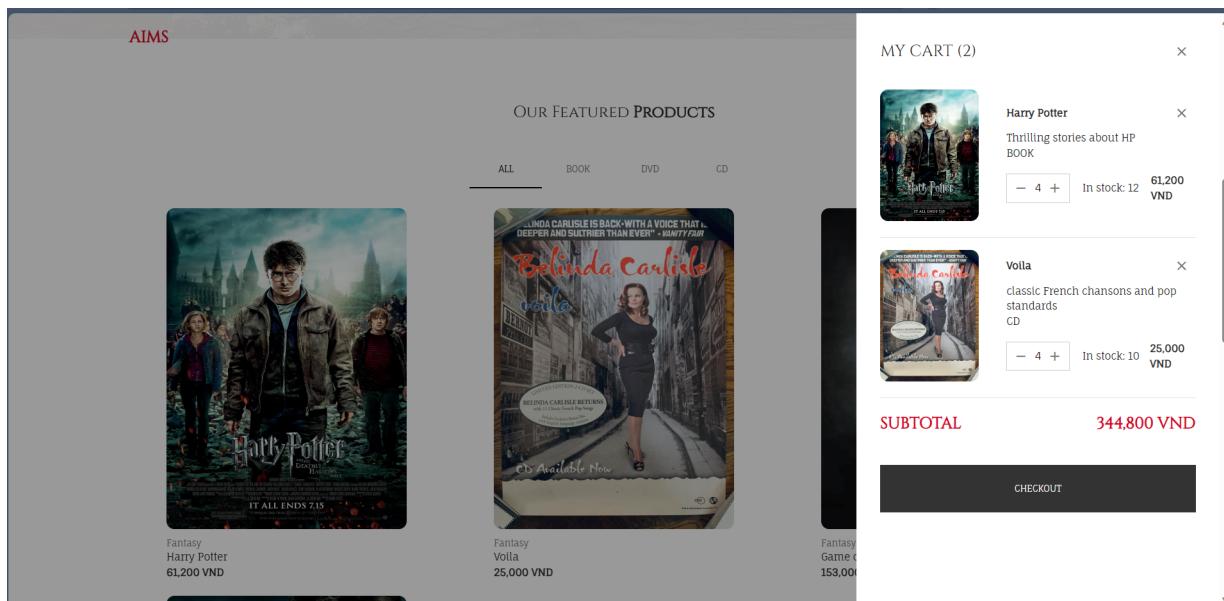
AIMS Project, an e-commerce software specializing in the buying and selling of media products.

Careers Help Center
 Our Stores Track Your Order
 Terms & Conditions Corporate & Bulk Purchasing
 Privacy Policy Returns & Refunds

O1 Tran Dai Nghia Street, Hai Ba Trung, Hanoi, Vietnam
 Email: duyanhvca110@gmail.com
 +84 944 162 021

Hình 5.4: Màn hình chi tiết sản phẩm

CHƯƠNG 5. THIẾT KẾ CHI TIẾT

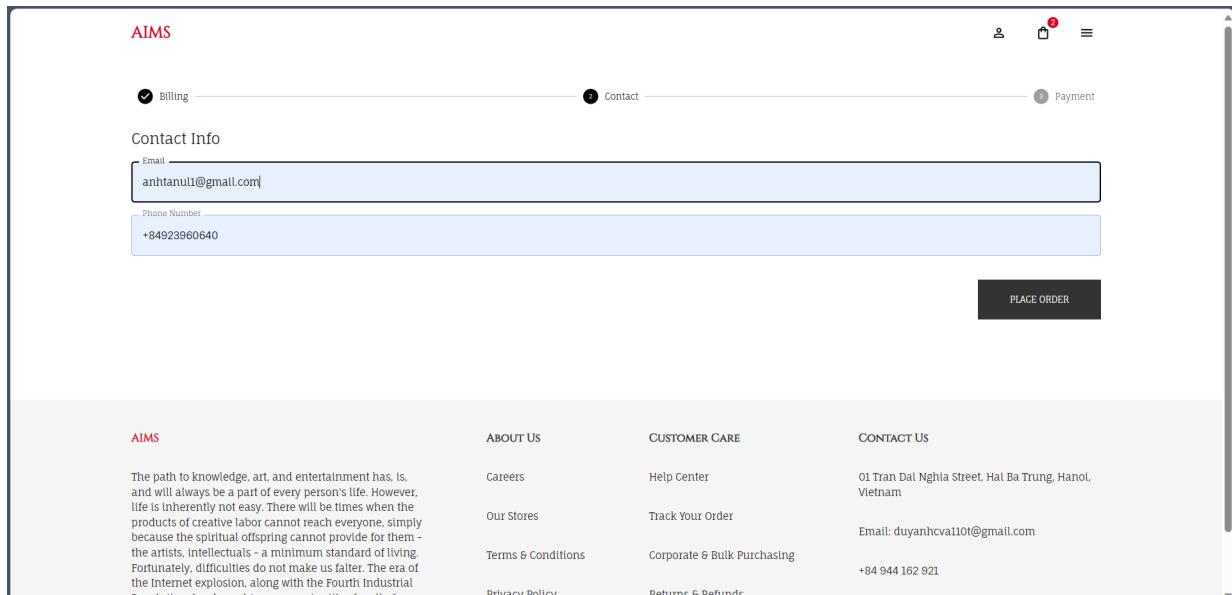


Hình 5.5: Màn hình giỏ hàng

The screenshot shows a "Billing Information" form. It includes fields for First Name (Nguyen), Last Name (anh), Province (Thành phố Hà Nội), District (Quận Đống Đa), Address (1 Đại Cồ Việt, Xuân Tảng - Bắc Phù - Sóc Sơn - Hà Nội), and Instructions (Optional) (Nhà). There is also a checked checkbox for "Rush Delivery" and a date input field set to 01/02/2024 12:00 AM. A "NEXT" button is at the bottom right.

Hình 5.6: Màn hình địa chỉ

CHƯƠNG 5. THIẾT KẾ CHI TIẾT



AIMS

The path to knowledge, art, and entertainment has, is, and will always be a part of every person's life. However, life is inherently not easy. There will be times when the products of creative labor cannot reach everyone, simply because the spiritual offspring cannot provide for them - the artists, intellectuals - a minimum standard of living. Fortunately, difficulties do not make us falter. The era of the Internet explosion, along with the Fourth Industrial Revolution, has brought new opportunities for all of us:

