

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

School of Information and Communications Technology

## **Capstone Project Report**

Subject: Thiết kế xây dựng phần mềm

Nhóm 4

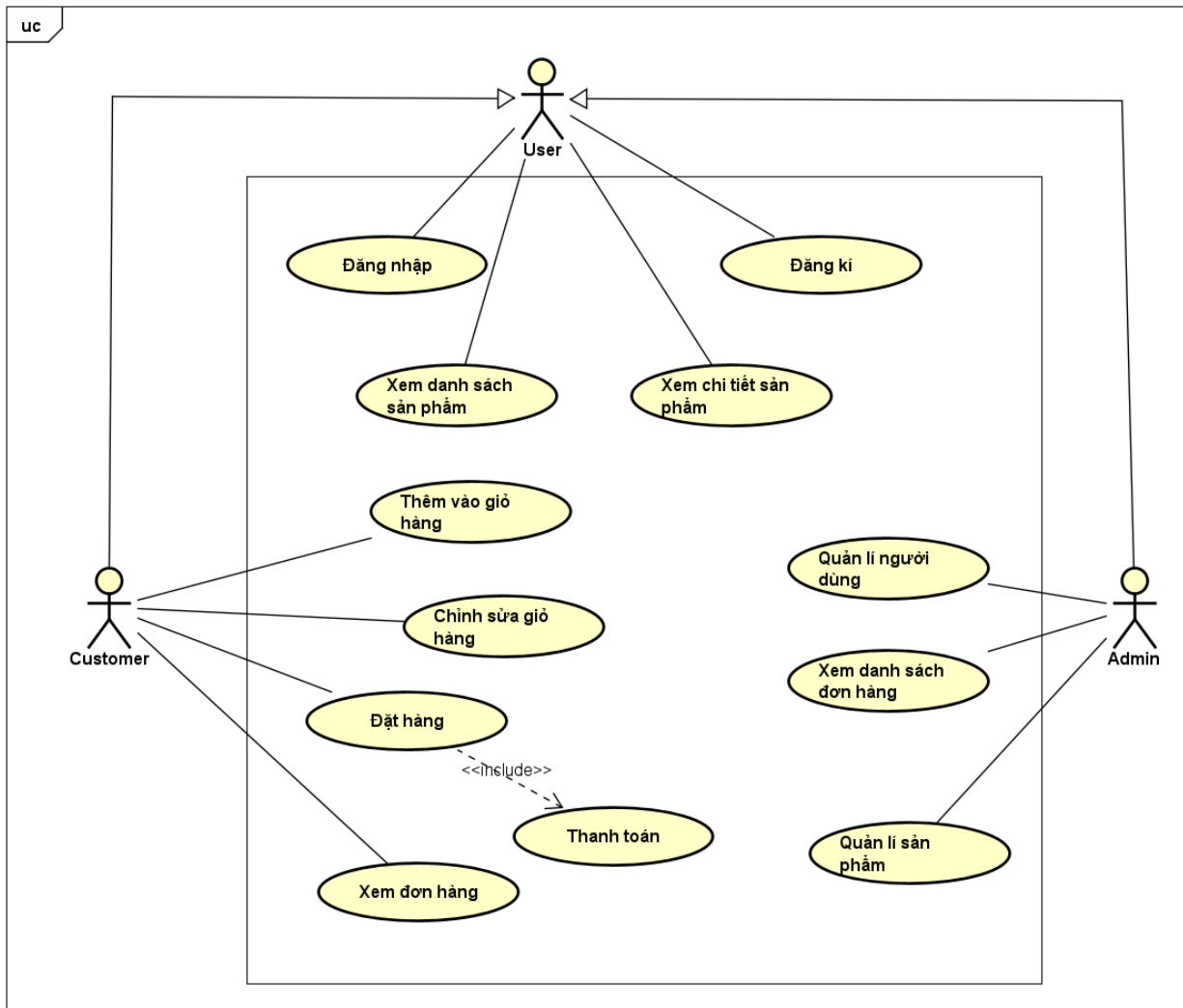
## Team's members contribution

Tên	MSSV	Nhiệm vụ	Đóng góp
Tạ Hữu Đăng	20194010	Sửa sản phẩm, Quản lí order, thanh toán	30%
Ngô Hoàng Hải Đăng	20204639	Thêm sản phẩm mới	20%
Trần Văn Điền	20204527	Đăng ký, Đăng nhập, Quản lý người dùng	20%
Trịnh Quốc Đạt	20194015	Giỏ hàng, Đặt hàng, Hóa đơn	30%

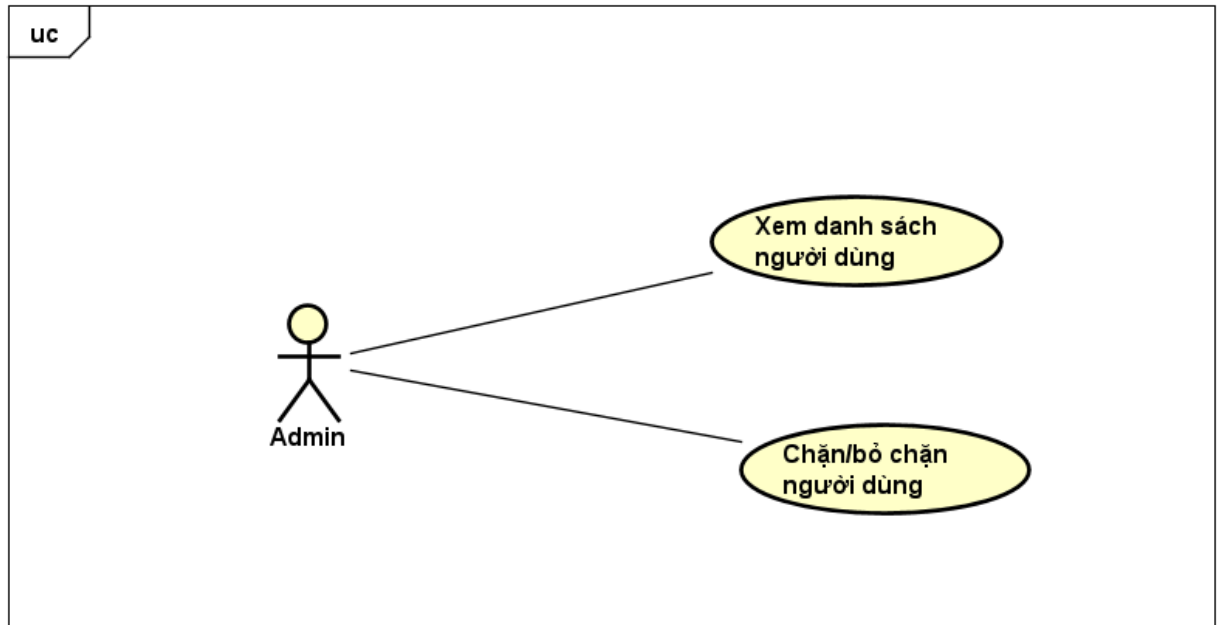
# I. Đặc tả thiết kế

## 1. UseCase Diagram

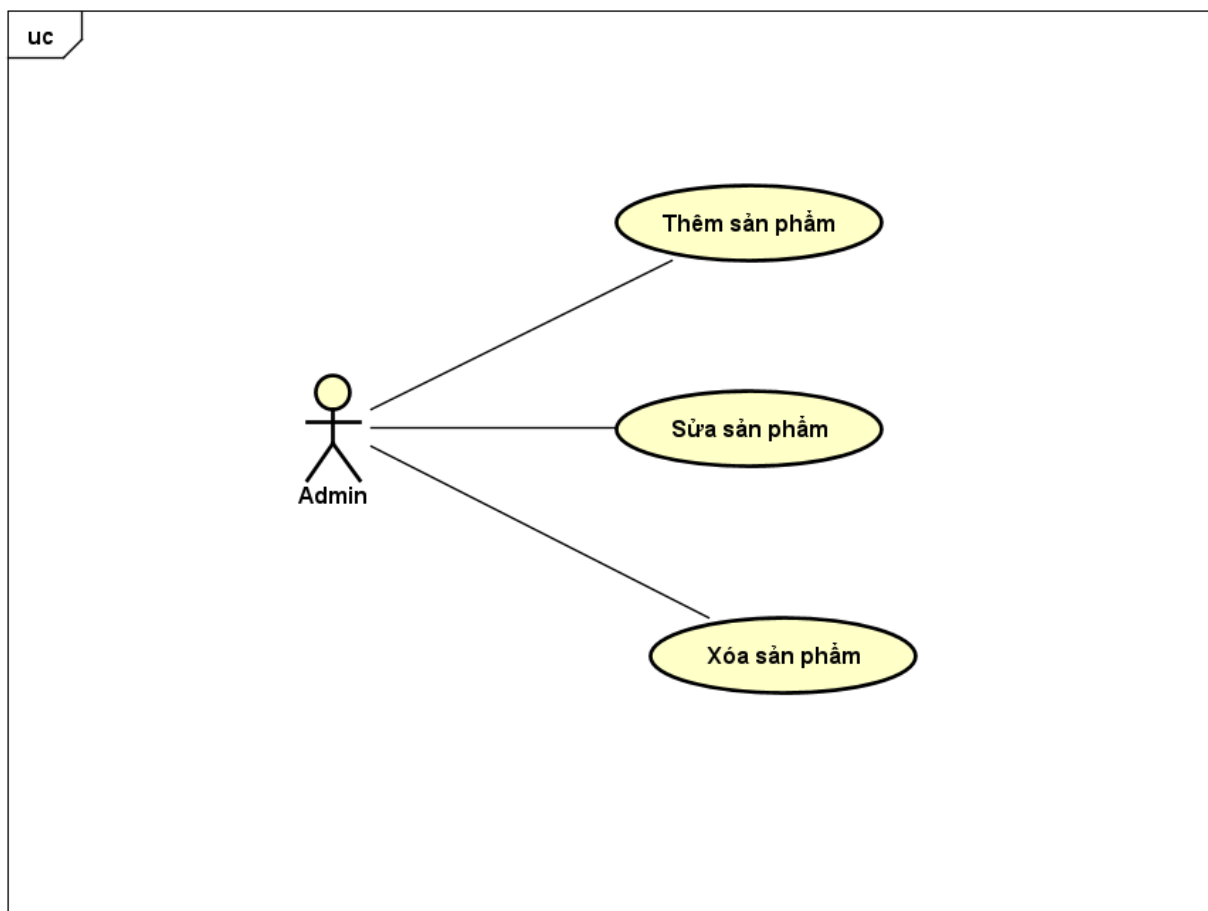
- Tổng quan



- Phân rã quản lí người dùng



Phân rã quản lí sản phẩm



Phân rã quản lí order

uc

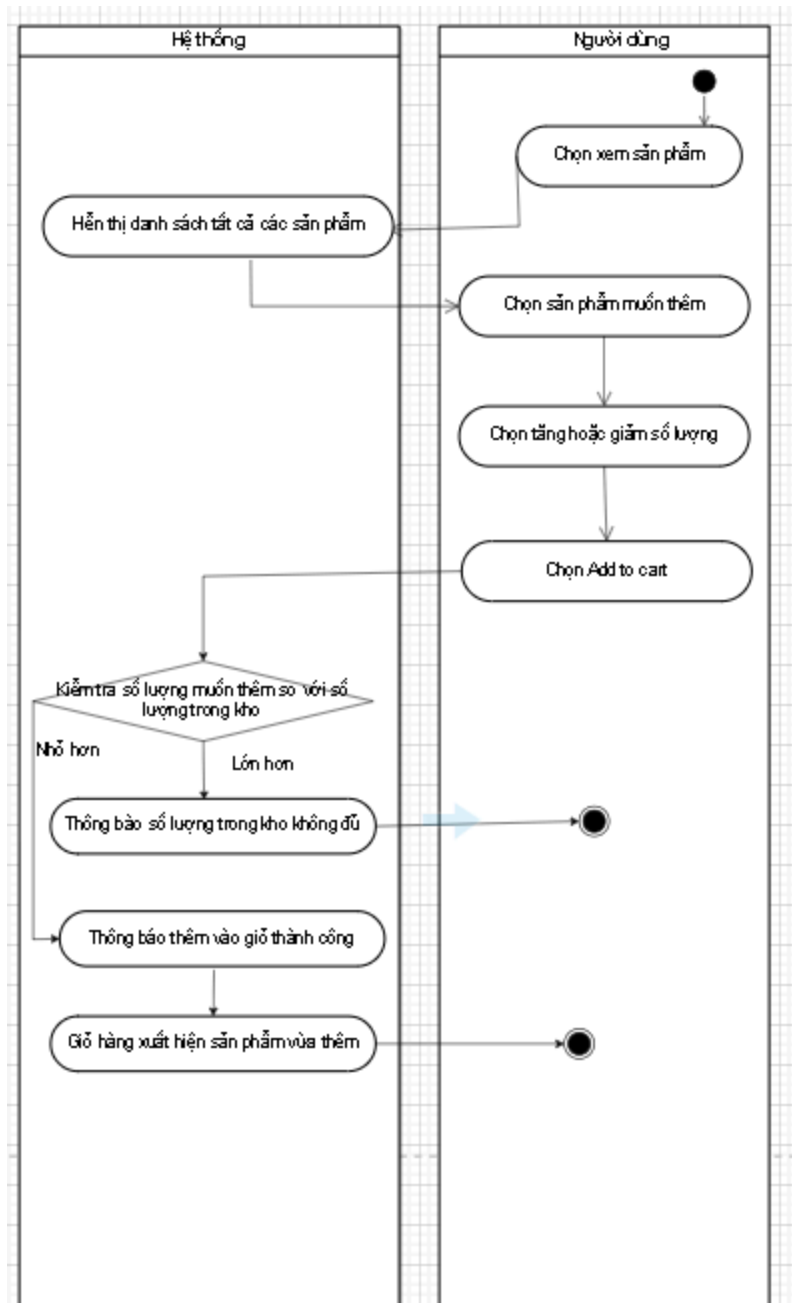


Xem danh sách  
order

Xem chi tiết  
order

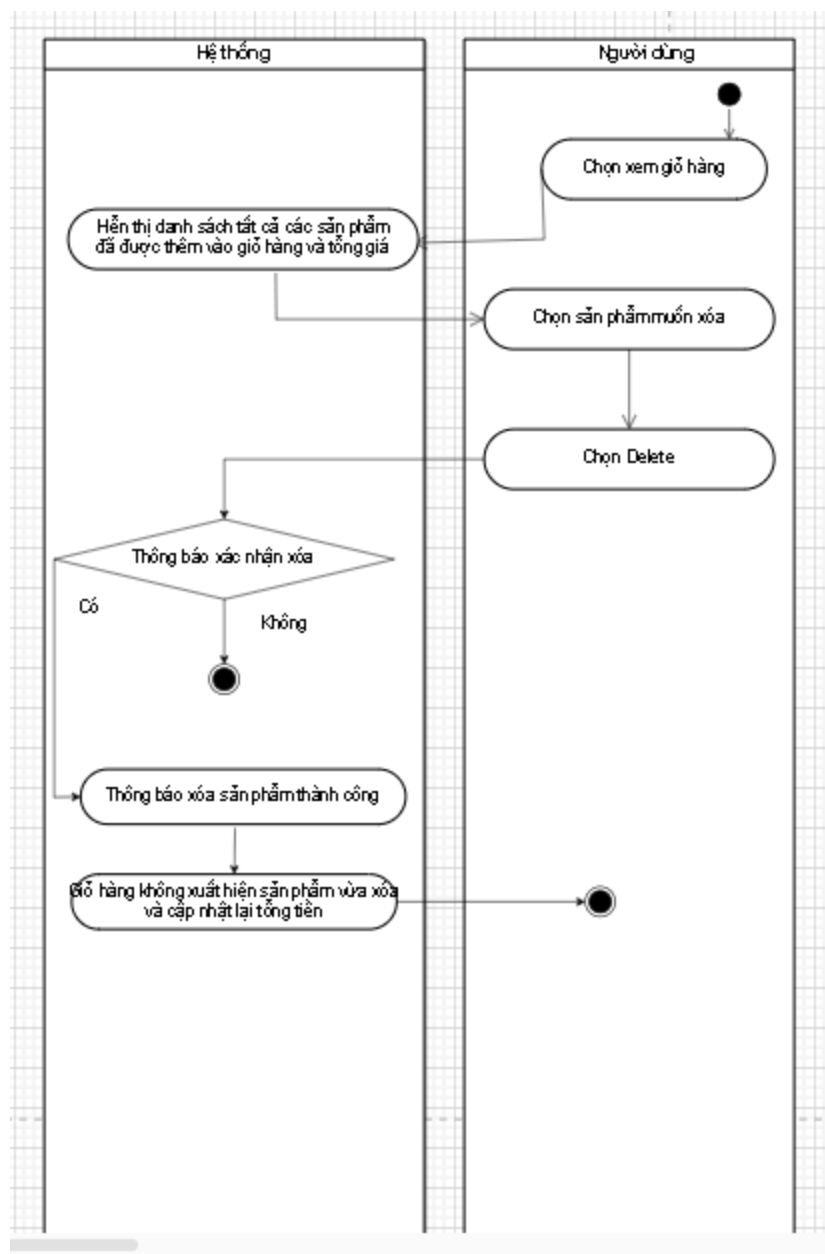
## **2. Activity Diagram**

- Thêm sản phẩm vào giỏ hàng:

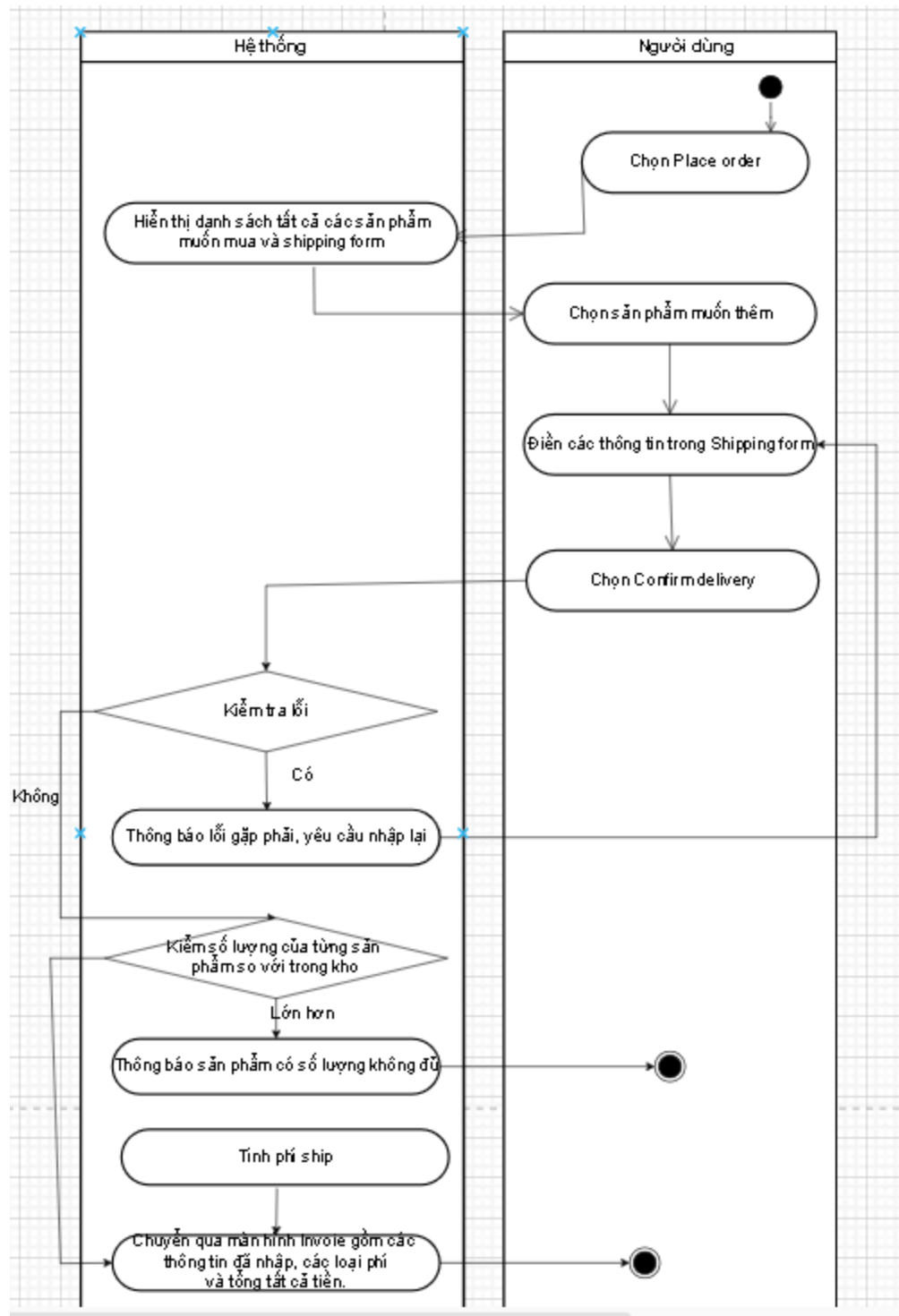


- Xóa sản phẩm khỏi giỏ hàng:

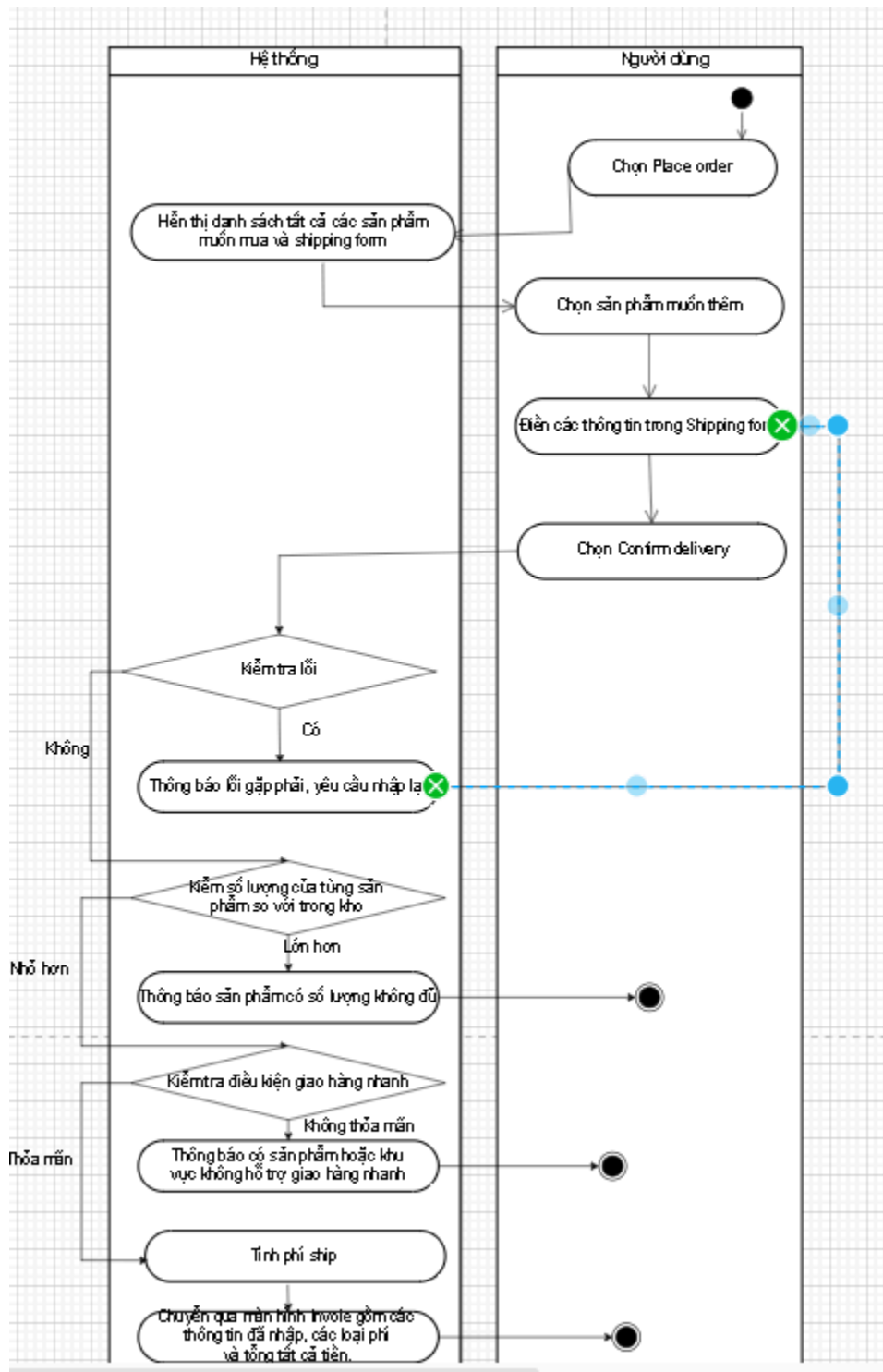




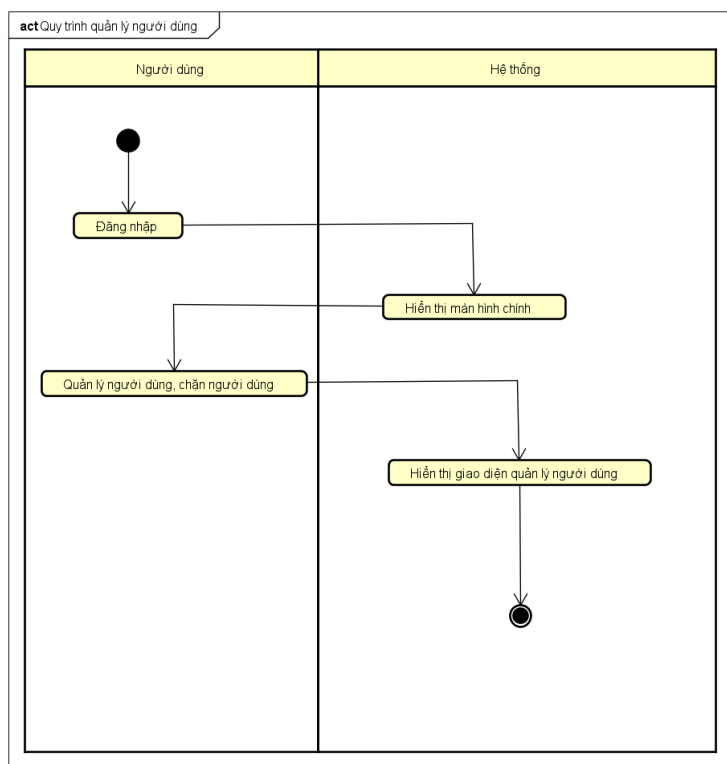
- Đặt hàng:



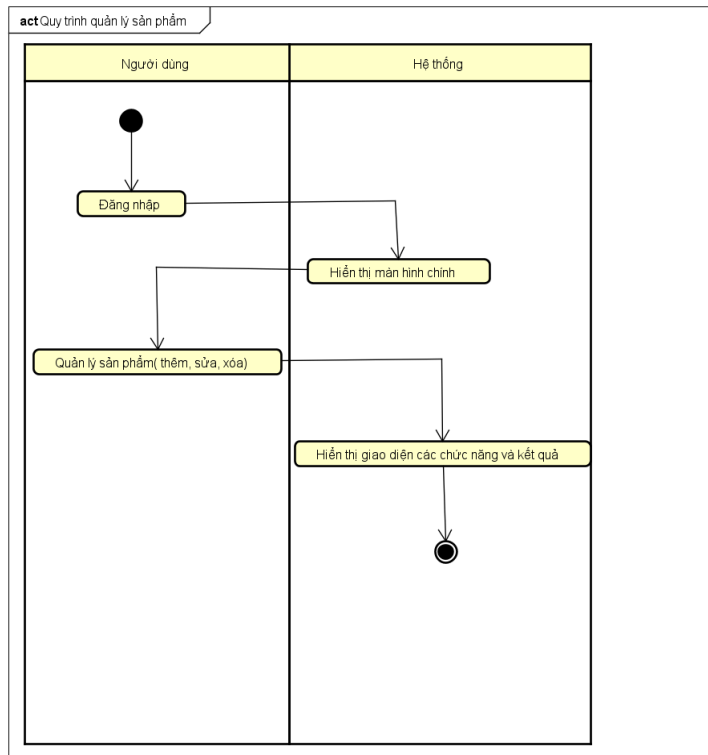
- Đặt hàng nhanh:



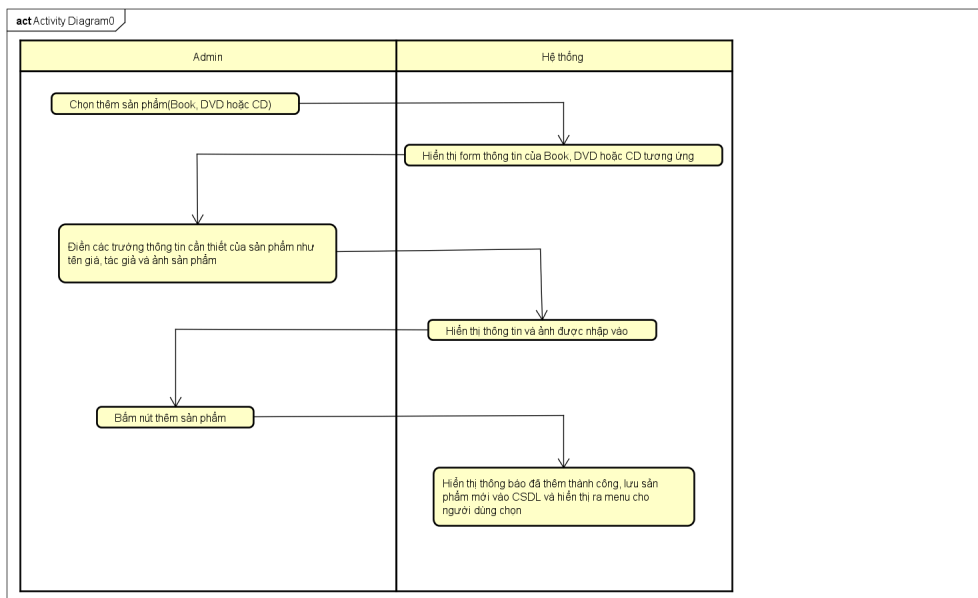
- Quản lý người dùng:



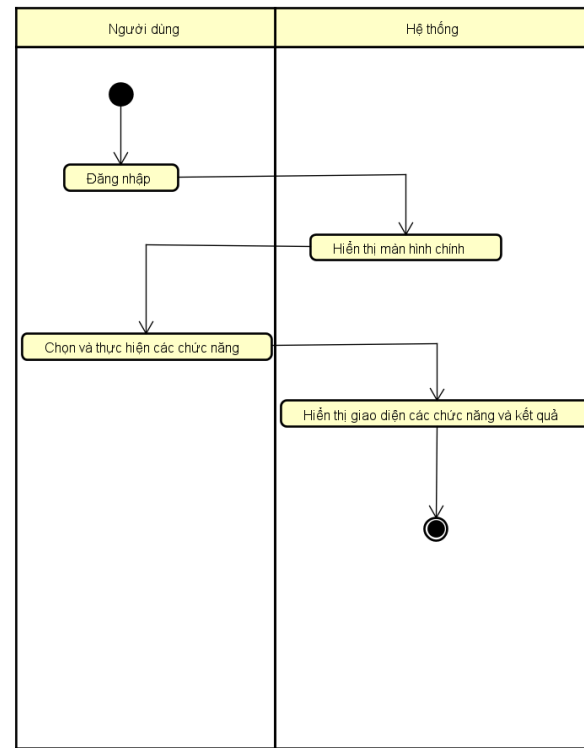
- Quản lý sản phẩm:



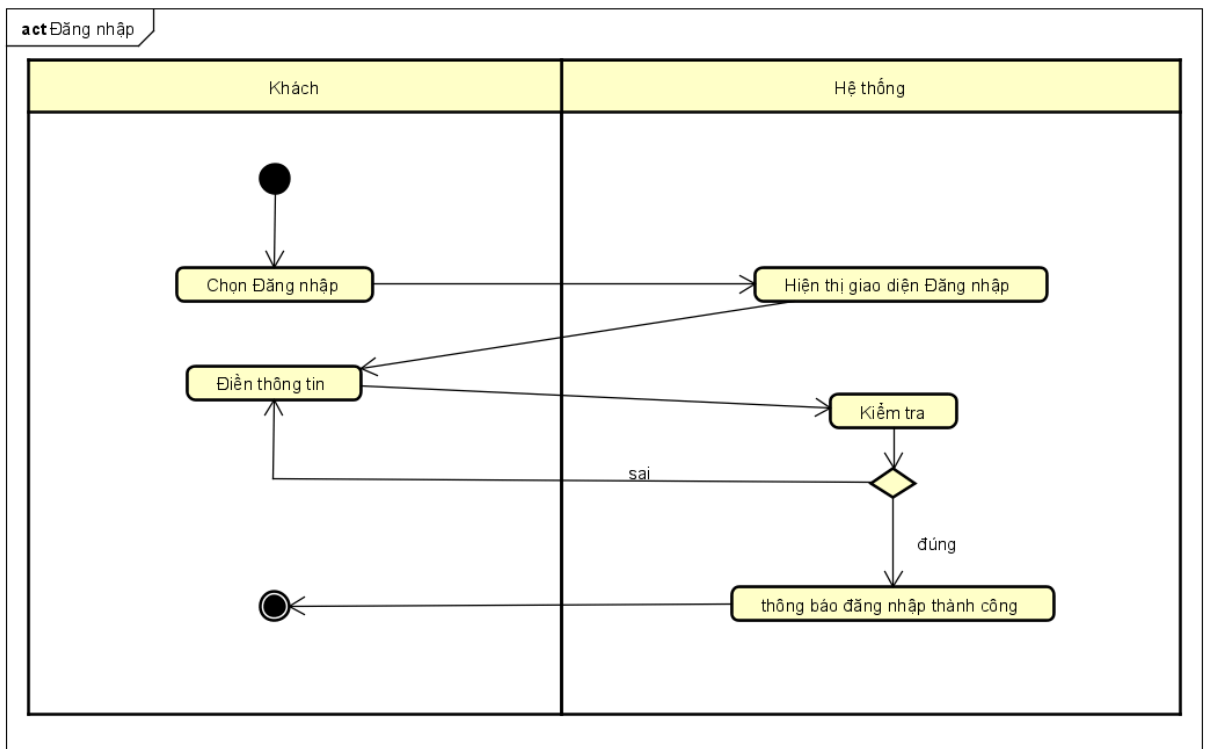
- Admin thêm sản phẩm mới vào Menu:



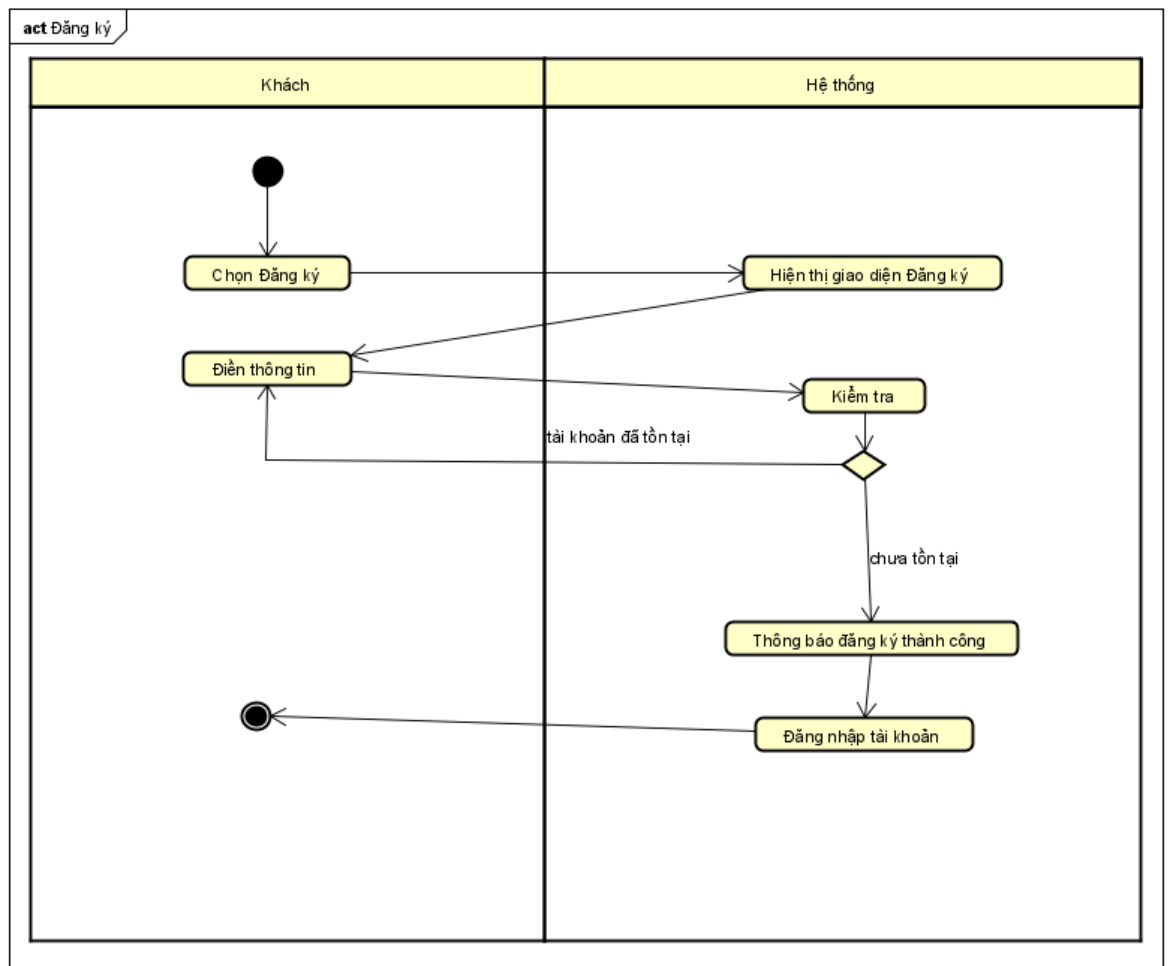
- Quản lý phần mềm:



- Đăng nhập:

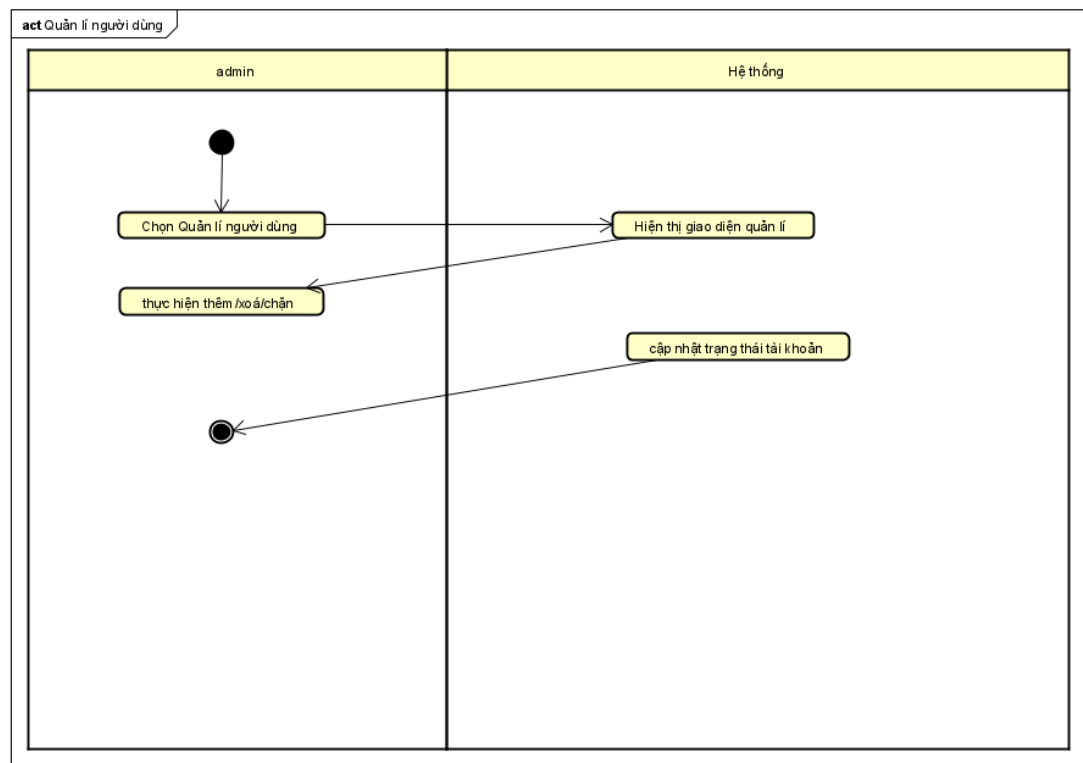


- Đăng ký:



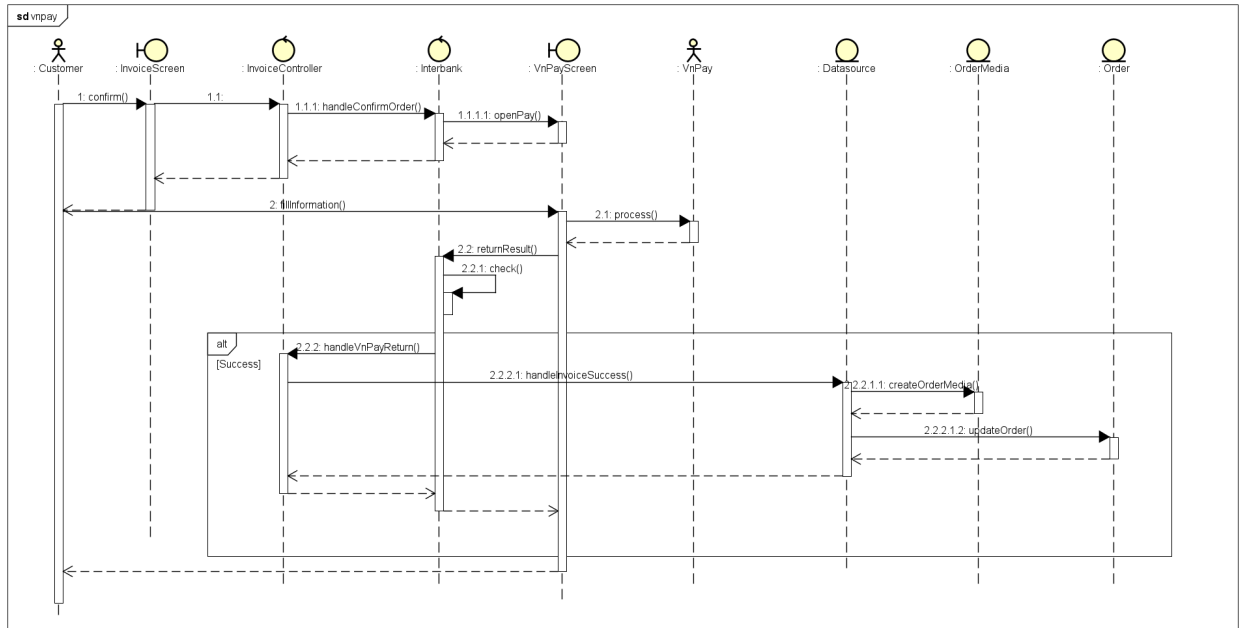
- Quản lý người dùng:



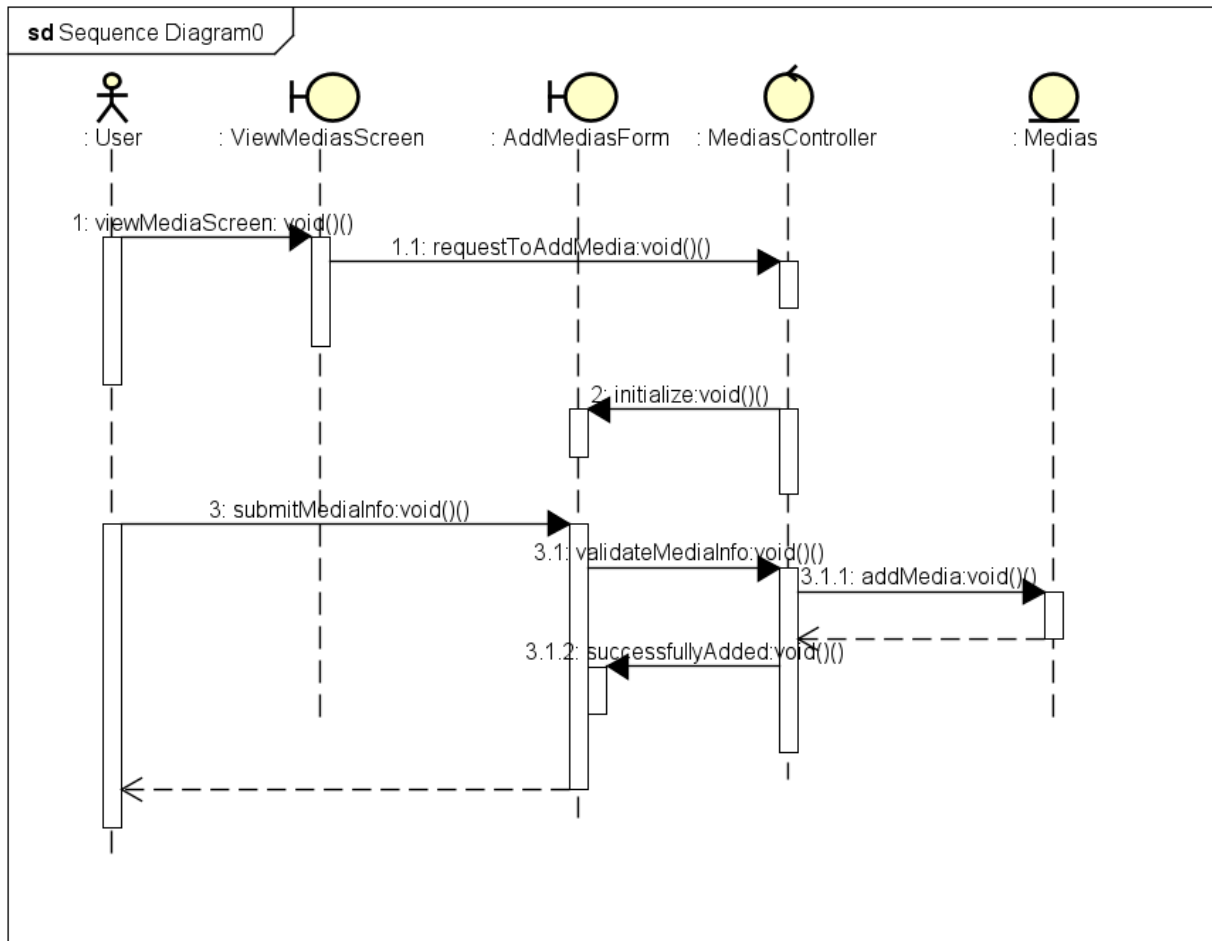


### 3. Sequence Diagram

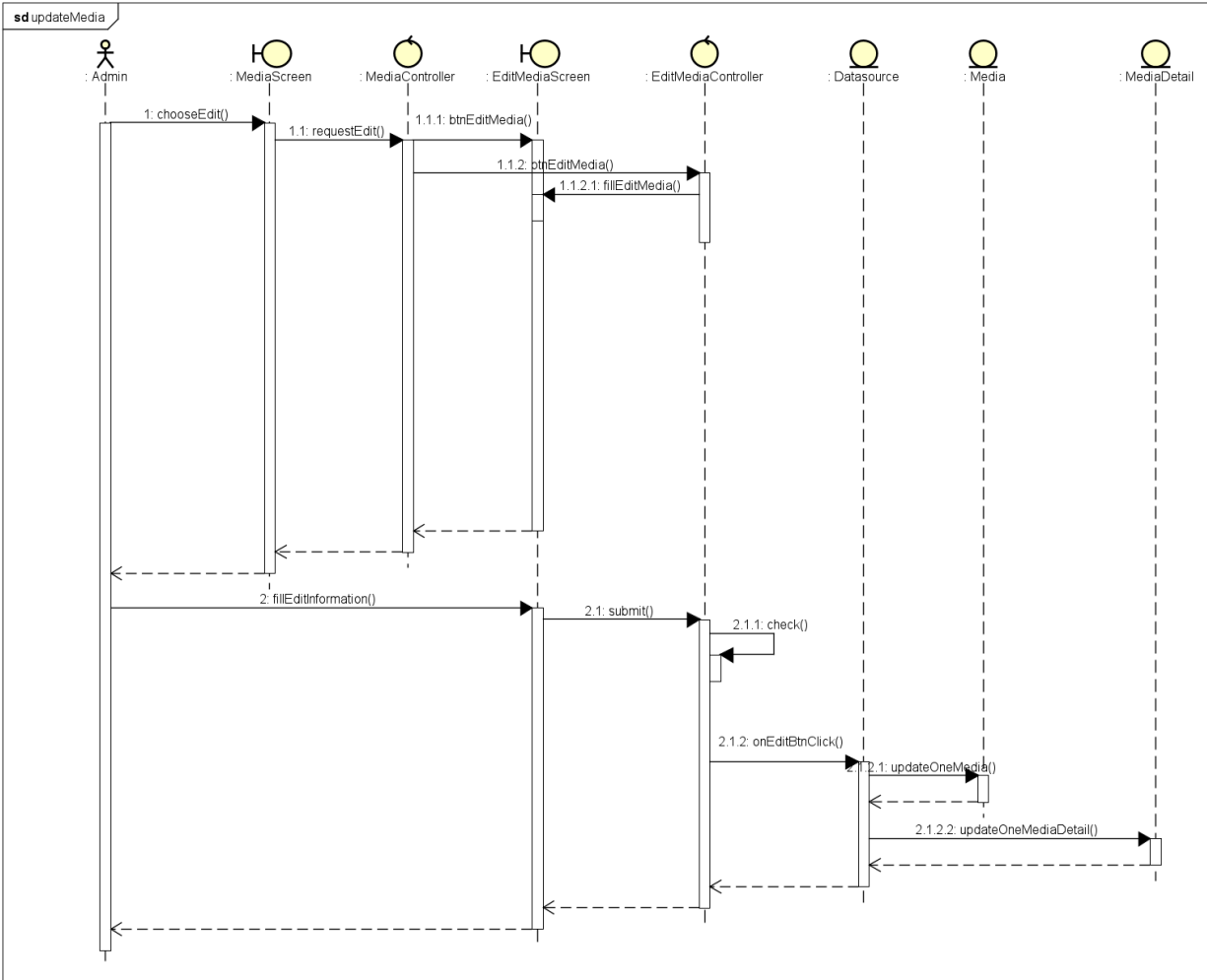
- Thanh toán bằng VNPay:



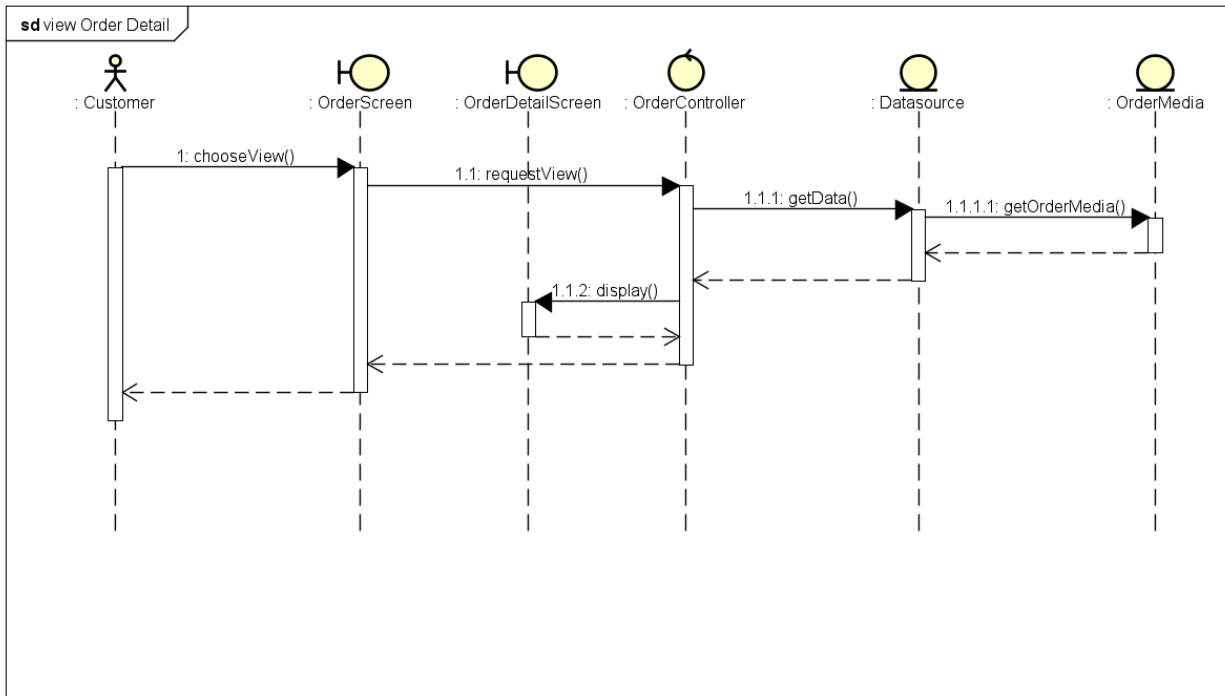
- Admin thêm sản phẩm:



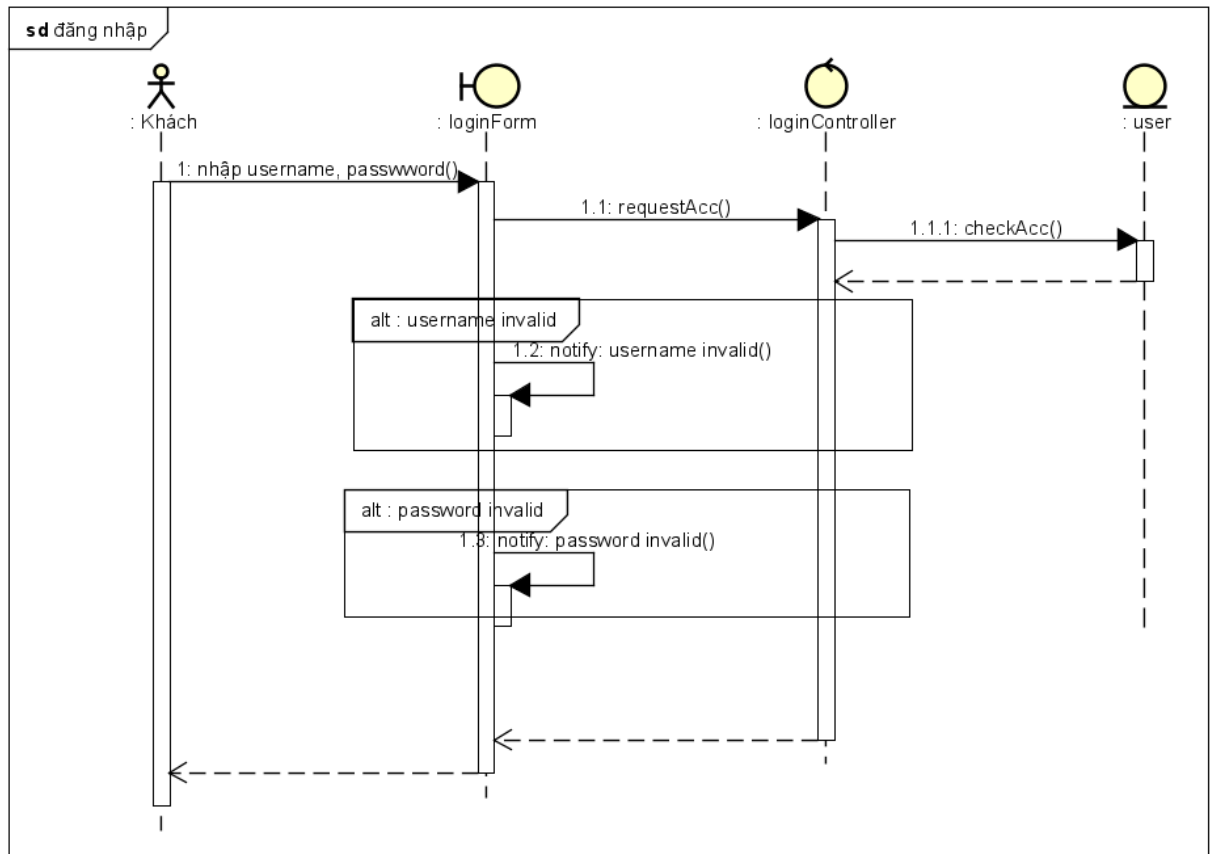
- Admin sửa sản phẩm



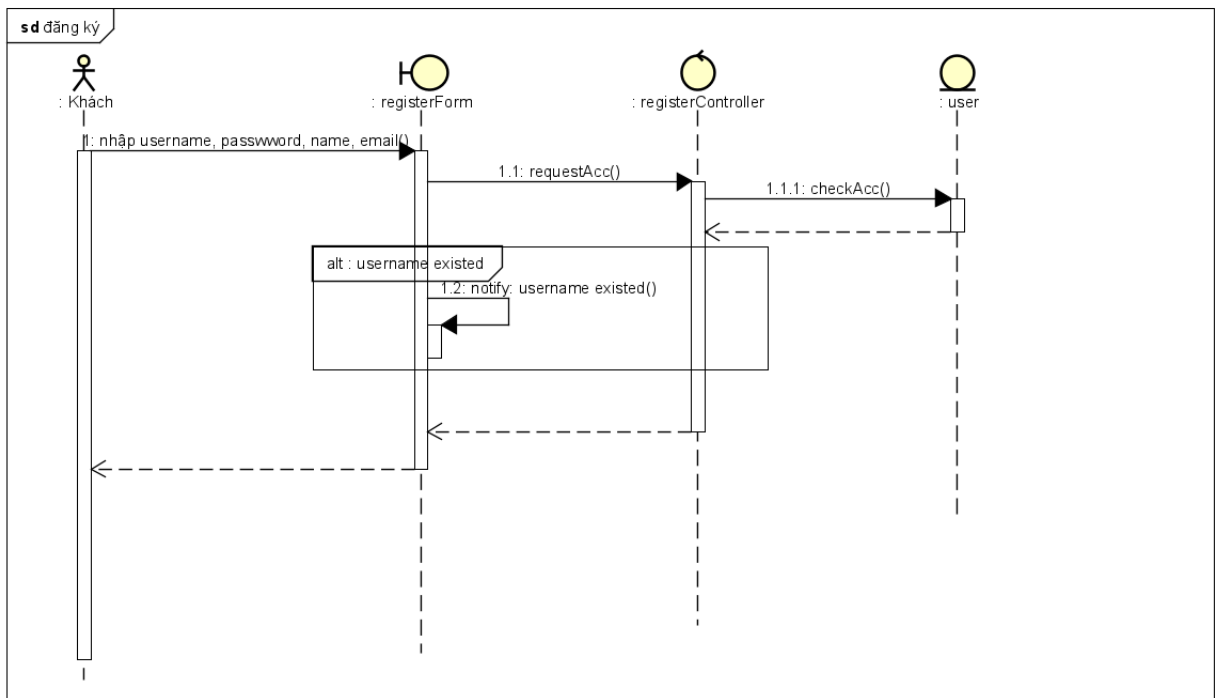
- Xem chi tiết order



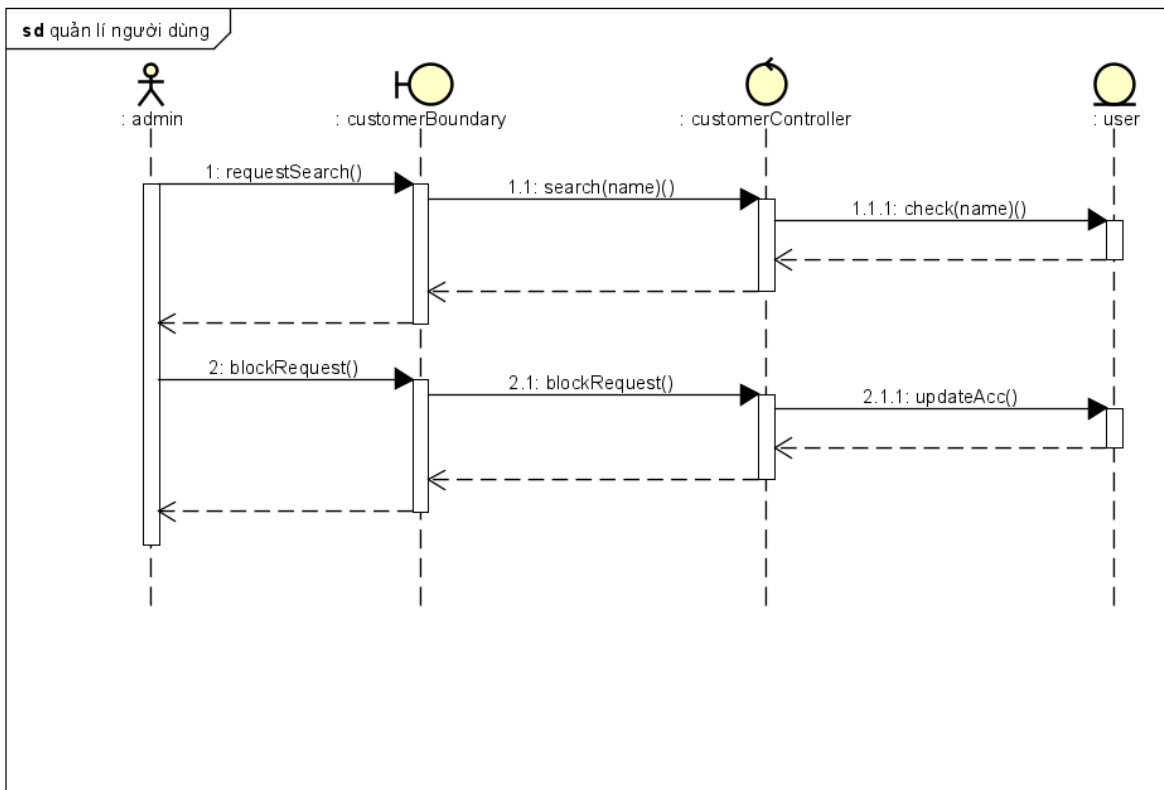
- Đăng nhập:



- Đăng ký:

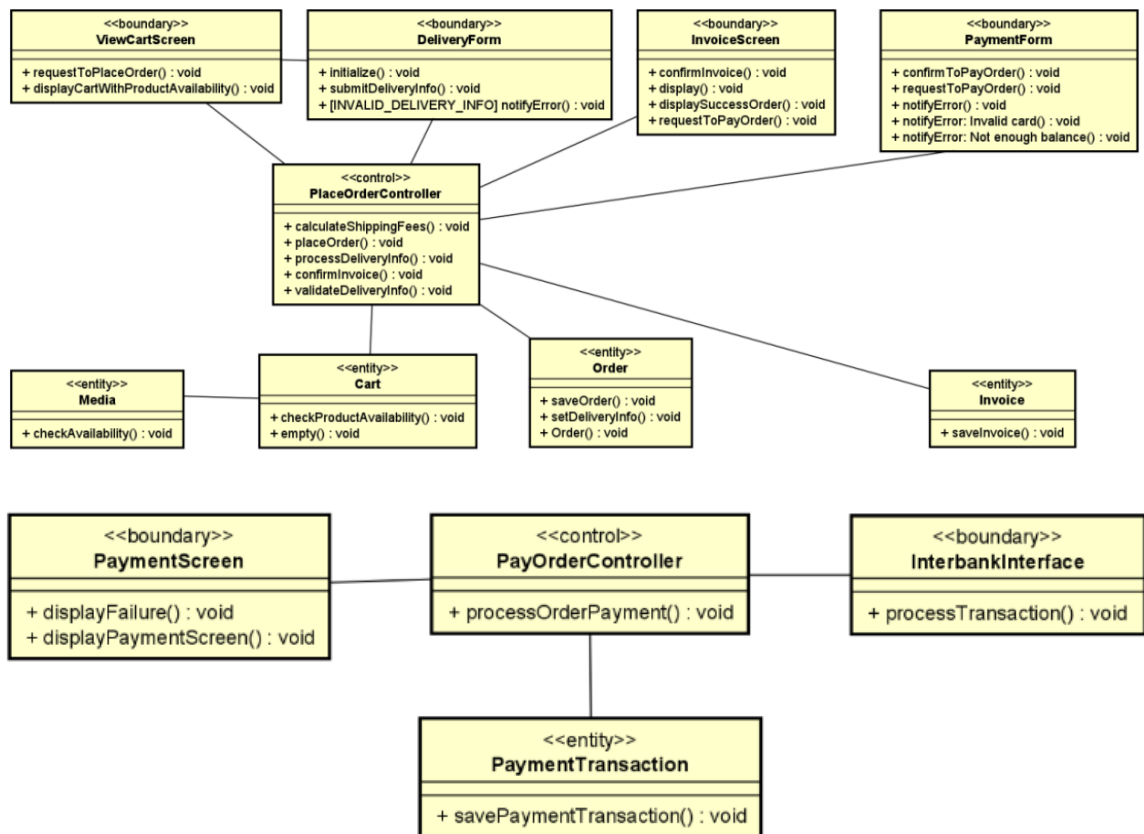


- Quản lý người dùng:



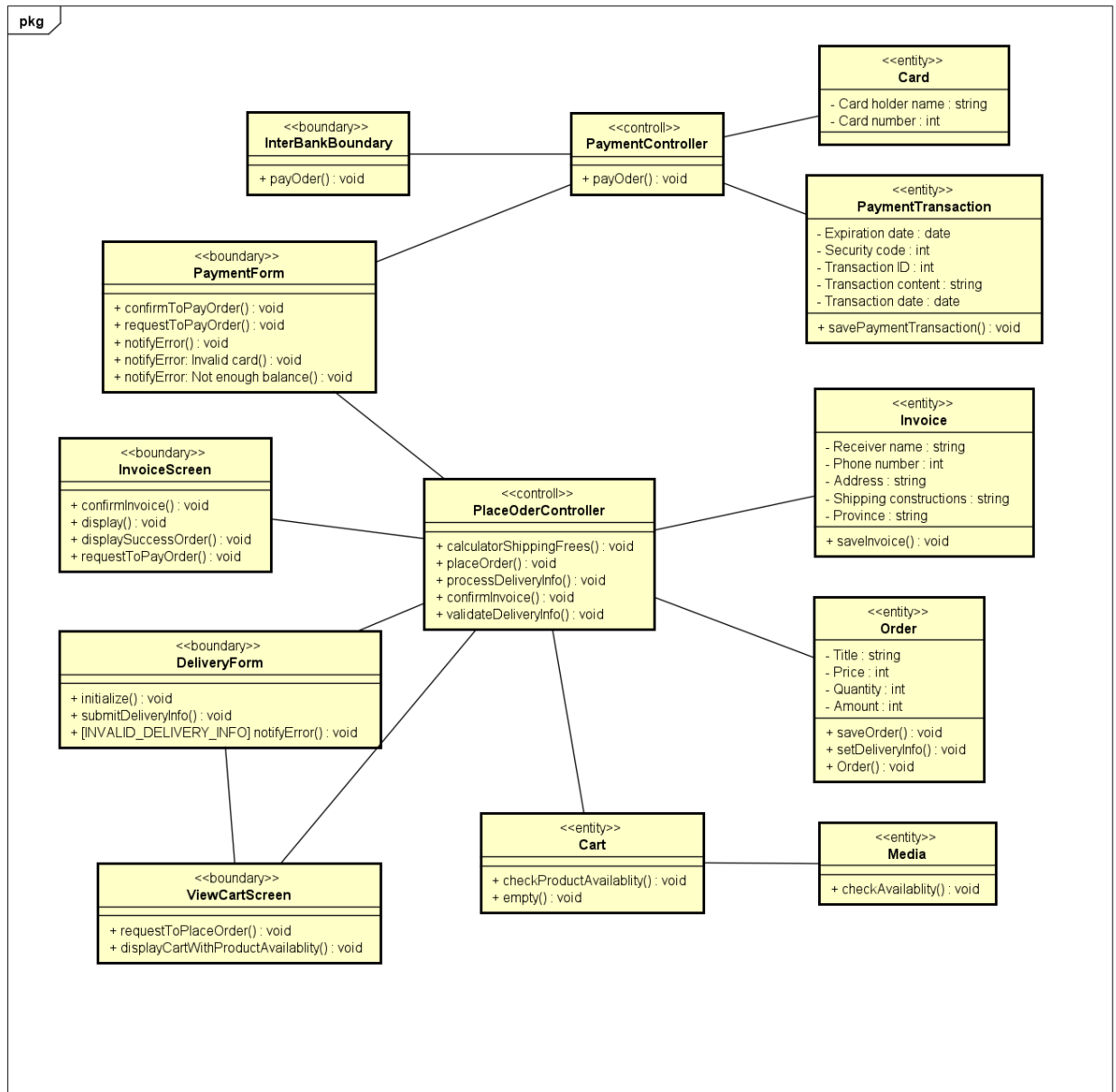
## 4. Class Diagram

- Analysis:

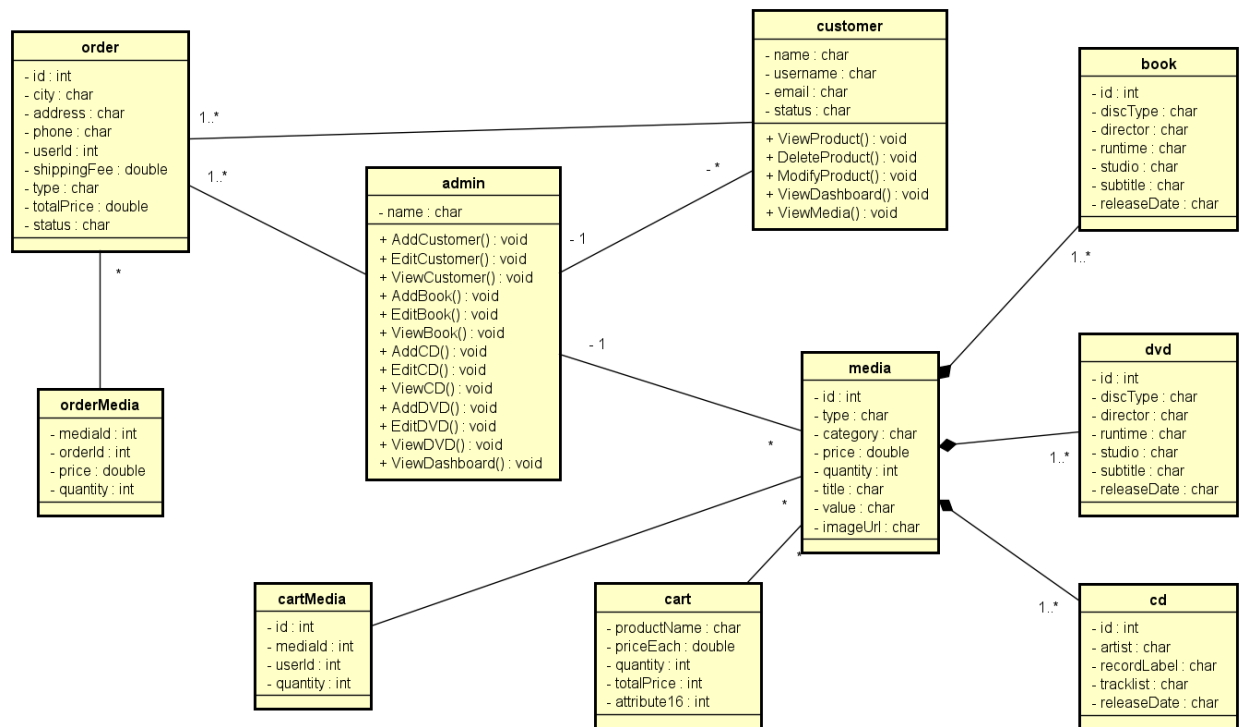




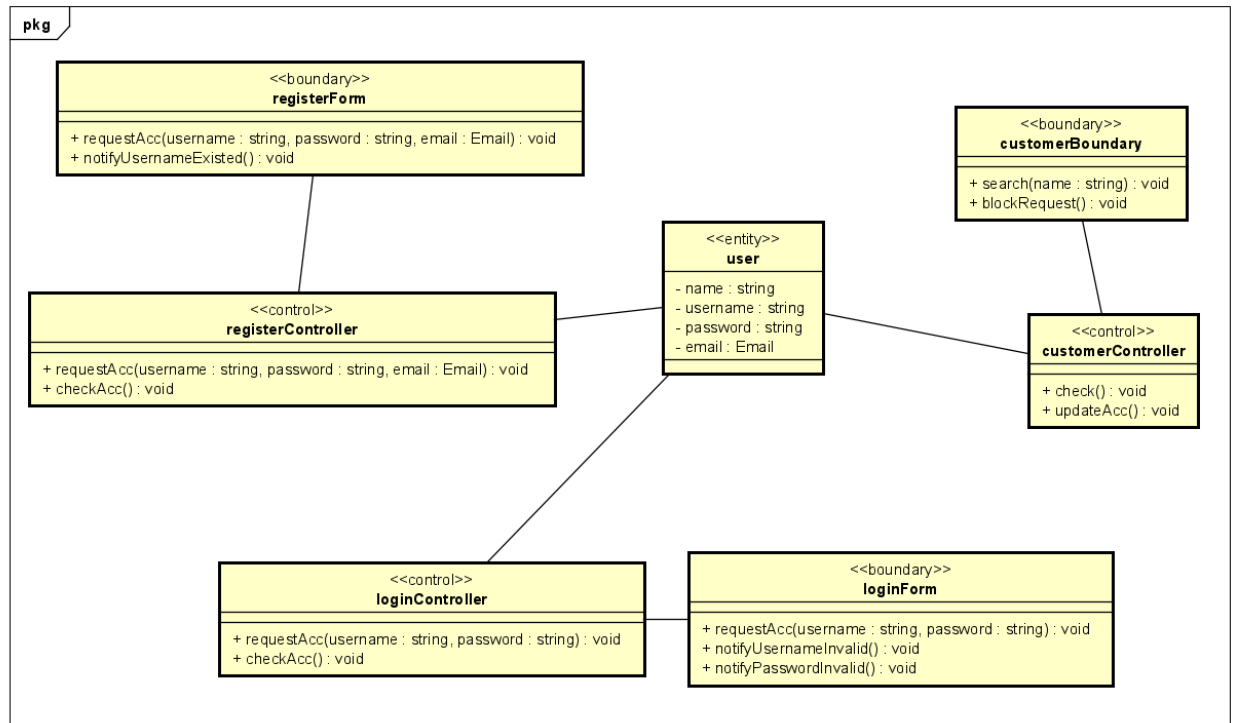
- Detail:



- Relationship:



- Quản lý tài khoản:



## 5. Đặc tả UseCase.

- Thêm sản phẩm vào giỏ hàng:

### 1. Use case code

UC 003

### 2. Brief Description

Use case mô tả cách khách hàng thêm sản phẩm vào giỏ hàng

### 3. Actors

Khách hàng

### 4. Preconditions

Khách hàng đã truy cập trang web và đăng nhập

### 5. Basic Flow of Events

1. Khách hàng xem danh sách sản phẩm hoặc chi tiết sản phẩm.
2. Chọn sản phẩm mà khách hàng muốn thêm vào giỏ hàng.
3. Tăng giảm số lượng muốn mua
4. Nhấn nút "Thêm vào giỏ hàng".
5. Hệ thống kiểm tra xem sản phẩm có đủ số lượng trong kho hay không.
  - Nếu có, thêm sản phẩm vào giỏ hàng và cập nhật số lượng.
  - Nếu không, hiển thị thông báo lỗi cho khách hàng.
6. Hệ thống cập nhật giỏ hàng và hiển thị thông báo xác nhận.

### 6. Alternative flows

STT	Vị trí	Điều kiện	Hành động	Vị trí tiếp tục
1	Tại bước 5	Nếu số lượng sản phẩm trong kho không đủ	<ul style="list-style-type: none"><li>· Hiển thị thông báo lỗi cho khách hàng.</li><li>· Khách hàng có thể quay lại danh sách sản phẩm hoặc tiếp tục mua sắm.</li></ul>	Tiếp tục tại bước 2

## **7. Input data**

- Danh sách sản phẩm
- Sản phẩm đã chọn
- Số lượng

## **8. Output data**

- Trạng thái giỏ hàng: Hệ thống cập nhật giỏ hàng với thông tin mới nhất.
- Thông báo xác nhận hoặc lỗi.

## **9. Postconditions**

Giỏ hàng của khách hàng được cập nhật với thông tin mới nhất về sản phẩm và số lượng.

## **- Đặt hàng:**

### **1. Use case code**

UC 001

### **2. Brief Description**

Use case mô tả tương tác giữa người dùng và hệ thống khi thanh toán.

### **3. Actors**

Khách hàng

### **4. Preconditions**

Khách hàng đã truy cập trang web và đăng nhập

### **5. Basic Flow of Events**

1. Người dùng chọn thanh toán đơn hàng
2. Hệ thống hiển thị giao diện thanh toán
3. Người dùng chọn phương thức thanh toán

4. Hệ thống yêu cầu khách hàng cho thêm thông tin nếu chọn thanh toán bằng thẻ tín dụng
5. Người dùng điền thông tin (nếu có)
6. Hệ thống hiển thị và cập nhật thông tin đơn hàng

## 6. Alternative flows

STT	Vị trí	Điều kiện	Hành động	Vị trí tiếp tục
1	Tại bước 6	Nếu có sản phẩm có số lượng cần mua lớn hơn số lượng trong kho thiếu sản phẩm so với đơn hàng	§ Hệ thống thông báo có sản phẩm không đủ trong kho	Tiếp tục tại bước 3
2	Tại bước 5	Nếu người dùng đã đăng nhập	§ Người dùng: có thể không cần nhập lại thông tin	Tiếp tục tại bước 6
3	Tại bước 6	Nếu thông tin không hợp lệ	§ Hệ thống yêu cầu người dùng nhập lại	Tiếp tục tại bước 5
4	Tại bước 6	Nếu khách hàng chọn Rush ở type mà city được chọn không ở Hà Nội hoặc có sản phẩm không hỗ trợ giao hàng nhanh.	§ Hệ thống thông báo không thể chọn giao hàng nhanh	Tiếp tục tại bước 5
5	Tại bước 8	Nếu người dùng chọn Thanh toán	§ Hệ thống : Hiện thị giao diện thanh toán	Sang giao diện “Thanh toán”

## 7. Input data

No	Data fields	Description	Mandatory	Valid condition	Example
1.	Receiver Name		Yes		Do Minh Hieu
2.	Phone Number		Yes	10 digits	0987654321
3.	Province	Choose from a list	Yes		Hanoi
4.	Address		Yes		12, 34 Alley of Tran Thai Tong street, Cau Giay district
5.	Shipping instructions		No		
6.	Type	Choose Normal or Rush	Yes		Normal

## 8. Output data

No	Data fields	Description	Display format	Example
1.	Title	Tên sản phẩm		DVD Phim Vượt ngục



2.	Price	Giá sản phẩm		123,000
3.	Quantity	Số lượng sản phẩm		2
4.	Amount	Giá sản phẩm		250,000
5.	Subtotal	Tổng số tiền trong giỏ hàng		500,000

6.	Shipping fees	Phí ship		30,000
7.	Total	Tổng tất cả tiền		2,346,600
8.	Name			Do Minh Hieu
9.	Phone number			0987654321
10.	Province	Choose from a list		Hanoi
11.	Address			12, 34 Alley of Tran Thai Tong street, Cau Giay district
12.	Shipping instructions			
13.	Type	Choose normal or rush		Rush

## **9. Postconditions**

### **- Xóa sản phẩm vào giỏ hàng:**

#### **1. Use case code**

UC 004

#### **2. Brief Description**

Use case Cho phép khách hàng xóa sản phẩm không mong muốn khỏi giỏ hàng của mình.

#### **3. Actors**

Khách hàng

#### **4. Preconditions**

Khách hàng đã truy cập trang giỏ hàng và đã thêm ít nhất một sản phẩm vào giỏ..

#### **5. Basic Flow of Events**

1. Khách hàng truy cập trang giỏ hàng.
2. Xem danh sách sản phẩm trong giỏ hàng.
3. Chọn sản phẩm mà khách hàng muốn xóa khỏi giỏ hàng.
4. Nhấn nút "Xóa" .
5. Hệ thống hiển thị thông báo xác nhận về việc xóa sản phẩm khỏi giỏ hàng.
6. Cập nhật giỏ hàng bằng cách loại bỏ sản phẩm đã chọn.
7. Cập nhật tổng giá tiền.

#### **6. Alternative flows**

Không

#### **7. Input data**

- Danh sách sản phẩm trong giỏ hàng
- Sản phẩm cần xóa

#### **8. Output data**

- Trạng thái giỏ hàng được cập nhật
- Thông báo xác nhận hoặc lỗi.

#### **9. Postconditions**

Giỏ hàng của khách hàng được cập nhật sau khi sản phẩm được xóa khỏi đó.

### **- Thêm sản phẩm mới:**

#### **3.1 Specification of Use case UC001 - “<Admin thêm sản phẩm mới>”**

##### **1. Use case code**

- UC001

##### **2. Brief Description**

- Use case để Admin thêm sản phẩm mới vào Medias

##### **3. Actors**

- Tác nhân: Admin

##### **4. Preconditions**

- Tiền điều kiện: Đã đăng nhập

##### **5. Basic Flow of Events**

1. Admin chọn “Medias” ở giao diện chính
2. Hệ thống hiển thị các media hiện có
3. Admin chọn Add CD hoặc Book hay DVD đều tương tự nhau
4. Hiển thị form thông tin để admin điền
5. Admin điền thông tin, thêm ảnh sản phẩm
6. Hệ thống hiển thị thông tin sản phẩm mới
7. Admin bấm vào thêm sản phẩm
8. Hệ thống xác nhận thêm sản phẩm thành công, thêm dữ liệu vào CSDL
9. Kết thúc

## 6. Alternative flows

Table N-Alternative flows of events for UC Place order

No	Location	Condition	Action	Resume location
1.	Ở bước 5	Nếu Admin thoát mà chưa điền hết thông tin	▪ Hệ thống thoát ra ngoài và sản phẩm sẽ chưa được thêm	Tiếp tục ở bước số 5

## 7. Input data

Table A-Input data of UC Place order

No	Data fields	Description	Mandatory	Valid condition	Example
1.	Media Name		Yes	Tối đa 100 kí tự	The Adventure of Remi
2.	Media Stock		Yes	Số	50
3.	Rush Support		Yes	Boolean	True
4.	Color Type		Yes	Tối đa 50 kí tự	White
5.	Publisher		Yes	Tối đa 200 kí tự	Đông Anh, Hà Nội
6.	Number of Pages		Yes	Số	100
7.	Media Price		Yes	Số	500000.0
8.	Media Category		Yes	Tối đa 100 kí tự	book
9.	Image		Yes	Image	Image
10.	Author		Yes	Tối đa 100 kí tự	Dang
11.	Language		Yes	Tối đa 50 kí tự	English
12.	Publish Date		Yes	Date	12/12/2020

## 8. Output data

- Đầu ra sẽ gồm thông tin như đã điền vào

## 9. Postconditions

- Không có

## -Đăng nhập

### 1. Use case code

### 2. Brief Description

Use case mô tả tương tác giữa khách và hệ thống khi đăng nhập.

### 3. Actors

khách

### 4. Preconditions

### 5. Basic Flow of Events

1. Khách nhập tên tài khoản, mật khẩu
2. Hệ thống kiểm tra thông tin
3. Hệ thống thông báo đăng nhập thành công

### 6. Alternative flows

STT	Vị trí	Điều kiện	Hành động	Vị trí tiếp tục
1	Tại bước 2	Nếu sai thông tin đăng nhập	§ Thông báo sai tên tài khoản/ mật khẩu	Tiếp tục tại bước 1

### 7. Input data

No	Data fields	Description	Mandatory	Valid condition	Example
1.	username	tên tài khoản	Yes		trandien

2.	password	mật khẩu	Yes		123
----	----------	----------	-----	--	-----

#### 8. Output data

#### 9. Postconditions

## -Đăng ký

#### 1. Use case code

#### 2. Brief Description

Use case mô tả tương tác giữa khách và hệ thống khi đăng ký.

#### 3. Actors

khách

#### 4. Preconditions

#### 5. Basic Flow of Events

1. Khách chọn đăng ký
2. Hệ thống hiện form đăng ký
3. Khách nhập tên tài khoản, mật khẩu, email
4. Hệ thống kiểm tra thông tin
5. Hệ thống thông báo đăng ký thành công

#### 6. Alternative flows

STT	Vị trí	Điều kiện	Hành động	Vị trí tiếp tục
1	Tại bước 4	Nếu tài khoản đã tồn tại	§ Thông báo tài khoản đã tồn tại	Tiếp tục tại bước 3

## 7. Input data

No	Data fields	Description	Mandatory	Valid condition	Example
1.	username	tên tài khoản	Yes		trandien
2.	password	mật khẩu	Yes		123
3.	email	email	Yes		test@gmail.com

## 8. Output data

## 9. Postconditions

## -Quản lí người dùng

### 1. Use case code

### 2. Brief Description

Use case mô tả tương tác giữa admin và hệ thống khi thao tác với tài khoản người dùng.

### 3. Actors

admin

### 4. Preconditions

Đăng nhập tài khoản admin

### 5. Basic Flow of Events

1. Admin tìm kiếm tài khoản
2. Hệ thống trả về các tài khoản
3. Admin block tài khoản

#### 4. Hệ thống cập nhật

#### 6. Alternative flows

STT	Vị trí	Điều kiện	Hành động	Vị trí tiếp tục
1	Tại bước 3	Admin unblock tài khoản	§ Hệ thống cập nhật	Kết thúc

#### 7. Input data

#### 8. Output data

#### 9. Postconditions

## Usecase “Thanh toán”

#### 1. Usecase code

UC

#### 2. Brief description

Cho phép khách hàng thanh toán

#### 3. Actors

Khách hàng

#### 4. Preconditions

Khách hàng ở trang hóa đơn

#### 5. Basic Flow of Events

1. Khách hàng chọn đặt hàng



2. Hệ thống mở cửa sổ trình duyệt giao diện thanh toán VNPay
3. Khách hàng điền thông tin và chọn thanh toán
4. VnPay kiểm tra thông tin
5. VnPay xử lí giao dịch và gửi kết quả cho hệ thống
6. Hệ thống xử lí kết quả
7. Hệ thống xác nhận dữ liệu cho đơn hàng
8. Vnpay hiển thị kết quả cho người dùng

#### 6. Alternative flows

No	Location	Condition	Action	Resume location
1.	Bước 3	Khách hàng nhập không chính xác	VnPay yêu cầu nhập lại	Bước 3
2.	Bước 5	Thanh toán Vnpay thất bại	Hệ thống không cập nhật trạng thái xác nhận cho đơn hàng,	Bước 8

## Usecase “Sửa thông tin Media”

### 1. Usecase code

UC

### 2. Brief description

Cho phép admin sửa thông tin Media

### 3. Actors

Admin

### 4. Preconditions

Admin ở trang Media

## 5. Basic Flow of Events

1. Admin chọn sửa Media
2. Hệ thống hiện form sửa Media
3. Admin nhập thông tin cần sửa và nhấn lưu
4. Hệ thống lưu thông tin và hiển thị kết quả

## 6. Alternative flows

No	Location	Condition	Action	Resume location
7.	Bước 3	Khách hàng không nhấn lưu	Hệ thống không lưu thông tin	Usecase kết thúc

# Usecase “Xem chi tiết order”

## 1. Usecase code

UC

## 2. Brief description

Cho phép khách hàng xem chi tiết order đã đặt

## 3. Actors

Khách hàng

## 4. Preconditions

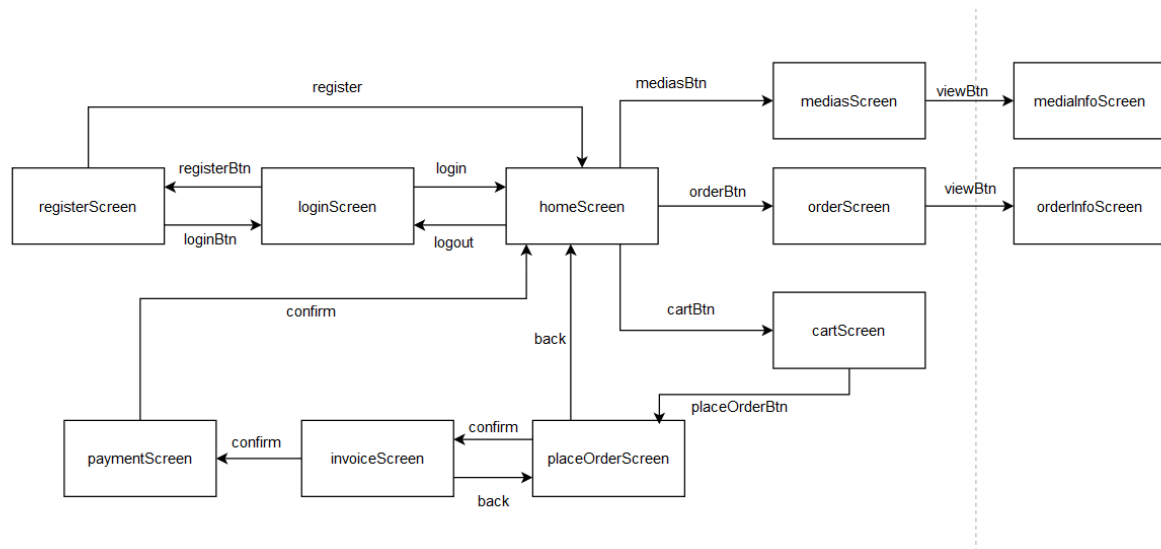
Khách hàng ở trang Orders

## 5. Basic Flow of Events

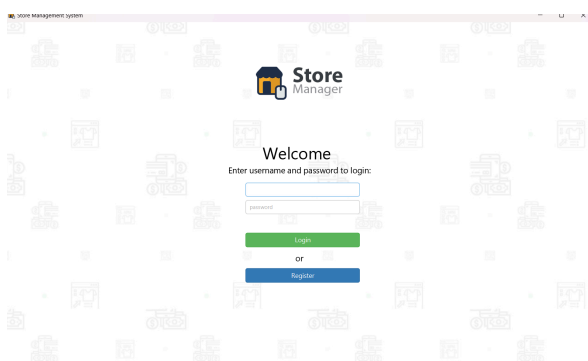
1. Khách hàng chọn xem chi tiết order
2. Hệ thống hiển thị chi tiết order

