HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

School of Information and Communications Technology

AIMS e-commerce software

Subject: ITSS Software Development

Group 6: Nguyễn Đình Dũng – 20210230
Vũ Minh Dũng - 20205179
Trần Nam Dương - 20210263
Nguyễn Mạnh Dũng - 20194745
Trịnh Tiến Dũng – 20215187

Hanoi, tháng 6 năm 2024

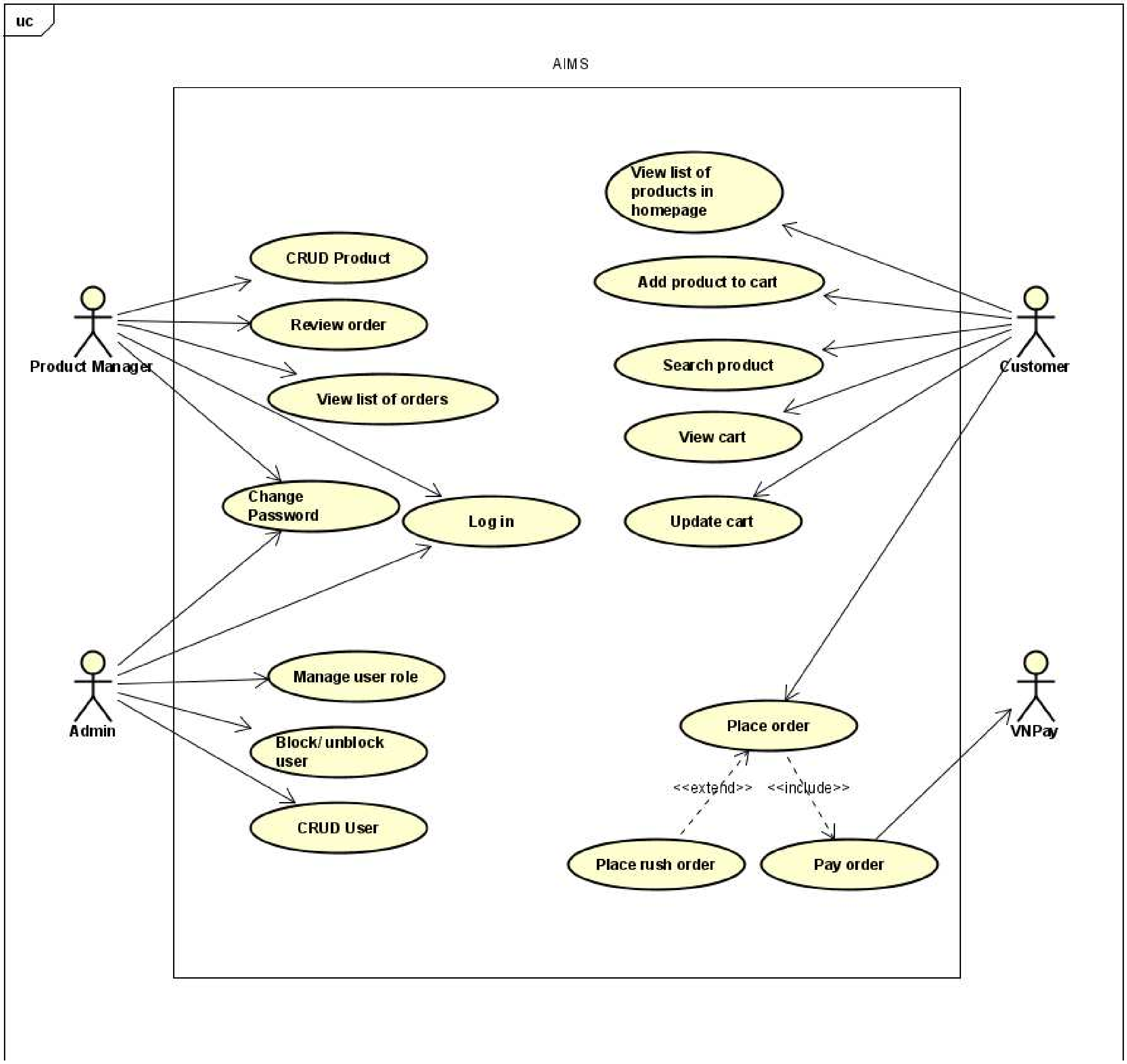
Mục lục

1. Assign working

Nguyễn Đình Dũng	Design place order and place rush order functions
Trần Nam Dương	Design payment functions, vnpay api , database
Nguyễn Mạnh Dũng	Design the product management functions of the product manager
Vũ Minh Dũng	Design the administrator's user management function, login
Trịnh Tiến Dũng	UC view list of products; view product detail; search, sort products; CRUD products in cart

2. Usecase Specification

2.1. General use case diagram



2.2. Usecase specification Search and sort product

Use Case "Search, Sort products"
• Use case code

UC002

- **Brief Description**

This use case describes the interaction between customer and AIMS when customer wish(es) to search or sort products on the home screen

- **Actors**
- **Customer**
- **Preconditions**

Customer successfully view products on the home screen.

- **Basic Flow of Events**
  - **Search product**
    1. The customer searches for products by title or category. (see Table A)
    2. AIMS looks up in database products satisfy the customer request.
    3. AIMS displays result to home screen. (see Table B)
  - **Sort products**
    - The customer requests ascending sort of products.
    - AIMS sorts products prices in ascending order.
    - AIMS displays sorted products on the home screen. (see Table B)
- **Alternative flows**

Table 1-Alternative flows of events for UC Search products

No	Location	Condition	Action	Resume location
• 1	At Step 3	If there is no product match search request	• System displays no product found	End use case

Table 2-Alternative flows of events for UC Sort products

No	Location	Condition	Action	Resume location
•	At Step 1	If customer requests descending sort.	• AIMS sorts products prices in descending order	Resumes at Step 3

- **Input data**

Table A-Input data of UC Search products

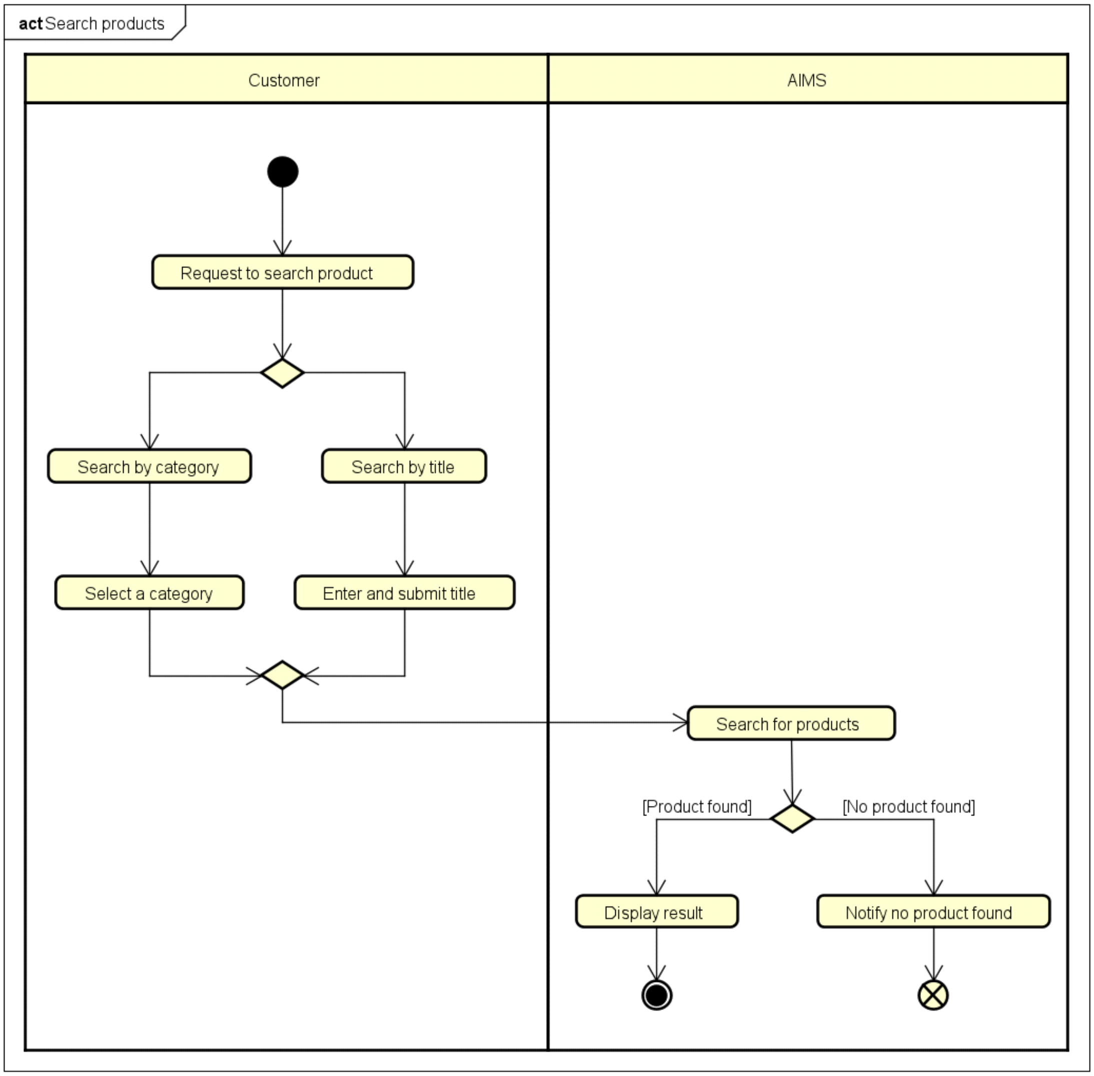
No	Data fields	Description	Mandatory	Valid condition	Example
•	Product title	Title of searched product	No	Text	Harry Potter
	Product category	Category of searched product	No	Choose from list	Book

- **Output data**

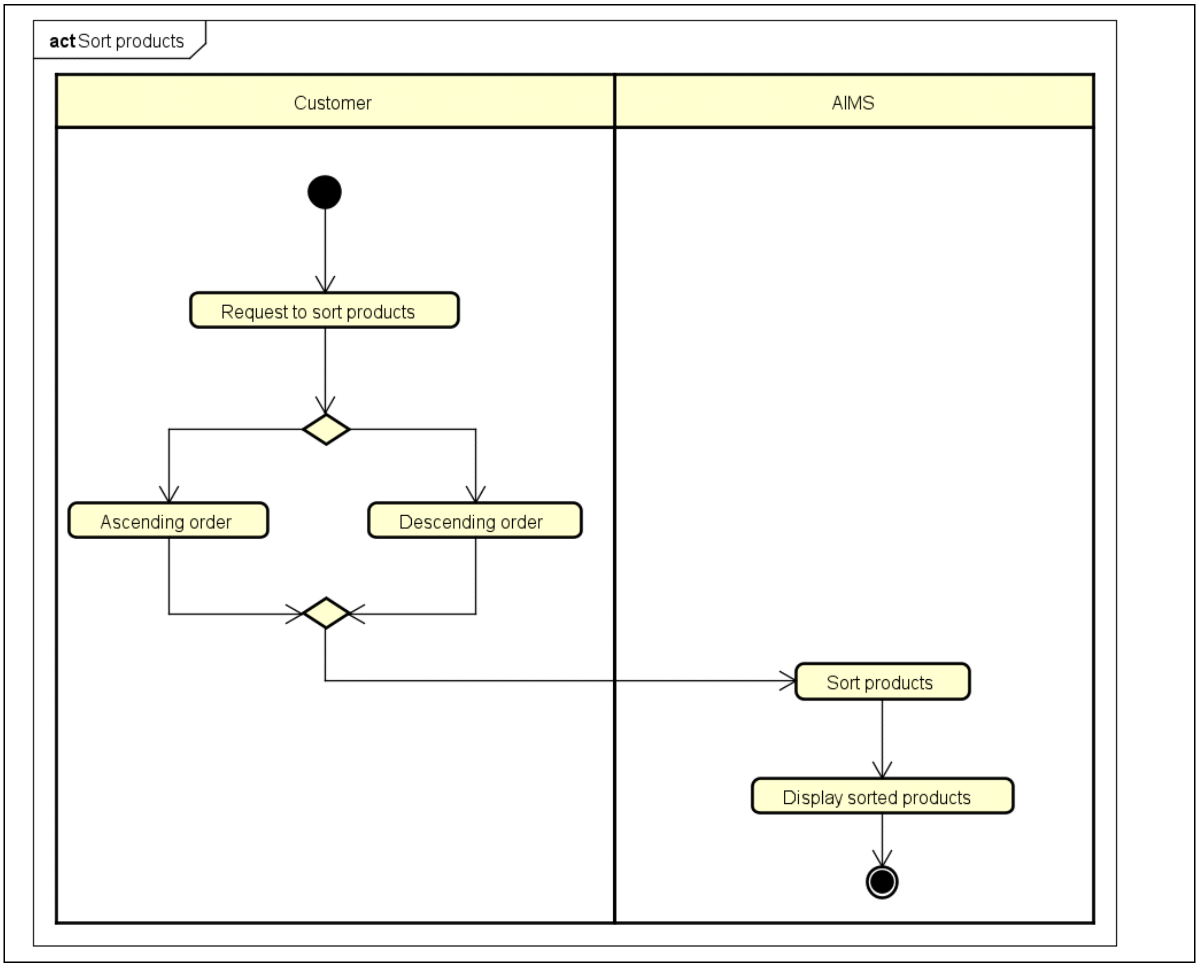
Table B-Output data of UC Searched products

No	Data fields	Description	Display format	Example
•	Image	Image of product's cover	Image	
•	Title	Title of product	Text	Harry Potter
•	Price	Price of product	- Comma for thousands Separator - Positive integer - Right alignment	100.000 vnd
•	Avail	Available quantity of product	- Positive integer - Right alignment	10

- **Postconditions**
- **Activity Diagrams**
- **Search products**



- **Sort products**



2.3. Usecase specification View product detail

Use Case “View product detail”

- **Use case code**

  UC003

- **Brief Description**

  This use case describes the interaction between customer and AIMS when customer wish(es) to view product detail

- **Actors**
- **Customer**
- **Preconditions**

  Customer successfully views list of products on the home screen.

- **Basic Flow of Events**
    - The customer requests to view detail information of a product
    - AIMS checks for category of the product
    - AIMS shows detail information of the product (if Book see Table A, if DVD see Table B, if CD see Table C)
- **Alternative flows**
- **Input data**
- **Output data**

Table A-Output data of Book's detail information

No	Data fields	Description	Display format	Example
•	Image	Image of book's cover	Image	
•	Title	Title of book	Text	Harry Potter
•	Author	Author of book	Text	J.K.Rowling
•	Publisher	Publisher of book	Text	Bloomsbury
•	Publish Date	Publish Date of book	dd/mm/yyyy	26/06/1997
•	Language	Language used in book	Text	English
•	Category	Category of book	Text	Fantasy
•	Numbers of Pages	Numbers of Pages of book	Positive Integer	700

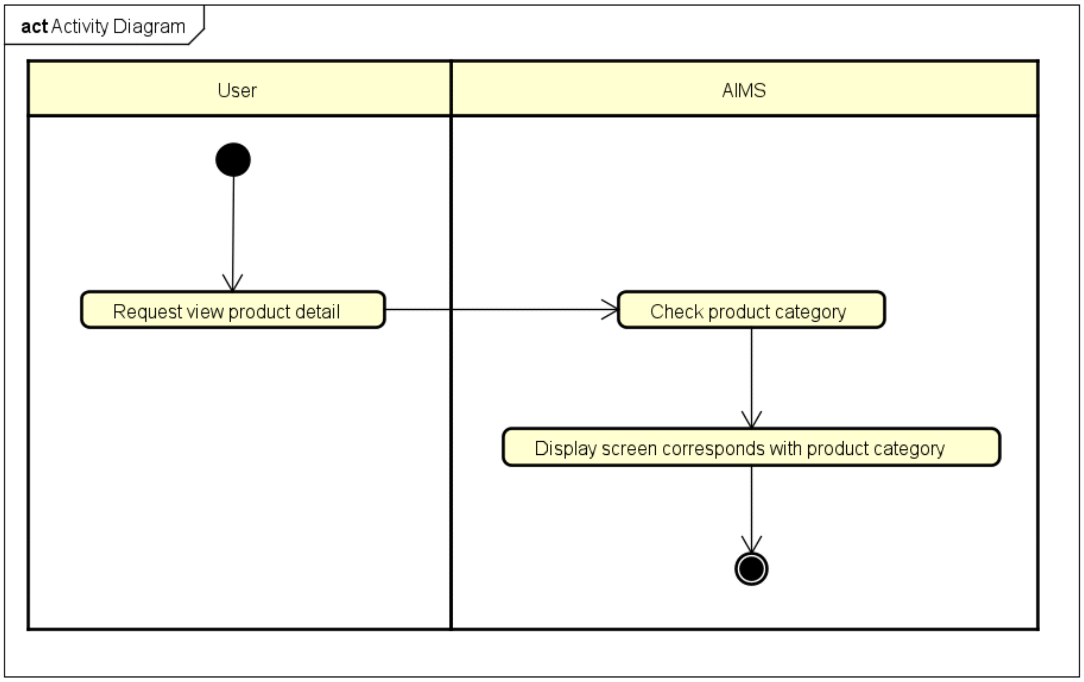
Table B-Output data of DVD's detail information

No	Data fields	Description	Display format	Example
•	Image	Image of DVD's cover	Image	
•	Title	Title of DVD	Text	Harry Potter
•	Director	Director of DVD	Text	Chris Columbus
•	Studio	Name of DVD production studio	Text	Warner Bros. Pictures
•	Released Date	Released Date of DVD	dd/mm/yyyy	11/05/2002
•	Type	Type of DVD	Text	Fantasy

Table C-Output data of CD's detail information

No	Data fields	Description	Display format	Example
•	Image	Image of CD's cover	Image	
•	Title	Title of CD	Text	1989
•	Artist	Artist of CD	Text	Taylor Swift
•	Record Label	Record Label of CD	Text	Big Machine Records
•	Record Date	Record Date of CD	dd/mm/yyyy	27/10/2014
•	Music Type	Music Type of CD	Text	Pop

- **Postconditions**
- **Activity Diagrams**


act Activity Diagram
User
AIMS
Request view product detail
Check product category
Display screen corresponds with product category

- **. Usecase Specification Manage Cart**

Use Case “CRUD product in Cart”

- **Use case code**

  UC00X

- **Brief Description**

  This use case describes the interaction between customer and AIMS when customer wish(es) to CRUD product in cart

- **Actors**
  - **Customer**
- **Preconditions**

  Customer successfully views list of products on the home screen.

- **Basic Flow of Events**
- **Add product to cart**

1. The customer selects a product and the quantity they want to purchase.
2. The customer requests to add the product to cart.
3. AIMS displays a notification that the product has been successfully added.