ABOUT US

Careers
Our Stores
Terms & Conditions
Privacy Policy

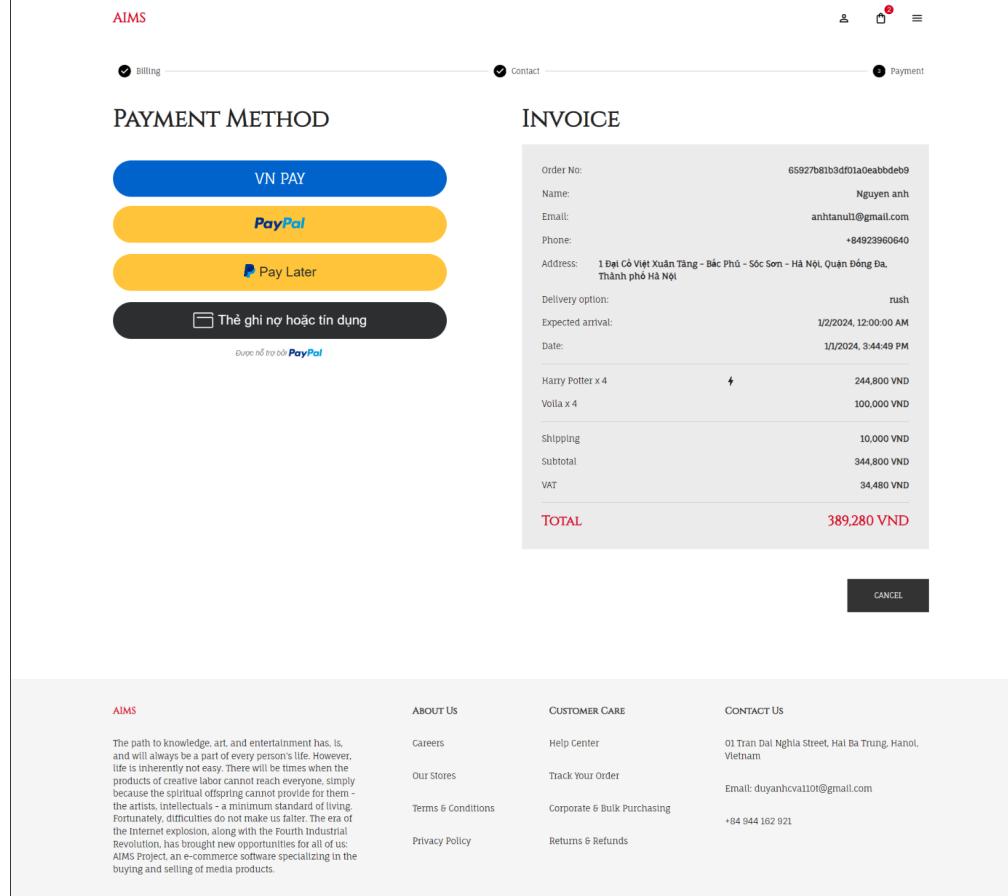
CUSTOMER CARE

Help Center
Track Your Order
Corporate & Bulk Purchasing
Returns & Refunds

CONTACT US

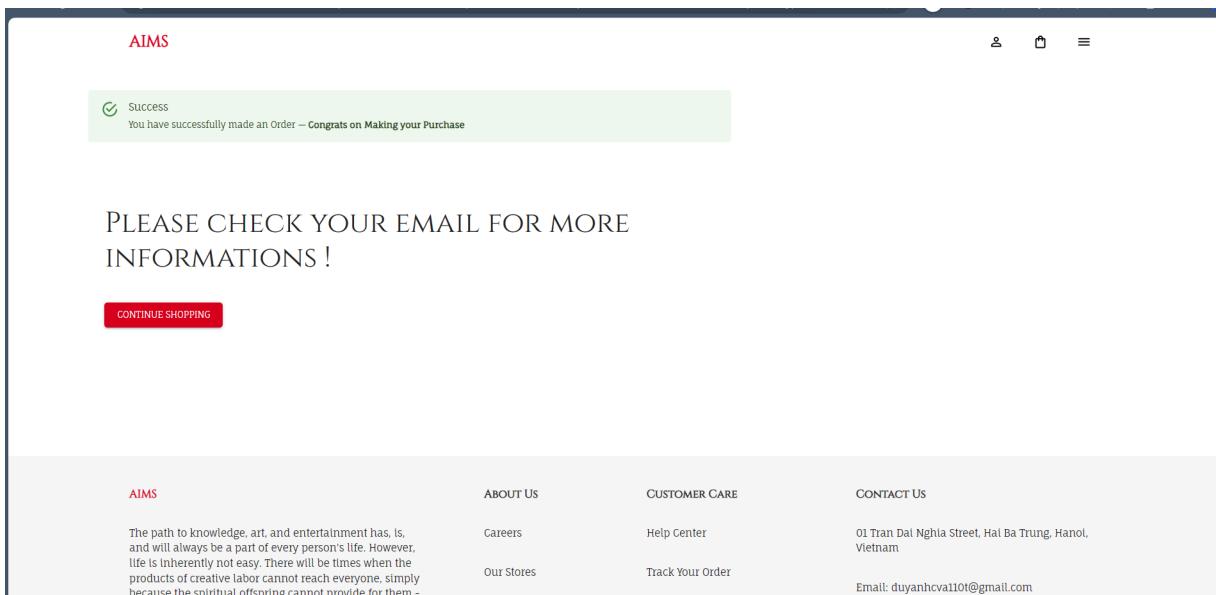
01 Tran Dai Nghia Street, Hal Ba Trung, Hanoi, Vietnam
Email: duyanhvca1101@gmail.com
+84 944 162 921

Hình 5.7: Màn hình thông tin liên hệ

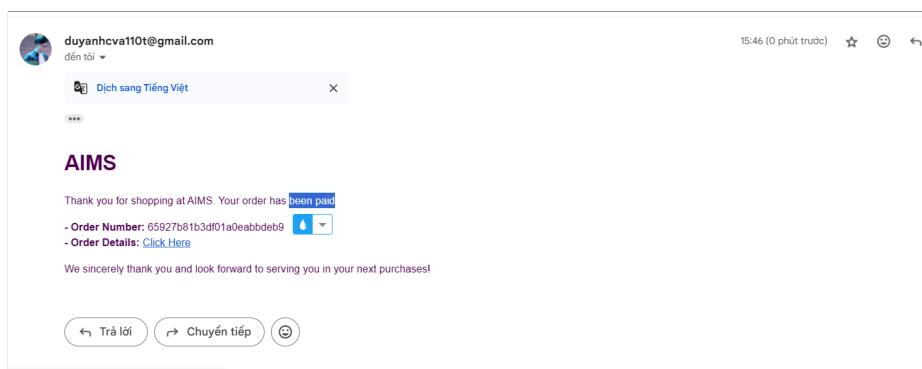


Hình 5.8: Màn hình thanh toán

CHƯƠNG 5. THIẾT KẾ CHI TIẾT



Hình 5.9: Màn hình xác nhận



Hình 5.10: Màn hình email

CHƯƠNG 5. THIẾT KẾ CHI TIẾT

The screenshot displays an order confirmation page from a platform named AIMS. At the top, there's a header with the AIMS logo and a navigation bar with icons for user profile, cart, and menu. Below the header, a green success message box says "SUCCESS You have successfully made an Order — Congrats on Making your Purchase".

ORDER DETAILS

ORDER NUMBER	65927b1b3d101a0eabbde9	ORDER STATUS	PAID
NAME	Nguyen anh	CONTACT	aanhvanull@gmail.com +84923960640
DELIVERY ADDRESS	1 Đường Cố Việt Xuân Tăng - Bắc Phù - Sóc Sơn - Hà Nội, Quận Đông Anh, Thành phố Hà Nội	DELIVERY OPTION	Rush
ORDER DATE	1/1/2024, 3:44:49 PM	PAYMENT METHOD	Vnpay
EXPECTED ARRIVAL	1/2/2024, 12:00:00 AM	TRANSACTION	389,280 VND

CONTINUE SHOPPING

ORDER SUMMARY

ITEMS	PRICE
Harry Potter x 4	61,200 VND
Volla	25,000 VND
SHIPPING	10,000 VND
SUBTOTAL	344,800 VND
VAT	34,480 VND
TOTAL	389,280 VND

REFUND

AIMS

The path to knowledge, art, and entertainment has, is, and will always be a part of every person's life. However, life is inherently not easy. There will be times when the process of seeking knowledge and entertainment becomes difficult because the spiritual offering cannot provide for them - the artists, intellectuals - a minimum standard of living. Fortunately, difficulties do not make us falter. The era of the Internet explosion, along with the Fourth Industrial Revolution, has brought new opportunities for all of us. AIMS Project, an e-commerce software specializing in the buying and selling of media products.