## II. Interface Design

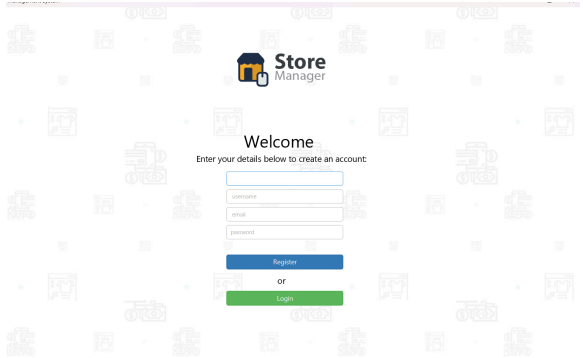
### 1. Screen transition diagram



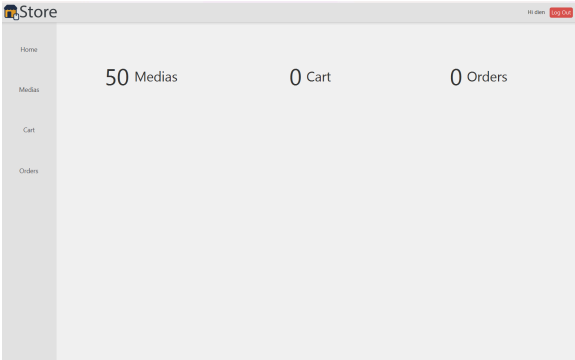
### 2. LoginScreen

Screen	Control	Operation	Function
	2 input login, password	nhập dữ liệu	nơi nhập thông tin tài khoản đăng nhập
	login button	click	đăng nhập, đi đến homeScreen
	register button	click	đến RegisterScreen

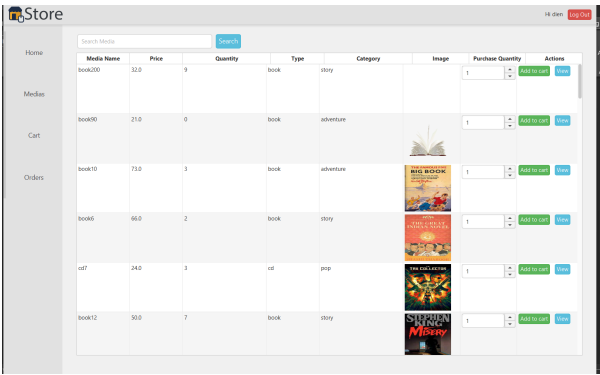
### 3. RegisterScreen

Screen	Control	Operation	Function
	4 input fullname, username, email, password	nhập dữ liệu	nơi nhập thông tin tài khoản đăng ký
	login button	click	đi đến LoginScreen
	register button	click	đăng ký thành công, đi đến homeScreen

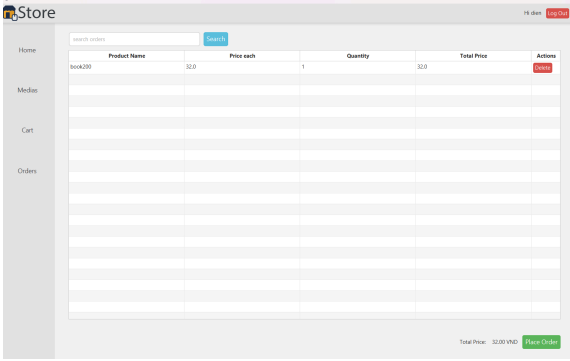
## 4. HomeScreen

Screen	Control	Operation	Function
	màn hình chính		hiển thị số lượng media, số sản phẩm trong cart và số order
	logout button	click	đi đến LoginScreen
	home button	click	ở tại homeScreen
	medias button	click	đi đến MediasScreen
	cart button	click	đi đến cartScreen
	orders button	click	đi đến OrdersScreen

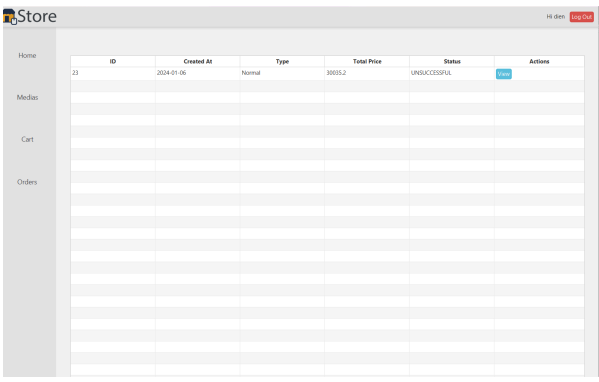
## 5. Medias Screen

Screen	Control	Operation	Function
	màn hình chính		hiển thị thông tin các media
	input search	nhập	nhập tên sản phẩm
	search button	click	hiển thị sản phẩm liên quan lên màn hình chính
	input quantity	nhập	nhập số lượng sản phẩm để thêm vào cart
	add to cart button	click	thêm sản phẩm vào cart
	view button	click	xem thông tin sản phẩm

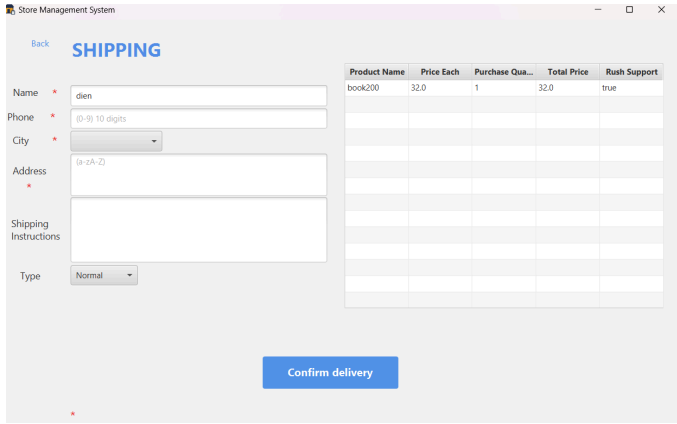
## 6. Cart Screen

Screen	Control	Operation	Function
	màn hình chính		hiển thị thông tin các media
	delete button	click	xoá sản phẩm khỏi cart
	place order button	click	đi đến PlaceOrderScreen

## 7. Order Screen

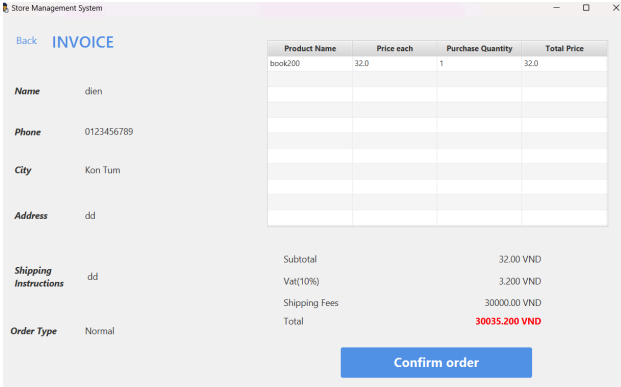
Screen	Control	Operation	Function
	màn hình chính		hiển thị thông tin đơn hàng
	view button	click	xem thông tin đơn hàng

## 8. Place Order Screen

Screen	Control	Operation	Function
	màn hình chính		hiển thị các input và các sản phẩm đặt hàng
	các input	nhập	nhập thông tin giao hàng
	back button	click	quay lại homeScreen
	confirm delivery button	click	đi đến invoiceScreen



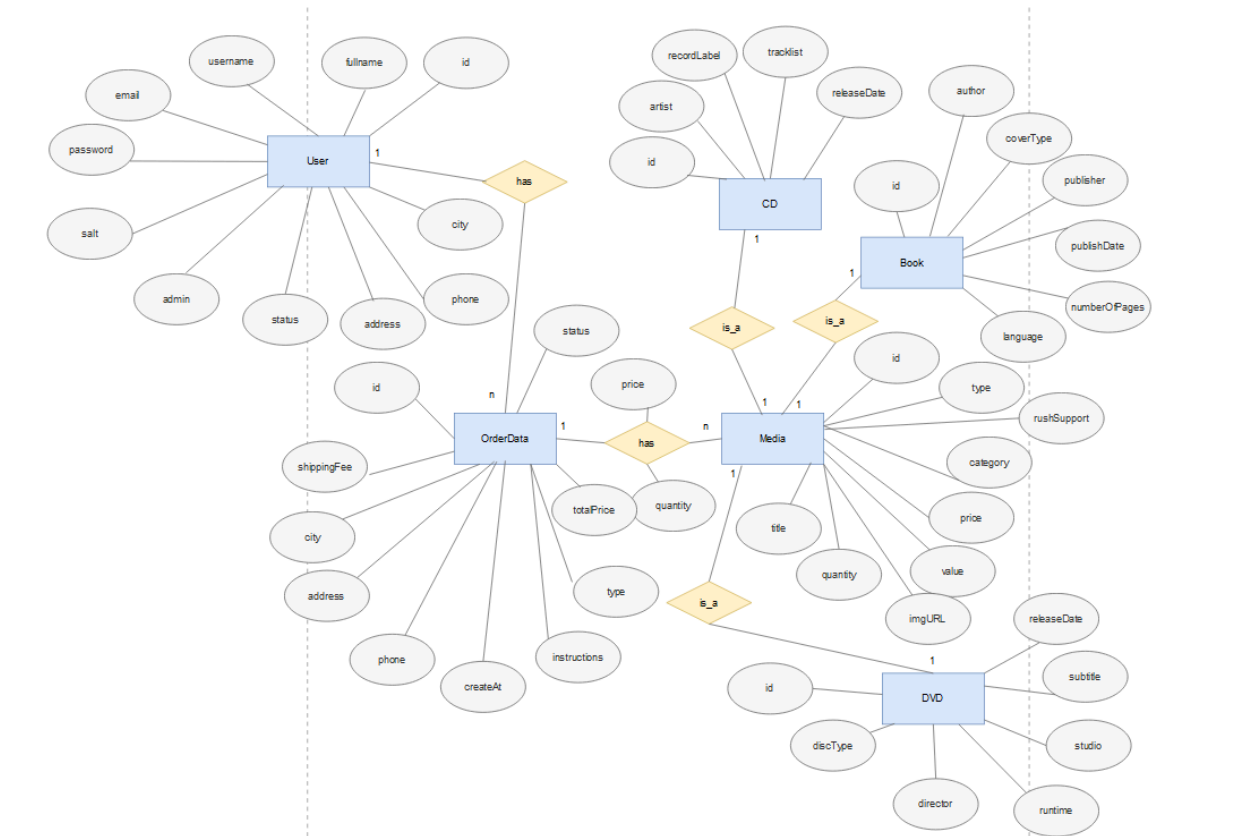
## 9. Invoice Screen

Screen	Control	Operation	Function
	màn hình chính		hiển thị thông tin đơn hàng
	back button	click	quay lại placeOrderScreen
	confirm order button	click	đi đến trang thanh toán

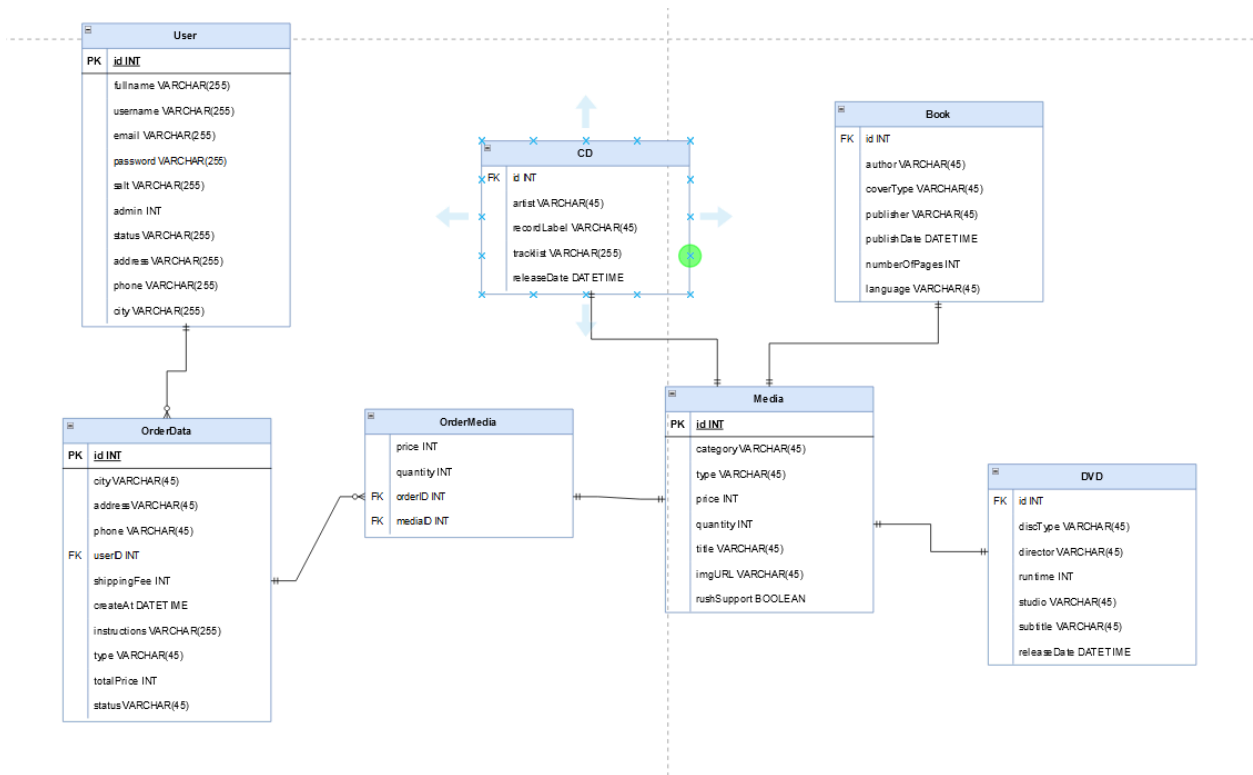
# III. Data Modeling

IV.

## 1. Entity-Relation Diagram



## 2. Database Design



## - Media

#	PK	FK	Column Name	Data type	Mandatory	Description
1.	x		id	INT	Yes	ID, auto increment
2.			category	VARCHAR(45)	Yes	Media type, e.g., CD, DVD
3.			price	INT	Yes	Current price
4.			quantity	INT	Yes	Number of products

#	PK	FK	Column Name	Data type	Mandatory	Description
5.			title	VARCHAR(45)	Yes	Product name
6.			rushSupport	BOOLEAN	Yes	rush support?
7.			imageUrl	VARCHAR(45)	Yes	Product image path

## - CD

#	PK	FK	Column Name	Data type	Mandatory	Description
1.		x	id	INT	Yes	ID, same as ID of Media of which type is CD
2.			artist	VARCHAR(45)	Yes	Artist's name
3.			recordLabel	VARCHAR(45)	Yes	Record label
4.			tracklist	VARCHAR(255)	Yes	List music
5.			releasedDate	DATETIME	No	Release date

## - Book

#	PK	FK	Column Name	Data type	Mandatory	Description
1.		x	id	INT(10)	Yes	ID, same as ID of Media of which type is Book
2.			author	VARCHAR(45)	Yes	Author
3.			coverType	VARCHAR(45)	Yes	Cover type
4.			publisher	VARCHAR(45)	Yes	Publishing house

5.			publishDate	DATETIME	Yes	Date of publishing
6.			numberOfPages	INT	Yes	Page number
7.			language	VARCHAR(45)	Yes	Language

**- DVD**

#	PK	FK	Column Name	Data type	Mandatory	Description
1.		x	id	INT	Yes	ID, same as ID of Media of which type is DVD
2.			discType	VARCHAR(45)	Yes	Disc type
3.			director	VARCHAR(45)	Yes	Director
4.			runtime	INT	Yes	Duration
5.			studio	VARCHAR(45)	Yes	Manufacturer
6.			subtitle	VARCHAR(45)	Yes	Subtitles
7.			releasedDate	DATETIME	Yes	Release date

- **OrderData**

#	PK	FK	Column Name	Data type	Mandatory	Description
1.	X		id	INT	Yes	ID
2.			shippingFees	INT	Yes	Shipping fee
3.			city	VARCHAR(45)	Yes	name city
4.			phone	VARCHAR(45)	Yes	number phone
5.		X	userId	VARCHAR(45)	Yes	user id
6.			createAt	DATETIME	Yes	
7.			instructions	VARCHAR(255)	No	
8.			type	VARCHAR(45)	Yes	
9.			totalPrice	INT	Yes	price of order
10			status	VARCHAR(45)	Yes	status of order

- **OrderMedia**

#	PK	FK	Column Name	Data type	Mandatory	Description
1.		X	mediaID	INT	Yes	Media ID
2.		X	orderID	INT	Yes	Order ID
3.			price	INT	Yes	Selling price
4.			quantity	INT	Yes	Number

- **User**

#	PK	FK	Column Name	Data type	Mandatory	Description
1.	X		id	INT	Yes	ID
2.			fullname	VARCHAR(255)	Yes	full name
3.			username	VARCHAR(255)	Yes	name account
4.			email	VARCHAR(255)	Yes	email
5.			password	VARCHAR(255)	Yes	password
6.			salt	VARCHAR(255)	No	



<b>7.</b>			admin	INT	Yes	user is admin?
<b>8.</b>			status	VARCHAR(255)	Yes	
<b>9.</b>			address	VARCHAR(255)	No	address
<b>10</b>			phone	VARCHAR(255))	No	number phone
<b>11</b>			city	VARCHAR(255)	No	city

### 3. SQL Scripts

- Bảng Book:

```
CREATE TABLE Book (  
    id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,  
    author VARCHAR (45) NOT NULL,  
    coverType VARCHAR (45) NOT NULL,  
    publisher VARCHAR (45) NOT NULL,  
    publishDate DATETIME NOT NULL,  
    numberOfPages INTEGER NOT NULL,  
    language VARCHAR (45) NOT NULL,  
    FOREIGN KEY (id)  
    REFERENCES Media (id) ON DELETE CASCADE  
);
```

- Bảng DVD:

```
CREATE TABLE DVD (  
    id INTEGER PRIMARY KEY NOT NULL,  
    discType VARCHAR (45) NOT NULL,  
    director VARCHAR (45) NOT NULL,  
    runtime INTEGER NOT NULL,  
    studio VARCHAR (45) NOT NULL,  
    subtitle VARCHAR (45) NOT NULL,  
    releaseDate DATETIME,  
    FOREIGN KEY (id)  
    REFERENCES Media (id) ON DELETE CASCADE);
```

- Bảng CD:

```
CREATE TABLE CD (  
    id INTEGER PRIMARY KEY NOT NULL,  
    artist VARCHAR (45) NOT NULL,  
    recordLabel VARCHAR (45) NOT NULL,  
    tracklist VARCHAR (255) NOT NULL,  
    releaseDate DATE,  
    FOREIGN KEY (id)  
    REFERENCES Media (id) ON DELETE CASCADE  
);
```

- Bảng CartMedia:

```
CREATE TABLE CartMedia (  
    id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,  
    mediaId INTEGER NOT NULL,  
    userId INTEGER NOT NULL,  
    quantity INTEGER NOT NULL,  
    FOREIGN KEY (mediaId) REFERENCES Media (id),  
    FOREIGN KEY (userId) REFERENCES User (id)  
);
```

- Bảng User:

```
CREATE TABLE User (  
    id INTEGER NOT NULL,  
    fullname VARCHAR (255) NOT NULL,  
    username VARCHAR (255) NOT NULL,  
    email VARCHAR (255) NOT NULL,  
    password VARCHAR (255) NOT NULL,  
    salt VARCHAR (255),  
    admin INTEGER NOT NULL DEFAULT (3),  
    status varchar (255) NOT NULL DEFAULT 'enabled',  
    address VARCHAR (255),  
    phone VARCHAR (255),  
    city VARCHAR (255),  
    PRIMARY KEY (id AUTOINCREMENT)  
);
```

- Bảng Media:

```
CREATE TABLE Media (  
    id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,  
    type VARCHAR (45) NOT NULL,  
    category VARCHAR (45) NOT NULL,  
    price INTEGER NOT NULL,  
    quantity INTEGER NOT NULL,  
    title VARCHAR (45) NOT NULL UNIQUE,  
    value INTEGER NOT NULL,
```

```
imageUrl VARCHAR (45) NOT NULL,  
rushSupport BOOLEAN NOT NULL DEFAULT (1)  
);
```

- **Bảng OrderData:**

```
CREATE TABLE OrderData (  
    id INTEGER PRIMARY KEY NOT NULL,  
    city VARCHAR (45) NOT NULL,  
    address VARCHAR (45) NOT NULL,  
    phone VARCHAR (45) NOT NULL,  
    userId INTEGER NOT NULL,  
    shippingFee INTEGER NOT NULL,  
    createdAt DATETIME NOT NULL,  
    instructions VARCHAR (255),  
    type VARCHAR (45) NOT NULL,  
    totalPrice INTEGER NOT NULL,  
    status VARCHAR (45) NOT NULL DEFAULT UNSUCCESSFUL,  
    CONSTRAINT fk_order_user  
    FOREIGN KEY (userId) REFERENCES User (id)  
);
```

- Bảng OrderMedia:

```
CREATE TABLE OrderMedia (  
    mediaID INTEGER NOT NULL,  
    orderID INTEGER NOT NULL,  
    price INTEGER NOT NULL,  
    quantity INTEGER NOT NULL,  
    PRIMARY KEY (mediaID, orderID),  
    CONSTRAINT fk_ordermedia_media  
    FOREIGN KEY (mediaID) REFERENCES Media (id),  
    CONSTRAINT fk_ordermedia_order  
    FOREIGN KEY (orderID) REFERENCES OrderData (id)  
);
```

## V. Design Concept and Principle

### 1. Media

controller/admin/./MediasController.java

#### - Coupling

##### 1. Data Coupling (Liên kết Dữ liệu):

- **Xác định:** Có sử dụng Datasource để lấy và xử lý dữ liệu từ nguồn dữ liệu.
- **Đánh giá:** Liên kết dữ liệu là trung bình. Lớp này phụ thuộc vào Datasource để truy xuất dữ liệu.
- **Đề xuất:** Có thể sử dụng dependency injection để giảm kết nối chặt chẽ với Datasource.

##### 2. Stamp Coupling (Liên kết Dấu):

- **Xác định:** Không có sử dụng dấu (stamp) hoặc sự liên quan đặc biệt đến nó.
- **Đánh giá:** Không có dấu hiệu của liên kết dấu trong mã nguồn.
- **Đề xuất:** Tiếp tục tránh sử dụng dấu hiệu để giữ cho liên kết không rõ ràng.

##### 3. Control Coupling (Liên kết Kiểm soát):

- **Xác định:** Có sử dụng cơ chế điều khiển trong phương thức btnViewMedia, btnEditMedia, btnMediasSearchOnAction.
- **Đánh giá:** Liên kết kiểm soát tồn tại do tương tác với các phương thức xử lý sự kiện.
- **Đề xuất:** Xem xét tái cấu trúc để giảm liên kết kiểm soát, có thể sử dụng gọi lại hoặc sự kiện để tách biệt logic kiểm soát.

#### 4. Common Coupling (Liên kết Chung):

- **Xác định:** Có sử dụng các biến như fieldMediasSearch, viewMediaResponse, formEditMediaView, mediasContent, tableMediasPage.
- **Đánh giá:** Có một số liên kết chung do sử dụng các biến chung.
- **Đề xuất:** Nếu có thể, đóng gói các biến chia sẻ vào một lớp cụ thể hoặc giảm sử dụng biến chia sẻ.

#### 5. Content Coupling (Liên kết Nội dung):

- **Xác định:** Các phương thức listMedias, addActionButtonsToTable, btnMediasSearchOnAction, btnViewMedia, btnEditMedia,... liên quan chặt chẽ đến nội dung xử lý của MediasController.
- **Đánh giá:** Có liên kết nội dung mạnh mẽ do các phương thức thực hiện các chức năng quản lý và hiển thị phương tiện.
- **Đề xuất:** Xem xét tách biệt các phương thức thành các lớp riêng biệt để giảm độ liên kết nội dung và tăng tính tái sử dụng.