- **View product in cart**

1. The customer requests to view the shopping cart.
2. AIMS displays a list of products in the shopping cart.
3. AIMS displays the total amount for the products. (see Table A)

- **Delete product in cart**

- The customer views products in cart
- The customer delete products from cart

3. AIMS displays list of products after delete.
4. AIMS updates and displays the total amount for the products. (see Table A)

- **Update product in cart**

1. The customer views products in cart.
2. The customer updates the products' quantity.
3. AIMS updates and displays the total amount for the products. (see Table A)

- **Alternative flows**

Table 1-Alternative flows of events for Add product to cart

No	Location	Condition	Action	Resume location
•	At Step 2	The quantity added to the cart exceeds the quantity in stock	• AIMS displays error message: The number of products in stock is not enough	Resumes at Step 1

Table 2-Alternative flows of events for Update product in cart

No	Location	Condition	Action	Resume location
•	At Step 2	The updated quantity exceeds the quantity in stock	• AIMS displays error message: The number of products in stock is not enough	Resumes at Step 1

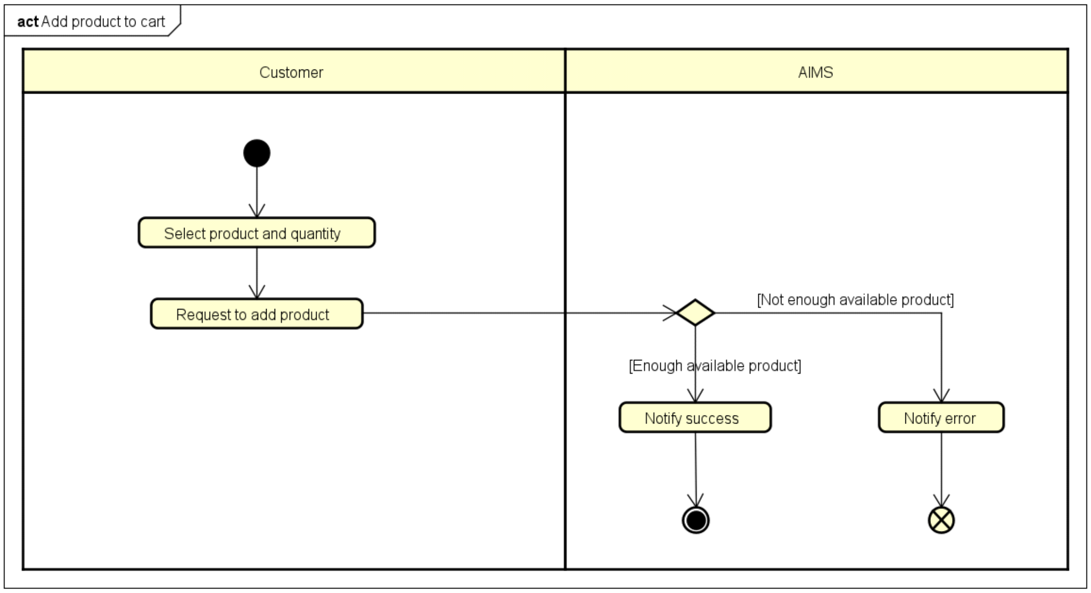
- **Input data**
- **Output data**

Table A-Output data of View products in cart

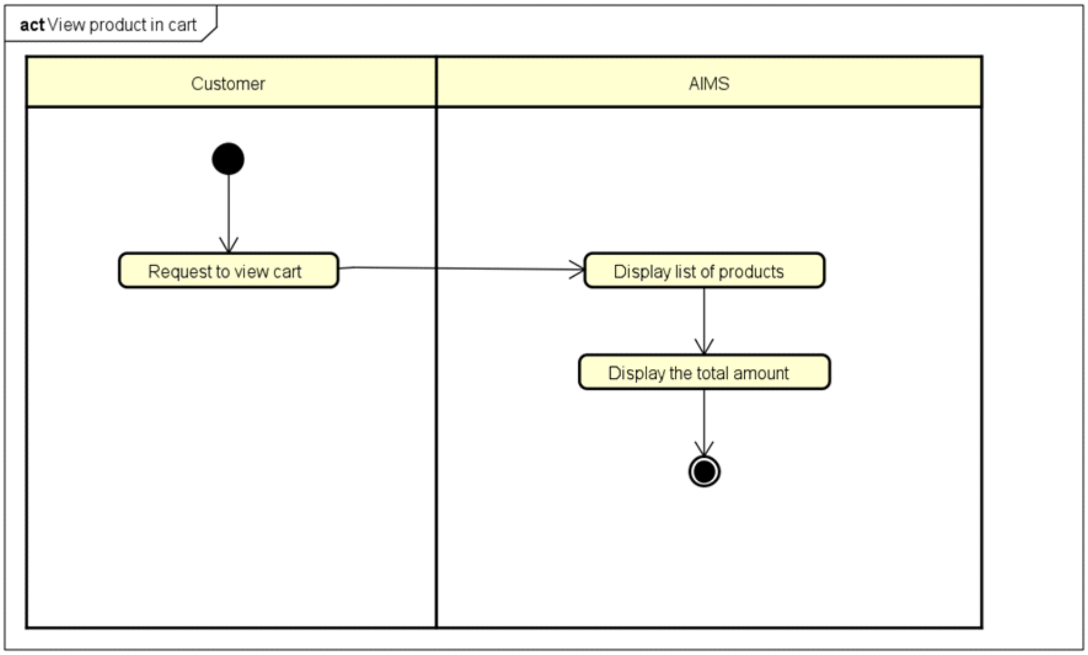
No	Data fields	Description	Display format	Example
•	Image	Image of product's cover	Image	
•	Title	Title of product	Text	Harry Potter
•	Quantity	Quantity of product	- Positive integer - Right alignment	10
•	Price	Price of product	- Comma for thousands Separator - Positive integer	100.000 vnd
•	Subtotal	Total price of products in the cart before VAT		100.000 vnd

•	VAT	Value Added Tax	- Right alignment	10.000 vnd
•	Amount	Total price of products in the cart after VAT		110.000 vnd

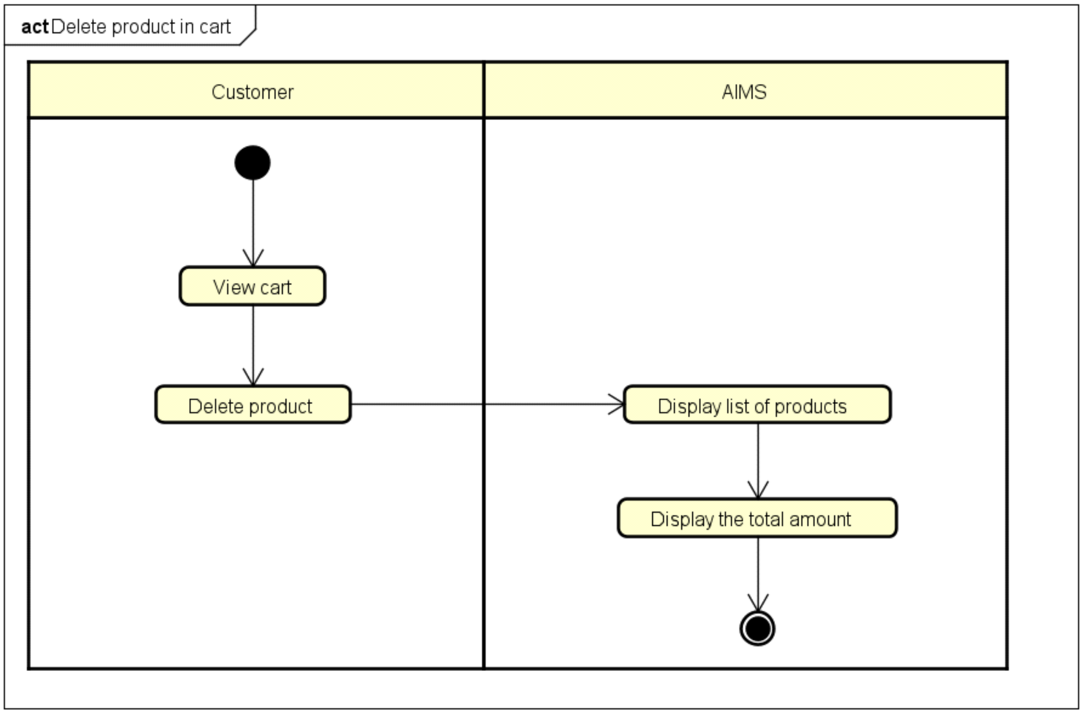
- **Postconditions**
- **Activity Diagrams**
  - **Add product to cart**


act Add product to cart
Customer
AIMS
Select product and quantity
Request to add product
[Not enough available product]
[Enough available product]
Notify success
Notify error

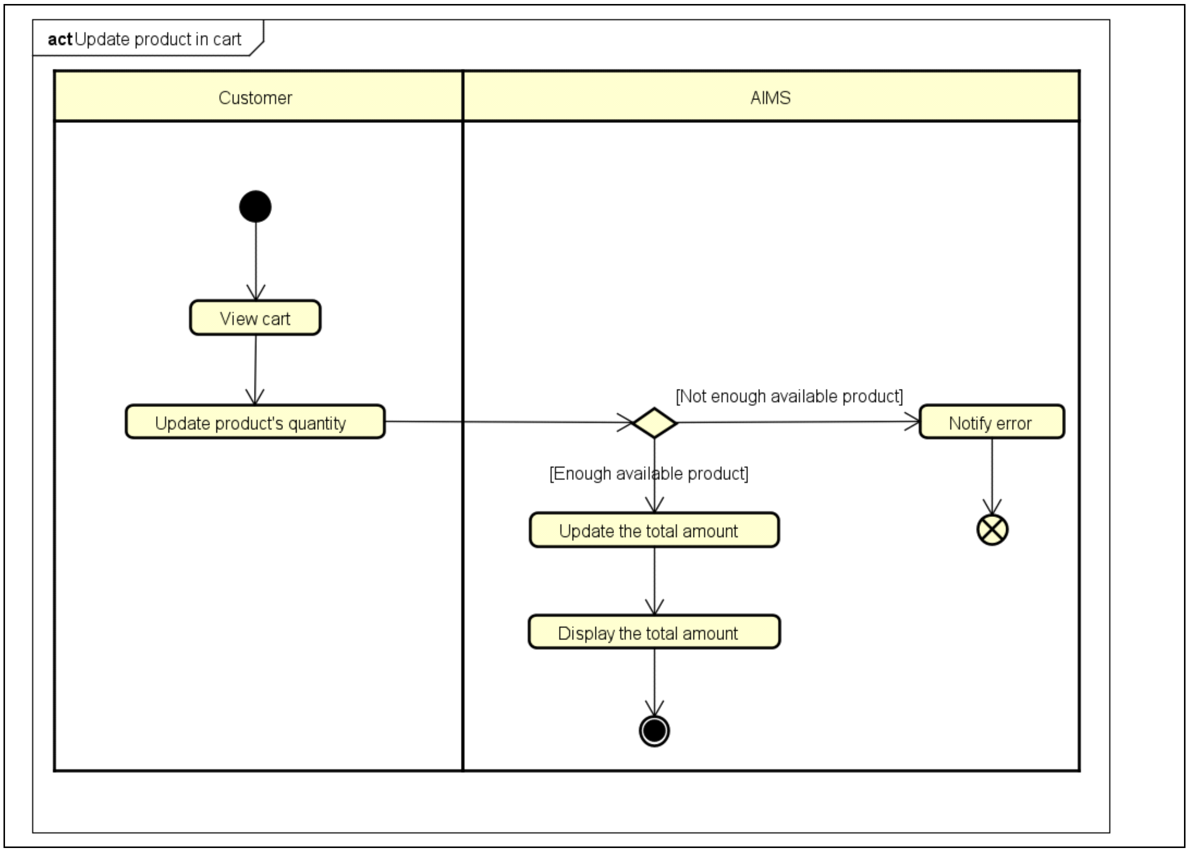
  - **View product in cart**



- **Delete product in cart**



- **Update product in cart**


act Update product in cart
Customer
AIMS
View cart
Update product's quantity
[Not enough available product]
Notify error
[Enough available product]
Update the total amount
Display the total amount

2.5. Usecase Specification Place Order

Use Case "Place Order"

- **Use case code**

  UC005
- **Brief Description**

  This use case describes the interaction between customer and AIMS when customer wish(es) to place order
- **Actors**
  - **Customer**
- **Preconditions**

None

- **Basic Flow of Events**
  - Customers click to view cart.
  - The system calculates the total product price.

- The system checks whether the products in the cart are still in stock.
- The system will display a list of items the customer wants to order (product name, quantity, and price).
- Customer clicks on the “Place Order” button.
- The system checks whether the products in the cart are still in stock.
- The system displays a delivery information form, asking the customer to update shipment details.
- Customers fill in the necessary information and do not select "fast delivery".
- Click “Update”.
- The system checks input information.
- Delivery fee calculation system.
- The system will display temporary order information.
- Call Usecase “Payment”.
- System empties cart.

- **Alternative flows**

No	Location	Condition	Action	Resume location
• 61	At Step 3	If system check in insufficient warehouse products.	• The system will notify customers the product is not enough in stock and requires the customer to update the cart with missing products.	Resumes at Step 4.
•	At step 8	If the customer misses the required information fields blank or write in the wrong format.	• The system will ask the customer to enter complete information.	Resumes at Step 8.
•	At step 8	If the customer selects “Place Rush Order”	• Insert into usecase “Place Rush Order”	Continue usecase “Place Rush Order”
•	At step 1	If there are no products in the cart.	• The system will notify you that there are no products in the cart product	Resumes at Step 1.

Table A-Input data of “delivery information form”

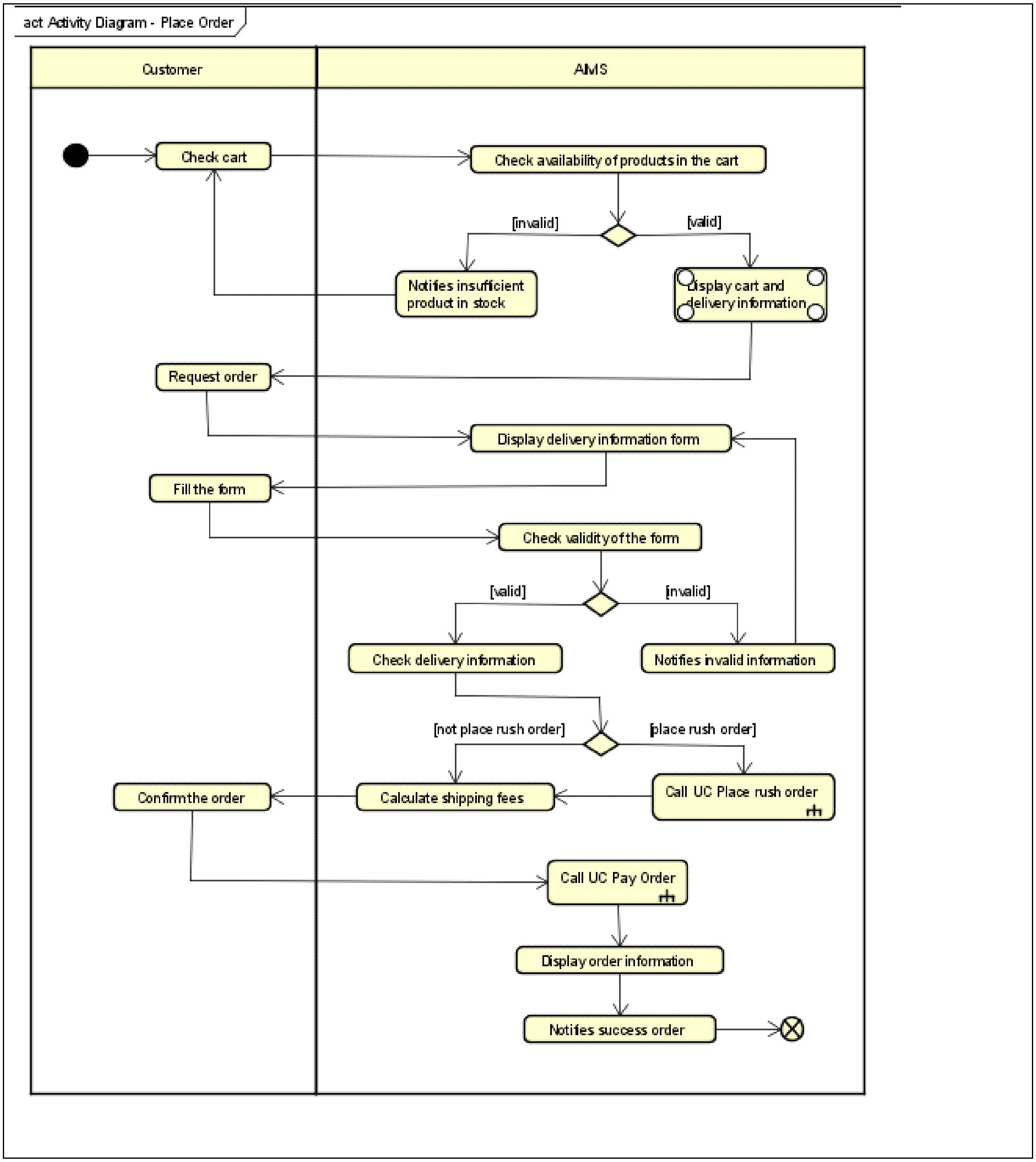
No	Data fields	Description	Mandatory	Valid condition	Example
	Address	Address of customer	Yes	Text	Số 1, Đường Tạ Quang Bửu, quận Hai Bà Trưng.
	Name of customer		Yes	Maximum 30 characters	Nguyen Van A
	Phone number		Yes	From 9 – 11 numbers (first number need be 0)	0998716388
	Province		Yes	Choose from list	Ha Noi

- ## Output data

Table B-Output data of "Temporary order information sheet"

No	Data fields	Description	Display format	Example
•	Title	Name of Product	Text	Harry Potter Book
•	Price	Price of product	- Comma for thousands Separator - Positive integer - Right alignment	100.000 vnd
•	Quanity	Quantity of product	- Positive integer	10
•	Sum price of product	Sum price of each product	- Comma for thousands Separator - Positive integer - Right alignment	300,00 vnđ
•	Total amount payable before calculation shipping fee	Total price of all products before calculation shipping fee	- Comma for thousands Separator - Positive integer - Right alignment	20,000 vnđ
•	Total amount money	Total amout money of all products after calculation shipping fee	- Comma for thousands Separator - Positive integer - Right alignment	320,000 vnđ

- ## Postconditions
- ## Activity Diagrams


act Activity Diagram - Place Order
Customer
AIMS
Check cart
Check availability of products in the cart
[invalid]
[valid]
Notifies insufficient product in stock
Display cart and delivery information
Request order
Display delivery information form
Fill the form
Check validity of the form
[valid]
[invalid]
Check delivery information
Notifies invalid information
[not place rush order]
[place rush order]
Confirm the order
Calculate shipping fees
Call UC Place rush order
Call UC Pay Order
Display order information
Notifies success order

2.6. Usecase Speccification Place Rush Order

Use Case “Place Rush Order”

- **Use case code**

  UC006

- **Brief Description**

  This use case describes the interaction between customer and AIMS when customer wish(es) to place order

- **Actors**
  - **Customer**
- **Preconditions**

None

- **Basic Flow of Events**
  - The system displays the delivery address information fieldsThe system calculates the total product price.
  - User enters information fields.
  - User confirms
  - The system checks whether the fields are valid.
  - The system switches to the delivery method selection screen.
  - The user selects the fast delivery method
  - User confirms.
  - The system calculates delivery fees and displays the invoice screen.
  - User confirms.
  - The system switches to payment.
- **Alternative flows**

No	Location	Condition	Action	Resume location
1	At step 4	If the customer misses the required information fields blank or write in the wrong format	• The system will ask the customer to enter complete information.	Resumes at Step 2.
2	At step 6	The address checking system does not support fast shipping	• Fast delivery is not allowed	Resumes at Step 6.
3	At step 6	There are no product which is fast dilivery sopported	• Fast delivery is not allowed products in the cart product	Resumes at Step 6.

Table A-Input data of “delivery information form”

No	Data fields	Description	Mandatory	Valid condition	Example

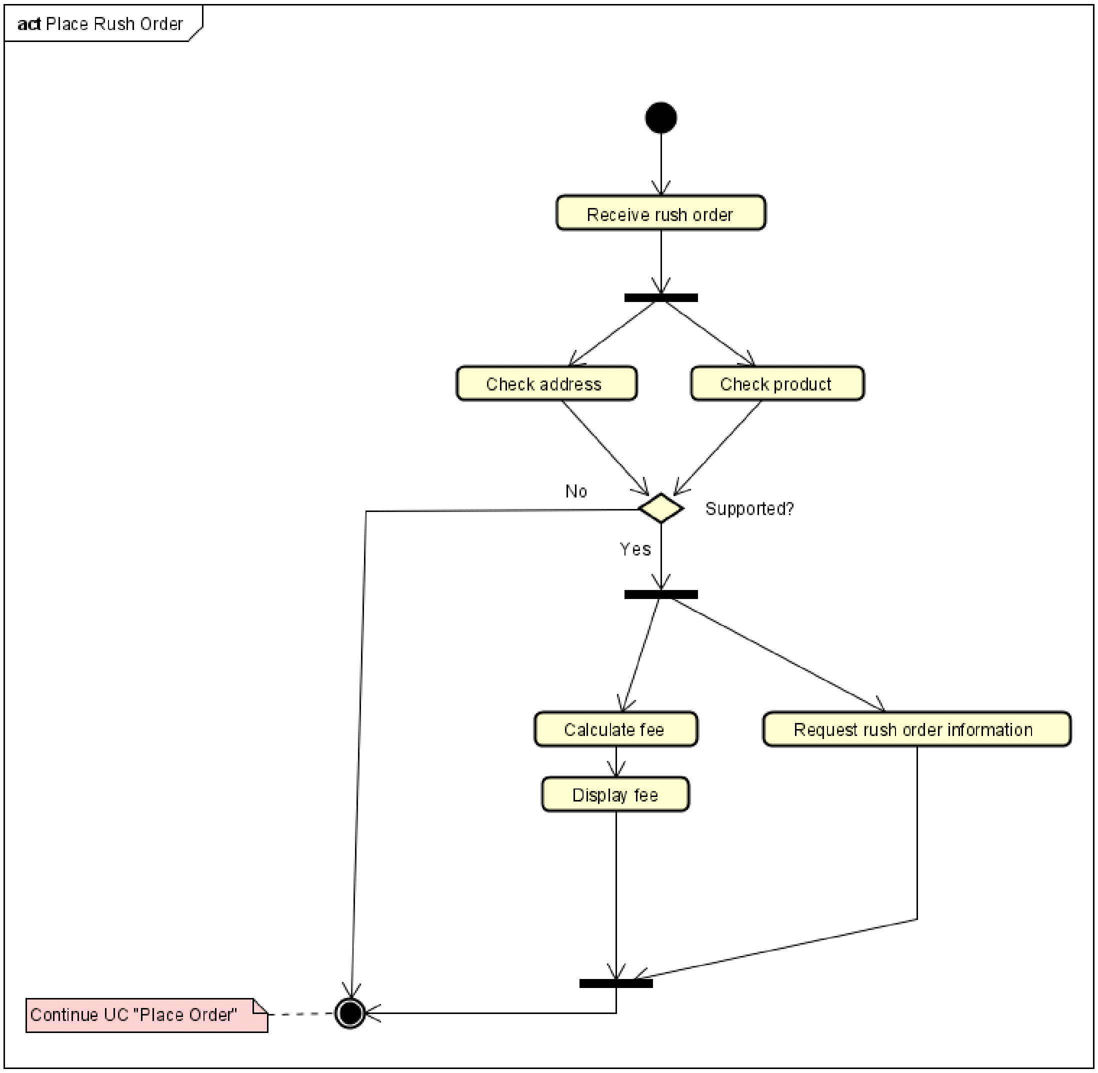
	Address	Address of customer	Yes	Text	Số 1, Đường Tạ Quang Bửu, quận Hai Bà Trưng.
	Name of customer		Yes	Maximum 30 characters	Nguyen Van A
	Phone number		Yes	From 9 – 11 numbers (first number need be 0)	0998716388
	Province		Yes	Choose from list	Ha Noi
	Shipping Instructions		No	Maximum 50 characters	Khong

- **Output data**

Table B-Output data of “Temporary rush order information sheet”

No	Data fields	Description	Display format	Example
•	Title	Name of Product	Text	Harry Potter Book
•	Price	Price of product	- Comma for thousands Separator - Positive integer - Right alignment	100.000 vnd
•	Quanity	Quantity of product	- Positive integer	10
•	Type of delivery		Normal or Fast	Fast
•	Sum price of product	Sum price of each product	- Comma for thousands Separator - Positive integer - Right alignment	300,00 vnđ
•	Total amount payable before calculation shipping fee	Total price of all products before calculation shipping fee	- Comma for thousands Separator - Positive integer - Right alignment	20,000 vnđ
•	Total amount fee of rush delivery	10,000 vnđ for each rush delivery product	- Comma for thousands Separator - Positive integer - Right alignment	50,000 vnđ
•	Total amount money	Total amout money of all products after calculation shipping fee	- Comma for thousands Separator - Positive integer - Right alignment	370,000 vnđ

- **Postconditions**
- **Activity Diagrams**



2.7 Use case specification: Pay order

Usecase code:

UC007

Brief Description:

This usecase describes the interaction between the customer and the AIMS system when the user pay for the order.