ABOUT US

Careers, Our Stores, Terms & Conditions, Privacy Policy

CUSTOMER CARE

Help Center, Track Your Order, Corporate & Bulk Purchasing, Returns & Refunds

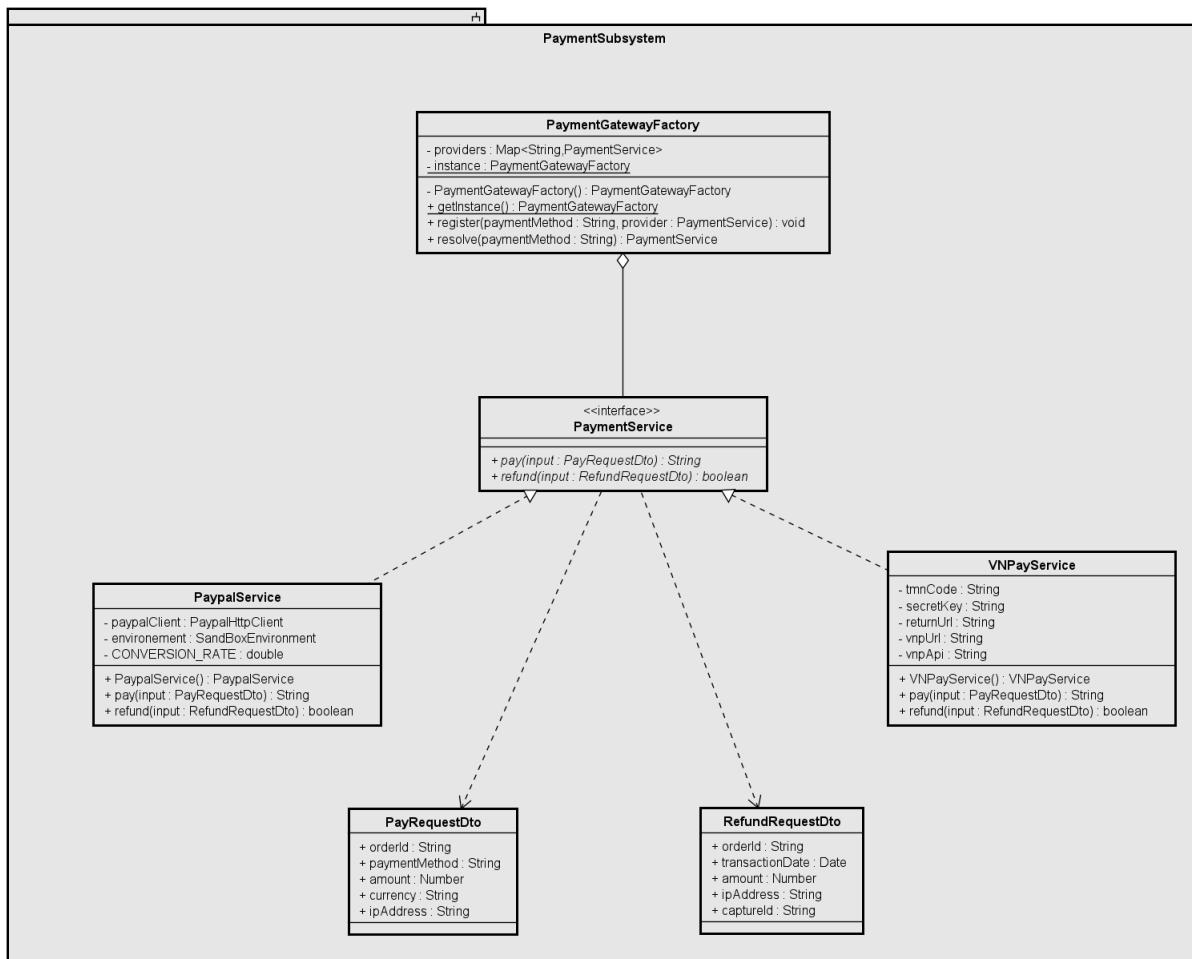
CONTACT US

01 Tran Dai Nghia Street, Hai Ba Trung, Hanoi, Vietnam
Email: duyanhvca110t@gmail.com
+84 944 162 921

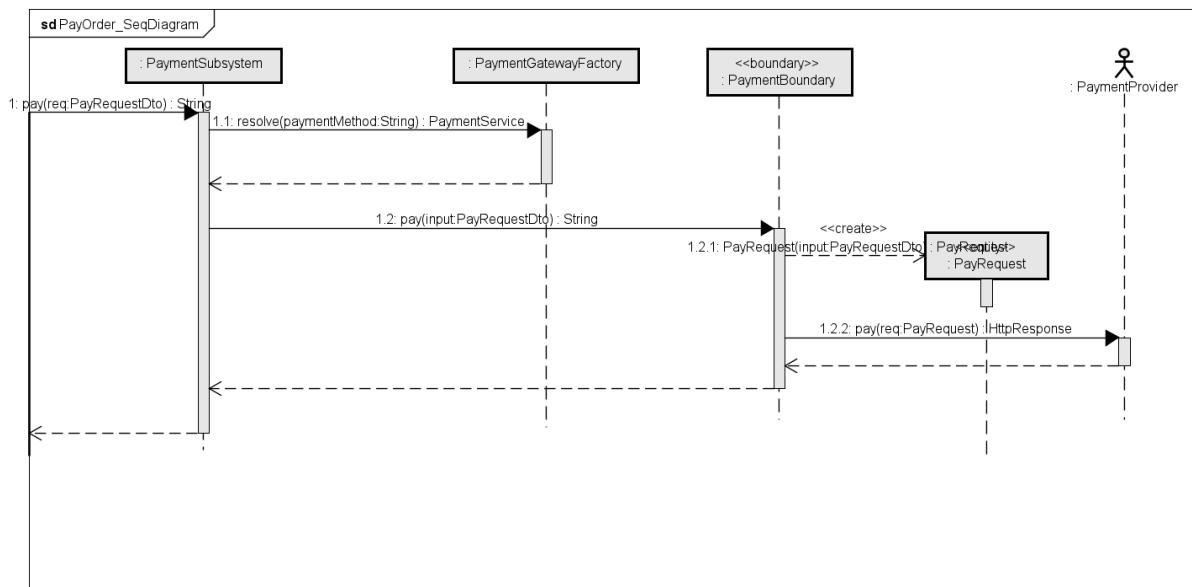