### - Cohesion

#### 1. Tính Gắn kết Chức năng (Functional Cohesion):

- **Xác định:** Các phương thức như listMedias, addActionButtonsToTable, btnMediasSearchOnAction, btnViewMedia, btnEditMedia liên quan chặt chẽ đến chức năng quản lý và hiển thị phương tiện.
- **Đánh giá:** Tính gắn kết chức năng cao khi các phương thức hoạt động chung để thực hiện một mục đích chung.
- **Đề xuất:** Tiếp tục duy trì tính gắn kết chức năng.



## 2. Tính Gắn kết Dữ liệu (Data Cohesion):

- **Xác định:** Có sử dụng Datasource để lấy và xử lý dữ liệu từ nguồn dữ liệu.
- **Đánh giá:** Liên kết dữ liệu là trung bình. Lớp này phụ thuộc vào Datasource để truy xuất dữ liệu.
- **Đề xuất:** Có thể sử dụng dependency injection để giảm kết nối chặt chẽ với Datasource.

## 3. Tính Gắn kết Liên quan (Coincidental Cohesion):

- **Xác định:** Không có dấu hiệu rõ ràng về tính gắn kết liên quan trong mã nguồn.
- **Đánh giá:** Không có sự gắn kết ngẫu nhiên hoặc không mong muốn trong lớp.
- **Đề xuất:** Tiếp tục tránh sử dụng các thành phần có mối quan hệ ngẫu nhiên.

## - SOLID

### 1. Nguyên tắc Single Responsibility (SRP):

- **Đánh giá:** Tập MediasController có trách nhiệm chính là quản lý giao diện người dùng và xử lý sự kiện liên quan đến các phương tiện như CD, DVD, và Book. Tuy nhiên, nó thực hiện nhiều chức năng, chẳng hạn như quản lý hình ảnh, kiểm tra tính hợp lệ của dữ liệu đầu vào, và các tác vụ khác.
- **Đề xuất cải tiến:** Có thể tách các chức năng khác nhau thành các lớp hoặc đối tượng riêng biệt để đảm bảo tính đơn trách nhiệm.

### 2. Nguyên tắc Open/Closed (OCP):

- **Đánh giá:** Mặc dù tập MediasController có thể mở rộng bằng cách thêm các chức năng mới (ví dụ: thêm loại phương tiện

khác), nhưng để thực hiện các thay đổi, bạn phải sửa đổi mã nguồn hiện tại.

- **Đề xuất cải tiến:** Sử dụng kế thừa và ghi đè để mở rộng chức năng mà không phải sửa đổi mã nguồn hiện tại.

### 3. Nguyên tắc Liskov Substitution (LSP):

- **Đánh giá:** Chưa có thông tin cụ thể về các lớp con hoặc việc sử dụng kế thừa trong tệp. Cần xem xét các lớp con và đảm bảo rằng chúng có thể thay thế cho các lớp cơ sở mà không làm thay đổi hành vi mong đợi.
- **Đề xuất cải tiến:** Kiểm tra các lớp con, đảm bảo rằng chúng tuân thủ nguyên tắc LSP và có thể thay thế cho lớp cơ sở một cách an toàn.

### 4. Nguyên tắc Interface Segregation (ISP):

- **Đánh giá:** Các interface không được hiển thị trong tệp, nhưng phương thức như `btnEditMedia()` và `btnViewMedia()` chấp nhận các đối tượng có nhiều chức năng.
- **Đề xuất cải tiến:** Tách các interface thành các interface nhỏ hơn, phản ánh các chức năng cụ thể của các đối tượng.

### 5. Nguyên tắc Dependency Inversion (DIP):

- **Đánh giá:** Tệp `MediasController` sử dụng `Datasource` trực tiếp trong phương thức `listMedias()`.
- **Đề xuất cải tiến:** Sử dụng dependency injection hoặc inversion of control để giảm sự phụ thuộc trực tiếp vào `Datasource` và làm cho `MediasController` dễ kiểm soát và kiểm thử hơn.

a. `controller/user/./UserMediasController.java`

## - Coupling

## 1. Data Coupling (Liên kết Dữ liệu):

- **Xác định:** Lớp này sử dụng lớp Datasource để tương tác với nguồn dữ liệu.
- **Đánh giá:** Liên kết dữ liệu là trung bình. Lớp này phụ thuộc vào Datasource để truy xuất và thao tác dữ liệu.
- **Đề xuất:** Xem xét sử dụng dependency injection hoặc mô hình repository để giảm kết nối lớp này với nguồn dữ liệu cụ thể.

## 2. Stamp Coupling (Liên kết Dấu):

- **Xác định:** Lớp này không cho thấy bất kỳ liên kết dấu nào rõ ràng.
- **Đánh giá:** Không có dấu hiệu của liên kết dấu trong mã nguồn được cung cấp.
- **Đề xuất:** Đảm bảo rằng lớp tiếp tục tránh liên kết dấu, vì đó là một khía cạnh tích cực của mã nguồn.

## 3. Control Coupling (Liên kết Kiểm soát):

- **Xác định:** Phương thức btnATCMedia dường như tương tác với người dùng bằng cách hiển thị cảnh báo và yêu cầu xác nhận.
- **Đánh giá:** Liên kết kiểm soát tồn tại do tương tác với cảnh báo người dùng.
- **Đề xuất:** Xem xét tái cấu trúc phương thức để tách biệt quyền truy cập UI và logic kinh doanh, có thể sử dụng gọi lại hoặc sự kiện.

## 4. Common Coupling (Liên kết Chung):

- **Xác định:** Lớp này sử dụng các biến chia sẻ như `pu_quantity`.
- **Đánh giá:** Có một số liên kết chung do sử dụng trạng thái chia sẻ.
- **Đề xuất:** Nếu có thể, đóng gói trạng thái chia sẻ vào một lớp cụ thể hoặc giảm sử dụng biến chia sẻ.

## 5. Content Coupling (Liên kết Nội dung):

- **Xác định:** Lớp có các phương thức (`listMedias`, `btnATCMedia`, v.v.) liên quan đến nội dung (hành vi) của `UserMediasController`.
- **Đánh giá:** Liên kết nội dung hiện diện khi các phương thức hoạt động cùng nhau để thực hiện chức năng xử lý phương tiện.
- **Đề xuất:** Đảm bảo rằng các phương thức có độ liên kết nội dung phù hợp và xem xét tách biệt chức năng riêng biệt thành các lớp khác nhau.

## - Cohesion

### 1. Functional Cohesion (Tính Gắn kết Chức năng):

- **Xác định:** Các phương thức như `listMedias`, `refresh`, `addSpinnerColumnToTable`, `removeColumnFromTable`, `addActionButtonsToTable`, và các phương thức khác có vẻ liên quan đến chức năng quản lý và hiển thị phương tiện.
- **Đánh giá:** Tính gắn kết chức năng được quan sát khi các phương thức liên quan đến một mục đích chung.
- **Đề xuất:** Tiếp tục rà soát các phương thức để duy trì tính gắn kết chức năng và có thể tách biệt quyền truy cập khi cần thiết.

## - SOLID

### 1. Nguyên tắc Đơn trách nhiệm (SRP):

- **Đánh giá:** Lớp có nhiều trách nhiệm, bao gồm xử lý sự kiện UI, quản lý dữ liệu phương tiện và tương tác UI.
- **Đề xuất:** Xem xét tái cấu trúc để tuân theo nguyên tắc SRP hơn bằng cách phân tách quyền truy cập thành các lớp khác nhau.

### 2. Nguyên tắc Mở Đóng (OCP):

- **Đánh giá:** Lớp có thể yêu cầu sửa đổi để thêm chức năng mới.
- **Đề xuất:** Xem xét sử dụng giao diện và trừu tượng hóa để làm cho lớp mở rộng mà không cần sửa đổi.

### 3. Nguyên tắc Thay thế Liskov (LSP):

- **Đánh giá:** Sự hiện diện của các lớp con và sự tuân thủ của chúng với LSP không rõ ràng trong mã nguồn được cung cấp.
- **Đề xuất:** Đảm bảo rằng các lớp con có thể sử dụng thay thế cho lớp cơ bản mà không ảnh hưởng đến tính chính xác.

### 4. Nguyên tắc Phân chia Giao diện (ISP):

- **Đánh giá:** Lớp sử dụng các thành phần JavaFX, nhưng không có cài đặt giao diện rõ ràng hiển thị trong mã nguồn được cung cấp.

### 5. Nguyên tắc đảo ngược phụ thuộc (DIP):

- **Đánh giá:** Lớp sử dụng trực tiếp Datasource, biểu thị sự phụ thuộc trực tiếp.

- **Đề xuất:** Áp dụng tính năng chèn phụ thuộc hoặc đảo ngược điều khiển để giảm sự phụ thuộc trực tiếp và làm cho lớp linh hoạt hơn.

## 2. Customer

controller/admin/./CutomersController.java

### - Coupling

#### 1. Data Coupling (Phụ thuộc dữ liệu):

- CustomersController sử dụng dữ liệu từ Datasource để lấy thông tin về khách hàng từ cơ sở dữ liệu.
- Mức độ phụ thuộc dữ liệu là khá rõ ràng, vì CustomersController phải biết về cấu trúc dữ liệu và phương thức từ Datasource.

#### 2. Stamp Coupling (Phụ thuộc tem):

- CustomersController gọi các phương thức như getAllCustomers, searchCustomers, deleteSingleCustomer từ Datasource thông qua một giao diện chung (giả định là có một giao diện hoặc lớp trừu tượng chung được triển khai bởi Datasource).
- Mức độ phụ thuộc tem được giảm bằng cách sử dụng giao diện chung giữa CustomersController và Datasource.

#### 3. Control Coupling (Phụ thuộc điều khiển):

- Các nút hành động (View, Edit, Block) trong bảng được kết nối với các phương thức xử lý tương ứng trong CustomersController (ví dụ: btnViewCustomer, btnEditCustomer).

- Mức độ phụ thuộc điều khiển không quá mức lớn, vì các nút chỉ gọi các phương thức trong chính CustomersController.

#### **4. Common Coupling (Phụ thuộc chung):**

- Không có sự phụ thuộc chung rõ ràng trong CustomersController. Các thành phần của CustomersController đang làm việc độc lập và không chia sẻ dữ liệu hoặc trạng thái chung nào đó.

#### **5. Content Coupling (Phụ thuộc nội dung):**

- Mức độ phụ thuộc nội dung ở đây không rõ ràng, vì không có việc chia sẻ nội dung đặc biệt giữa các thành phần. Tuy nhiên, có thể xem xét các FXML files được sử dụng để xem xét sự phụ thuộc của giao diện người dùng.

### **- Cohesion**

#### **1. Logical Cohesion (Liên kết Logic):**

- Các phương thức listCustomers, addActionButtonsToTable, btnCustomersSearchOnAction, btnEditCustomer, btnViewCustomer, fillEditCustomer đều liên quan đến quản lý và hiển thị dữ liệu của khách hàng trong giao diện người dùng.
- Các phương thức này chia sẻ mục tiêu chung và làm việc cùng nhau để thực hiện các chức năng liên quan đến quản lý khách hàng.

#### **2. Coincidental Cohesion (Liên kết Tình cờ):**

- Có một số phương thức như btnEditCustomer, btnViewCustomer, fillEditCustomer có thể được xem là tình cờ liên kết, vì chúng thực hiện các chức năng không liên quan trực tiếp đến quản lý khách hàng.

- Ví dụ, btnEditCustomer và btnViewCustomer thực hiện điều hướng đến các trang chỉnh sửa và xem, trong khi fillEditCustomer được thiết kế để điền dữ liệu vào giao diện chỉnh sửa.

### **3. Temporal Cohesion (Liên kết Thời gian):**

- Có một số phương thức như listCustomers và btnCustomersSearchOnAction có thể được xem như liên kết thời gian, vì chúng thực hiện các chức năng tương tự liên quan đến hiển thị dữ liệu khách hàng trong giao diện người dùng.

### **4. Procedural Cohesion (Liên kết Thủ tục):**

- Các phương thức addActionButtonsToTable, btnEditCustomer, btnViewCustomer, và fillEditCustomer đều thực hiện các chức năng liên quan đến xử lý sự kiện và thủ tục cho các hành động người dùng cụ thể.

## **- SOLID**

### **1. Nguyên tắc Đơn trách nhiệm (Single Responsibility Principle - SRP):**

- Mã nguồn tuân theo SRP. CustomersController chịu trách nhiệm về hiển thị và quản lý dữ liệu liên quan đến khách hàng trong giao diện người dùng.

### **2. Nguyên tắc Mở rộng đóng (Open/Closed Principle - OCP):**

- Mã nguồn có thể mở rộng được. Các chức năng như hiển thị và quản lý khách hàng có thể được mở rộng thông qua việc thêm mới các phương thức hoặc chức năng mới mà không làm ảnh hưởng đến mã hiện tại.

### **3. Nguyên tắc Liskov Substitution (Liskov Substitution Principle - LSP):**



- Mã nguồn không có vấn đề nổi lên liên quan đến việc thay thế các đối tượng con (ví dụ, User) cho các đối tượng cha mà không làm thay đổi tính đúng đắn của chương trình.

#### **4. Nguyên tắc Interface Segregation (Interface Segregation Principle - ISP):**

- Mã nguồn không triển khai việc sử dụng quá nhiều phương thức trong một interface. Các interface trong mã nguồn đều tương đối nhỏ và chỉ chứa những phương thức cần thiết.

#### **5. Nguyên tắc Phụ thuộc đảo ngược (Dependency Inversion Principle - DIP):**

- Mã nguồn sử dụng một số nguyên tắc đảo ngược phụ thuộc, đặc biệt là khi sử dụng Datasource thông qua interface. Điều này giúp giảm độ phụ thuộc vào cụ thể và tăng tính linh hoạt.

### 3. Màn Home

controller/admin/./HomeController.java

#### - Coupling

##### 1. Data Coupling:

- Mức độ data coupling trong HomeController thấp. Các phương thức chỉ chia sẻ dữ liệu thông qua giá trị trả về của `Datasource.getInstance().countAllMedias()` và `Datasource.getInstance().countAllCustomers()`. Các phương thức chỉ làm việc với dữ liệu số nguyên và không phụ thuộc trực tiếp vào cấu trúc dữ liệu nào khác.

##### 2. Stamp Coupling:

- Stamp coupling không áp dụng trong trường hợp này. Stamp coupling xảy ra khi một module truyền một dạng dữ liệu vào một phần khác của chương trình mà không sử dụng nó.

##### 3. Control Coupling:

- Các phương thức trong HomeController không phụ thuộc vào việc điều khiển logic của nhau. Không có sự phụ thuộc giữa các phương thức trong việc điều khiển luồng chương trình.

##### 4. Common Coupling:

- Common coupling cũng thấp. Các thành phần chỉ giao tiếp thông qua việc trả về giá trị và không có sự chia sẻ trạng thái nào ngoài việc gán giá trị cho các nhãn (`mediasCount` và `customersCount`).

##### 5. Content Coupling:

- Các phương thức trong HomeController chỉ chịu trách nhiệm về việc cập nhật dữ liệu và giao diện người dùng liên quan đến

tổng số media và số lượng khách hàng. Nó không chứa nhiều nội dung logic khác.

## - Cohesion

### 1. Functional Cohesion:

- Mức độ cohesion chức năng cao trong HomeController. Cả hai phương thức getDashboardProdCount và getDashboardCostCount đều liên quan đến chức năng của việc đếm và hiển thị tổng số lượng media và số lượng khách hàng. Cả hai phương thức đều thực hiện các chức năng tương tự.

### 2. Sequential Cohesion:

- Có mức độ cohesion tuần tự ở đây, vì cả hai phương thức đều thực hiện các bước theo thứ tự: bắt đầu một nhiệm vụ bất đồng bộ (Task), đợi cho kết quả, sau đó cập nhật giao diện người dùng.

### 3. Communicational Cohesion:

- Mức độ cohesion giao tiếp ở mức độ trung bình. Cả hai phương thức đều liên quan đến việc giao tiếp với Datasource để lấy thông tin cần thiết.

### 4. Temporal Cohesion:

- Có mức độ temporal cohesion, vì cả hai phương thức đều liên quan đến việc cập nhật giao diện người dùng về tổng số lượng media và số lượng khách hàng.

### 5. Procedural Cohesion:

- Có mức độ procedural cohesion, vì cả hai phương thức đều thực hiện các bước cụ thể để đạt được mục tiêu của mình.

## - SOLID

### 1. Nguyên tắc Đơn trách nhiệm (Single Responsibility Principle - SRP):

- HomeController chịu trách nhiệm cho việc hiển thị thông tin về số lượng media và số lượng khách hàng trong trang admin dashboard. Điều này tuân thủ nguyên tắc SRP.

### 2. Nguyên tắc Mở rộng (Open/Closed Principle - OCP):

- Hiện tại, HomeController không có nhiều phương thức hoặc tính năng để mở rộng hoặc thay đổi. Nếu cần thêm chức năng trong tương lai, có thể cần chỉnh sửa mã nguồn. Điều này có thể được cải thiện bằng cách sử dụng một giao diện và triển khai nhiều hơn một lớp để đạt được nguyên tắc OCP.

### 3. Nguyên tắc Chuyển giao (Liskov Substitution Principle - LSP):

- Không có sự kế thừa hoặc sự mở rộng nào trong HomeController đến nay. Do đó, không có vi phạm nguyên tắc LSP.

### 4. Nguyên tắc Interface Segregation Principle (ISP):

- HomeController không sử dụng giao diện, do đó không có vi phạm nguyên tắc ISP. Tuy nhiên, nếu có sự mở rộng trong tương lai, việc sử dụng giao diện có thể giúp tuân thủ nguyên tắc này.

### 5. Nguyên tắc Phụ thuộc đảo ngược (Dependency Inversion Principle - DIP):

- HomeController phụ thuộc vào Datasource để lấy dữ liệu. Tuy nhiên, việc phụ thuộc trực tiếp vào một lớp cụ thể có thể làm giảm khả năng tái sử dụng. Sử dụng dependency injection có thể làm giảm độ phụ thuộc và cải thiện nguyên tắc DIP.

- Tóm lại, HomeController có thể cần một số cải thiện nhỏ để tuân thủ đầy đủ các nguyên tắc SOLID, nhưng nó đã bắt đầu với một cấu trúc tương đối tốt từ các khía cạnh của SRP và không có vi phạm lớn đối với các nguyên tắc khác.

controller/user/./UserHomeController.java

## - Coupling

### 1. Data Coupling:

- UserHomeController sử dụng Datasource.getInstance() để lấy dữ liệu về số lượng medias và số lượng đơn đặt hàng. Điều này tạo ra một mức độ phụ thuộc dữ liệu, nhưng đối tượng UserHomeController chỉ phụ thuộc vào giao diện Datasource chứ không phụ thuộc vào chi tiết cụ thể của lớp Datasource. Điều này giảm mức độ coupling.

### 2. Stamp Coupling:

- Không có sự phụ thuộc nào dựa trên "stamp" (nhãn) trong mã nguồn này. "Stamp coupling" xảy ra khi một phương thức yêu cầu đối số có kiểu dữ liệu dựa trên một nhãn hoặc đánh dấu cụ thể.

### 3. Control Coupling:

- Không có sự phụ thuộc kiểm soát (control coupling) rõ ràng. Các phương thức trong UserHomeController không trực tiếp tương tác với các phương thức của các đối tượng khác để kiểm soát luồng thực thi.

### 4. Common Coupling:

- Không có dấu hiệu của common coupling. Common coupling xảy ra khi nhiều lớp chia sẻ một số biến toàn cục hoặc cấu trúc dữ liệu chung.

## 5. Content Coupling:

- UserHomeController sử dụng Datasource để truy xuất dữ liệu, nhưng không phụ thuộc trực tiếp vào cách dữ liệu được xử lý bên trong Datasource. Do đó, không có content coupling rõ ràng.

### - Cohesion

#### 1. Functional Cohesion (Cohesion chức năng):

- getDashboardProdCount chịu trách nhiệm lấy và hiển thị số lượng medias.
- getDashboardOrdersCount chịu trách nhiệm lấy và hiển thị số lượng đơn đặt hàng.

#### 2. Sequential Cohesion (Cohesion tuần tự):

- Các phương thức đều được thực thi tuần tự, không có sự phụ thuộc rõ ràng giữa chúng.

#### 3. Communicational Cohesion (Cohesion giao tiếp):

- Các phương thức không trực tiếp liên lạc với nhau.

#### 4. Procedural Cohesion (Cohesion thủ tục):

- Mỗi phương thức thực hiện một công việc cụ thể, không chia sẻ nhiều dữ liệu hoặc logic.

### - SOLID

#### 1. Nguyên tắc Đơn trách nhiệm (SRP):

- Các phương thức getDashboardProdCount và getDashboardOrdersCount đều thực hiện một nhiệm vụ cụ thể và chỉ liên quan đến đếm số lượng. Do đó, chúng tuân thủ nguyên tắc SRP.

## **2. Nguyên tắc Mở rộng đóng (OCP):**

- Mã nguồn của bạn không chứa các biểu hiện rõ ràng của việc áp dụng nguyên tắc OCP. Điều này có thể phụ thuộc vào ngữ cảnh lớn hơn của ứng dụng.

## **3. Nguyên tắc Liskov thay thế (LSP):**

- Không có biểu hiện rõ ràng về việc áp dụng nguyên tắc LSP trong mã nguồn này.

## **4. Nguyên tắc Interface phân chia (ISP):**

- Mã nguồn không sử dụng giao diện, nhưng trong trường hợp này, việc sử dụng giao diện có thể không cần thiết do tính chất đơn giản của các nhiệm vụ.

## **5. Nguyên tắc Đảo ngược phụ thuộc (DIP):**

- Sử dụng Datasource để truy cập dữ liệu là một hình thức tương đối tốt của nguyên tắc DIP, vì phương thức của bạn không phụ thuộc trực tiếp vào cơ sở dữ liệu.

## 4. Order

controller/admin/./OrdersController.java

### - Coupling

#### 1. Data Coupling:

- Các phương thức trong OrdersController sử dụng lớp Datasource để truy cập dữ liệu, nhưng chúng không trực tiếp tương tác với dữ liệu, mà chỉ sử dụng các phương thức của Datasource. Điều này giảm mức độ data coupling, vì OrdersController không phụ thuộc trực tiếp vào cấu trúc dữ liệu của Datasource.

#### 2. Stamp Coupling:

- Không có biểu hiện rõ ràng về stamp coupling trong mã nguồn này. Stamp coupling thường xảy ra khi một thành phần phụ thuộc vào cấu trúc nội dung (cấu trúc dữ liệu) của một thành phần khác. Trong trường hợp này, không có dấu hiệu đó.