Actor:

Customer, VNPay subsystem.

Preconditions:

UC Place Order after user confirm shipping method.

Flow of action:

- The customer confirm the order.

- AIMS redirect the customer to VNPay payment page.
- The customer finish payment for the order through VNPay
- AIMS display Payment Successful page.

Alternate flows for Searching

No	Location	Condition	Action	Resume location
•	At Step 3	User cancel the payment	• AIMS display Payment Failed Page	End usecase

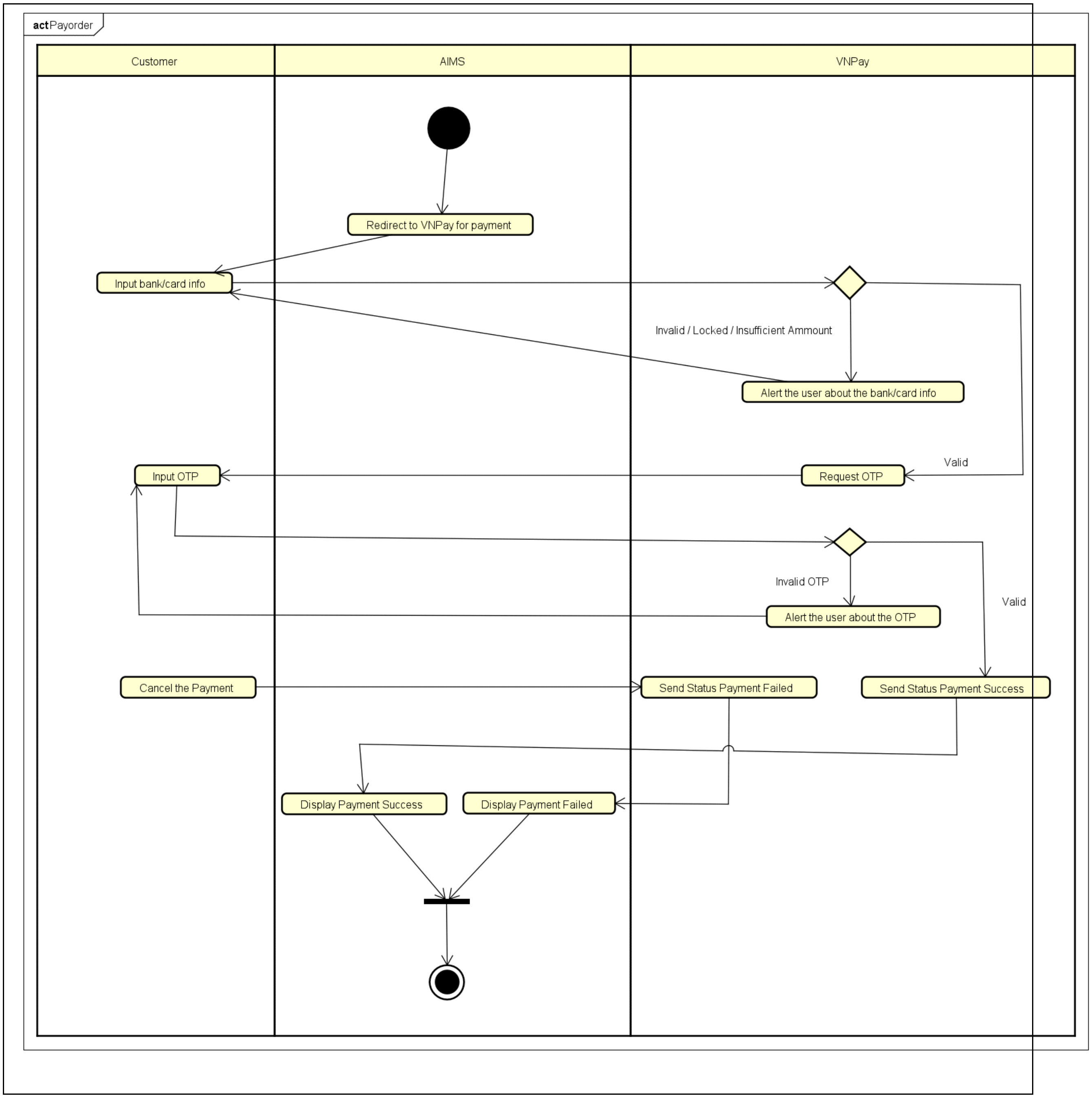
Input data

Output data

Postconditions:

None

Activity Diagram:


actPayorder
Customer
AIMS
VNPay
Redirect to VNPay for payment
Input bank/card info
Invalid / Locked / Insufficient Ammount
Alert the user about the bank/card info
Valid
Input OTP
Request OTP
Invalid OTP
Alert the user about the OTP
Valid
Cancel the Payment
Send Status Payment Failed
Send Status Payment Success
Display Payment Success
Display Payment Failed

2.6. Usecase Specification View list of products on home screen

Use Case “View list of products”

1. **Use case code**

UC008

2. Brief Description

This use case describes the interaction between customer and AIMS when customer wish(es) to view product detail

3. Actors

3.1 Customer

4. Preconditions

5. Basic Flow of Events

1. The customer access AIMS
2. AIMS initialize home screen
3. AIMS get all media information from database
4. AIMS displays list of 20 products on each page

6. Alternative flows

7. Input data

8. Output data

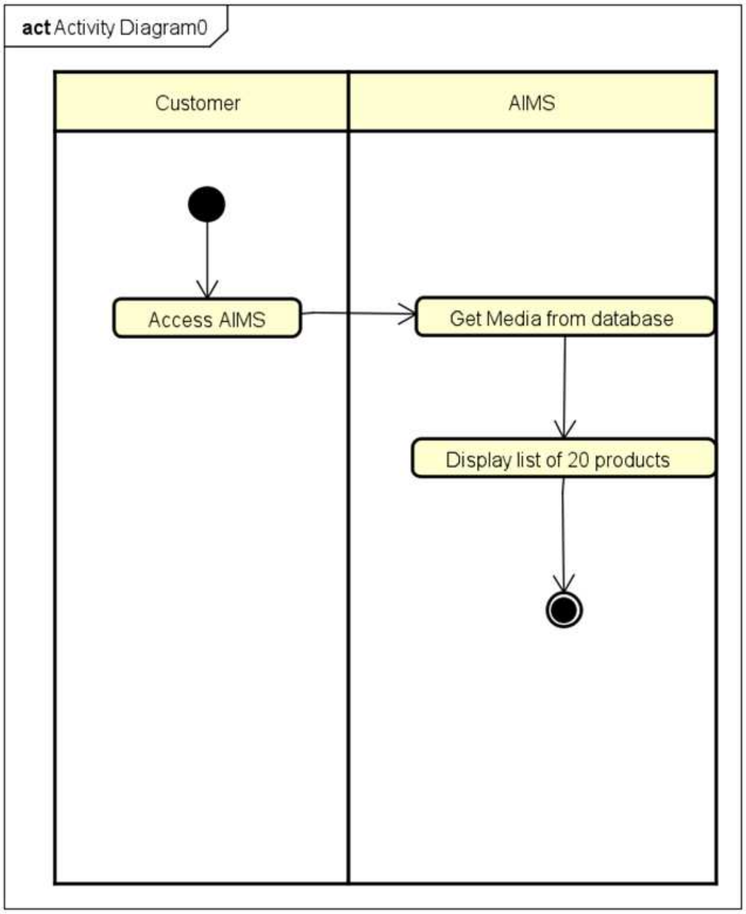
Table A-Output data of Book's detail information

No	Data fields	Description	Display format	Example
1.	Image	Image of product's cover	Image	
2.	Title	Title of product	Text	Harry Potter
3.	Price	Price of product	- Comma for thousands Separator - Positive integer - Right alignment	100.000 vnd

4.	Avail	Available quantity of product	- Positive integer - Right alignment	10

9. Postconditions

10. Activity Diagrams

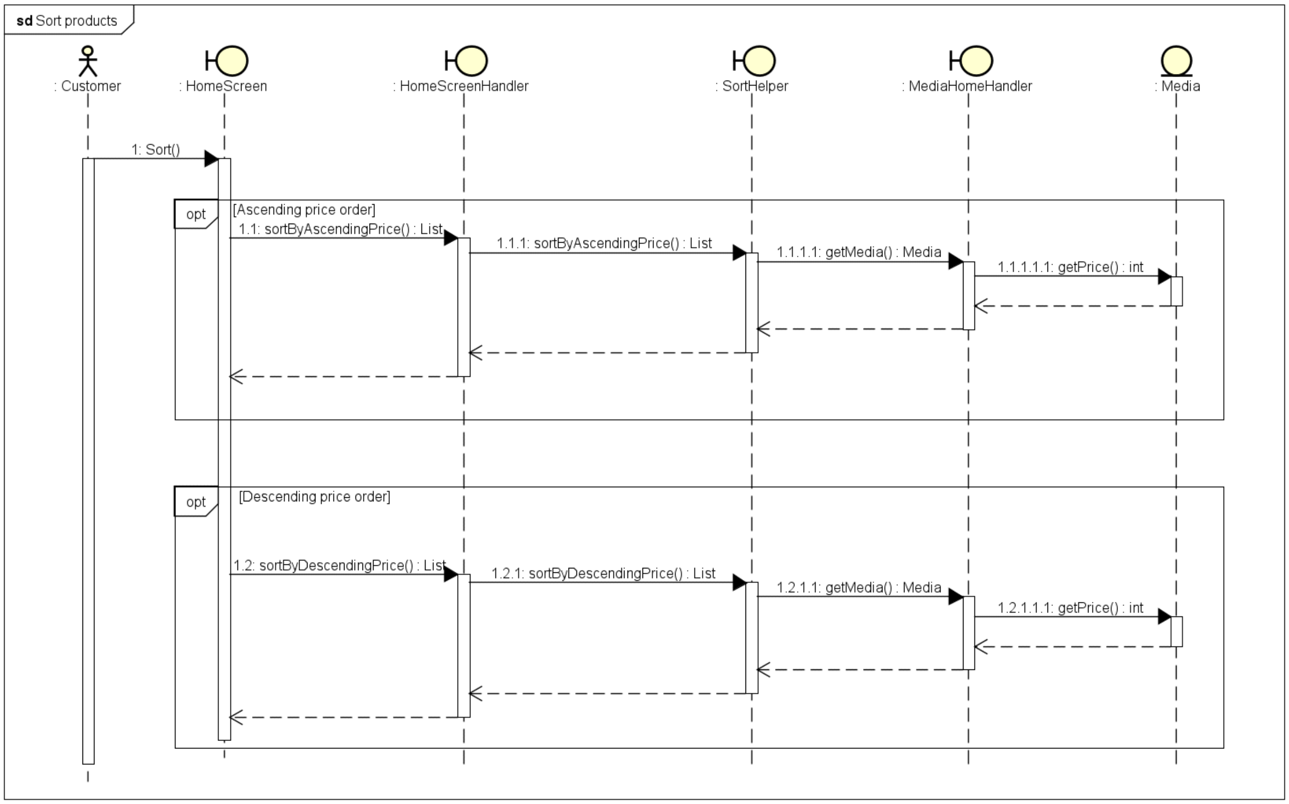


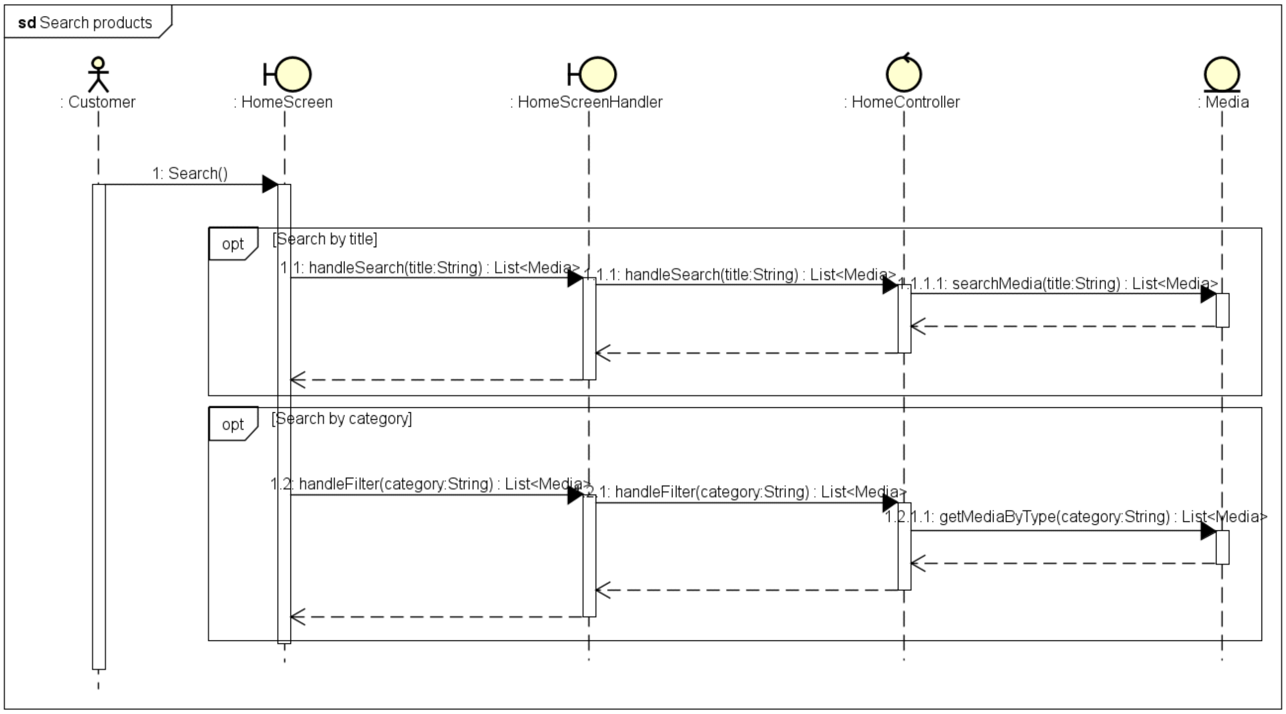
2.7. Usecase Specification for Login

3. Usecase Analysis

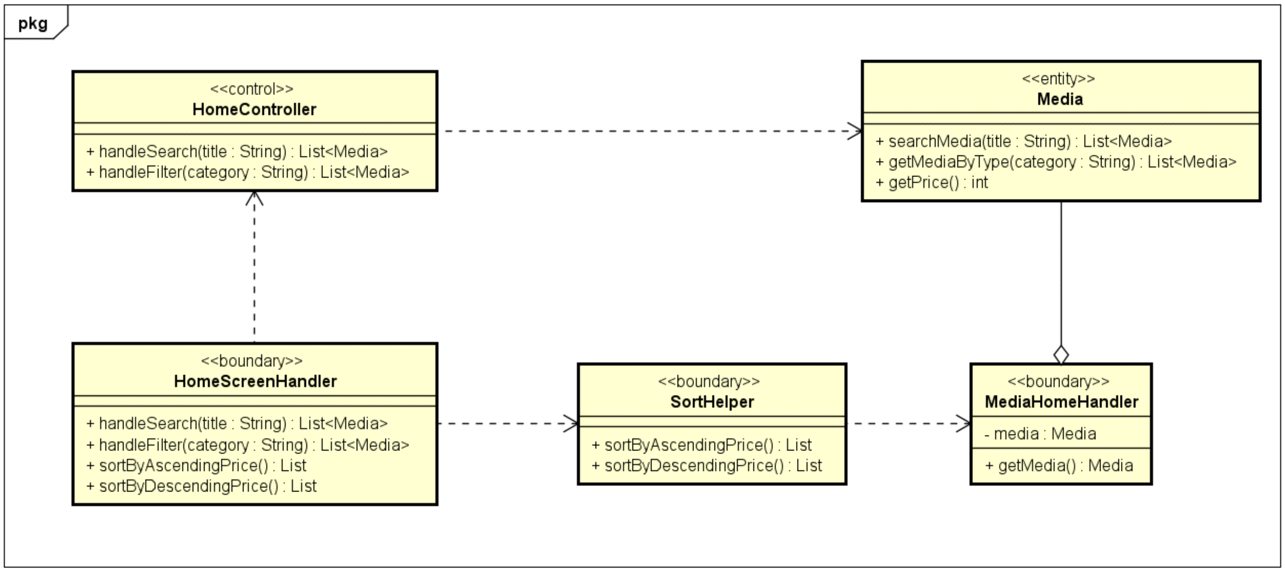
3.1. Usecase Analysis Search/Sort product.

Sequence Diagram:


sd Sort products
: Customer
: HomeScreen
: HomeScreenHandler
: SortHelper
: MediaHomeHandler
: Media
1: Sort()
opt
[Ascending price order]
1.1: sortByAscendingPrice() : List
1.1.1: sortByAscendingPrice() : List
1.1.1.1: getMedia() : Media
1.1.1.1.1: getPrice() : int
opt
[Descending price order]
1.2: sortByDescendingPrice() : List
1.2.1: sortByDescendingPrice() : List
1.2.1.1: getMedia() : Media
1.2.1.1.1: getPrice() : int

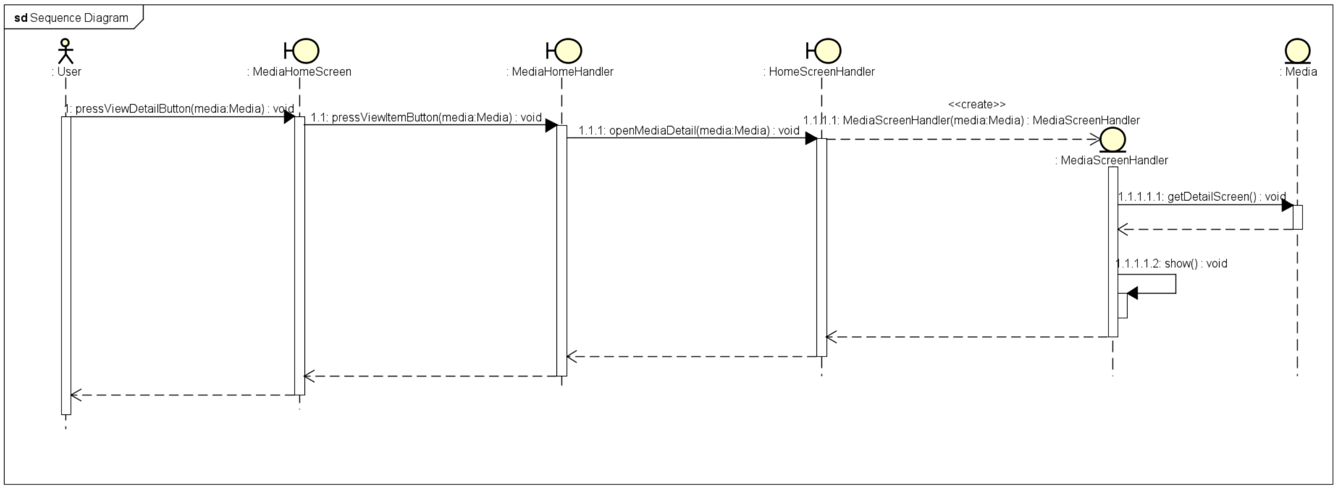

sd Search products
: Customer
: HomeScreen
: HomeScreenHandler
: HomeController
: Media
1: Search()
opt
[Search by title]
1.1: handleSearch(title:String) : List<Media>
1.1.1: handleSearch(title:String) : List<Media>
1.1.1.1: searchMedia(title:String) : List<Media>
opt
[Search by category]
1.2: handleFilter(category:String) : List<Media>
1.2.1: handleFilter(category:String) : List<Media>
1.2.1.1: getMediaByType(category:String) : List<Media>

Class Diagram:

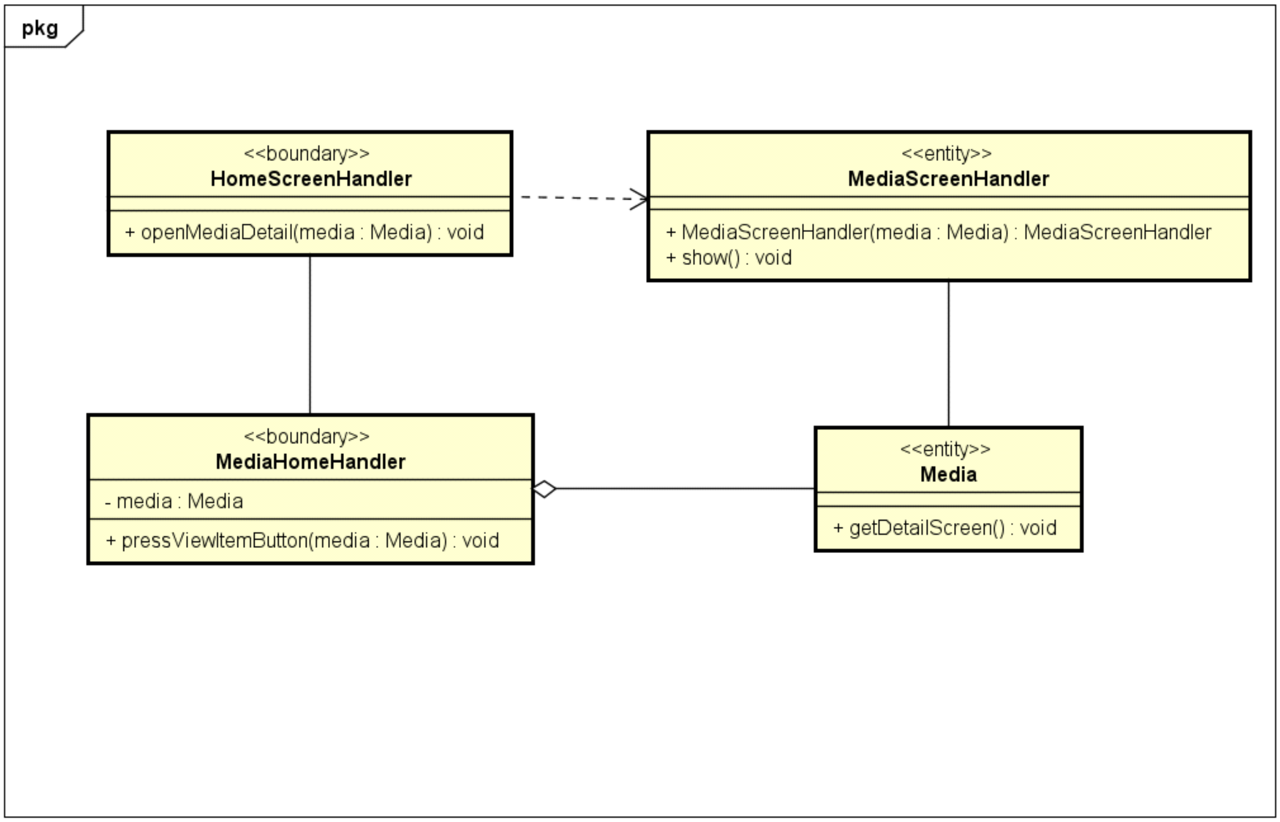

pkg
<<control>>
HomeController
+ handleSearch(title : String) : List<Media>
+ handleFilter(category : String) : List<Media>
<<entity>>
Media
+ searchMedia(title : String) : List<Media>
+ getMediaByType(category : String) : List<Media>
+ getPrice() : int
<<boundary>>
HomeScreenHandler
+ handleSearch(title : String) : List<Media>
+ handleFilter(category : String) : List<Media>
+ sortByAscendingPrice() : List
+ sortByDescendingPrice() : List
<<boundary>>
SortHelper
+ sortByAscendingPrice() : List
+ sortByDescendingPrice() : List
<<boundary>>
MediaHomeHandler
- media : Media
+ getMedia() : Media