Hình 5.11: Màn hình thông tin đơn hàng

5.2.2 System Interface Design

a, Payment Subsystem

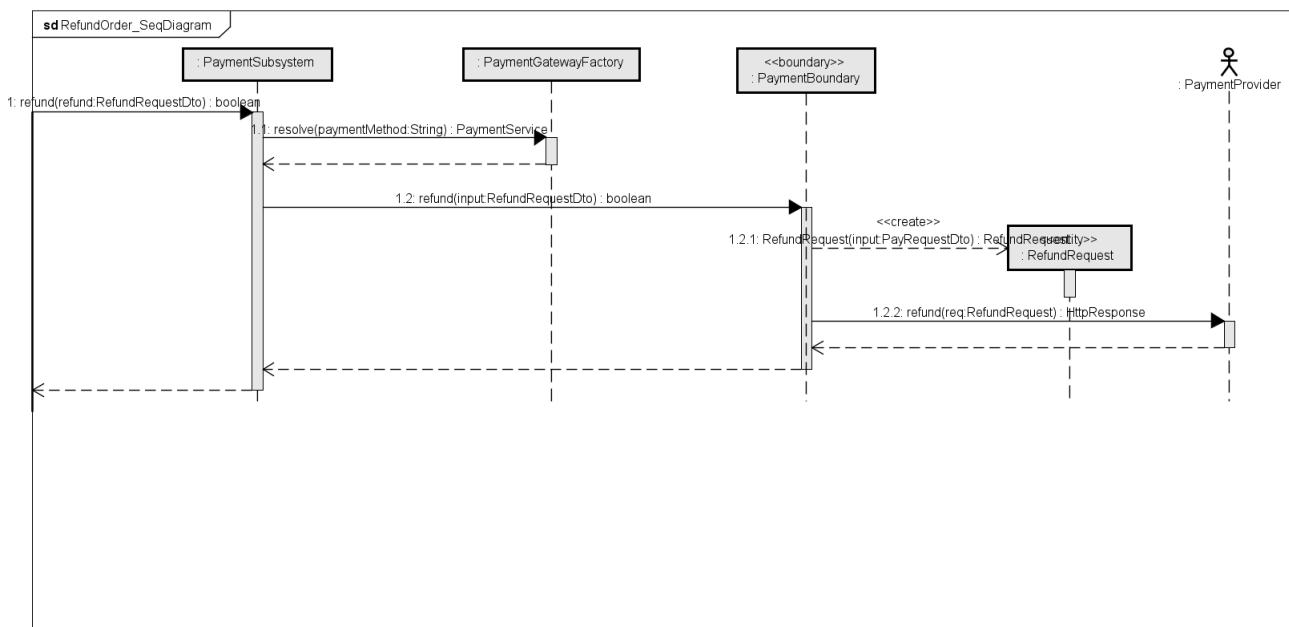


Hình 5.12: Biểu đồ gói payment subsystem



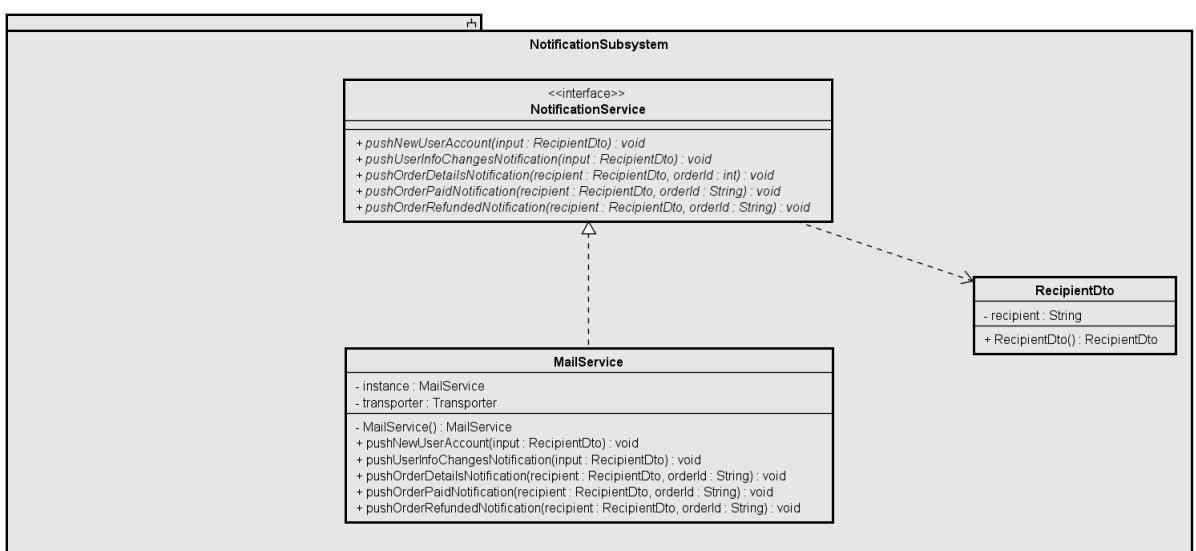
Hình 5.13: Biểu đồ tuần tự nghiệp vụ thanh toán

CHƯƠNG 5. THIẾT KẾ CHI TIẾT



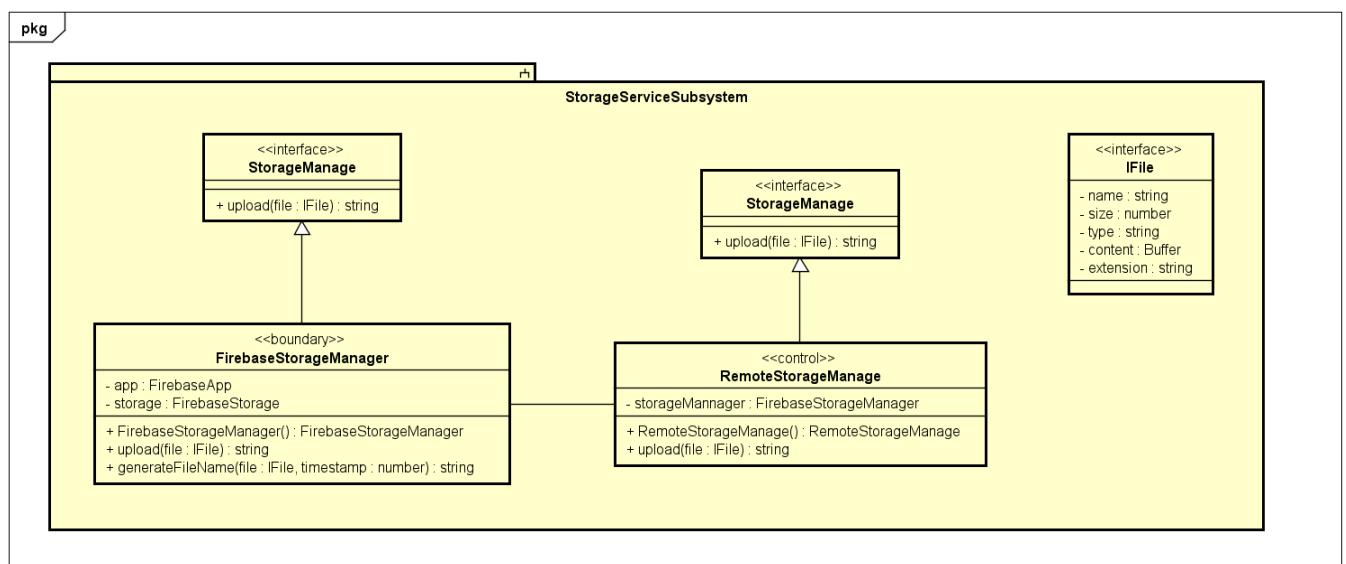
Hình 5.14: Biểu đồ tuần tự nghiệp vụ hoàn tiền