#### 3. Control Coupling:

- Phương thức listOrders sử dụng Datasource để lấy danh sách đơn hàng và addActionButtonsToTable để thêm các nút thao tác vào bảng. Sự tương tác giữa các phương thức trong OrdersController không phụ thuộc mạnh mẽ lẫn nhau, điều này giảm mức độ control coupling.

#### 4. Common Coupling:

- Không có biểu hiện rõ ràng về common coupling trong mã nguồn này. Common coupling xảy ra khi các thành phần chia sẻ một số dữ liệu hoặc trạng thái chung, nhưng trong trường hợp này, không có dấu hiệu của sự chia sẻ đó.



## **5. Content Coupling:**

- Mức độ content coupling được giữ ở mức thấp. Các phương thức chỉ chịu ảnh hưởng của cấu trúc dữ liệu và phương thức của lớp Datasource, và không chịu ảnh hưởng từ cấu trúc nội dung của các phương thức khác trong cùng một lớp.

## **- Cohesion**

### **1. Functional Cohesion:**

- Cao: Các phương thức trong OrdersController đều liên quan đến việc hiển thị và quản lý đơn hàng. Chúng thực hiện các chức năng liên quan đến đơn hàng như lấy danh sách đơn hàng, hiển thị các hành động trên bảng đơn hàng.

### **2. Sequential Cohesion:**

- Trung bình: Các phương thức không chủ yếu phụ thuộc vào kết quả của nhau. Tuy nhiên, có một sự liên kết tuần tự trong việc thực hiện các bước để hiển thị và quản lý đơn hàng.

### **3. Communicational Cohesion:**

- Cao: Lớp này tương tác với Datasource để lấy danh sách đơn hàng từ cơ sở dữ liệu.

### **4. Procedural Cohesion:**

- Cao: Các phương thức thực hiện một chuỗi các bước để đạt được một mục tiêu cụ thể, đó là hiển thị và quản lý đơn hàng.

### **5. Temporal Cohesion:**

- Trung bình: Có sự liên quan về thời gian trong việc thực hiện các bước để hiển thị đơn hàng.

## - SOLID

### 1. Nguyên tắc Đơn trách nhiệm (Single Responsibility Principle - SRP):

- **Đánh giá:** Lớp này chịu trách nhiệm quản lý và hiển thị đơn hàng, cũng như thực hiện các hành động liên quan. Tuy nhiên, có thể tách chức năng hiển thị và quản lý đơn hàng thành các lớp khác nhau để tuân thủ tốt hơn nguyên tắc này.
- **Đề xuất cải tiến:** Tách phần hiển thị và quản lý thành các lớp riêng biệt để cải thiện SRP.

### 2. Nguyên tắc Mở rộng (Open/Closed Principle - OCP):

- **Đánh giá:** Hiện tại, lớp không có sự mở rộng dễ dàng cho các thay đổi hoặc bổ sung chức năng mới.
- **Đề xuất cải tiến:** Sử dụng các mô hình linh hoạt hơn để mở rộng hoặc mô-đun hóa các phần của lớp.

### 3. Nguyên tắc Lặp lại (Liskov Substitution Principle - LSP):

- **Đánh giá:** Chưa có thông tin đủ để đánh giá việc tuân thủ nguyên tắc này.
- **Đề xuất cải tiến:** Đảm bảo rằng mọi lớp con có thể thay thế lớp cơ sở mà không làm thay đổi tính đúng đắn của chương trình.

### 4. Nguyên tắc Interface đồng nhất (Interface Segregation Principle - ISP):

- **Đánh giá:** Hiện tại, không có thông tin về việc lớp này sử dụng các giao diện.
- **Đề xuất cải tiến:** Nếu có giao diện, đảm bảo rằng chúng được thiết kế một cách hợp lý và không chứa nhiều phương thức không liên quan.

## 5. Nguyên tắc Phụ thuộc nghịch đảo (Dependency Inversion Principle - DIP):

- **Đánh giá:** Lớp này có phụ thuộc vào Datasource để lấy dữ liệu đơn hàng.
- **Đề xuất cải tiến:** Sử dụng dependency injection hoặc các mô hình phụ thuộc ngược để giảm độ phụ thuộc cứng.

controller/user/./UserOrdersController.java

### - Coupling

#### 1. Data Coupling:

- **Đánh giá:** Lớp này sử dụng dữ liệu từ UserSessionController, Datasource, và CartMedia.
- **Giải pháp:** Điều này có thể dẫn đến mức độ kết nối dữ liệu cao. Đối với một mức độ coupling thấp hơn, bạn có thể cân nhắc sử dụng dependency injection để truyền các phụ thuộc vào thay vì sử dụng trực tiếp từ các lớp khác.

#### 2. Stamp Coupling:

- **Đánh giá:** Lớp này sử dụng orderData và getAllOrderMedias trong sự kiện của nút "View".
- **Giải pháp:** Nếu cần thực hiện nhiều công việc khác nhau, có thể xem xét việc tách chúng thành các phương thức riêng biệt để giảm coupling.

#### 3. Control Coupling:

- **Đánh giá:** Lớp này gửi một sự kiện (hiển thị danh sách đơn hàng) khi nút "View" được nhấp.
- **Giải pháp:** Có thể giảm coupling bằng cách sử dụng mô hình kiến trúc sự kiện hoặc callback.

#### 4. Common Coupling:

- **Đánh giá:** Lớp này có sử dụng các phương thức chung (ví dụ: `calculateTotalPrice`) cho việc tính toán giá trị tổng cần thiết.
- **Giải pháp:** Nếu các phương thức này được sử dụng rộng rãi trong hệ thống, có thể đưa chúng vào một lớp tiện ích hoặc lớp dịch vụ chung.

#### 5. Content Coupling:

- **Đánh giá:** Lớp này tạo và hiển thị một cửa sổ mới (`UserInvoiceController`) khi nút "View" được nhấp.
- **Giải pháp:** Điều này là một loại coupling content khi lớp này biết cụ thể về cách hiển thị thông tin cho người dùng. Nếu cần, có thể xem xét việc sử dụng mô hình chiến lược để giảm coupling.

### - Cohesion

#### 1. Functional Cohesion:

- **Đánh giá:** Phương thức `listOrders` thực hiện một chức năng cụ thể - hiển thị danh sách đơn hàng cho người dùng.
- **Mô tả:** Các phương thức trong lớp này liên quan đến chức năng cụ thể của việc quản lý đơn hàng người dùng.

#### 2. Sequential Cohesion:

- **Đánh giá:** Các phương thức `listOrders` và `addActionButtonsToTable` được gọi theo trình tự nhất định.
- **Mô tả:** Các phương thức được gọi tuần tự để thực hiện nhiệm vụ hiển thị đơn hàng và thêm các nút hành động vào bảng.

### 3. Communicational Cohesion:

- **Đánh giá:** Phương thức `addActionButtonsToTable` và `calculateTotalPrice` liên quan đến việc giao tiếp với thành phần giao diện người dùng và tính toán giá trị tổng.
- **Mô tả:** Các phương thức liên quan chặt chẽ đến việc truyền thông tin giữa giao diện người dùng và logic tính toán.

### 4. Procedural Cohesion:

- **Đánh giá:** Các phương thức trong lớp này được sắp xếp theo trình tự nhất định để thực hiện chức năng liên quan đến đơn hàng người dùng.
- **Mô tả:** Các phương thức chia sẻ một mức độ chức năng và được tổ chức để thực hiện một dãy các bước liên quan đến đơn hàng.

### 5. Temporal Cohesion:

- **Đánh giá:** Phương thức `addActionButtonsToTable` và `calculateTotalPrice` thực hiện các nhiệm vụ liên quan đến xử lý giao diện người dùng và tính toán trong cùng một khoảng thời gian.
- **Mô tả:** Các phương thức được gọi cùng một lúc để thực hiện các công việc có liên quan đến đơn hàng.

## - SOLID

### 1. Nguyên tắc Single Responsibility (SRP):

- **Đánh giá:** Lớp có vẻ tuân theo nguyên tắc SRP vì nó chịu trách nhiệm hiển thị danh sách đơn hàng và thêm các nút hành động vào bảng.

- **Mô tả:** Phương thức listOrders chịu trách nhiệm lấy dữ liệu đơn hàng và hiển thị chúng. Phương thức addActionButtonsToTable chịu trách nhiệm thêm các nút hành động vào bảng.

## 2. Nguyên tắc Open/Closed (OCP):

- **Đánh giá:** Lớp có thể mở rộng để thêm các chức năng mới mà không cần sửa đổi mã nguồn hiện tại.
- **Mô tả:** Bất kỳ thay đổi nào trong chức năng hiển thị đơn hàng có thể được thêm vào mà không cần sửa đổi lớp UserOrdersController chính.

## 3. Nguyên tắc Liskov Substitution (LSP):

- **Đánh giá:** Dường như không có việc thực hiện hoặc ghi đè các phương thức của lớp cha trong lớp UserOrdersController.
- **Mô tả:** Điều này làm cho nó tuân theo nguyên tắc LSP.

## 4. Nguyên tắc Interface Segregation (ISP):

- **Đánh giá:** Lớp không triển khai các giao diện nên không cần phải quan tâm đến nguyên tắc này.
- **Mô tả:** Không có sự phụ thuộc vào các giao diện, vì vậy không có vấn đề với nguyên tắc ISP.

## 5. Nguyên tắc Dependency Inversion (DIP):

- **Đánh giá:** Lớp UserOrdersController có thể nhận dữ liệu từ các nguồn khác nhau mà không cần biết chi tiết về cách dữ liệu được truy xuất.
- **Mô tả:** Phương thức listOrders sử dụng Datasource để lấy dữ liệu, nhưng lớp không phụ thuộc trực tiếp vào cụ thể của Datasource.

## 5. Dashboard

controller/admin/MainDashboardController.java

### - Coupling

#### 1. Data Coupling:

- **Đánh giá:** Không có dấu hiệu của Data Coupling.
- **Mô tả:** Lớp MainDashboardController không trực tiếp truy cập hoặc chia sẻ dữ liệu với các lớp khác thông qua các biến toàn cục. Việc truyền dữ liệu thường thông qua tham số của các phương thức.

#### 2. Stamp Coupling:

- **Đánh giá:** Không có dấu hiệu của Stamp Coupling.
- **Mô tả:** Lớp MainDashboardController không chứa bất kỳ loại dữ liệu dấu hiệu (stamp data) cụ thể của bất kỳ lớp hay giao diện nào khác.

#### 3. Control Coupling:

- **Đánh giá:** Có mức độ control coupling nhỏ.
- **Mô tả:** Mức độ control coupling là nhỏ do lớp MainDashboardController gọi các phương thức trong các lớp khác để thực hiện các chức năng như hiển thị danh sách, đăng xuất, và điều này xảy ra thông qua các phương thức được kích hoạt bởi các sự kiện người dùng.

#### 4. Common Coupling:

- **Đánh giá:** Không có dấu hiệu của Common Coupling.

- **Mô tả:** Lớp MainDashboardController không chia sẻ dữ liệu với các lớp khác thông qua các biến toàn cục.

## 5. Content Coupling:

- **Đánh giá:** Mức độ content coupling là nhỏ.
- **Mô tả:** Lớp MainDashboardController gọi các phương thức trong các lớp khác để thực hiện các chức năng như hiển thị danh sách, đăng xuất, nhưng nó không phụ thuộc vào cụ thể của nội dung được hiển thị.

## - Cohesion

### 1. Functional Cohesion:

- **Đánh giá:** Phần lớn có tính functional cohesion.
- **Mô tả:** Các phương thức trong MainDashboardController thực hiện các chức năng liên quan đến quản lý trang chính của ứng dụng quản lý, chẳng hạn như hiển thị thông tin trang chính, danh sách đơn hàng, khách hàng, và quản lý hệ thống.

### 2. Sequential Cohesion:

- **Đánh giá:** Có một số dấu hiệu của sequential cohesion.
- **Mô tả:** Các phương thức có thể được gọi theo một thứ tự cụ thể để thực hiện các chức năng liên quan đến việc hiển thị và quản lý dữ liệu.

### 3. Communicational Cohesion:

- **Đánh giá:** Có mức độ communicational cohesion nhỏ.
- **Mô tả:** Mức độ communicational cohesion xuất hiện khi lớp gọi các phương thức trong các lớp khác để thực hiện các chức năng cụ thể như hiển thị danh sách đơn hàng, medias, và quản lý khách hàng.



#### 4. Procedural Cohesion:

- **Đánh giá:** Có một số phần có procedural cohesion.
- **Mô tả:** Có những phần trong lớp có thể được nhìn nhận như có procedural cohesion khi chúng thực hiện một loạt các bước để hoàn thành một nhiệm vụ nhất định, chẳng hạn như đăng xuất và hiển thị danh sách.

#### 5. Temporal Cohesion:

- **Đánh giá:** Không có dấu hiệu của temporal cohesion.
- **Mô tả:** Không có dấu hiệu rõ ràng rằng các phần của lớp chỉ liên quan đến nhau trong một khoảng thời gian cụ thể.

### - SOLID

#### 1. Single Responsibility Principle (SRP):

- **Đánh giá:** Một số phương thức thực hiện nhiều chức năng.
- **Mô tả:** Một số phương thức trong MainDashboardController thực hiện nhiều nhiệm vụ, chẳng hạn như đăng xuất, hiển thị trang chính, và hiển thị danh sách khách hàng. Điều này có thể làm tăng độ phức tạp của lớp và làm giảm tính SRP.

#### 2. Open/Closed Principle (OCP):

- **Đánh giá:** Khó mở rộng và thay đổi một số chức năng.
- **Mô tả:** Đối với việc thêm chức năng mới, có thể cần sửa đổi các phương thức hiện tại hoặc thêm mã mới vào lớp. Điều này có thể làm giảm tính OCP, vì không dễ dàng mở rộng hệ thống.

#### 3. Liskov Substitution Principle (LSP):

- **Đánh giá:** Không có dấu hiệu rõ ràng về vi phạm LSP.

- **Mô tả:** Các phương thức của lớp có vẻ không gây ra vấn đề nào đối với việc thay thế lớp bằng các lớp con của nó. Tính tuân thủ LSP khá tốt.

#### 4. Interface Segregation Principle (ISP):

- **Đánh giá:** Lớp không triển khai các giao diện.
- **Mô tả:** Không có dấu hiệu rõ ràng về vi phạm ISP, vì lớp không triển khai giao diện nào cả. Tuy nhiên, việc triển khai các giao diện có thể cải thiện khả năng mở rộng.

#### 5. Dependency Inversion Principle (DIP):

- **Đánh giá:** Phương thức sử dụng giá trị độc lập với các lớp cụ thể.
- **Mô tả:** Các phương thức trong lớp không phụ thuộc trực tiếp vào các lớp cụ thể mà thay vào đó chúng tương tác thông qua các giao diện (ví dụ: `UserController`). Điều này tuân thủ DIP, làm cho lớp trở nên linh hoạt hơn.

`controller/user/UserMainDashboardController.java`

### - Coupling

#### 1. Data Coupling:

- **Đánh giá:** Thấp.
- **Mô tả:** Lớp này sử dụng dữ liệu từ `UserController` để lấy tên người dùng và có thể tương tác với các dữ liệu cụ thể trong các controller khác như `UserHomeController`, `UserCartController`, `UserMediasController`, `UserOrdersController` thông qua các phương thức như `getDashboardProdCount`, `getDashboardOrdersCount`, `listCartMedias`, `listMedias`, `listOrders`.

#### 2. Stamp Coupling:

- **Đánh giá:** Thấp.
- **Mô tả:** Lớp này không yêu cầu bất kỳ đánh dấu nào (stamp) từ các lớp khác.

### 3. Control Coupling:

- **Đánh giá:** Trung bình.
- **Mô tả:** Lớp này liên kết với nhiều controller khác nhau (được gọi từ nút nhấn), như HomeController, CartController, MediaController, OrdersController. Sự phụ thuộc này là do nó cần tương tác với các chức năng khác nhau trong ứng dụng.

### 4. Common Coupling:

- **Đánh giá:** Thấp.
- **Mô tả:** Lớp này không chia sẻ bất kỳ biến hoặc trạng thái nào với các lớp khác, ngoại trừ việc sử dụng SessionController để lấy thông tin người dùng.

### 5. Content Coupling:

- **Đánh giá:** Trung bình.
- **Mô tả:** Có mức độ sự phụ thuộc vào cấu trúc nội dung của các trang (pages) khác nhau. Sự phụ thuộc này thể hiện qua việc gọi các phương thức như getDashboardProdCount, getDashboardOrdersCount, listCartMedias, listMedias, listOrders để cập nhật nội dung.

## - Cohesion

### 1. Functional Cohesion:

- **Đánh giá:** Cao.
- **Mô tả:** Lớp này chịu trách nhiệm cho các chức năng liên quan đến giao diện người dùng của dashboard người dùng. Các phương thức như `btnHomeOnClick`, `btnCartOnClick`, `btnMediasOnClick`, `btnOrdersOnClick`, `btnLogOutOnClick` đều liên quan đến các chức năng của dashboard người dùng.

### 2. Sequential Cohesion:

- **Đánh giá:** Cao.
- **Mô tả:** Các phương thức của lớp này thường liên quan đến các bước tuần tự trong việc xử lý sự kiện người dùng, từ việc nhấn nút đến hiển thị nội dung tương ứng.

### 3. Communicational Cohesion:

- **Đánh giá:** Trung bình.
- **Mô tả:** Lớp này thực hiện giao tiếp với các controller khác như `UserHomeController`, `UserCartController`, `UserMediasController`, `UserOrdersController` để yêu cầu cập nhật nội dung.

### 4. Procedural Cohesion:

- **Đánh giá:** Trung bình.
- **Mô tả:** Lớp này chứa các phương thức thực hiện các thao tác cụ thể như `loadFXMLPage`, `initialize`.

## - SOLID

### 1. Single Responsibility Principle (SRP):

- **Đánh giá:** Tốt.
- **Mô tả:** Lớp này có một nhiệm vụ cụ thể - quản lý tương tác giữa giao diện người dùng và các controller khác. Các phương thức của nó liên quan đến việc hiển thị các trang và xử lý sự kiện người dùng.

### 2. Open/Closed Principle (OCP):

- **Đánh giá:** Tốt.
- **Mô tả:** Lớp này có thể mở rộng bằng cách thêm các phương thức mới để xử lý các chức năng mới mà không cần sửa đổi mã nguồn hiện tại.

### 3. Liskov Substitution Principle (LSP):

- **Đánh giá:** Đạt yêu cầu cơ bản.
- **Mô tả:** Không có sự kế thừa đặc biệt nào trong lớp này, vì vậy không có vấn đề lớn liên quan đến LSP.

### 4. Interface Segregation Principle (ISP):

- **Đánh giá:** Đạt yêu cầu cơ bản.
- **Mô tả:** Lớp này không triển khai bất kỳ giao diện nào, vì vậy không có vấn đề liên quan đến ISP.

### 5. Dependency Inversion Principle (DIP):

- **Đánh giá:** Đạt yêu cầu cơ bản.
- **Mô tả:** Lớp này phụ thuộc vào một số controller khác thông qua interface của chúng (UserHomeController,

UserCartController, UserMediasController, UserOrdersController). Việc này đáp ứng yêu cầu của DIP.

## 6. Cart

controller/user/./UserCartController.java

### - Coupling

#### 1. Data Coupling:

- **Đánh giá:** Coupling dữ liệu khá thấp.
- **Mô tả:** Lớp UserCartController sử dụng dữ liệu từ UserSessionController và Datasource, nhưng chúng được sử dụng thông qua các phương thức chung và không chặt chẽ kết nối với nhau.

#### 2. Stamp Coupling:

- **Đánh giá:** Coupling dựa trên stamp khá cao.
- **Mô tả:** Lớp UserCartController sử dụng nhiều stamp như Task, ActionEvent, FXML, và Stage. Mặc dù đây là các phần cần thiết để xử lý sự kiện trong JavaFX, nhưng chúng tạo ra một loại coupling thông qua sự phụ thuộc vào các thành phần cụ thể của framework.

#### 3. Control Coupling:

- **Đánh giá:** Coupling kiểm soát là thấp.
- **Mô tả:** Lớp UserCartController không chứa nhiều kiểm soát trực tiếp về các quyết định của lớp khác. Các phương thức chủ yếu thực hiện logic liên quan đến hiển thị và xử lý sự kiện.

#### 4. Common Coupling:

- **Đánh giá:** Không có coupling chung đáng kể.

- **Mô tả:** Lớp này không chia sẻ nhiều dữ liệu hoặc trạng thái với các lớp khác. Các phương thức và thuộc tính đều liên quan đến chức năng cụ thể của lớp.

## 5. Content Coupling:

- **Đánh giá:** Coupling nội dung thấp.
- **Mô tả:** Lớp này không đặc biệt phụ thuộc vào cấu trúc hoặc triển khai nội dung cụ thể của các lớp khác. Nó tương tác với các đối tượng thông qua các giao diện chung như ObservableList, Task, và Button.

## - Cohesion

### 1. Functional Cohesion:

- **Đánh giá:** Cohesion chức năng là cao.
- **Mô tả:** Lớp UserController tập trung vào chức năng hiển thị và xử lý các sự kiện liên quan đến giỏ hàng của người dùng. Các phương thức như listCartMedias, refresh, btnPlaceOrderAction đều liên quan chặt chẽ đến nhiệm vụ hiển thị và xử lý giỏ hàng.

### 2. Sequential Cohesion:

- **Đánh giá:** Cohesion tuần tự là cao.
- **Mô tả:** Các phương thức trong lớp thường xuyên được gọi theo chuỗi logic liên tục, như listCartMedias gọi addActionButtonsToTable, và btnPlaceOrderAction gọi refresh.

### 3. Communicational Cohesion:

- **Đánh giá:** Cohesion giao tiếp là cao.

- **Mô tả:** Các phương thức trong lớp thường xuyên tương tác với nhau thông qua truyền tham số, ví dụ như `setOrdersData` và `setTotalPrice` được gọi trong `btnPlaceOrderAction`.

#### 4. Procedural Cohesion:

- **Đánh giá:** Cohesion thủ tục là cao.
- **Mô tả:** Các phương thức của lớp thực hiện các bước theo thứ tự cụ thể để thực hiện các chức năng như là hiển thị giỏ hàng, làm mới dữ liệu giỏ hàng, và chuyển hướng đến trang giao hàng.

### - SOLID

#### 1. Nguyên tắc Single Responsibility (SRP):

- **Đánh giá:** Lớp này tuân thủ SRP.
- **Mô tả:** `UserCartController` chịu trách nhiệm cho việc hiển thị và quản lý giỏ hàng của người dùng. Nó thực hiện một chức năng cụ thể: liên quan đến giỏ hàng và hiển thị nó.

#### 2. Nguyên tắc Open/Closed (OCP):

- **Đánh giá:** Lớp này không mở rộng được mà không làm thay đổi mã nguồn nguyên tắc OCP.
- **Mô tả:** Hiện tại, không có cơ chế nào mở rộng hay mở lại các chức năng của `UserCartController`.

#### 3. Nguyên tắc Liskov Substitution (LSP):

- **Đánh giá:** Không áp dụng mạnh mẽ nguyên tắc LSP.
- **Mô tả:** Lớp này không thừa kế từ lớp nào khác và không ghi đè các phương thức.