3.2. Usecase Analysis View product detail.

Sequence Diagram:

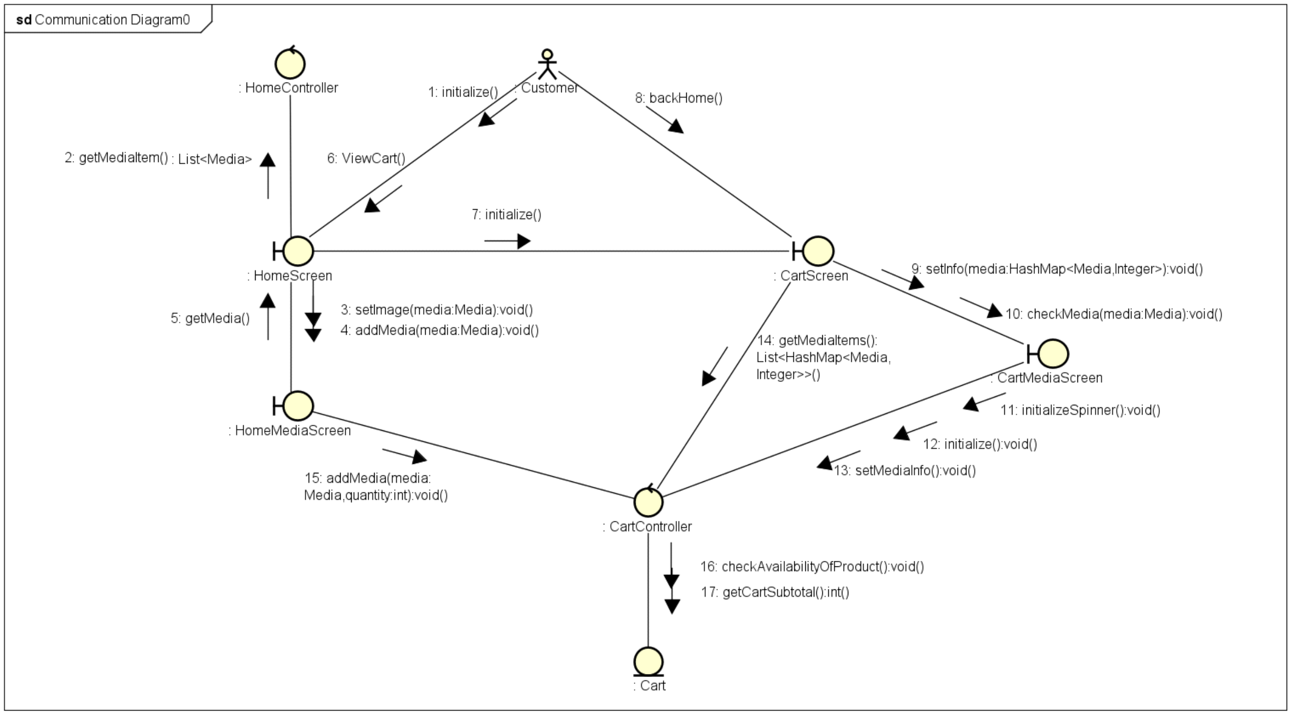

sd Sequence Diagram
: User
: MediaHomeScreen
: MediaHomeHandler
: HomeScreenHandler
: Media
1: pressViewDetailButton(media:Media) : void
1.1: pressViewItemButton(media:Media) : void
1.1.1: openMediaDetail(media:Media) : void
<<create>>
1.1.1.1: MediaScreenHandler(media:Media) : MediaScreenHandler
: MediaScreenHandler
1.1.1.1.1: getDetailScreen() : void
1.1.1.1.2: show() : void

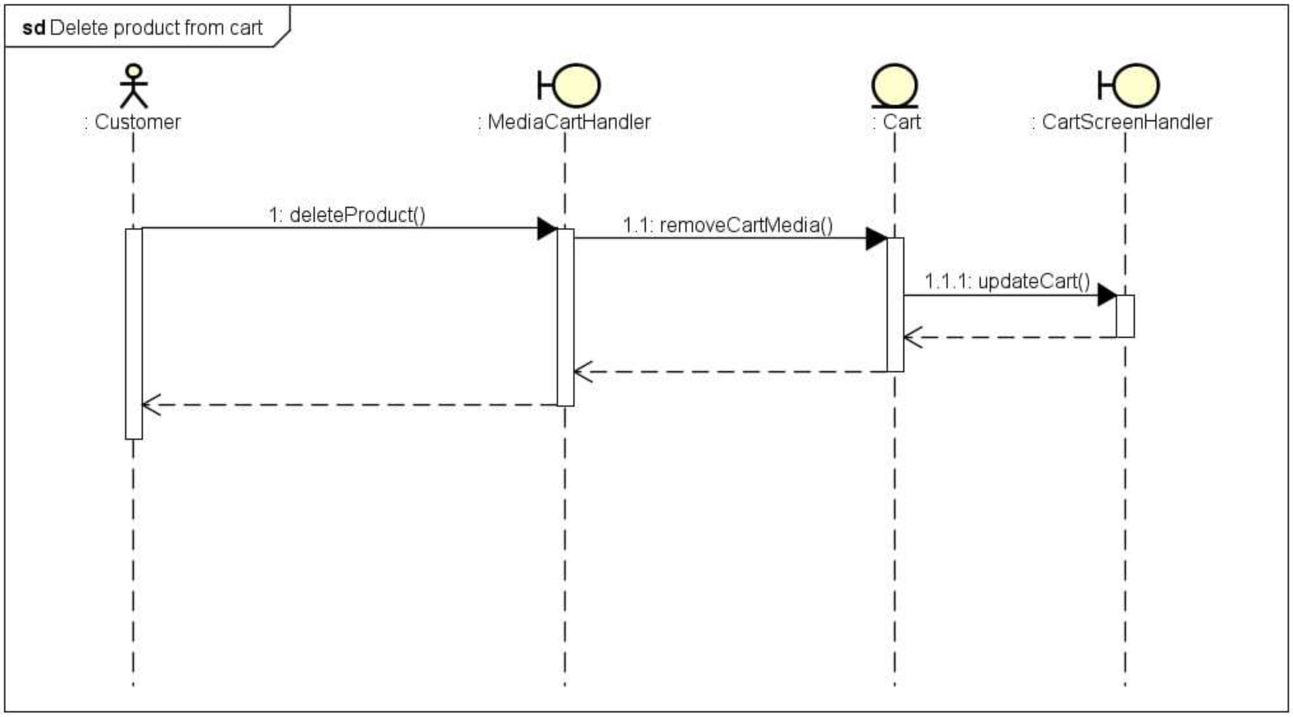
Class Diagram:

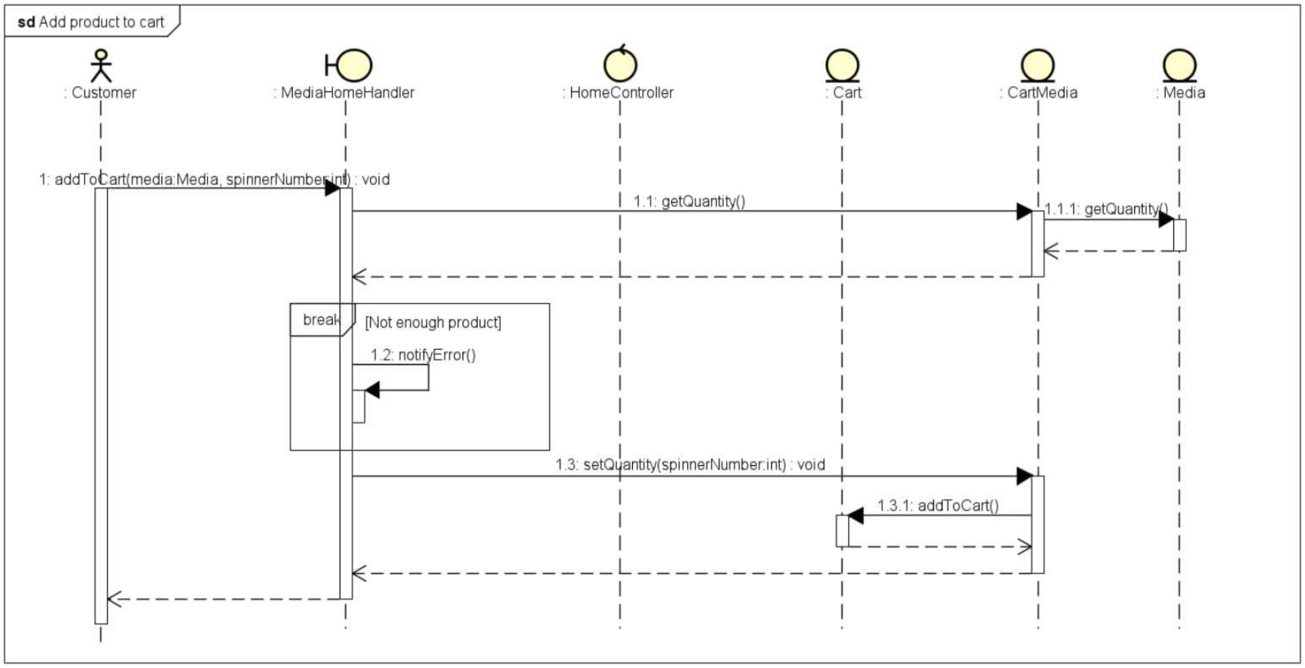

pkg
<<boundary>>
HomeScreenHandler
+ openMediaDetail(media : Media) : void
<<entity>>
MediaScreenHandler
+ MediaScreenHandler(media : Media) : MediaScreenHandler
+ show() : void
<<boundary>>
MediaHomeHandler
- media : Media
+ pressViewItemButton(media : Media) : void
<<entity>>
Media
+ getDetailScreen() : void

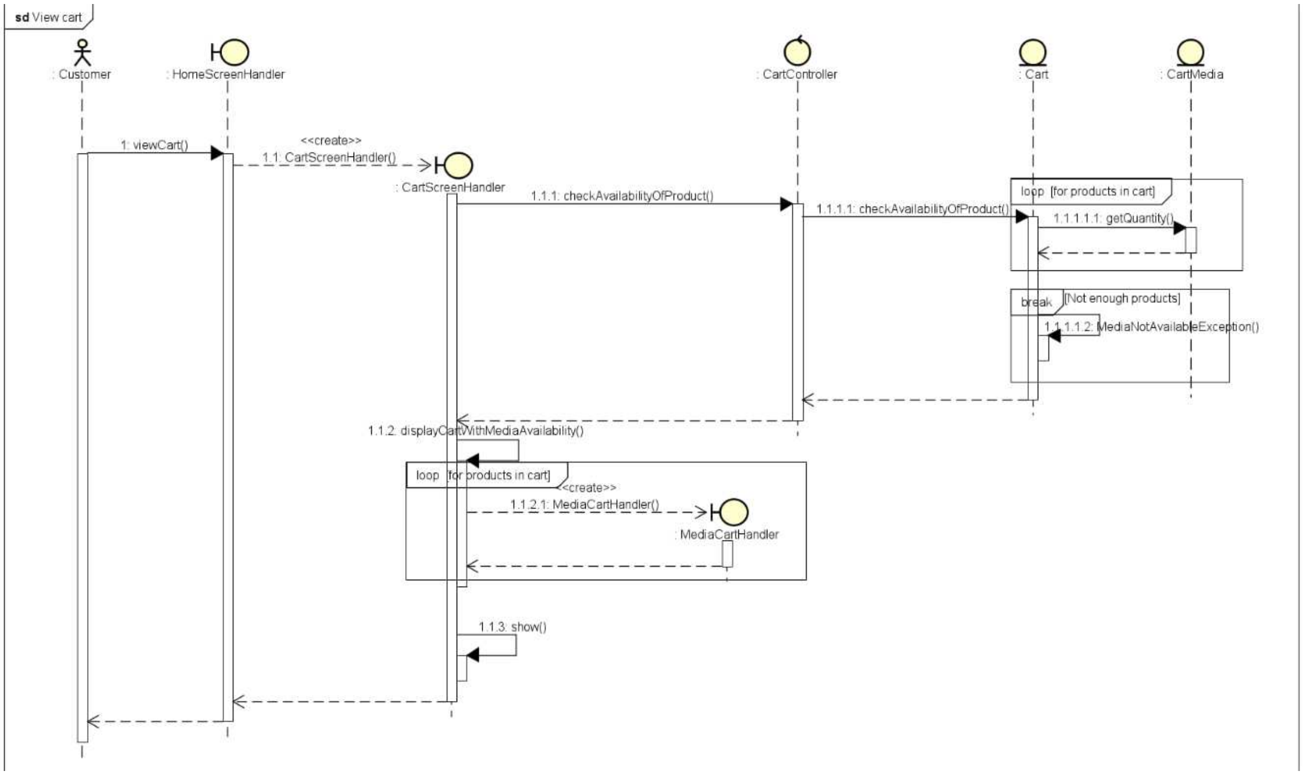
3.3. Usecase Analysis Manage Cart

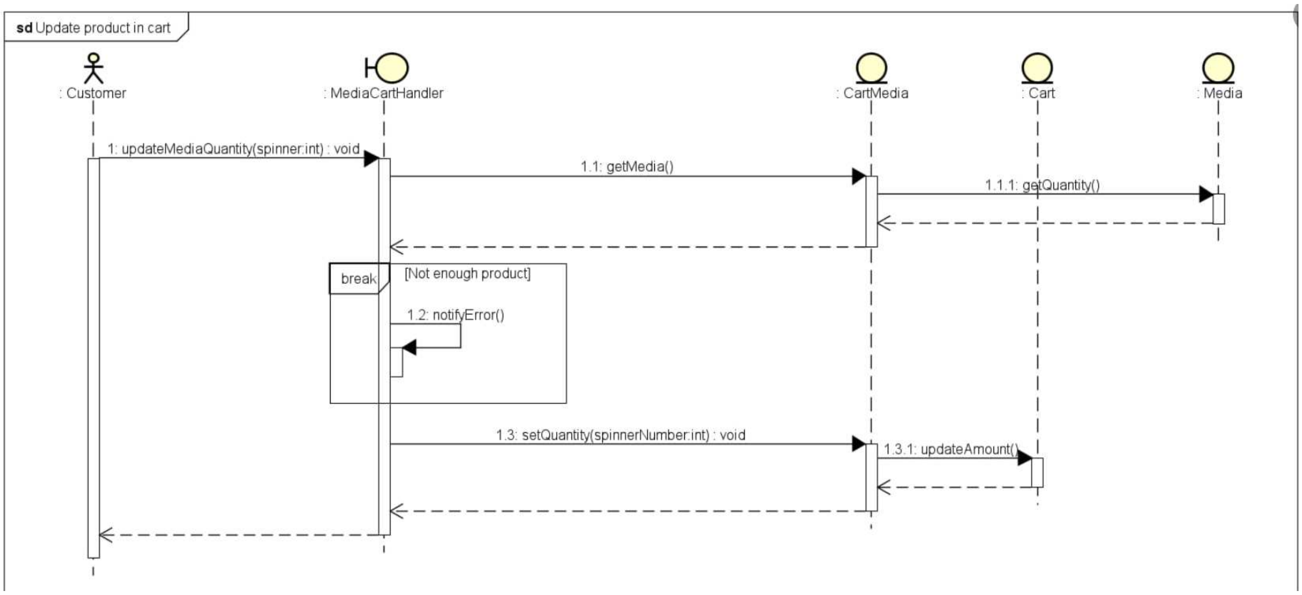
Sequence Diagram:


sd Communication Diagram0
: HomeController
: Customer
1: initialize()
8: backHome()
2: getMediaItem() : List<Media>
6: ViewCart()
7: initialize()
: HomeScreen
: CartScreen
9: setInfo(media:HashMap<Media,Integer>):void()
10: checkMedia(media:Media):void()
5: getMedia()
3: setImage(media:Media):void()
4: addMedia(media:Media):void()
14: getMediaItems():
List<HashMap<Media,
Integer>>()
: CartMediaScreen
11: initializeSpinner():void()
12: initialize():void()
13: setMediaInfo():void()
: HomeMediaScreen
15: addMedia(media:
Media,quantity:int):void()
: CartController
16: checkAvailabilityOfProduct():void()
17: getCartSubtotal():int()
: Cart

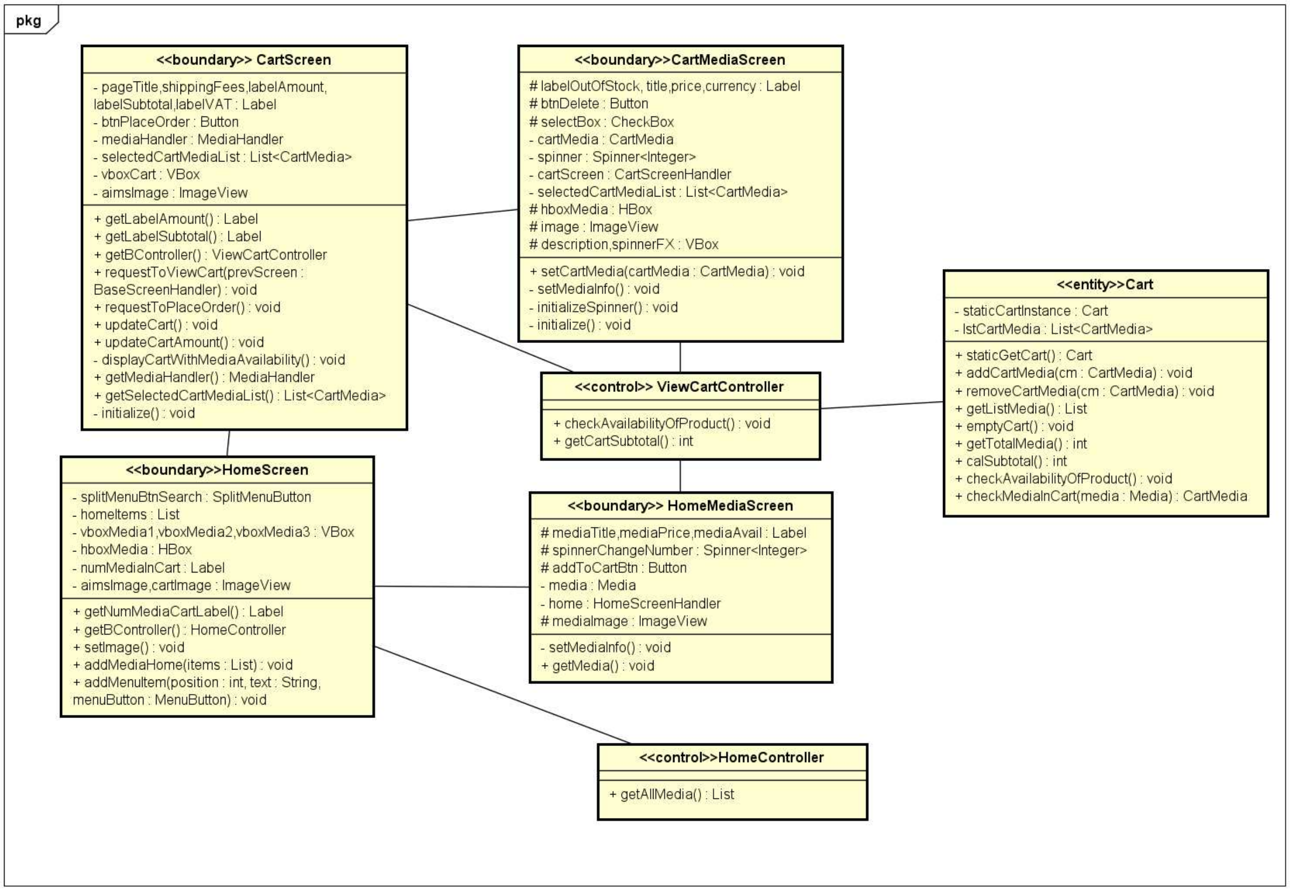

sd Delete product from cart
: Customer
: MediaCartHandler
: Cart
: CartScreenHandler
1: deleteProduct()
1.1: removeCartMedia()
1.1.1: updateCart()


sd Add product to cart
: Customer
: MediaHomeHandler
: HomeController
: Cart
: CartMedia
: Media
1: addToCart(media:Media, spinnerNumber:int) : void
1.1: getQuantity()
1.1.1: getQuantity()
break [Not enough product]
1.2: notifyError()
1.3: setQuantity(spinnerNumber:int) : void
1.3.1: addToCart()


sd View cart
: Customer
: HomeScreenHandler
: CartController
: Cart
: CartMedia
1: viewCart()
<<create>>
1.1: CartScreenHandler()
: CartScreenHandler
1.1.1: checkAvailabilityOfProduct()
1.1.1.1: checkAvailabilityOfProduct()
loop [for products in cart]
1.1.1.1.1: getQuantity()
break [Not enough products]
1.1.1.1.2: MediaNotAvailableException()
1.1.2: displayCartWithMediaAvailability()
loop [for products in cart]
<<create>>
1.1.2.1: MediaCartHandler()
: MediaCartHandler
1.1.3: show()