b, Notification Subsystem



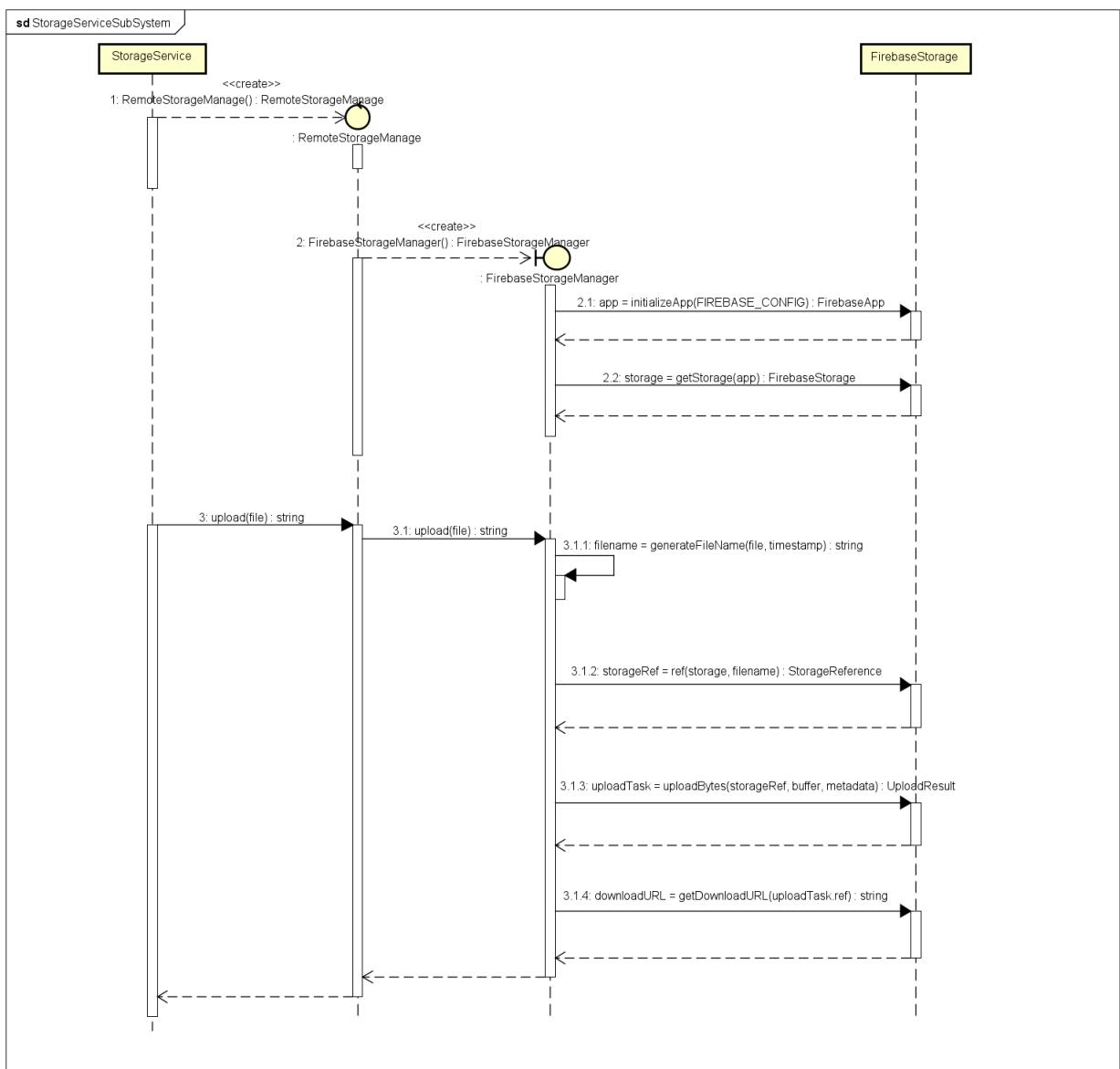
Hình 5.15: Biểu đồ gói notification subsystem

c, Storage Service Subsystem



Hình 5.16: Biểu đồ gói Storage Service Subsystem

CHƯƠNG 5. THIẾT KẾ CHI TIẾT



Hình 5.17: Biểu đồ tuần tự quá trình upload

CHƯƠNG 6. LUU Ý THIẾT KẾ

6.1 Architectural Strategies

Sinh viên so sánh kết quả nghiên cứu hoặc sản phẩm của mình với các nghiên cứu hoặc sản phẩm tương tự.

Sinh viên phân tích trong suốt quá trình thực hiện ĐATN, mình đã làm được gì, chưa làm được gì, các đóng góp nổi bật là gì, và tổng hợp những bài học kinh nghiệm rút ra nếu có.

6.2 Goals and Guidelines

6.3 Coupling and Cohesion

6.3.1 App.ts

- Cohesion: Functional Cohesion

- File này có mức độ kết dính cao vì nó chỉ thực hiện một nhiệm vụ chính là khởi tạo ứng dụng. Nó tạo ra một kết nối cơ sở dữ liệu thông qua MongooseConnection :

```
constructor(controllers: controller[]) {
    // THE ORDER OF THESE FUNCTIONS IS IMPORTANT
    new MongooseConnection()
    const expressServer = new ExpressServer(controllers)
    expressServer.listen();
}
```

- Khởi tạo máy chủ Express thông qua ExpressServer và để bắt đầu ứng dụng: và

```
const expressServer = new ExpressServer(controllers)
expressServer.listen();
```

- Tất cả các chức năng trong file này đều liên quan mật thiết đến việc khởi tạo ứng dụng

- Coupling: Data Coupling

- File này chỉ trao đổi dữ liệu đơn giản với các module khác (MongooseConnection và ExpressServer) và không kiểm soát cách thức hoạt động của chúng. Điều này được thể hiện qua việc:

- Tạo kết nối với mongoose

```
constructor(controllers: controller[]) {
    // THE ORDER OF THESE FUNCTIONS IS IMPORTANT
    new MongooseConnection()
```

- Truyền controllers vào ExpressServer

```
const expressServer = new ExpressServer(controllers)
expressServer.listen();
```

- Nó không kiểm soát cách thức hoạt động của MongooseConnection hoặc ExpressServer

6.3.2 express.server.ts

- Cohesion: Functional Cohesion

- Tất cả các chức năng trong file này đều liên quan mật thiết đến việc khởi tạo và cấu hình máy chủ Express. Điều này được thể hiện qua việc :
- Các hàm initializeMiddleware, initializeControllers, và initializeErrorHandler đều liên quan đến việc cấu hình máy chủ Express.
- Khởi tạo middleware ở constructor:

```
constructor(controllers: Controller[]) {
    this.express = express()
    this.port = Number(process.env.PORT) || 8080

    // THE ORDER OF THESE FUNCTIONS IS IMPORTANT
    this.initializeMiddleware()
    this.initializeControllers(controllers)
    this.initializeErrorHandler()
}
```

- Đoạn code khởi tạo các middleware cho máy chủ Express

```
private initializeMiddleware(): void {
    this.express.set('view engine', 'ejs')
    this.express.use(cors(this.CORS_OPTIONS))
    this.express.use(express.static('public'))
    this.express.use(express.json())
    this.express.use(express.urlencoded({ extended: true }))
```

- Đoạn code khởi tạo các controller cho máy chủ Express:

```

    }

    private initializeControllers(controllers: Controller[]): void {
        controllers.forEach(controller: Controller) => {
            this.express.use('/api', controller.router)
        })
    }
}

```

- Đoạn code khởi tạo xử lý lỗi cho máy chủ Express

```

private initializeErrorHandler(): void {
    this.express.all('*', (req, res, next) => {
        throw new NotFoundError()
    })
    this.express.use(ErrorMiddleware)
}

```

- Đoạn code để bắt đầu ứng dụng

```

public listen(): void {
    this.express.listen(this.port, () => {
        console.log(`[INFO] Application is listening on port ${this.port}`)
    })
}

```

- Coupling: Control Coupling

- File này kiểm soát các module khác bằng cách quyết định những gì sẽ được gọi và cách thức hoạt động của chúng. Điều này được thể hiện qua các đoạn code sau:
- Ở đây, ExpressServer quyết định sử dụng middleware cors, express.json, và express.urlencoded.

```

private initializeMiddleware(): void {
    this.express.set('view engine', 'ejs')
    this.express.use(cors(this.CORS_OPTIONS))
    this.express.use(express.static('public'))
    this.express.use(express.json())
    this.express.use(express.urlencoded({ extended: true }))
}

```

6.3.3 mongoose.connection.ts:

- Cohesion: Functional Cohesion

- File này có mức độ kết dính cao vì tất cả các chức năng trong file này đều liên quan mật thiết đến việc khởi tạo kết nối cơ sở dữ liệu MongoDB. Điều này được thể hiện qua các đoạn code sau
- Đoạn code khởi tạo URI cho kết nối MongoDB

```
constructor(controllers: Controller[]) {
    this.express = express()
    this.port = Number(process.env.PORT) || 8080

    // THE ORDER OF THESE FUNCTIONS IS IMPORTANT
    this.initializeMiddleware()
    this.initializeControllers(controllers)
    this.initializeErrorHandler()
}
```

- Đoạn code gọi hàm initializeDatabaseConnection để khởi tạo và kết nối cơ sở dữ liệu.

```
private initializeDatabaseConnection(): void {
    mongoose
        .connect(this.mongodbUri)
        .then(() => {
            console.log('[INFO] Database connection established')
        })
        .catch((err) =>
            console.log('[ERROR] Database connection failed: ' + err)
        )
}
```

- Coupling: Data Coupling.