#### 4. Nguyên tắc Interface Segregation (ISP):



- **Đánh giá:** Lớp này không tham gia vào nguyên tắc ISP.
- **Mô tả:** Không có giao diện nào được thực hiện bởi lớp UserController.

## 5. Nguyên tắc Dependency Inversion (DIP):

- **Đánh giá:** Lớp này tuân thủ nguyên tắc DIP đến một mức độ nhất định.
- **Mô tả:** Phương thức listCartMedias sử dụng Datasource mà không phải làm việc trực tiếp với các đối tượng cụ thể mà nó thao tác, điều này có thể coi là áp dụng DIP.

## 7. Invoice

controller/user/./UserInvoiceController.java

### - Coupling

#### 1. Data Coupling:

- **Mức độ:** Có mức độ coupling thấp.
- **Mô tả:** Phương thức setData đưa dữ liệu vào controller từ bên ngoài thông qua các tham số. Dữ liệu được truyền dưới dạng tham số, không có sự phụ thuộc cứng vào cấu trúc nội bộ của dữ liệu bên trong controller.

#### 2. Stamp Coupling:

- **Mức độ:** Không có stamp coupling.
- **Mô tả:** Không có sự phụ thuộc vào dữ liệu đánh dấu (stamp) giữa các phần của mã nguồn. Không có thông tin đánh dấu cụ thể nào được chuyển giữa các phương thức.

#### 3. Control Coupling:

- **Mức độ:** Có mức độ control coupling.
- **Mô tả:** Phương thức setData chịu trách nhiệm thiết lập dữ liệu cho các thành phần giao diện người dùng. Nó có sự kiểm soát (control) đối với việc hiển thị dữ liệu cho người dùng.

#### 4. Common Coupling:

- **Mức độ:** Có mức độ common coupling.
- **Mô tả:** Có sự chia sẻ dữ liệu thông qua ObservableList<CartMedia> ordersData, nhưng không có sự phụ thuộc cứng vào cấu trúc nội bộ của dữ liệu.

#### 5. Content Coupling:

- **Mức độ:** Có mức độ content coupling.
- **Mô tả:** Có một số sự giữa việc xử lý nội dung (content) cụ thể như tính toán giá cả và quản lý đặt hàng. Tuy nhiên, các chi tiết cụ thể vẫn được giữ tách biệt một cách tương đối.

## - Cohesion

### 1. Functional Cohesion:

- **Mức độ:** Có mức độ functional cohesion.
- **Mô tả:** Các phương thức như `handleBackButtonClick`, `handleBacktoOrder`, `listOrders`, `getShippingFee`, `calculateTotalPrice`, `calculateTotalQuantity`, `insertNewMediaOrder`, `decreaseStock`, `setData`, `handleConfirmOrder`, `handleSuccessInvoice` có chức năng chính là xử lý một khía cạnh cụ thể của quá trình đặt hàng và hiển thị thông tin đơn hàng.

### 2. Sequential Cohesion:

- **Mức độ:** Có mức độ sequential cohesion.
- **Mô tả:** Các phương thức như `handleBackButtonClick`, `handleBacktoOrder`, `listOrders`, `getShippingFee`, `calculateTotalPrice`, `calculateTotalQuantity`, `insertNewMediaOrder`, `decreaseStock`, `setData`, `handleConfirmOrder`, `handleSuccessInvoice` có thể được gọi theo một chuỗi logic tuần tự, ví dụ: từ xem giỏ hàng đến xác nhận đơn hàng.

### 3. Communicational Cohesion:

- **Mức độ:** Có mức độ communicational cohesion.
- **Mô tả:** Các phương thức như `handleBackButtonClick`, `handleBacktoOrder`, `listOrders`, `getShippingFee`,

calculateTotalPrice, calculateTotalQuantity, insertNewMediaOrder, decreaseStock, setData, handleConfirmOrder, handleSuccessInvoice có thể chia sẻ dữ liệu thông qua tham số và biến cục bộ, như việc chia sẻ thông tin đơn hàng giữa chúng.

#### 4. **Procedural Cohesion:**

- **Mức độ:** Có mức độ procedural cohesion.
- **Mô tả:** Các phương thức như handleBackButtonClick, handleBacktoOrder, listOrders, getShippingFee, calculateTotalPrice, calculateTotalQuantity, insertNewMediaOrder, decreaseStock, setData, handleConfirmOrder, handleSuccessInvoice đều thực hiện các bước cụ thể trong quá trình xử lý đơn hàng.

#### 5. **Temporal Cohesion:**

- **Mức độ:** Có mức độ temporal cohesion.
- **Mô tả:** Các phương thức như handleBackButtonClick, handleBacktoOrder, listOrders, getShippingFee, calculateTotalPrice, calculateTotalQuantity, insertNewMediaOrder, decreaseStock, setData, handleConfirmOrder, handleSuccessInvoice có thể liên quan đến cùng một khía cạnh của thời gian, như xử lý đơn hàng.

### - **SOLID**

#### 1. **Single Responsibility Principle (S - Nguyên tắc Đơn trách nhiệm):**

- **Ưu điểm:**
  - § Mỗi phương thức trong các lớp có vẻ chỉ thực hiện một chức năng cụ thể, ví dụ handleBackButtonClick xử lý việc quay lại trang địa chỉ.

- **Khuyết điểm:**

- § Một số phương thức như setData thực hiện nhiều nhiệm vụ, ví dụ như hiển thị thông tin và thiết lập giá trị.

## **2. Open/Closed Principle (O - Nguyên tắc Mở rộng/Đóng gói):**

- **Ưu điểm:**

- § Các lớp có thể mở rộng thông qua việc thêm các phương thức mới, ví dụ như thêm các phương thức xử lý thanh toán mới.

- **Khuyết điểm:**

- § Một số phương thức có thể cần sửa đổi nếu có thay đổi yêu cầu, ví dụ như việc thêm chức năng mới.

## **3. Liskov Substitution Principle (L - Nguyên tắc Thay thế Liskov):**

- **Ưu điểm:**

- § Không có sự thay thế giữa các lớp trong mã nguồn.

- § Các lớp như Interbank chưa rõ trong mã nguồn được gọi có thể được thay thế bằng các lớp thực hiện cùng một giao diện.

- **Khuyết điểm:**

- § Không có sự kế thừa hoặc thay thế rõ ràng trong mã nguồn.

## **4. Interface Segregation Principle (I - Nguyên tắc Phân tách Giao diện):**

- **Ưu điểm:**

§ Không có sự tuyến tính và tách biệt giữa các giao diện trong mã nguồn.

- **Khuyết điểm:**

§ Các giao diện có thể chưa đúng mức độ của nguyên tắc này, chúng có thể chứa nhiều phương thức không liên quan.

## **5. Dependency Inversion Principle (D - Nguyên tắc Đảo Ngược Phụ thuộc):**

- **Ưu điểm:**

§ Các lớp như Interbank có vẻ giữ được nguyên tắc này khi không phụ thuộc trực tiếp vào các lớp cụ thể.

- **Khuyết điểm:**

§ Các lớp khác có thể phụ thuộc trực tiếp vào cài đặt cụ thể, không tuân theo nguyên tắc này.

## 8. Shipping

controller/user/./UserShippingController.java

### - Coupling

#### 1. Data Coupling:

- **Điểm chung:**

- § Các phương thức như setData trong UserShippingController chấp nhận và sử dụng dữ liệu từ bên ngoài, nhưng không phụ thuộc vào cài đặt cụ thể của dữ liệu đó.

- § Các lớp như Datasource được sử dụng để truy vấn và cập nhật dữ liệu, nhưng không có sự phụ thuộc trực tiếp vào cài đặt cụ thể của lớp đó.

#### 2. Stamp Coupling:

- **Điểm chung:**

- § Không có dấu hiệu rõ ràng về sự liên kết thông qua sự đánh dấu (stamp).

- § Các lớp và phương thức không yêu cầu các "đánh dấu" cụ thể để tương tác với nhau.

#### 3. Control Coupling:

- **Điểm chung:**

- § Một số phương thức như handleDeliveryButtonClick trong UserShippingController nhận sự kiện (event) từ giao diện người dùng và thực hiện kiểm soát dòng chạy của ứng dụng.

§ Sự kiện được truyền vào phương thức để xử lý các hành động liên quan đến việc nhấn nút giao hàng.

#### 4. **Common Coupling:**

- **Điểm chung:**

§ Không có sự chia sẻ dữ liệu thông qua các biến toàn cục hoặc các thành phần chung khác.

§ Mỗi lớp và phương thức có vẻ độc lập và không chia sẻ trạng thái hoặc biến với các thành phần khác.

#### 5. **Content Coupling:**

- **Điểm chung:**

§ Không có sự phụ thuộc vào cấu trúc nội dung của các lớp khác.

§ Các phương thức như `handleDeliveryButtonClick` chủ yếu tập trung vào xử lý logic kinh doanh và không phụ thuộc vào cách cấu trúc hoặc triển khai cụ thể của các lớp khác.

### - **Cohesion**

#### 1. **Cohesion Hàm (Functional Cohesion):**

- **Điểm chung:**

§ Các phương thức như `handleDeliveryButtonClick`, `checkRush`, `checkQuantity`, và `handleBackButtonClick` trong `UserShippingController` chủ yếu thực hiện một chức năng cụ thể liên quan đến xử lý đơn hàng và giao hàng.

- **Nhận xét:**



§ Các phương thức này hợp nhất và tập trung vào các nhiệm vụ có liên quan đến quy trình giao hàng và xác nhận đơn hàng.

## 2. Cohesion Dữ Liệu (Data Cohesion):

- **Điểm chung:**

§ Các phương thức như `setTotalPrice`, `setOrdersData`, và `setData` chủ yếu thao tác với dữ liệu và có mục tiêu chung là cập nhật hoặc truy xuất thông tin từ các thành phần khác.

- **Nhận xét:**

§ Các phương thức này có liên quan đến quản lý dữ liệu và tạo liên kết giữa các thành phần.

## 3. Cohesion Sequential (Sequential Cohesion):

- **Điểm chung:**

§ Các phương thức trong `UserShippingController` thường xuyên gọi lẫn nhau theo thứ tự cụ thể để thực hiện một nhiệm vụ hoặc chức năng.

- **Nhận xét:**

§ Các bước thực hiện được tổ chức theo một luồng tuần tự trong quá trình xử lý đơn hàng và giao hàng.

## 4. Cohesion Communicational (Communicational Cohesion):

- **Điểm chung:**

§ Các phương thức như `handleDeliveryButtonClick` và `setOrdersData` có sự tương tác thông qua dữ liệu, chẳng hạn như cập nhật thông tin đơn hàng và truyền dữ liệu giữa các thành phần.

- **Nhận xét:**

- § Sự tương tác này tạo ra một cấu trúc giao tiếp chặt chẽ giữa các phương thức.

## **5. Cohesion Temporal (Temporal Cohesion):**

- **Điểm chung:**

- § Các phương thức trong UserShippingController thường liên quan đến các công đoạn thực hiện giao hàng và xử lý đơn hàng tại thời điểm gọi phương thức.

- **Nhận xét:**

- § Các bước thực hiện được tổ chức để liên kết thời gian trong quá trình xử lý đơn hàng.

## **- SOLID**

### **1. Nguyên tắc Single Responsibility (SRP):**

- *Đánh giá:* UserShippingController giữ một trách nhiệm duy nhất - quản lý đơn hàng và thông tin giao hàng.
- *Gợi ý:* Nếu trong quá trình phát triển, có thêm các trách nhiệm khác không liên quan đến quản lý đơn hàng và giao hàng, có thể xem xét chia thành các lớp mới để tuân thủ SRP.

### **2. Nguyên tắc Open/Closed (OCP):**

- *Đánh giá:* Mã nguồn có thể mở rộng để thêm các chức năng mới mà không cần sửa đổi mã nguồn hiện tại.
- *Gợi ý:* Các phương thức như handleDeliveryButtonClick có thể được mở rộng mà không cần sửa đổi quá nhiều.

### **3. Nguyên tắc Liskov Substitution (LSP):**

- *Đánh giá:* Không có dấu hiệu rõ ràng về việc vi phạm nguyên tắc LSP trong mã nguồn này.
- *Gợi ý:* Kiểm tra lại trong quá trình mở rộng các phương thức và lớp để đảm bảo tính thay thế linh hoạt.

#### **4. Nguyên tắc Interface Segregation (ISP):**

- *Đánh giá:* Mã nguồn không sử dụng giao diện, nên không có vấn đề liên quan đến ISP.
- *Gợi ý:* Nếu có một số lớp chỉ sử dụng một phần của các phương thức của giao diện, có thể xem xét việc chia thành các giao diện nhỏ hơn.

#### **5. Nguyên tắc Dependency Inversion (DIP):**

- *Đánh giá:* Các phương thức trong UserShippingController sử dụng một số phụ thuộc như Datasource và HelperMethods, nhưng không tạo ra phụ thuộc cấp cao.
- *Gợi ý:* Nếu có sự mở rộng, đảm bảo rằng các phụ thuộc cấp cao được chuyển vào thông qua các phương thức hoặc xây dựng để duy trì nguyên tắc DIP.

## 9. Login

controller/LoginController.java

### - Coupling

#### 1. Data Coupling:

- Phương thức `handleLoginButtonAction` sử dụng dữ liệu từ các trường `usernameField` và `passwordField`. Dữ liệu này được lấy để xác định thông tin đăng nhập người dùng. Các trường này thuộc loại dữ liệu nền tảng JavaFX.
- Phương thức `handleLoginButtonAction` sử dụng dữ liệu từ đối tượng `User` để kiểm tra thông tin đăng nhập và trạng thái tài khoản của người dùng.

#### 2. Stamp Coupling:

- Có dấu hiệu của stamp coupling khi phương thức `handleLoginButtonAction` sử dụng đối tượng `User` để kiểm tra trạng thái tài khoản của người dùng (`user.getStatus().equals("blocked")`).

#### 3. Control Coupling:

- Phương thức `handleLoginButtonAction` kiểm soát chuyển đổi giữa các màn hình và cấu hình hành vi của ứng dụng dựa trên thông tin người dùng.
- Phương thức `handleRegisterButtonAction` kiểm soát việc chuyển đổi sang màn hình đăng ký.

#### 4. Common Coupling:

- Không có dấu hiệu rõ ràng của common coupling trong mã nguồn này. Mỗi phương thức có nhiệm vụ rõ ràng và không chia sẻ dữ liệu không cần thiết.

## 5. Content Coupling:

- Có dấu hiệu của content coupling khi phương thức `handleLoginButtonAction` trực tiếp liên quan đến xác định xem người dùng có quyền admin hay không (`user.isAdmin()`). Có thể làm cho mã nguồn trở nên phức tạp nếu có sự thay đổi trong cách quyền admin được quản lý.

## - Cohesion

### 1. Functional Cohesion:

- Phương thức `handleLoginButtonAction` thực hiện một nhiệm vụ cụ thể là xử lý đăng nhập người dùng, kiểm tra thông tin và chuyển đổi màn hình.
- Phương thức `handleRegisterButtonAction` thực hiện nhiệm vụ cụ thể là chuyển đến màn hình đăng ký.

### 2. Sequential Cohesion:

- Các phương thức thường sắp xếp các bước thực hiện theo một trình tự nhất định, ví dụ, kiểm tra thông tin đăng nhập và chuyển đến màn hình tương ứng trong `handleLoginButtonAction`.

### 3. Communicational Cohesion:

- Phương thức `handleLoginButtonAction` tương tác với giao diện người dùng để lấy thông tin đăng nhập (`usernameField` và `passwordField`) và hiển thị thông báo.
- Phương thức `handleRegisterButtonAction` tương tác với giao diện người dùng để chuyển đến màn hình đăng ký.

### 4. Procedural Cohesion:

- Các phương thức thường thực hiện các bước để đạt được một mục tiêu cụ thể, chẳng hạn kiểm tra thông tin đăng nhập và chuyển đến màn hình tương ứng.

## **5. Temporal Cohesion:**

- Các bước thực hiện trong phương thức `handleLoginButtonAction` có thể được thực hiện theo một trình tự thời gian cụ thể, chẳng hạn như kiểm tra đăng nhập trước và sau đó chuyển đến màn hình.

## **- SOLID**

### **1. Single Responsibility Principle (SRP):**

- `LoginController` có vẻ tuân thủ SRP. Cả hai phương thức `handleLoginButtonAction` và `handleRegisterButtonAction` đều thực hiện một nhiệm vụ cụ thể và không chứa quá nhiều chức năng.

### **2. Open/Closed Principle (OCP):**

- Mã nguồn không thể mở rộng một cách dễ dàng cho các thay đổi hoặc mở rộng trong tương lai. Tuy nhiên, đối với một phần mềm đăng nhập cơ bản, việc này có thể không cần thiết.

### **3. Liskov Substitution Principle (LSP):**

- Không có sự mở rộng hoặc kế thừa đặc biệt trong mã nguồn hiện tại. Các nguyên tắc kế thừa cơ bản đều được tuân thủ.

### **4. Interface Segregation Principle (ISP):**

- Không có giao diện nào được sử dụng trong mã nguồn hiện tại. Tuy nhiên, trong trường hợp đơn giản như này, việc này có thể không cần thiết.

### **5. Dependency Inversion Principle (DIP):**

- Mã nguồn không chứa nhiều phụ thuộc và không có việc sử dụng Dependency Injection. Tuy nhiên, trong một ứng dụng lớn hơn, việc này có thể làm cho mã nguồn trở nên linh hoạt hơn.

## 10. Register

controller/RegisterController.java

### - Coupling

#### 1. Single Responsibility Principle (SRP):

- RegisterController có thể được coi là tuân thủ SRP vì nó chịu trách nhiệm cho việc đăng ký người dùng và chuyển hướng giữa các màn hình. Phương thức handleRegisterButtonAction chủ yếu thực hiện các chức năng liên quan đến đăng ký.

#### 2. Open/Closed Principle (OCP):

- Mã nguồn không có sự mở rộng dễ dàng cho các thay đổi hoặc mở rộng trong tương lai. Các nghiệp vụ đăng ký có thể được mở rộng với việc thêm chức năng đặc biệt hoặc kiểm tra thêm điều kiện.

#### 3. Liskov Substitution Principle (LSP):

- Không có vấn đề lớn với LSP trong mã nguồn này. Các phương thức thực hiện chức năng của mình mà không gây ra sự độn độ về logic.

#### 4. Interface Segregation Principle (ISP):

- Mã nguồn không sử dụng giao diện. Điều này là đúng vì không có cần thiết phải tách giao diện cho một controller đăng ký đơn giản.

#### 5. Dependency Inversion Principle (DIP):

- Các phương thức của RegisterController sử dụng Datasource trực tiếp. Việc này có thể làm cho mã nguồn trở nên cứng nhắc. Sử dụng Dependency Injection có thể làm cho mã nguồn linh hoạt hơn.



## - Cohesion

### 1. Functional Cohesion (Cohesion chức năng):

- **Đánh giá:** Các phương thức trong RegisterController liên quan chặt chẽ đến việc đăng ký người dùng. Các chức năng chính như xử lý đăng ký, kiểm tra thông tin người dùng đều liên quan đến một mục đích chung.

- **Đánh giá cụ thể:**

- § handleLoginButtonAction: Chuyển đến màn hình đăng nhập.

- § handleRegisterButtonAction: Xử lý đăng ký người dùng.

### 2. Sequential Cohesion (Cohesion tuần tự):

- **Đánh giá:** Các phương thức của RegisterController thực hiện theo một chuỗi tuần tự, từ việc kiểm tra dữ liệu nhập vào, kiểm tra điều kiện, thực hiện đăng ký, và chuyển hướng người dùng.

- **Đánh giá cụ thể:**

- § handleRegisterButtonAction: Các bước thực hiện đăng ký người dùng theo trình tự nhất định.

### 3. Communicational Cohesion (Cohesion giao tiếp):

- **Đánh giá:** Các phương thức tương tác với các thành phần khác (như Datasource, UserSessionController) để thực hiện chức năng đăng ký.

- **Đánh giá cụ thể:**

- § Sử dụng Datasource để kiểm tra và lưu trữ thông tin người dùng.

### 4. Procedural Cohesion (Cohesion thủ tục):

- **Đánh giá:** Các phương thức được sắp xếp theo các bước cụ thể để thực hiện chức năng đăng ký.

- **Đánh giá cụ thể:**

- § handleRegisterButtonAction: Thực hiện các bước cụ thể liên quan đến đăng ký người dùng.

## 5. Temporal Cohesion (Cohesion thời gian):

- **Đánh giá:** Các phương thức trong RegisterController liên quan đến cùng một thời điểm - thực hiện đăng ký khi người dùng ấn nút đăng ký.

- **Đánh giá cụ thể:**

- § handleRegisterButtonAction: Thực hiện tất cả các nhiệm vụ liên quan đến đăng ký khi được gọi.

## - SOLID

### 1. Single Responsibility Principle (Nguyên tắc Đơn trách nhiệm):

- **Đánh giá:** Class RegisterController có trách nhiệm chính là xử lý các chức năng đăng ký người dùng, từ việc kiểm tra dữ liệu đầu vào đến lưu trữ thông tin người dùng vào cơ sở dữ liệu.

- **Ưu điểm:** Class giữ một trách nhiệm chính, dễ hiểu và bảo trì.

### 2. Open/Closed Principle (Nguyên tắc Mở đóng):

- **Đánh giá:** Class không mở rộng hoặc thay đổi mã nguồn khi cần thêm chức năng mới. Điều này có thể đạt được bằng cách sử dụng extension (kế thừa, implement interface) để thêm các chức năng mới.

- **Ưu điểm:** Dễ mở rộng với các chức năng mới mà không làm ảnh hưởng đến mã nguồn hiện tại.

### 3. Liskov Substitution Principle (Nguyên tắc Thay thế Liskov):

- **Đánh giá:** Không có vi phạm nguyên tắc thay thế Liskov trong class RegisterController. Các phương thức thực hiện chính xác như được mong đợi và có thể thay thế cho nhau.
- **Ưu điểm:** Tăng tính linh hoạt và tái sử dụng.

### 4. Interface Segregation Principle (Nguyên tắc Phân chia Interface):

- **Đánh giá:** Class không triển khai các interface lớn và không sử dụng các phương thức không cần thiết. Tuy nhiên, nếu chức năng phức tạp hơn xuất hiện, việc sắp xếp lại các interface có thể cần thiết.
- **Ưu điểm:** Các class chỉ cần triển khai những phương thức liên quan đến chức năng của mình.

### 5. Dependency Inversion Principle (Nguyên tắc Phản đảo Phụ thuộc):

- **Đánh giá:** Class RegisterController tương tác với Datasource thông qua interface, giảm sự phụ thuộc trực tiếp vào implementaion cụ thể của Datasource. Điều này làm giảm độ phức tạp và làm cho class trở nên dễ kiểm thử và tái sử dụng.
- **Ưu điểm:** Giảm độ phụ thuộc giữa các class và tăng khả năng mở rộng và bảo trì.