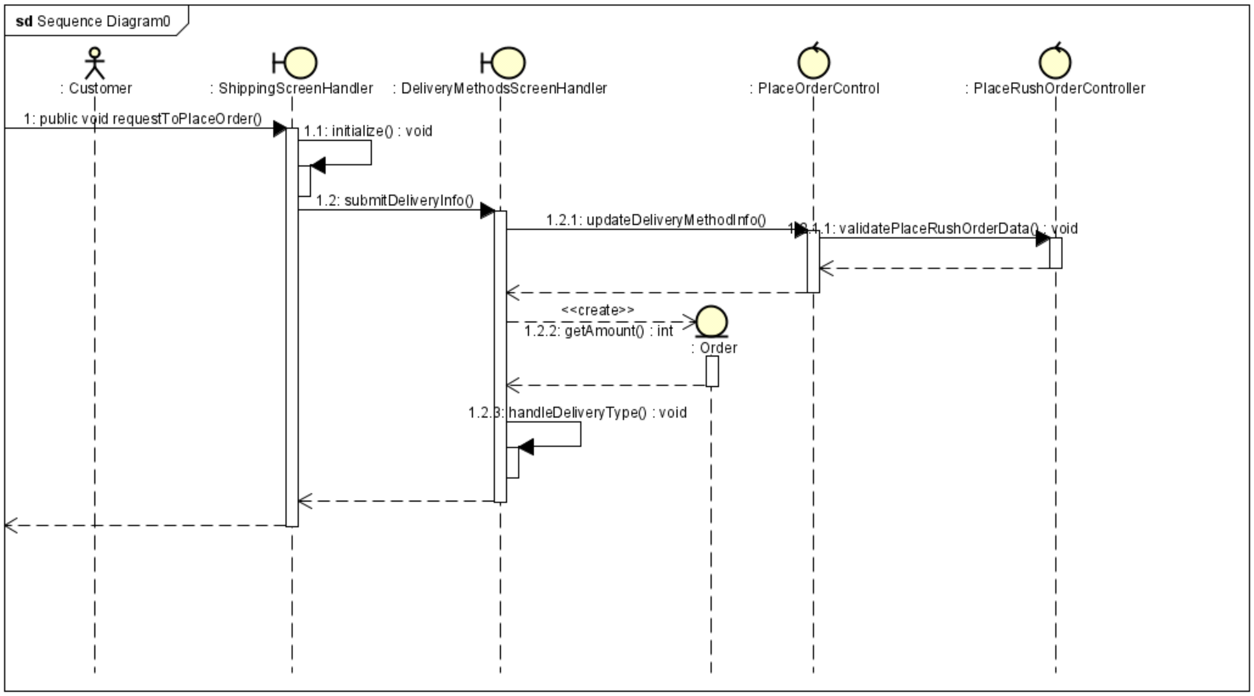


Class Diagram:

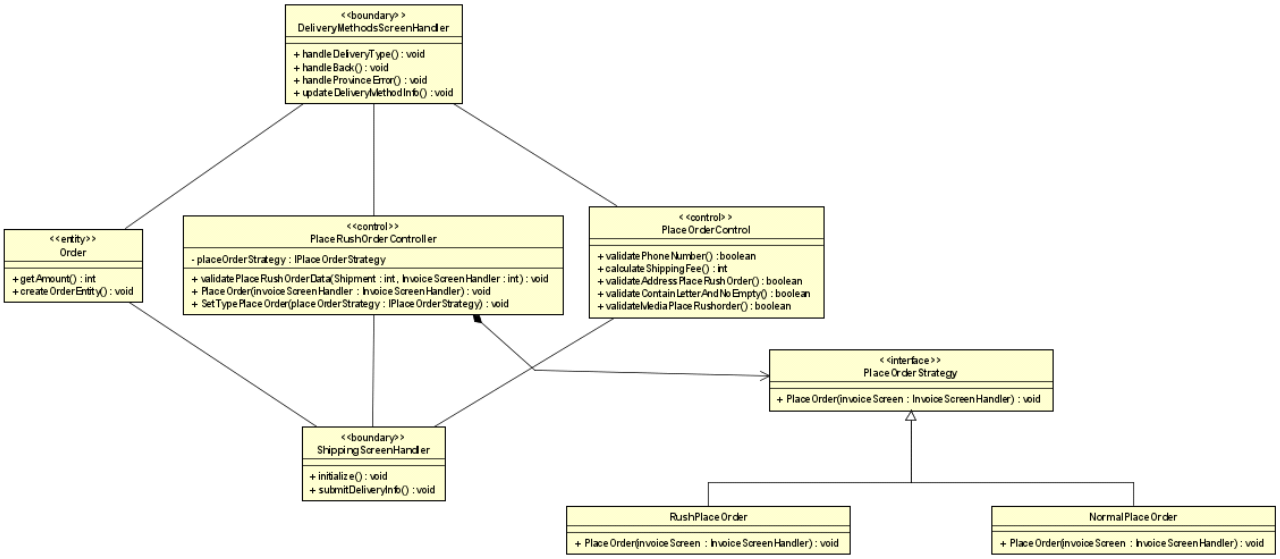

pkg
<<boundary>> CartScreen
- pageTitle,shippingFees,labelAmount,
labelSubtotal,labelVAT : Label
- btnPlaceOrder : Button
- mediaHandler : MediaHandler
- selectedCartMediaList : List<CartMedia>
- vboxCart : VBox
- aimsImage : ImageView
+ getLabelAmount() : Label
+ getLabelSubtotal() : Label
+ getBController() : ViewCartController
+ requestToViewCart(prevScreen :
BaseScreenHandler) : void
+ requestToPlaceOrder() : void
+ updateCart() : void
+ updateCartAmount() : void
- displayCartWithMediaAvailability() : void
+ getMediaHandler() : MediaHandler
+ getSelectedCartMediaList() : List<CartMedia>
- initialize() : void
<<boundary>>CartMediaScreen
# labelOutOfStock, title,price,currency : Label
# btnDelete : Button
# selectBox : CheckBox
- cartMedia : CartMedia
- spinner : Spinner<Integer>
- cartScreen : CartScreenHandler
- selectedCartMediaList : List<CartMedia>
# hboxMedia : HBox
# image : ImageView
# description,spinnerFX : VBox
+ setCartMedia(cartMedia : CartMedia) : void
- setMediaInfo() : void
- initializeSpinner() : void
- initialize() : void
<<entity>>Cart
- staticCartInstance : Cart
- lstCartMedia : List<CartMedia>
+ staticGetCart() : Cart
+ addCartMedia(cm : CartMedia) : void
+ removeCartMedia(cm : CartMedia) : void
+ getListMedia() : List
+ emptyCart() : void
+ getTotalMedia() : int
+ calSubtotal() : int
+ checkAvailabilityOfProduct() : void
+ checkMediaInCart(media : Media) : CartMedia
<<control>> ViewCartController
+ checkAvailabilityOfProduct() : void
+ getCartSubtotal() : int
<<boundary>>HomeScreen
- splitMenuBtnSearch : SplitMenuButton
- homeItems : List
- vboxMedia1,vboxMedia2,vboxMedia3 : VBox
- hboxMedia : HBox
- numMediaInCart : Label
- aimsImage,cartImage : ImageView
+ getNumMediaCartLabel() : Label
+ getBController() : HomeController
+ setImage() : void
+ addMediaHome(items : List) : void
+ addMenuItem(position : int, text : String,
menuButton : MenuButton) : void
<<boundary>> HomeMediaScreen
# mediaTitle,mediaPrice,mediaAvail : Label
# spinnerChangeNumber : Spinner<Integer>
# addToCartBtn : Button
- media : Media
- home : HomeScreenHandler
# mediaImage : ImageView
- setMediaInfo() : void
+ getMedia() : void
<<control>>HomeController
+ getAllMedia() : List

3.4. Usecase Analysis Place Order.

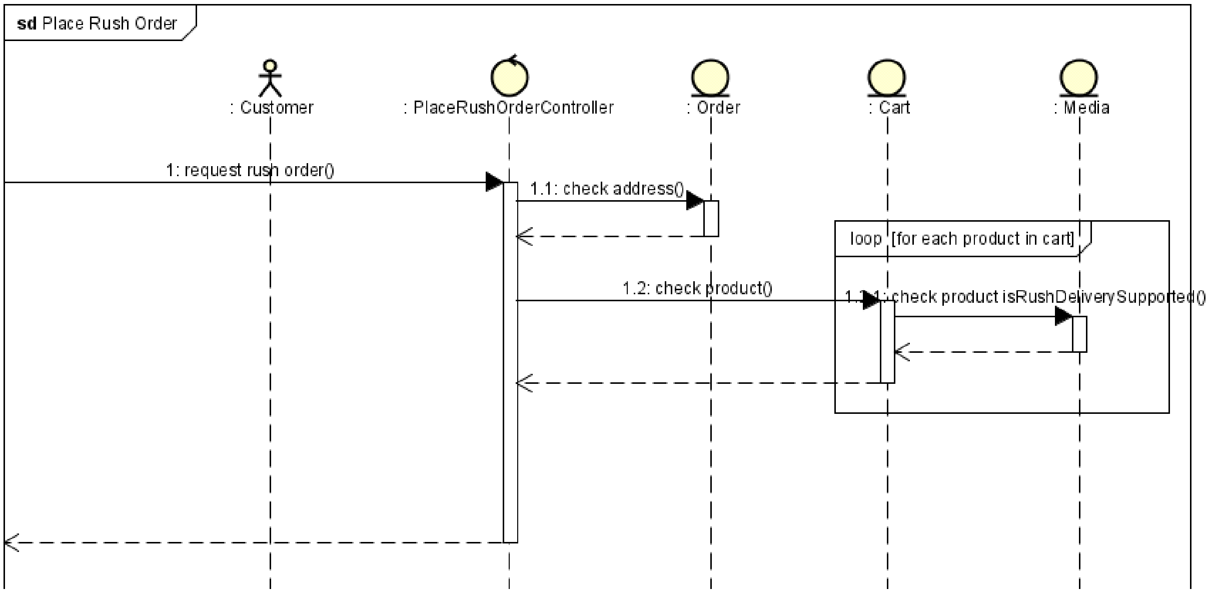
Sequence Diagram:


sd Sequence Diagram0
: Customer
: ShippingScreenHandler
: DeliveryMethodsScreenHandler
: PlaceOrderControl
: PlaceRushOrderController
1: public void requestToPlaceOrder()
1.1: initialize() : void
1.2: submitDeliveryInfo()
1.2.1: updateDeliveryMethodInfo()
1.2.1.1: validatePlaceRushOrderData() : void
<<create>>
1.2.2: getAmount() : int
: Order
1.2.3: handleDeliveryType() : void

Class diagram:

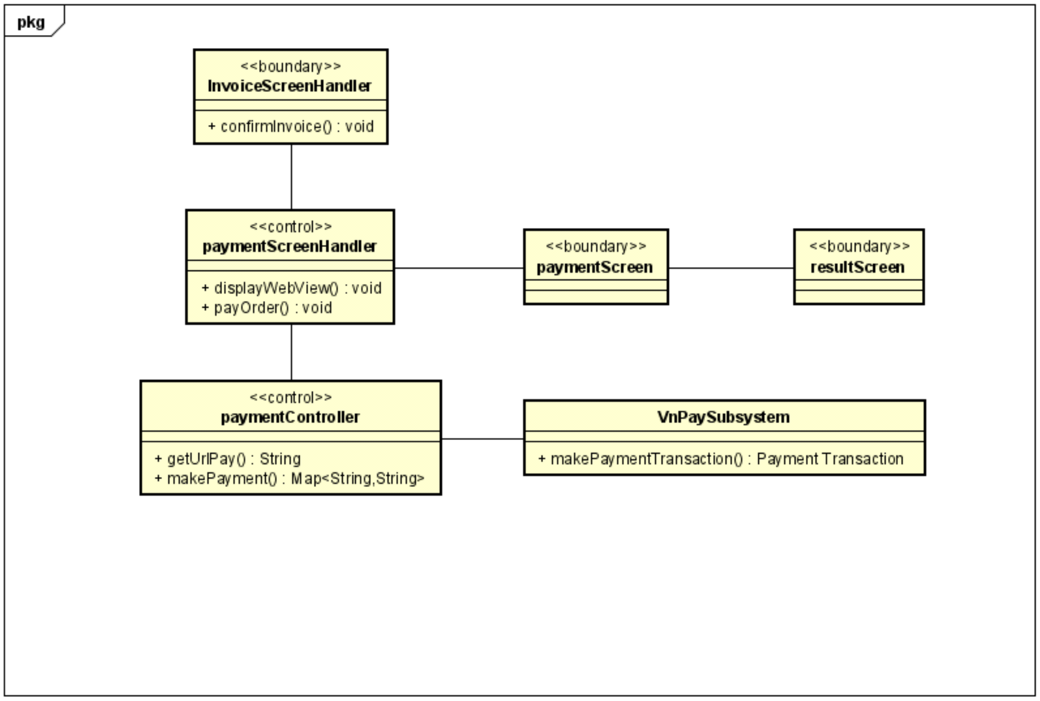

<<boundary>>
DeliveryMethodsScreenHandler
+ handleDeliveryType() : void
+ handleBack() : void
+ handleProvinceError() : void
+ updateDeliveryMethodInfo() : void
<<entity>>
Order
+ getAmount() : int
+ createOrderEntity() : void
<<control>>
PlaceRushOrderController
- placeOrderStrategy : IPlaceOrderStrategy
+ validatePlaceRushOrderData(Shipment : int, InvoiceScreenHandler : int) : void
+ PlaceOrder(invoiceScreenHandler : InvoiceScreenHandler) : void
+ SetTypePlaceOrder(placeOrderStrategy : IPlaceOrderStrategy) : void
<<control>>
PlaceOrderControl
+ validatePhoneNumber() : boolean
+ calculateShippingFee() : int
+ validateAddressPlaceRushOrder() : boolean
+ validateContainLetterAndNoEmpty() : boolean
+ validateMediaPlaceRushorder() : boolean
<<interface>>
PlaceOrderStrategy
+ PlaceOrder(invoiceScreen : InvoiceScreenHandler) : void
<<boundary>>
ShippingScreenHandler
+ initialize() : void
+ submitDeliveryInfo() : void
RushPlaceOrder
+ PlaceOrder(invoiceScreen : InvoiceScreenHandler) : void
NormalPlaceOrder
+ PlaceOrder(invoiceScreen : InvoiceScreenHandler) : void

- **Usecase Analysis Place Rush Order**

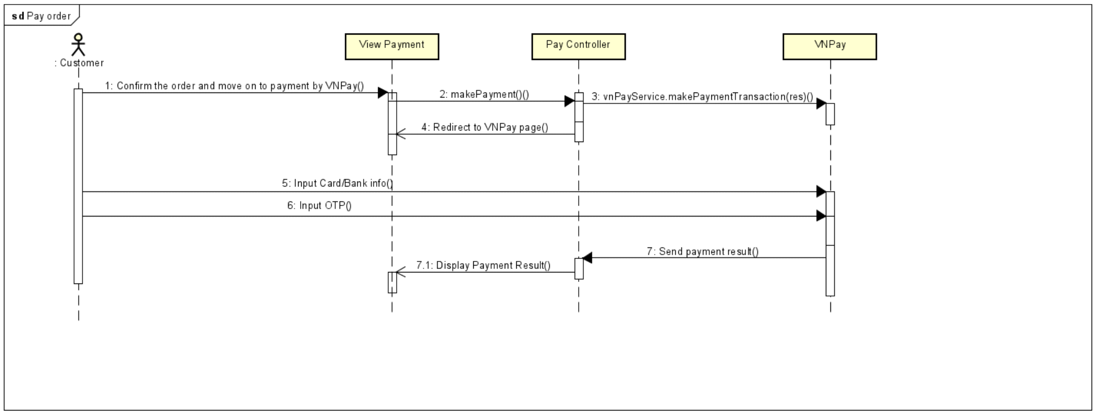


- **Usecase Analysis Pay Order**

Class diagram:

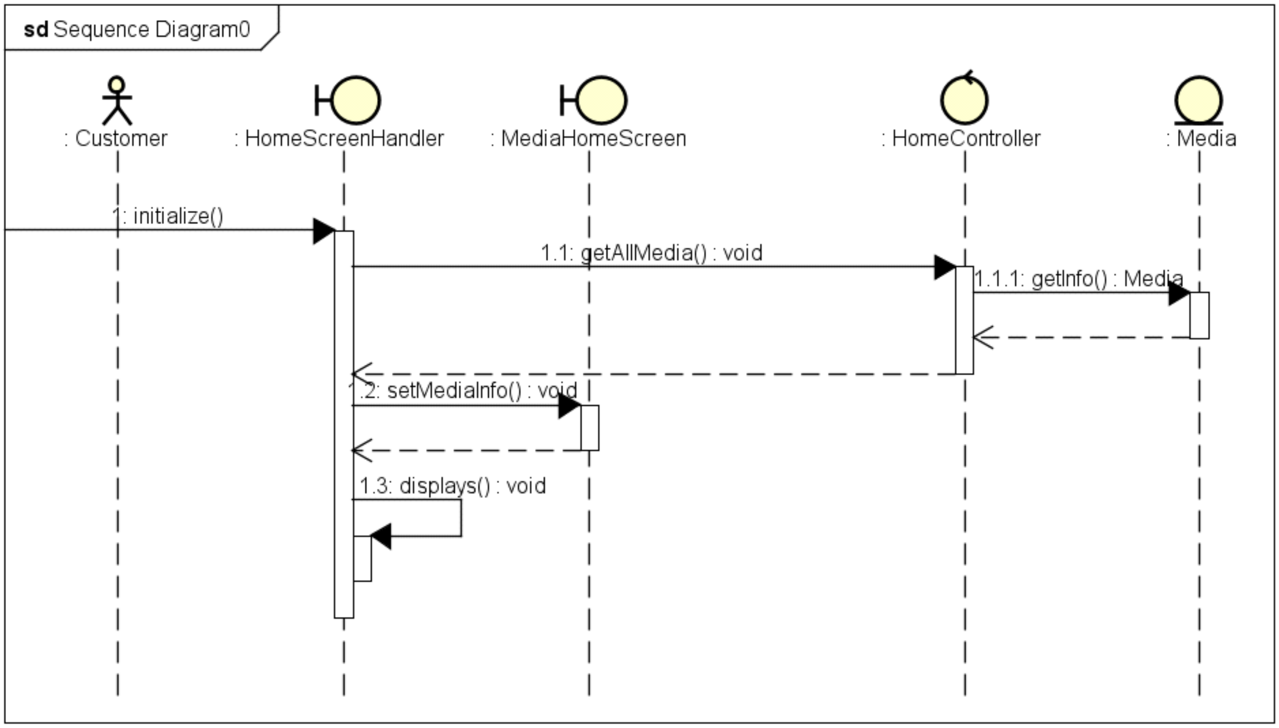


Sequence Diagram:

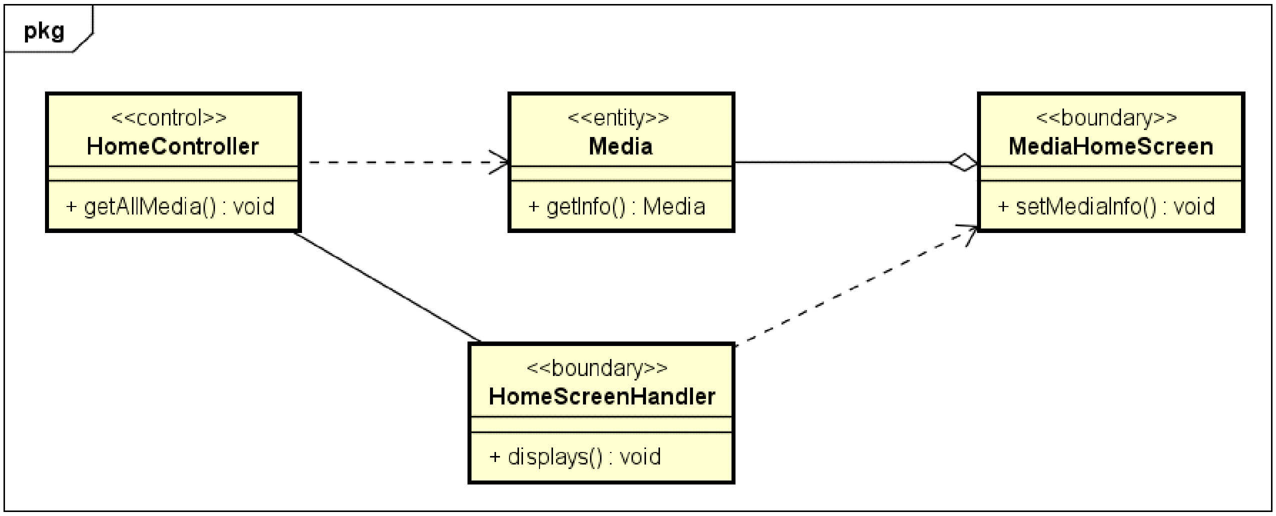

sd Pay order
: Customer
View Payment
Pay Controller
VNPay
1: Confirm the order and move on to payment by VNPay()
2: makePayment()()
3: vnPayService.makePaymentTransaction(res)()
4: Redirect to VNPay page()
5: Input Card/Bank info()
6: Input OTP()
7: Send payment result()
7.1: Display Payment Result()