- File này có mức độ kết nối thấp vì File này chỉ trao đổi dữ liệu đơn giản với module mongoose. Điều này được thể hiện qua đoạn code sau:

```
private initializeDatabaseConnection(): void {
    mongoose
        .connect(this.mongodbUri)
        .then(() => {
            console.log('[INFO] Database connection established')
        })
}
```

- MongooseConnection chỉ trao đổi dữ liệu đơn giản với module mongoose bằng cách truyền mongodbUri vào hàm connect. Nó không kiểm soát cách thức hoạt động của mongoose.

6.4 Design Principles

6.4.1 Single Responsibility

Nguyên lý Single Responsibility Principle (SRP) trong SOLID nói rằng mỗi module, class hoặc hàm trong chương trình chỉ nên chịu trách nhiệm cho một nhiệm vụ cụ thể. Nếu một module, class hoặc hàm chịu trách nhiệm cho nhiều hơn một nhiệm vụ, nó trở nên phức tạp hơn và khó bảo dưỡng hơn. Dưới đây là cách SRP được áp dụng

Lớp ProductController được thiết kế với một mục đích chính, đó là xử lý các yêu cầu liên quan đến đối tượng sản phẩm trong hệ thống thông qua việc quản lý các tuyến đường (router). Chức năng chính của lớp này là định nghĩa các đường dẫn (endpoints) và kết nối chúng với các hàm xử lý logic tương ứng.

```

23  export class ProductController implements Controller {
24      public readonly path = '/products'
25      public readonly router = Router()
26      public readonly productDaoFactory: ProductDaoFactory
27      public readonly productDao: ProductsDao
28      public readonly storageManage: StorageManage
29
30      constructor() {
31          this.productDaoFactory = new ProductDaoFactory()
32          this.productDao = new ProductsMongooseDao(ProductModel.getInstance())
33          this.storageManage = new RemoteStorageManage()
34          this.initializeRoutes()
35      }
36
37  >    private initializeRoutes() { ... }
38
39  >    private createProduct = async ( ... )
40
41  >    private updateProduct = async ( ... )
42
43  >    private getProduct = async ( ... )
44
45  >    private getProducts = async ( ... )
46
47  >    private deleteProduct = async ( ... )
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192 }
```

Hình 6.1: Product Controller

6.4.2 Open Closed

Nguyên lý Open-Closed Principle (OCP) trong SOLID nói rằng các module, class hoặc hàm trong chương trình nên được mở (open) cho việc mở rộng, nhưng

đóng (closed) cho việc sửa đổi. Điều này có nghĩa là chúng ta nên thiết kế chương trình sao cho khi cần thêm chức năng mới, chúng ta chỉ cần thêm code mới mà không cần sửa đổi code hiện có. Dưới đây là cách OCP được áp dụng:

Tính đóng mở của Storage Service Subsystem để có thể mở rộng thêm các hình thức lưu trữ khác ngoài Remote Storage như hiện tại. Hơn nữa có thể mở rộng thêm công nghệ khác thay vì Firebase như hiện tại

Định nghĩa interface IFile chung cho hệ thống Định nghĩa các interface cho Stor-

```

1  export interface IFile {
2    ...
3    name: string;
4    ...
5    size: number;
6    ...
7    type: string;
8    ...
9    content: Buffer;
10   ...
11   extension: string;
12 }
```

Hình 6.2: File Interface

ageManage (Hình thức quản lý lưu trữ local/remote/...) và StorageManager (Công cụ quản lí lưu trữ FileSystem/Firebase/AWS/...) Implement StorageManager với

```

4  export interface StorageManager {
5    ...
6    upload(
7      ...
8      file: IFile
9    ): Promise<UploadedFile>
10 }
```



```

4  export interface StorageManager {
5    ...
6    upload(
7      ...
8      file: IFile
9    ): Promise<UploadedFile | undefined>
10 }
```

Hình 6.3: Storage Manage Interfaces

thư viện firebase storage Trong lớp RemoteStorageManage khởi tạo FirebaseStorageManager và gọi method upload

Đối với lớp PaymentGatewayFactory, để thêm mới các phương thức thanh toán chỉ cần thêm cài đặt cho phương thức đó kèm với decorator "@PaymentProvider",

```

14  export class FirebaseStorageManager implements StorageManager {
15    private static app: FirebaseApp
16    private static storage: FirebaseStorage
17
18    constructor() {
19      FirebaseStorageManager.app = initializeApp(FIREBASE_CONFIG)
20      FirebaseStorageManager.storage = getStorage(FirebaseStorageManager.app)
21    }
22
23    async upload(file: IFile): Promise<UploadedFile | undefined> {
24      const storageRef = ref(
25        FirebaseStorageManager.storage,
26        'images/' + this.generateFileName(file, Date.now())
27      )
28
29      const metadata = {
30        contentType: file.type,
31      }
32      const buffer = Buffer.from(file.content)
33      const uploadTask = await uploadBytes(storageRef, buffer, metadata)
34
35      const downloadURL = await getDownloadURL(uploadTask.ref)
36
37      return { path: downloadURL }
38    }
39
40    generateFileName(file: IFile, timestamp: number): string {
41      return `${file.name}-${timestamp}.${file.extension}`
42    }
43  }

```

Hình 6.4: Firebase Storage Manager

```

8  export class RemoteStorageManager implements StorageManager {
9    constructor()
10   private readonly storageManager: StorageManager = new FirebaseStorageManager()
11  }
12
13  async upload(file: IFile): Promise<UploadedFile> {
14    const uploadedFile = await this.storageManager.upload(file)
15
16    if (!uploadedFile) {
17      throw new BadRequestError('File upload failed')
18    }
19
20    return uploadedFile
21  }
22

```

Hình 6.5: Remote Storage Manager

phương thức thanh toán mới này sẽ được tự động đăng ký thông qua hàm register, không cần chỉnh sửa thêm gì ở các dòng code cũ.

```

3
4  export class PaymentGatewayFactory {
5      private providers: Record<string, PaymentService> = {}
6      private static instance: PaymentGatewayFactory
7
8      private constructor() {}
9
10     public static getInstance(): PaymentGatewayFactory {
11         if (!PaymentGatewayFactory.instance) {
12             PaymentGatewayFactory.instance = new PaymentGatewayFactory()
13         }
14
15         return PaymentGatewayFactory.instance
16     }
17
18     public register(payment_method: string, provider: any): void {
19         this.providers[payment_method] = new provider()
20     }
21
22     public resolve(payment_method: string): any {
23         const matchedProvider = this.providers[payment_method]
24
25         if (!matchedProvider) {
26             throw new BadRequestError(
27                 `This payment method is not supported: ${payment_method}`
28             )
29         }
30
31         return matchedProvider
32     }
33 }
34
35 export enum PAYMENT_METHOD {
36     PAYPAL = 'paypal',
37     VNPAY = 'vnpay',
38 }
39