- **Usecase Analysis View list of products**

Sequence Diagram:

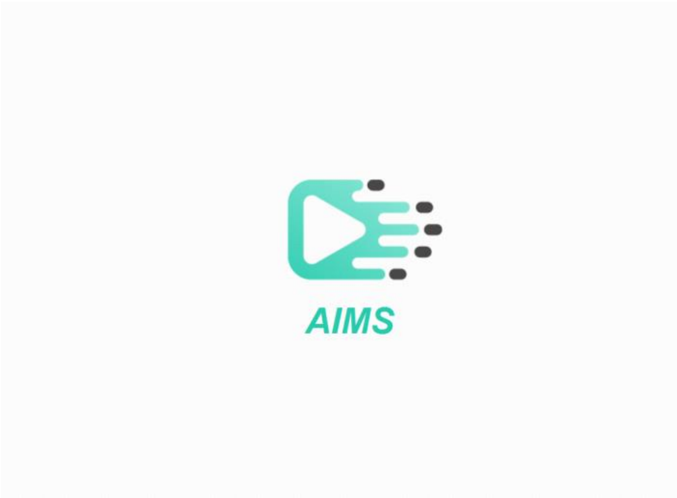


Class Diagram:

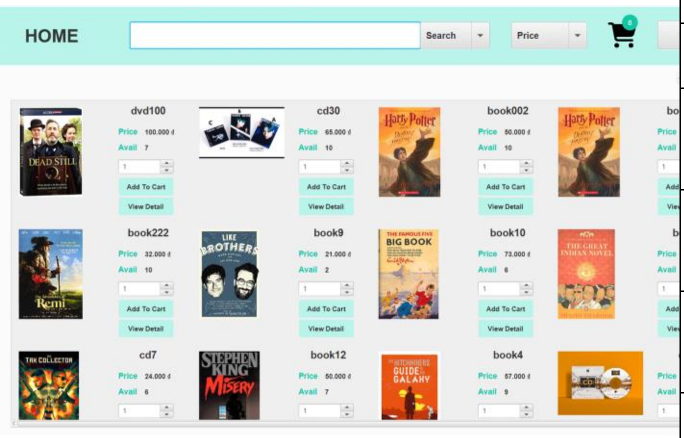


4. Interface Design

4.1. Splash Screen

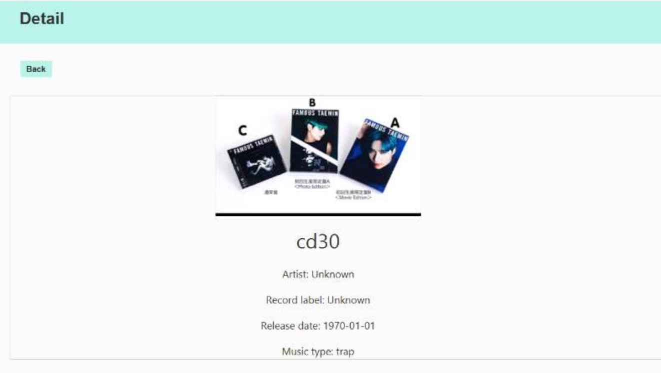
Aims Software		**Date of creation**	**Approved by**	**Reviewed by**	**Persion in charge**
Screen specification	**Splash screen**	17/06/2024			Trinh Tien Dung
		**Control**	**Operation**	**Function**	
		Main area	None	Loading	

4.2. Home Screen

<table>
<tr><td colspan="2"><b>Aims Software</b></td><td><b>Date of creation</b></td><td><b>Approved by</b></td><td><b>Reviewed by</b></td><td><b>Persion in charge</b></td></tr>
<tr><td><b>Screen specification</b></td><td><b>Home screen</b></td><td>17/06/2024</td><td></td><td></td><td>Trinh Tien Dung</td></tr>
<tr><td colspan="2" rowspan="9">
dvd100 Price 100.000 đ Avail 7
cd30 Price 65.000 đ Avail 10
book002 Price 50.000 đ Avail 10
book222 Price 32.000 đ Avail 10
book9 Price 21.000 đ Avail 2
book10 Price 73.000 đ Avail 6
cd7 Price 24.000 đ Avail 6
book12 Price 50.000 đ Avail 7
book4 Price 57.000 đ Avail 9</td><td><b>Control</b></td><td><b>Operation</b></td><td colspan="2"><b>Function</b></td></tr>
<tr><td>Page title</td><td>Click</td><td colspan="2">Reload home screen</td></tr>
<tr><td>Search bar</td><td>Type and Click</td><td colspan="2">Category or title of search items and search</td></tr>
<tr><td>Cart logo</td><td>Click</td><td colspan="2">View items in the cart</td></tr>
<tr><td>Scroll bar</td><td>Scroll</td><td colspan="2">Scroll to see hidden items</td></tr>
<tr><td>Login button</td><td>Click</td><td colspan="2">Login as user</td></tr>
<tr><td>Price button</td><td>Select and click</td><td colspan="2">Sort items by price</td></tr>
<tr><td>Pagination</td><td>Click</td><td colspan="2">Go to other pages</td></tr>
<tr><td>Main area</td><td>Click</td><td colspan="2">Items detail info and add to cart</td></tr>
</table>

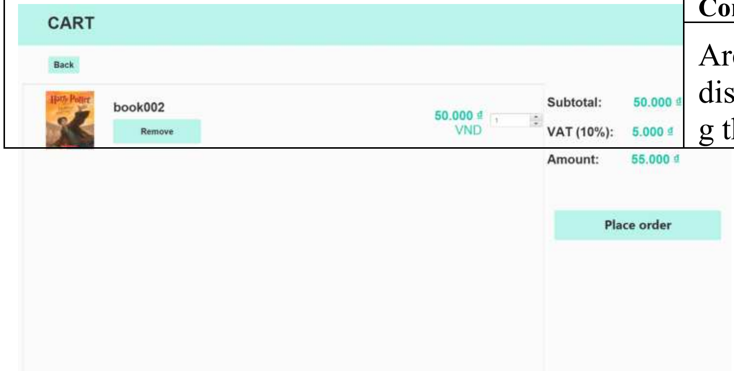
Screen	**Main screen**			
Field name	**Type**	**Limitation**	**Attribute**	**Remarks**
Media title	Text	40 characters	Bold, black	Left-justified
Price	Digits	10 digits	Bold, black	Left-justified Dot for thousand separation
Avail	Digits	5 digits	Bold, black	Left-justified
Item image	Image	158x178 pixels	None	None

4.3. Product detail screen

Aims Software		**Date of creation**	**Approved by**	**Reviewed by**	**Persion in charge**
Screen specification	**Product detail screen**	17/06/2024			Trinh Tien Dung
 Detail Back cd30 Artist: Unknown Record label: Unknown Release date: 1970-01-01 Music type: trap		**Control**	**Operation**	**Function**	
		Area for display items in the cart	Initial	Displays detail information base on category	
		Back button	Click	Go to back to home screen	

Screen	**Main screen**			
Field name	**Type**	**Limitation**	**Attribute**	**Remarks**
Media title	Text	40 characters	Black	None
Item image	Image	400 width pixels	None	None
Media detail information	Text	1000 characters	Black	None

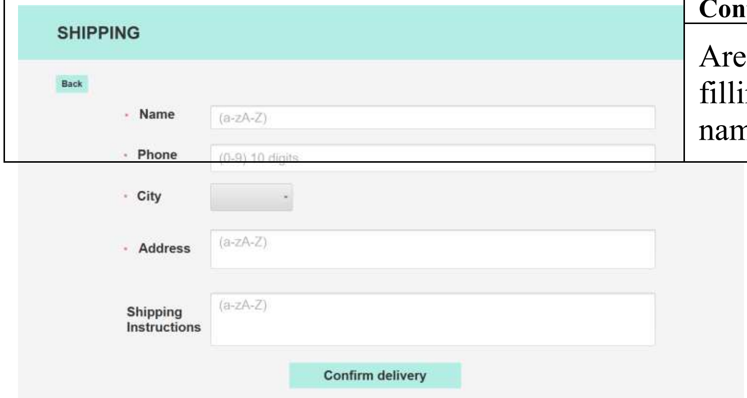
4.4. Cart

Aims Software		**Date of creation**	**Approved by**	**Reviewed by**	**Persion in charge**
Screen specification	**View cart screen**	17/06/2024			Trinh Tien Dung
 CART Back book002 Remove 50.000 đ VND 1 Subtotal: 50.000 đ VAT (10%): 5.000 đ Amount: 55.000 đ Place order		**Control**	**Operation**	**Function**	
		Area for displaying the	Initial	Display the subtotal	

	subtotal		
	Area for display items in the cart	Initial	***Display the product with corresponding information***
	Place order button	Click	Display the Delivery form
	Edit number button	Click/Input	Change number of selected products
	Remove button	Click	Remove product from cart
	Back button	Click	Back to home screen

Screen	**Cart screen**			
Field name	**Type**	**Limitation**	**Attribute**	**Remarks**
Media title	Text	40 characters	Bold, black	Left-justified
Quantity	Digits	5 digits	None	Left-justified
Price	Digits	10 digits	Green	Left-justified Dot for thousand separation
Subtotal				
Currency	Text	3 characters	All caps	None
Item thumbnail	Image	95x103 pixels	None	None

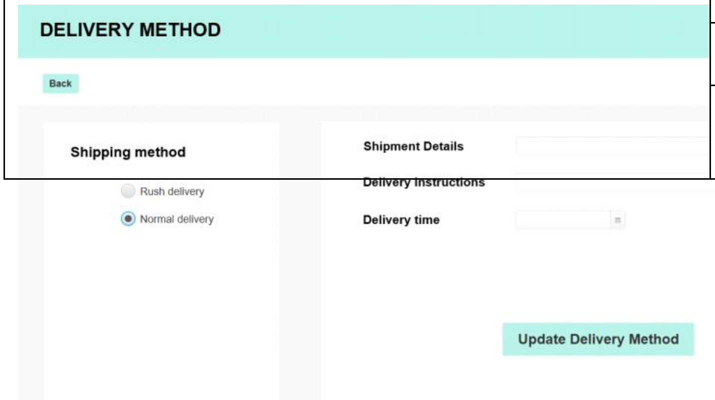
4.5. Shipping information Screen

Aims Software		**Date of creation**	**Approved by**	**Reviewed by**	**Persion in charge**
Screen specification	**Delivery Form**	17/06/2024			Trinh Tien Dung
		**Control**	**Operation**	**Function**	
		Area for filling the name	Input	***Fill name of the customer***	

	Area for filling phone	Input	***Fill phone of the customer***
	Area for filling address	Input	***Fill specific address of the customer***
	City selecting dropdown	Select	Change city of customer
	Area for filling shipping instruction	Input	***Fill shipping instruction of the customer***
	Submit button	Click	Save information and process to check rush order
	Back button	Click	Back to previous page

Screen	**Deliver screen**			
Field name	**Type**	**Limitation**	**Attribute**	**Remarks**
Province/City	Select from list	None	None	None
Delivery option	Select from list	None	None	None

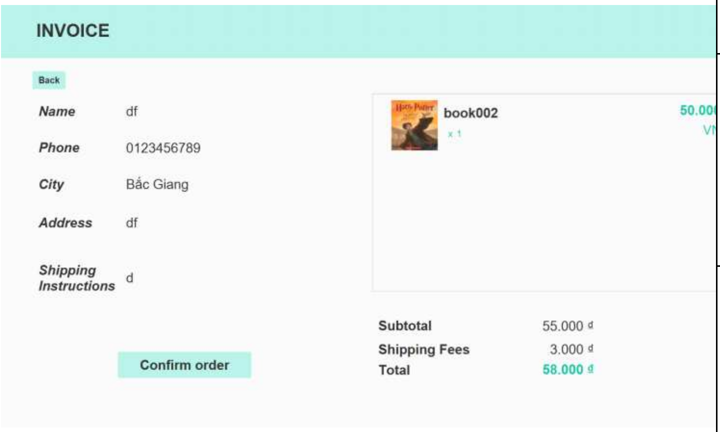
4.6. Delivery Method

Aims Software		**Date of creation**	**Approved by**	**Reviewed by**	**Persion in charge**
Screen specification	**Delivery Method**	17/06/2024			Trinh Tien Dung
		**Control**	**Operation**	**Function**	
		Shipping method	Select	Select delivery method	
		Area for inputting Shipment	Input	Input shipment detail	

	Detail		
	Area for inputting delivery instruction	Input	***Update the delivery instruction of order***
	Area for selected expected delivery time	Select	***Update the delivery expected time of order***
	Confirm button	Click	Save information and process to invoice screen
	Back button	Click	Back to previous screen

4.7. Invoice Screen

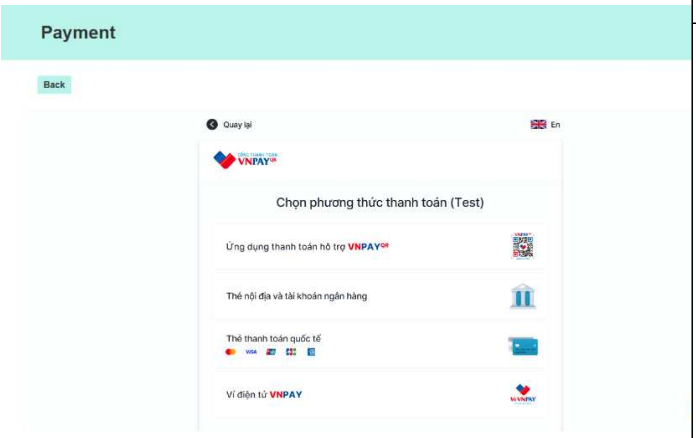
Aims Software		**Date of creation**	**Approved by**	**Reviewed by**	**Person in charge**
Screen specification	**Invoice Screen**	07/04/2024			Trinh Tien Dung

	Control	**Operation**	**Function**
 INVOICE Back Name df Phone 0123456789 City Bắc Giang Address df Shipping Instructions d book002 x 1 50.00 VN Subtotal 55.000 đ Shipping Fees 3.000 đ Total 58.000 đ Confirm order	Area for displaying the selected address	Initial	Display the selected shipping address
	Area for displaying the customer's phone number	Initial	Display the customer's phone number
	Area for displaying the	Initial	Display the customer's name

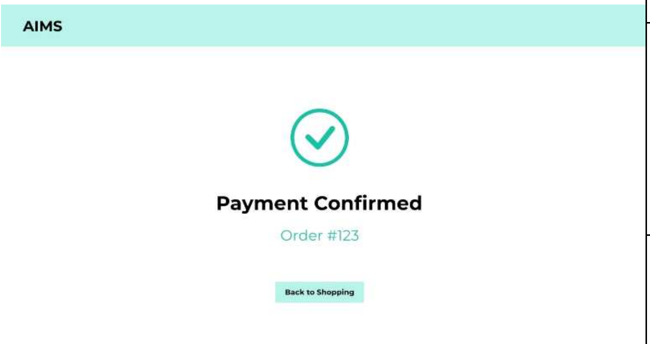
	customer's name		
	Area for displaying the customer's instruction	Initial	Display the customer's instruction
	Area for displaying the shipping fee	Initial	Display the customer's shipping fee
	Area for displaying the product list with specified info for each product	Initial	Display the customer's product list
	Confirm button	Click	Save information and process to invoice screen

Screen	**Normal order invoice & Rush order invoice screen**			
Field name	**Type**	**Limitation**	**Attribute**	**Remarks**
Media title	Text	40 characters	Bold, black	Left-justified
Subtotal	Digits	10 digits	None	Left-justified Dot for thousand separation
Shipping fee				
Price			Green	
Items				
Total				
Item thumbnail	Image	163x128 pixels	None	None
Customer name	Text	50 characters	Italic	None
Province/City		30 characters		
Address		100 characters		
Shipping instruction		100 characters		
Customer phone	Digits	10 digits	Italic	None

4.8. PaymentScreen

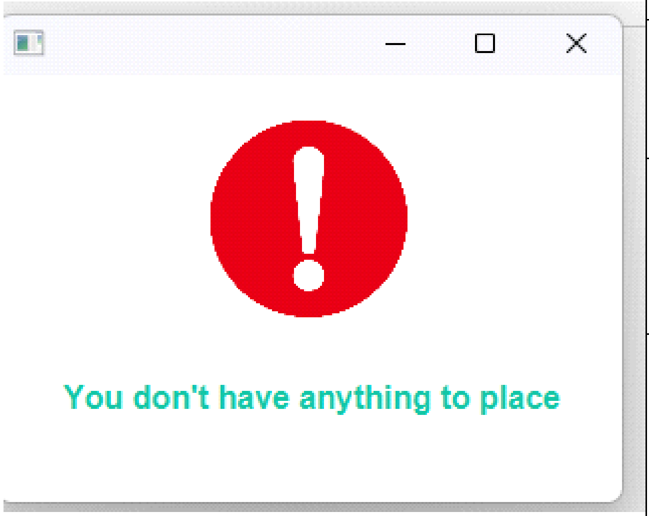
Aims Software		Date of creation	Approved by	Reviewed by	Persion in charge
Screen specification	**Payment Screen**	17/06/2024			Trinh Tien Dung
 Payment Back Quay lại En CỔNG THANH TOÁN VNPAY QR Chọn phương thức thanh toán (Test) Ứng dụng thanh toán hỗ trợ VNPAY QR Thẻ nội địa và tài khoản ngân hàng Thẻ thanh toán quốc tế Ví điện tử VNPAY		**Control**	**Operation**	**Function**	
		Area for redirect to VNPay	***Initial***	***Pay order through VNPay***	

4.10. PaymentResult

Aims Software		Date of creation	Approved by	Reviewed by	Persion in charge
Screen specification	**Result Screen**	17/06/2024			Trinh Tien Dung
 AIMS Payment Confirmed Order #123 Back to Shopping		**Control**	**Operation**	**Function**	
		Area for displaying successful order	***Initial***	***Display successful order***	
		Area for displaying message	***Initial***	***Displays message***	

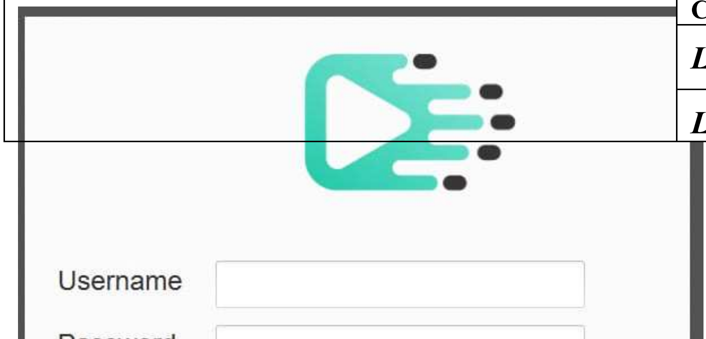
	Button	***Click***	***Back to home screen***

4.10. Popup Screen

Aims Software		**Date of creation**	**Approved by**	**Reviewed by**	**Person in charge**
Screen specification	**Popup Screen**	17/06/2024			Trinh Tien Dung
 You don't have anything to place		**Control**	**Operation**	**Function**	
		Area for displaying icon	***Initial***	***Display icon***	
		Area for displaying message	***Initial***	***Displays message***	
		Button	***Click***	***Close popup window***	

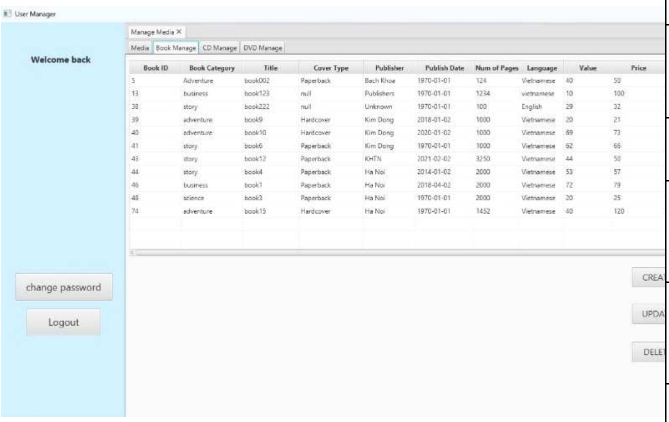
4.11. Login Screen

Aims Software		**Date of creation**	**Approved by**	**Reviewed by**	**Person in charge**
Screen specification	**Login Screen**	17/06/2024			Trinh Tien Dung
		Control	**Operation**	**Function**	
		Logo	Click	***Go to home screen***	
		Login	***Click***	***Login with username***	


Username

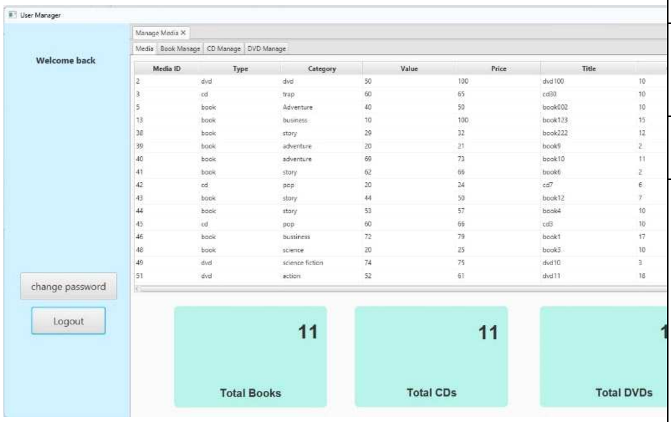
	button		***and password***
	Area for inputting username	***Input***	***Fill username***
	Area for inputting password	***Input***	***Fill password***

4.12.Product Manage

Aims Software		**Date of creation**	**Approved by**	**Reviewed by**	**Persion in charge**
Screen specification	**Product Manager Screen**	17/06/2024			Trinh Tien Dung
		**Control**	**Operation**	**Function**	
		Change password button	Click	***Change password***	
		Logout button	***Click***	***Logout***	
		Create button	***Click***	***Create new media***	
		Update button	***Click***	***Update selected media***	
		Delete button	***Click***	***Delete selected media***	
		Area for displaying media information	***Initial***	***Display all media by category***	