```

Hình 6.6: Class PaymentGatewayFactory

6.4.3 Liskov substitution

Với cách áp dụng factory pattern cho nhiều kiểu Products khác nhau (book, cd, dvd, long play), các instance của các class ProductsDao có thể thay thế cho nhau linh hoạt trong nhiều trường hợp. Các phương thức CRUD được downcasting mà không làm mất tính đúng đắn của chương trình.

```

11  class ProductDaoFactory {
12      private productDaos: Record<string, ProductsDao>
13      private bookDao: ProductsDao
14      private cdDao: ProductsDao
15      private dvdDao: ProductsDao
16      private longPlayDao: ProductsDao
17
18      constructor() {
19          this.bookDao = new BooksMongooseDao()
20          this.cdDao = new CdsMongooseDao()
21          this.dvdDao = new DvdsMongooseDao()
22          this.longPlayDao = new LongPlaysMongooseDao()
23          this.productDaos = {}
24          this.productDaos[KIND.BOOK] = this.bookDao
25          this.productDaos[KIND.CD] = this.cdDao
26          this.productDaos[KIND.DVD] = this.dvdDao
27          this.productDaos[KIND.LONG_PLAY] = this.longPlayDao
28      }
29
30      public getInstance(productType: KIND) {
31          if (!this.productDaos[productType]) {
32              throw new BadRequestError(
33                  'Request for quotation type not supported'
34              )
35          }
36          return this.productDaos[productType]
37      }
38  }

```

Hình 6.7: Product Dao Factory

6.4.4 Interface segregation

Nguyên lý Interface Segregation Principle (ISP) trong SOLID nói rằng "khách hàng không nên bị buộc phải thuộc vào các interfaces mà họ không sử dụng". Điều này có nghĩa là mỗi interface nên giữ cho mình một số lượng nhỏ các phương thức cần thiết, thay vì có một interface lớn với nhiều phương thức không liên quan đến nhau. Dưới đây là cách ISP được áp dụng:

Các interface trong ứng dụng được phân chia rõ ràng, chỉ định các chức năng cần thiết cho các class sử dụng chúng. Đơn cử là ở lớp PaymentService chỉ định các phương thức cần thiết chỉ để phục vụ chức năng là thanh toán hay hoàn tiền. Các lớp như VNPayService hay PayPalService sẽ implements PaymentService và cài đặt các phương thức trên.

order.dao.ts: Interface OrderDao định nghĩa các phương thức cần thiết cho việc truy vấn dữ liệu order từ cơ sở dữ liệu. Mỗi phương thức trong interface này đều liên quan trực tiếp đến việc quản lý order, không có phương thức nào không liên

CHƯƠNG 6. LUU Ý THIẾT KẾ

```
1 import { PayRequestDto } from '../dtos/pay.dto'
2 import { RefundRequestDto } from '../dtos/refund.dto'
3
4 export interface PaymentService {
5     pay(input: PayRequestDto): Promise<string>
6     refund(input: RefundRequestDto): Promise<boolean>
7 }
8
```

Hình 6.8: Class PaymentService

quan hoặc không cần thiết.

```
export interface OrderDao {
    findById(id: string | ObjectId): Promise<IOrder | null>
    create(createOrderDto: CreateOrderDto): Promise<IOrder>
    getLatestOrderId(): Promise<String>
    updateStatus(id: string | ObjectId, status: number): Promise<IOrder>
    updateOrder([
        id: string | ObjectId,
        updateOrderDto: CreateOrderDto
    ]): Promise<IOrder | null>
    findAll(): Promise<IOrder[]>
}
```

Hình 6.9: OrderDaoOpen

order-management.controller.ts: Class OrderManagementController sử dụng OrderDao thông qua dependency injection. Nó chỉ phụ thuộc vào các phương thức cần thiết từ OrderDao, không phụ thuộc vào các phương thức không liên quan từ các interface khác.

```
export class OrderManagementController implements Controller {
    private orderDao: OrderDao
    constructor() {
        this.orderDao = new OrderMongooseDao()
    }
    // Use methods from OrderDao
}
```

Hình 6.10: OrderDaoController

Như vậy, mỗi file đều tuân thủ nguyên lý Interface Segregation Principle, giúp giảm sự phụ thuộc không cần thiết và làm cho code dễ hiểu hơn.

```

8  @PaymentProvider(PAYMENT_METHOD.PAYPAL)
9  export class PaypalService implements PaymentService {
10    private readonly paypalClient
11    private readonly environment
12    private readonly CONVERSION_RATE = 0.000041
13
14  >   public constructor() { ... }
15
16
17  >   public async pay(input: PayRequestDto): Promise<string> { ... }
18
19
20  >   public async refund(input: RefundRequestDto): Promise<boolean> { ... }
21
22
23
24  >   private convertVndToUsd(total: number): string { ... }
25
26
27  }
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58

```

Hình 6.11: Class PayPalService

6.4.5 Dependency Inversion

Các class trong ứng dụng được thiết kế để phụ thuộc vào lớp abstraction, không phải phụ thuộc vào implementations cụ thể. Các module cấp cao không phụ thuộc vào các module cấp thấp. Cả hai phụ thuộc vào các abstraction. DIP giúp giảm sự phụ thuộc giữa các module cấp cao và cấp thấp, giúp code dễ dàng thay đổi và phát triển hơn.

Dưới đây là cách DIP được áp dụng trong các file bạn đã cung cấp:

- order.dao.ts và order.mongoose.dao.ts: OrderMongooseDao implement interface OrderDao. Điều này có nghĩa là OrderMongooseDao (chi tiết) phụ thuộc vào OrderDao (abstraction), không phải ngược lại. Điều này tuân thủ DIP.

```

export class OrderMongooseDao implements OrderDao {
  constructor(private orderModel: Document = OrderModel.getInstance()) {}
  public async findById(id: string): Promise<IOOrder | null> { ... }
  public async create(createOrderDto: CreateOrderDto): Promise<IOOrder> { ... }
  public async getLatestOrderId(filter?: object): Promise<string> { ... }
  public async updateOrder(id: string, updateOrderDto: CreateOrderDto): Promise<IOOrder>
  public async updateStatus(id: string, status: number): Promise<IOOrder> { ... }
  public async findAll(): Promise<IOOrder[]> { ... }
}

```

Hình 6.12: Class OrderMongooseDaoDIP

- order-management.controller.ts: Class OrderManagementController sử dụng

OrderDao thông qua dependency injection. Điều này có nghĩa là OrderManagementController (module cấp cao) không phụ thuộc trực tiếp vào OrderMongooseDao (module cấp thấp), mà phụ thuộc vào OrderDao (abstraction). Điều này tuân thủ DIP.

```
export class OrderManagementController implements Controller {
    private orderDao: OrderDao
    constructor() {
        this.orderDao = new OrderMongooseDao()
    }
    // Use methods from OrderDao
}
```

Hình 6.13: Class OrderControllerDIP

Như vậy, mỗi file đều tuân thủ nguyên lý Dependency Inversion Principle, giúp giảm sự phụ thuộc giữa các module và làm cho code dễ dàng thay đổi hơn.

6.5 Design Patterns

6.5.1 Singleton Pattern

Singleton pattern đảm bảo rằng một class chỉ có duy nhất một instance và cung cấp phương thức để truy cập tới instance đó.

Pattern này được áp dụng cho các model class, PaymentGatewayFactory.

6.5.2 Factory Pattern

a, Product

Làm cho tất cả các sản phẩm theo cùng một interface IProduct. Interface ProductsDao sẽ khai báo các phương thức có ý nghĩa trong mọi sản phẩm.

```

6  /**
7  * The Singleton class defines the `getInstance` method that lets clients access
8  * the unique singleton instance.
9  */
10 export class TransactionModel {
11     private static instance: Model<ITransaction>
12
13     /**
14      * The Singleton's constructor should always be private to prevent direct
15      * construction calls with the `new` operator.
16      */
17     private constructor() {}
18
19     /**
20      * The static method that controls the access to the singleton instance.
21      *
22      * This implementation let you subclass the Singleton class while keeping
23      * just one instance of each subclass around.
24      */
25     public static getInstance(): Model<ITransaction> {
26         if (!TransactionModel.instance) {
27             const transactionSchema = new Schema<ITransaction>(
28                 {
29                     status: {
30                         type: Number,
31                         default: TRANSACTION_STATUS.SUCCEED,
32                     },
33                     orderId: {
34                         type: mongoose.Schema.Types.ObjectId,
35                         ref: 'order',
36                     },
37                     captureId: String,
38
39                     paymentMethod: {
40                         type: String,
41                         required: true,
42                     },
43                     amount: {
44                         type: Number,
45                         required: true,
46                     },
47                     currency: {
48                         type: String,
49                         required: true,
50                     },
51                     content: String,
52                 },
53                 { timestamps: true }
54             )
55
56             TransactionModel.instance = model('Transaction', transactionSchema)
57         }
58     }
59
60     return TransactionModel.instance
61 }

```

Hình 6.14: Class TransactionModel

```

3  export interface IProduct {
4      id: number | ObjectId | string
5      title: string
6      category: string
7      price: number
8      value: number
9      importDate: Date
10     quantity: number
11     description: string
12     productDimensions: ProductDimension
13     barcode: string
14     kind: string
15     supportRush: boolean
16     image: string
17 }

```

```

4  export interface ProductsDao {
5      create(createProductDto: CreateProductDto): Promise<ObjectId>
6      update(id: string, updateProductDto: CreateProductDto): Promise<boolean>
7      findById(id: string): Promise<IProduct | null>
8      findAll(query: QueryProductDto): Promise<IProduct[]>
9      delete(id: string): Promise<boolean>
10     isBarcodeExist(barCode: string, id?: string): Promise<boolean>
11     findMany(ids: string[]): Promise<IProduct[]>
12 }

```

ProductDaoFactory là nơi lưu tất cả các instance cho các loại Products

```

11  class ProductDaoFactory {
12      private productDaos: Record<string, ProductsDao>
13      private bookDao: ProductsDao
14      private cdDao: ProductsDao
15      private dvdDao: ProductsDao
16      private longPlayDao: ProductsDao
17
18      constructor() {
19          this.bookDao = new BooksMongooseDao()
20          this.cdDao = new CdsMongooseDao()
21          this.dvdDao = new DvdsMongooseDao()
22          this.longPlayDao = new LongPlaysMongooseDao()
23          this.productDaos = {}
24          this.productDaos[KIND.BOOK] = this.bookDao
25          this.productDaos[KIND.CD] = this.cdDao
26          this.productDaos[KIND.DVD] = this.dvdDao
27          this.productDaos[KIND.LONG_PLAY] = this.longPlayDao
28      }
29
30      public getInstance(productType: KIND) {
31          if (!this.productDaos[productType]) {
32              throw new BadRequestError(
33                  'Request for quotation type not supported'
34              )
35          }
36          return this.productDaos[productType]
37      }
38  }

```

Hình 6.16: Product Factory

Lấy ra instance của kiểu product tương ứng

```
84  const productDao = this.productDaoFactory.getInstance(kind)
```

Hình 6.17: Product Factory Usage

b, PaymentGatewayFactory

Factory pattern cung cấp 1 nơi tập trung để khởi tạo ra các instance của các class liên quan đến nghiệp vụ tương tác với bên thứ 3 cung cấp dịch vụ thanh toán như PayPal hay VNPay.

```

3
4  export class PaymentGatewayFactory {
5      private providers: Record<string, PaymentService> = {}
6      private static instance: PaymentGatewayFactory
7
8      private constructor() {}
9
10     public static getInstance(): PaymentGatewayFactory {
11         if (!PaymentGatewayFactory.instance) {
12             PaymentGatewayFactory.instance = new PaymentGatewayFactory()
13         }
14
15         return PaymentGatewayFactory.instance
16     }
17
18     public register(payment_method: string, provider: any): void {
19         this.providers[payment_method] = new provider()
20     }
21
22     public resolve(payment_method: string): any {
23         const matchedProvider = this.providers[payment_method]
24
25         if (!matchedProvider) {
26             throw new BadRequestError(
27                 `This payment method is not supported: ${payment_method}`
28             )
29         }
30
31         return matchedProvider
32     }
33 }
34
35 export enum PAYMENT_METHOD {
36     PAYPAL = 'paypal',
37     VNPay = 'vnpay',
38 }
39

```

Hình 6.18: Class PaymentGatewayFactory

Các providers sẽ được register thông qua decorator "@PaymentProvider" khi chương trình được khởi tạo.

CHƯƠNG 6. LƯU Ý THIẾT KẾ

```
3   export function PaymentProvider(payment_method: string): Function {
4     return function (target: any): void {
5       PaymentGatewayFactory.getInstance().register(payment_method, target)
6     }
7   }
8 
```

Hình 6.19: Decorator PaymentProvider

```
8   @PaymentProvider(PAYMENT_METHOD.PAYPAL)
9   export class PaypalService implements PaymentService {
10     private readonly paypalClient
11     private readonly environment
12     private readonly CONVERSION_RATE = 0.000041
13
14   >   public constructor() { ... }
15   >
16   >   public async pay(input: PayRequestDto): Promise<string> { ... }
17   >
18   >   public async refund(input: RefundRequestDto): Promise<boolean> { ... }
19   >
20   >   private convertVndToUsd(total: number): string { ... }
21   >
22   >
23   >
24   >
25   >
26   >
27   >
28   >
29   >
30   >
31   >
32   >
33   >
34   >
35   >
36   >
37   >
38   >
39   >
40   >
41   >
42   >
43   >
44   >
45   >
46   >
47   >
48   >
49   >
50   >
51   >
52   >
53   >
54   >
55   >
56   >
57   >
58 } 
```

Hình 6.20: Class PaypalService sử dụng decorator PaymentProvider để tự động register với PaymentGatewayFactory

CHƯƠNG 7. ĐÁNH GIÁ CÔNG VIỆC

Bảng phân công công việc

Name - MSSV	Công việc	Đóng góp(%)
Vũ Thành An - 20205197	<ul style="list-style-type: none"> - Use case: Quản lý Sản phẩm, Tìm kiếm sản phẩm - Subsystem: Storage Service Subsystem - Repository: CD-Track - Database models 	33.3%
Đỗ Duy Anh - 20205198	<ul style="list-style-type: none"> - Use case: Thanh toán, Quản lý người dùng, Đăng nhập - Subsystem: Payment Subsystem, Notification Subsystem - Codebase 	33.3%
Nguyễn Ngọc Ánh - 20205228	<ul style="list-style-type: none"> - Use case: Giỏ hàng, đặt hàng, đặt hàng nhanh - Repository: Order-repository 	33.3%