Book ID	Book Category	Title	Cover Type	Publisher	Publish Date	Num of Pages	Language	Value	Price
5	Adventure	book002	Paperback	Bach Khoa	1970-01-01	124	Vietnamese	40	50
13	business	book123	null	Publishers	1970-01-01	1234	vietnamese	10	100
38	story	book222	null	Unknown	1970-01-01	100	English	29	32
39	adventure	book9	Hardcover	Kim Dong	2018-01-02	1000	Vietnamese	20	21
40	adventure	book10	Hardcover	Kim Dong	2020-01-02	1000	Vietnamese	69	73
41	story	book6	Paperback	Kim Dong	1970-01-01	1000	Vietnamese	62	66
43	story	book12	Paperback	KHTN	2021-02-02	3250	Vietnamese	44	50
44	story	book4	Paperback	Ha Noi	2014-01-02	2000	Vietnamese	53	57
46	business	book1	Paperback	Ha Noi	2018-04-02	2000	Vietnamese	72	79
48	science	book3	Paperback	Ha Noi	1970-01-01	2000	Vietnamese	20	25
74	adventure	book15	Hardcover	Ha Noi	1970-01-01	1452	Vietnamese	40	120

4.13.Media Manage

Aims Software		**Date of creation**	**Approved by**	**Reviewed by**	**Persion in charge**
Screen specification	**View Product Screen**	17/06/2024			Trinh Tien Dung
		**Control**	**Operation**	**Function**	
		Change password button	Click	***Change password***	
		Logout button	***Click***	***Logout***	
		Area for displaying media information	***Initial***	***Display all media by category***	

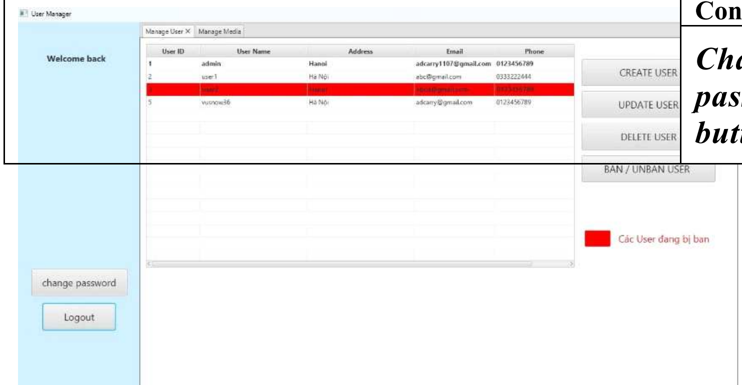
Media ID	Type	Category	Value	Price	Title	
2	dvd	dvd	50	100	dvd100	10
3	cd	trap	60	65	cd30	10
5	book	Adventure	40	50	book002	10
13	book	business	10	100	book123	15
38	book	story	29	32	book222	12
39	book	adventure	20	21	book9	2
40	book	adventure	69	73	book10	11
41	book	story	62	66	book6	2
42	cd	pop	20	24	cd7	6
43	book	story	44	50	book12	7
44	book	story	53	57	book4	10
45	cd	pop	60	66	cd3	10
46	book	bussiness	72	79	book1	17
48	book	science	20	25	book3	10
49	dvd	science fiction	74	75	dvd10	3
51	dvd	action	52	61	dvd11	18

11 Total Books

11 Total CDs

1 Total DVDs

4.14. User Manage

Aims Software		**Date of creation**	**Approved by**	**Reviewed by**	**Persion in charge**
Screen specification	**User Manager Screen**	17/06/2024			Trinh Tien Dung
		**Control**	**Operation**	**Function**	
		Change password button	Click	***Change password***	

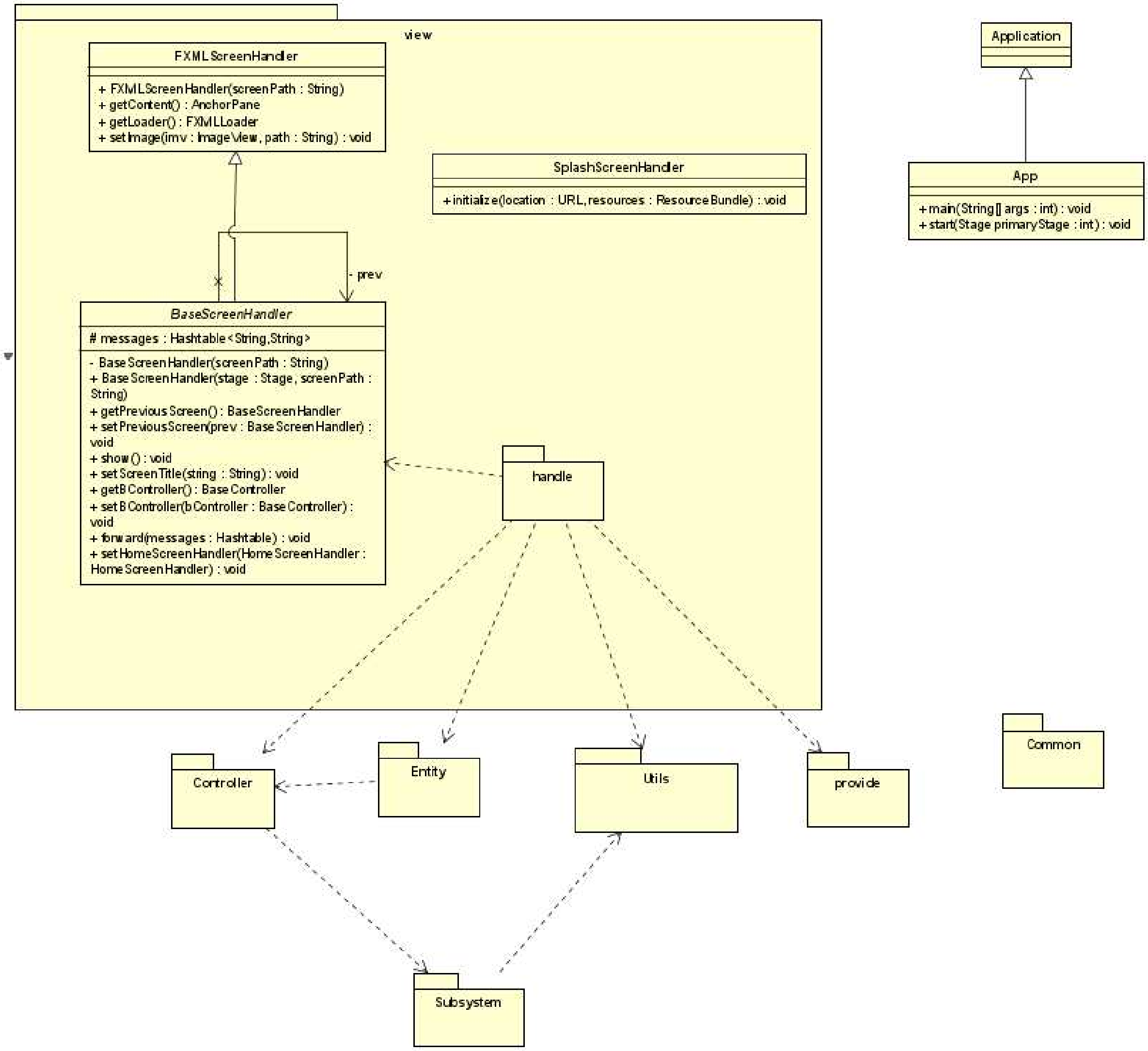
User ID	User Name	Address	Email	Phone
1	admin	Hanoi	adcarry1107@gmail.com	0123456789
2	user1	Hà Nội	abc@gmail.com	0333222444
3	user2	Hanoi	[illegible]	[illegible]
5	vusnow36	Hà Nội	adcarry@gmail.com	0123456789

🟥 Các User đang bị ban

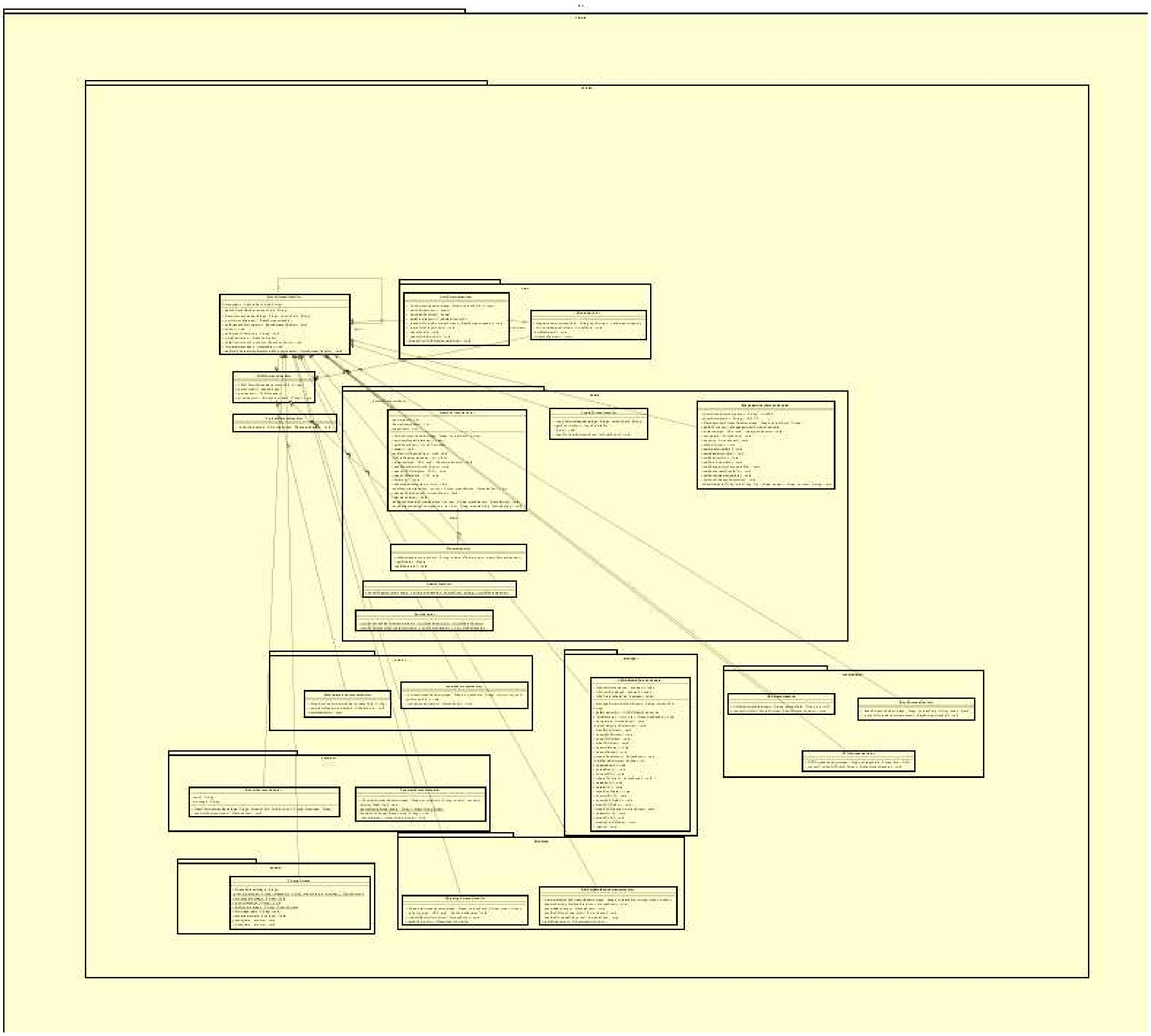
	Logout button	***Click***	***Logout***
	Create user button	Click	***Create new user***
	Update user button	Click	***Update selected user***
	Delete user button	Click	***Delete selected user***
	Ban/Unban an user button	Click	***Ban/Unbanselected user***
	Area for displaying users	***Initial***	***Display all users information***

5. ANALYSIS CLASS DIAGRAM

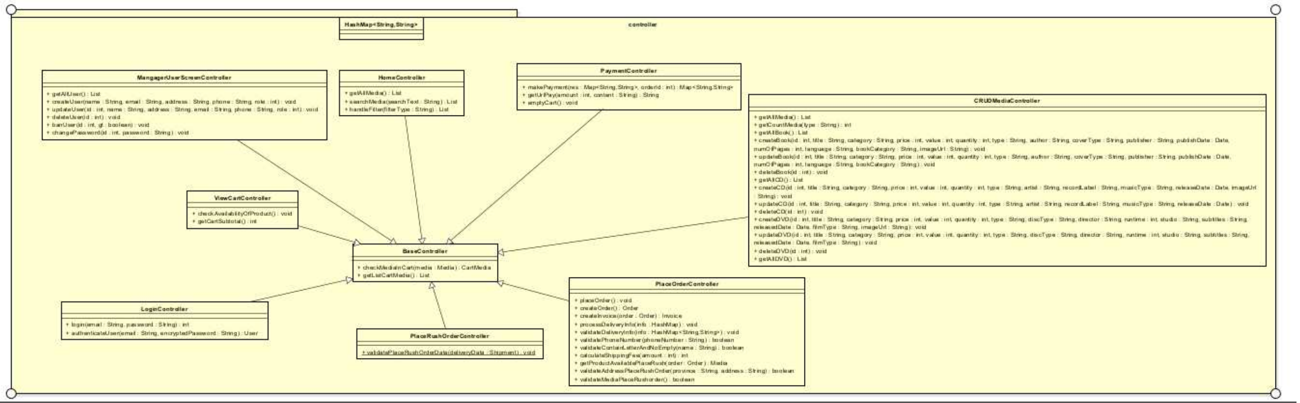
5.1. General Class Diagram


view
FXMLScreenHandler
+ FXMLScreenHandler(screenPath : String)
+ getContent() : AnchorPane
+ getLoader() : FXMLLoader
+ setImage(imv : ImageView, path : String) : void
SplashScreenHandler
+initialize(location : URL, resources : ResourceBundle) : void
- prev
BaseScreenHandler
# messages : Hashtable<String,String>
- BaseScreenHandler(screenPath : String)
+ BaseScreenHandler(stage : Stage, screenPath : String)
+ getPreviousScreen() : BaseScreenHandler
+ setPreviousScreen(prev : BaseScreenHandler) : void
+ show() : void
+ setScreenTitle(string : String) : void
+ getBController() : BaseController
+ setBController(bController : BaseController) : void
+ forward(messages : Hashtable) : void
+ setHomeScreenHandler(HomeScreenHandler : HomeScreenHandler) : void
handle
Application
App
+main(String[] args : int) : void
+start(Stage primaryStage : int) : void
Controller
Entity
Utils
provide
Common
Subsystem

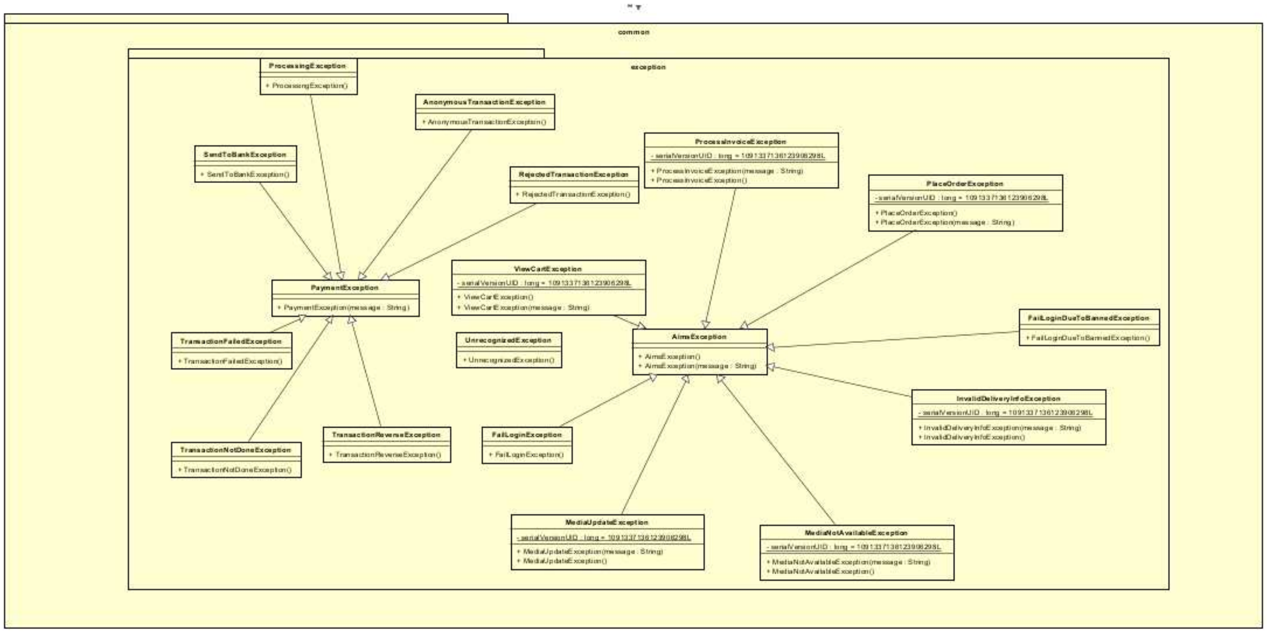
5.2.1 Class Diagram for Package View



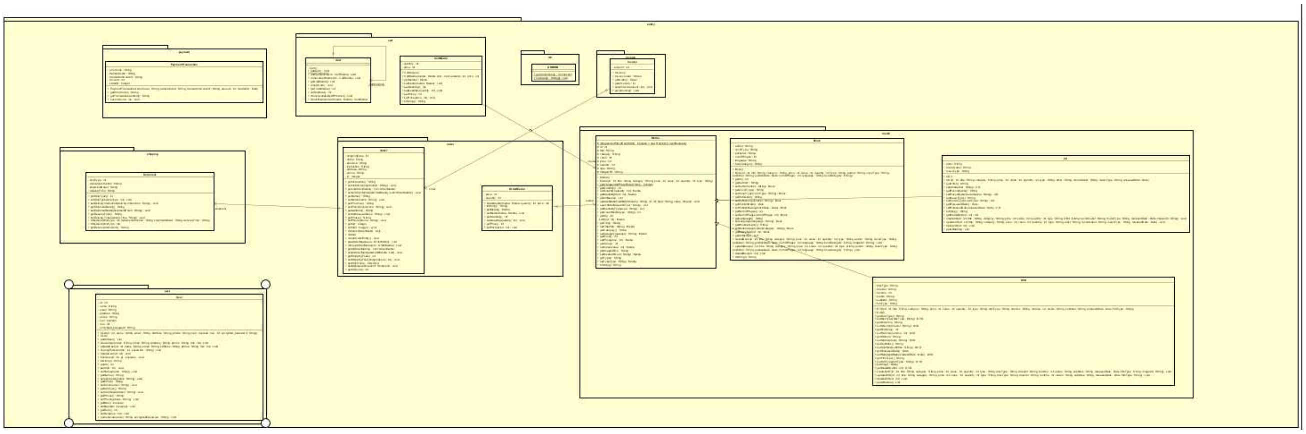
5.2.2 Class Diagram for Package Controller


controller
HashMap<String,String>
MangagerUserScreenController
+ getAllUser() : List
+ createUser(name : String, email : String, address : String, phone : String, role : int) : void
+ updateUser(id : int, name : String, address : String, email : String, phone : String, role : int) : void
+ deleteUser(id : int) : void
+ banUser(id : int, gt : boolean) : void
+ changePassword(id : int, password : String) : void
HomeController
+ getAllMedia() : List
+ searchMedia(searchText : String) : List
+ handleFilter(filterType : String) : List
PaymentController
+ makePayment(res : Map<String,String>, orderId : int) : Map<String,String>
+ getUrlPay(amount : int, content : String) : String
+ emptyCart() : void
CRUDMediaController
+ getAllMedia() : List
+ getCountMedia(type : String) : int
+ getAllBook() : List
+ createBook(id : int, title : String, category : String, price : int, value : int, quantity : int, type : String, author : String, coverType : String, publisher : String, publishDate : Date, numOfPages : int, language : String, bookCategory : String, imageUrl : String) : void
+ updateBook(id : int, title : String, category : String, price : int, value : int, quantity : int, type : String, author : String, coverType : String, publisher : String, publishDate : Date, numOfPages : int, language : String, bookCategory : String) : void
+ deleteBook(id : int) : void
+ getAllCD() : List
+ createCD(id : int, title : String, category : String, price : int, value : int, quantity : int, type : String, artist : String, recordLabel : String, musicType : String, releaseDate : Date, imageUrl : String) : void
+ updateCD(id : int, title : String, category : String, price : int, value : int, quantity : int, type : String, artist : String, recordLabel : String, musicType : String, releaseDate : Date) : void
+ deleteCD(id : int) : void
+ createDVD(id : int, title : String, category : String, price : int, value : int, quantity : int, type : String, discType : String, director : String, runtime : int, studio : String, subtitles : String, releasedDate : Date, filmType : String, imageUrl : String) : void
+ updateDVD(id : int, title : String, category : String, price : int, value : int, quantity : int, type : String, discType : String, director : String, runtime : int, studio : String, subtitles : String, releasedDate : Date, filmType : String) : void
+ deleteDVD(id : int) : void
+ getAllDVD() : List
ViewCartController
+ checkAvailabilityOfProduct() : void
+ getCartSubtotal() : int
BaseController
+ checkMediaInCart(media : Media) : CartMedia
+ getListCartMedia() : List
LoginController
+ login(email : String, password : String) : int
+ authenticateUser(email : String, encryptedPassword : String) : User
PlaceRushOrderController
+ validatePlaceRushOrderData(deliveryData : Shipment) : void
PlaceOrderController
+ placeOrder() : void
+ createOrder() : Order
+ createInvoice(order : Order) : Invoice
+ processDeliveryInfo(info : HashMap) : void
+ validateDeliveryInfo(info : HashMap<String,String>) : void
+ validatePhoneNumber(phoneNumber : String) : boolean
+ validateContainLetterAndNoEmpty(name : String) : boolean
+ calculateShippingFee(amount : int) : int
+ getProductAvailablePlaceRush(order : Order) : Media
+ validateAddressPlaceRushOrder(province : String, address : String) : boolean
+ validateMediaPlaceRushorder() : boolean

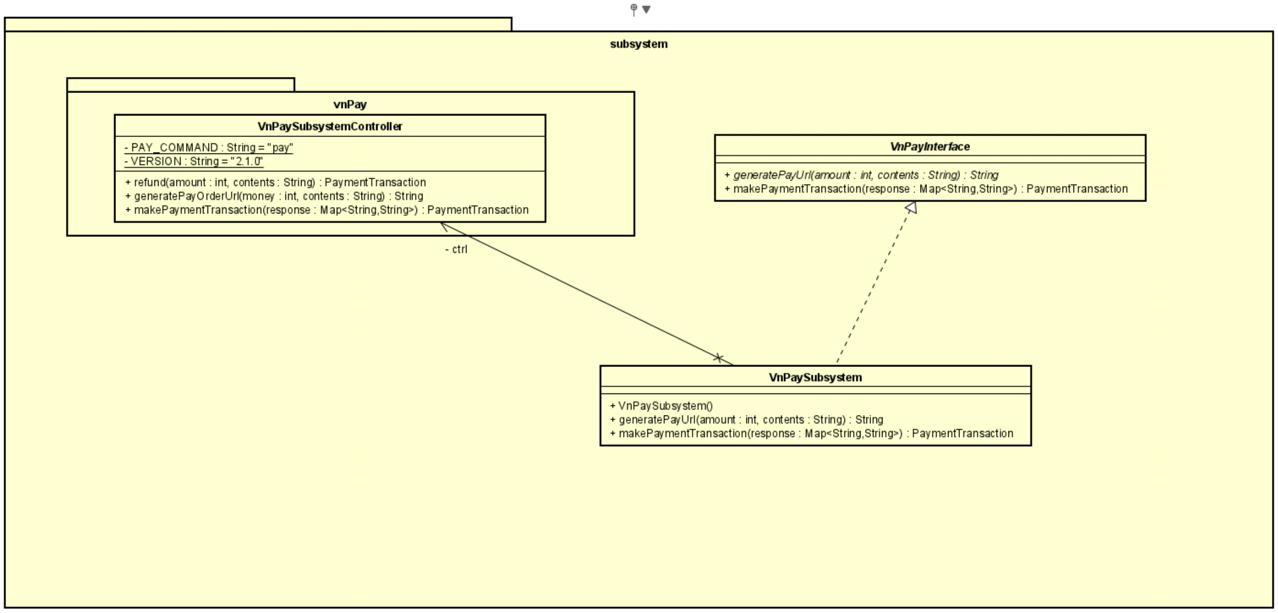
5.2.3 Class Diagram for Package Common


common
exception
ProcessingException
+ ProcessingException()
AnonymousTransactionException
+ AnonymousTransactionException()
SendToBankException
+ SendToBankException()
RejectedTransactionException
+ RejectedTransactionException()
ProcessInvoiceException
- serialVersionUID : long = 1091337136123906298L
+ ProcessInvoiceException(message : String)
+ ProcessInvoiceException()
PlaceOrderException
- serialVersionUID : long = 1091337136123906298L
+ PlaceOrderException()
+ PlaceOrderException(message : String)
PaymentException
+ PaymentException(message : String)
ViewCartException
- serialVersionUID : long = 1091337136123906298L
+ ViewCartException()
+ ViewCartException(message : String)
TransactionFailedException
+ TransactionFailedException()
UnrecognizedException
+ UnrecognizedException()
AimsException
+ AimsException()
+ AimsException(message : String)
FailLoginDueToBannedException
+ FailLoginDueToBannedException()
InvalidDeliveryInfoException
- serialVersionUID : long = 1091337136123906298L
+ InvalidDeliveryInfoException(message : String)
+ InvalidDeliveryInfoException()
TransactionNotDoneException
+ TransactionNotDoneException()
TransactionReverseException
+ TransactionReverseException()
FailLoginException
+ FailLoginException()
MediaUpdateException
- serialVersionUID : long = 1091337136123906298L
+ MediaUpdateException(message : String)
+ MediaUpdateException()
MediaNotAvailableException
- serialVersionUID : long = 1091337136123906298L
+ MediaNotAvailableException(message : String)
+ MediaNotAvailableException()

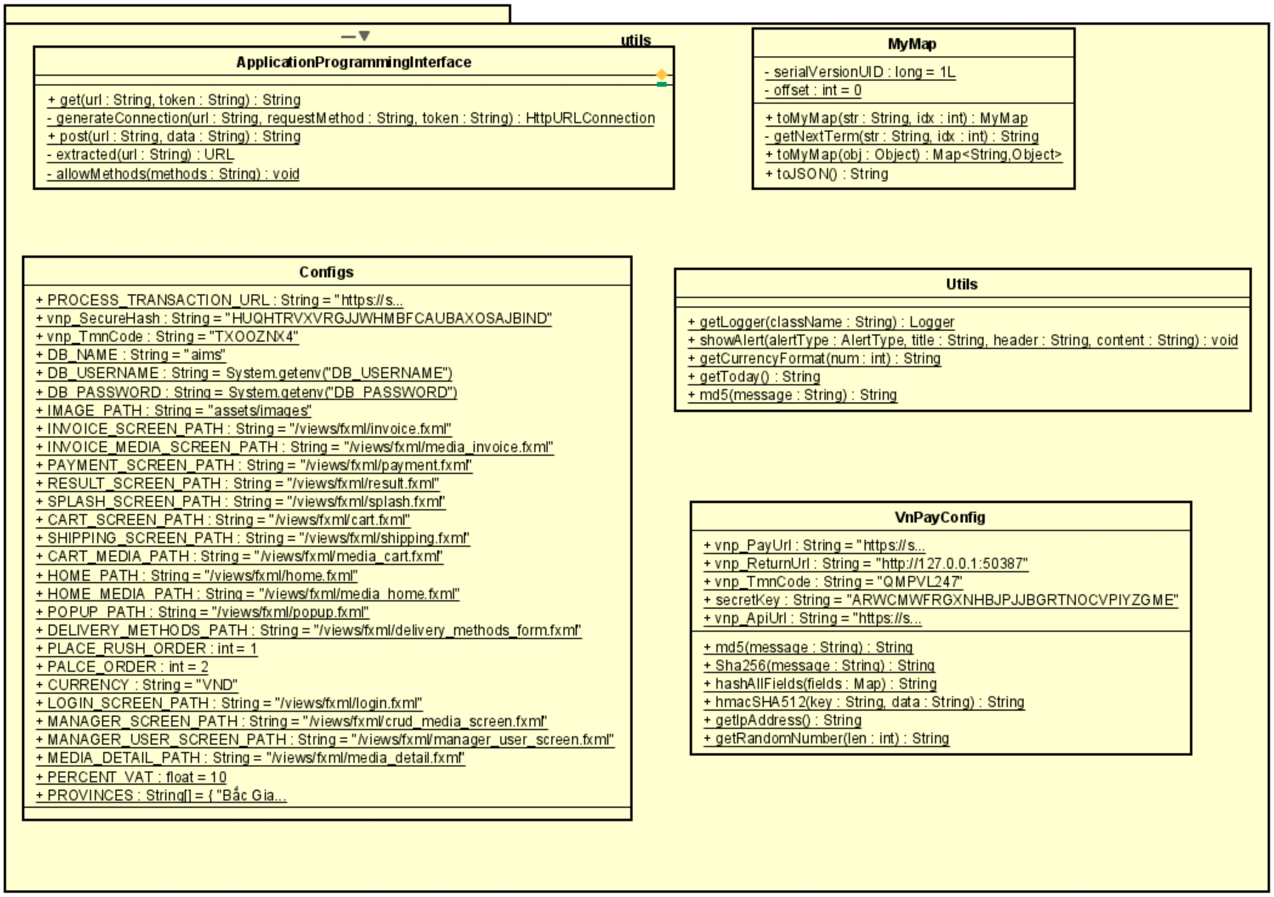
5.2.4 Class Diagram for Package Entity



5.2.5 Class Diagram for Package Subsystem

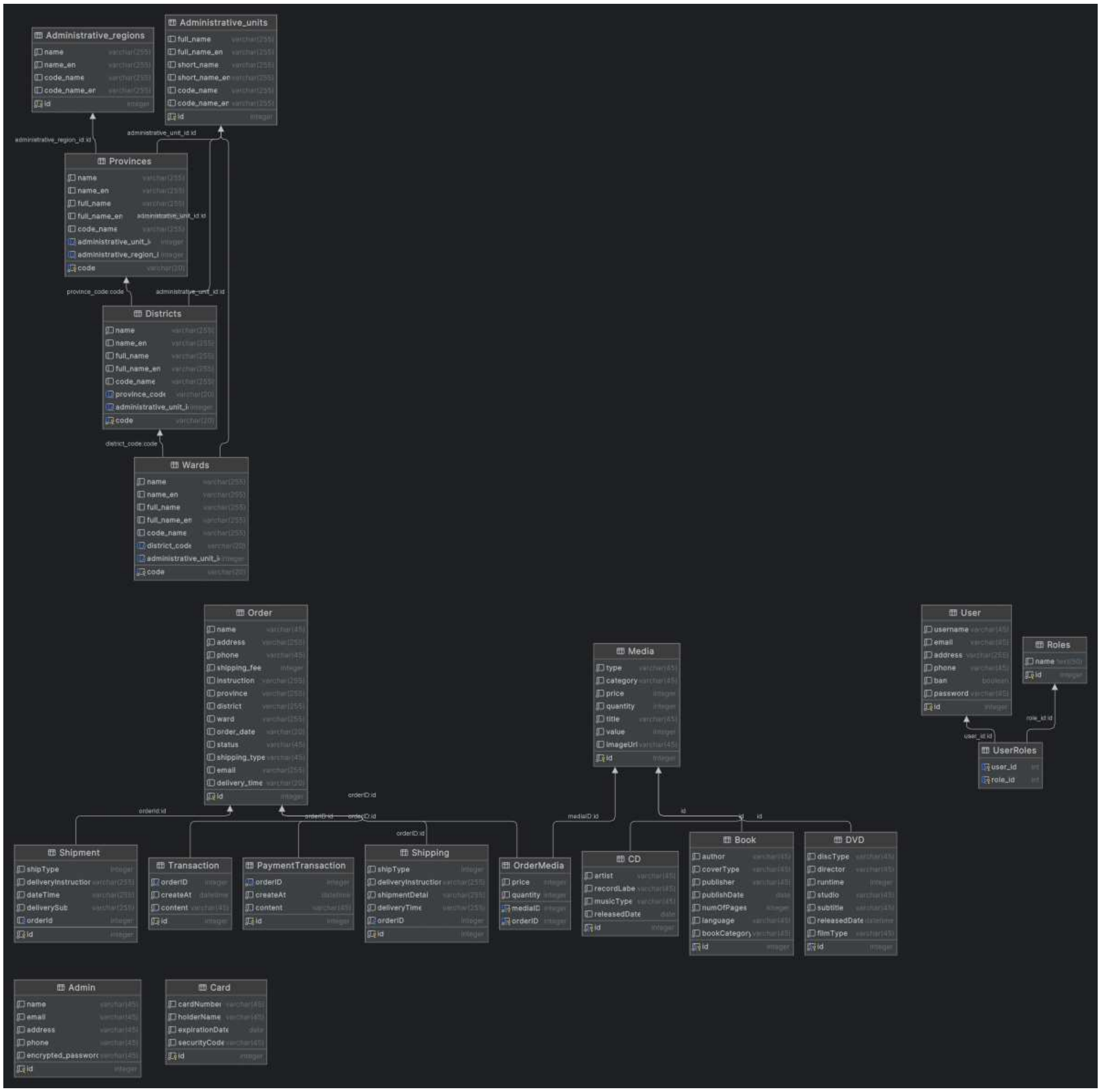

subsystem
vnPay
VnPaySubsystemController
- PAY_COMMAND : String = "pay"
- VERSION : String = "2.1.0"
+ refund(amount : int, contents : String) : PaymentTransaction
+ generatePayOrderUrl(money : int, contents : String) : String
+ makePaymentTransaction(response : Map<String,String>) : PaymentTransaction
VnPayInterface
+ generatePayUrl(amount : int, contents : String) : String
+ makePaymentTransaction(response : Map<String,String>) : PaymentTransaction
- ctrl
VnPaySubsystem
+ VnPaySubsystem()
+ generatePayUrl(amount : int, contents : String) : String
+ makePaymentTransaction(response : Map<String,String>) : PaymentTransaction

5.2.6 Class Diagram for Package Utils

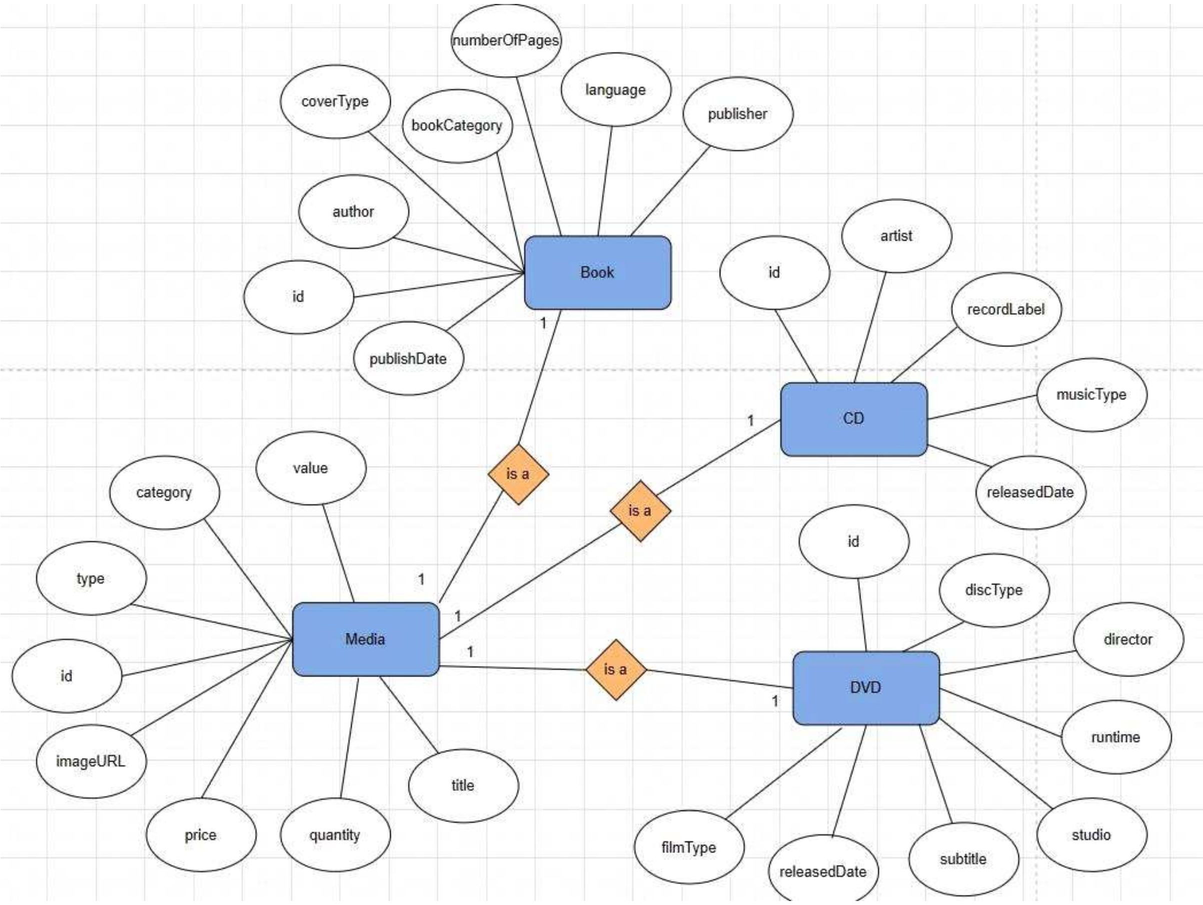

utils
ApplicationProgrammingInterface
+ get(url : String, token : String) : String
- generateConnection(url : String, requestMethod : String, token : String) : HttpURLConnection
+ post(url : String, data : String) : String
- extracted(url : String) : URL
- allowMethods(methods : String) : void
MyMap
- serialVersionUID : long = 1L
- offset : int = 0
+ toMyMap(str : String, idx : int) : MyMap
- getNextTerm(str : String, idx : int) : String
+ toMyMap(obj : Object) : Map<String,Object>
+ toJSON() : String
Configs
+ PROCESS_TRANSACTION_URL : String = "https://s...
+ vnp_SecureHash : String = "HUQHTRVXVRGJJWHMBFCAUBAXOSAJBIND"
+ vnp_TmnCode : String = "TXOOZNX4"
+ DB_NAME : String = "aims"
+ DB_USERNAME : String = System.getenv("DB_USERNAME")
+ DB_PASSWORD : String = System.getenv("DB_PASSWORD")
+ IMAGE_PATH : String = "assets/images"
+ INVOICE_SCREEN_PATH : String = "/views/fxml/invoice.fxml"
+ INVOICE_MEDIA_SCREEN_PATH : String = "/views/fxml/media_invoice.fxml"
+ PAYMENT_SCREEN_PATH : String = "/views/fxml/payment.fxml"
+ RESULT_SCREEN_PATH : String = "/views/fxml/result.fxml"
+ SPLASH_SCREEN_PATH : String = "/views/fxml/splash.fxml"
+ CART_SCREEN_PATH : String = "/views/fxml/cart.fxml"
+ SHIPPING_SCREEN_PATH : String = "/views/fxml/shipping.fxml"
+ CART_MEDIA_PATH : String = "/views/fxml/media_cart.fxml"
+ HOME_PATH : String = "/views/fxml/home.fxml"
+ HOME_MEDIA_PATH : String = "/views/fxml/media_home.fxml"
+ POPUP_PATH : String = "/views/fxml/popup.fxml"
+ DELIVERY_METHODS_PATH : String = "/views/fxml/delivery_methods_form.fxml"
+ PLACE_RUSH_ORDER : int = 1
+ PALCE_ORDER : int = 2
+ CURRENCY : String = "VND"
+ LOGIN_SCREEN_PATH : String = "/views/fxml/login.fxml"
+ MANAGER_SCREEN_PATH : String = "/views/fxml/crud_media_screen.fxml"
+ MANAGER_USER_SCREEN_PATH : String = "/views/fxml/manager_user_screen.fxml"
+ MEDIA_DETAIL_PATH : String = "/views/fxml/media_detail.fxml"
+ PERCENT_VAT : float = 10
+ PROVINCES : String[] = { "Bắc Gia...
Utils
+ getLogger(className : String) : Logger
+ showAlert(alertType : AlertType, title : String, header : String, content : String) : void
+ getCurrencyFormat(num : int) : String
+ getToday() : String
+ md5(message : String) : String
VnPayConfig
+ vnp_PayUrl : String = "https://s...
+ vnp_ReturnUrl : String = "http://127.0.0.1:50387"
+ vnp_TmnCode : String = "QMPVL247"
+ secretKey : String = "ARWCMWFRGXNHBJPJJBGRTNOCVPIYZGME"
+ vnp_ApiUrl : String = "https://s...
+ md5(message : String) : String
+ Sha256(message : String) : String
+ hashAllFields(fields : Map) : String
+ hmacSHA512(key : String, data : String) : String
+ getIpAddress() : String
+ getRandomNumber(len : int) : String

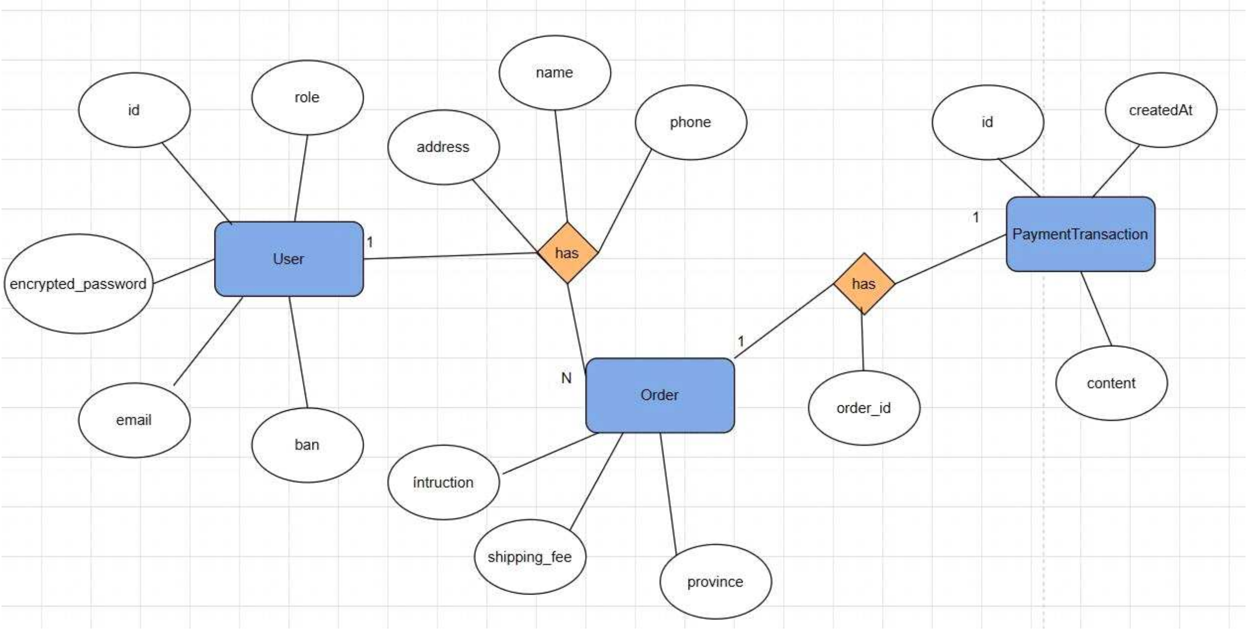
6. Data Modeling

6.1 Database Schema Diagram:

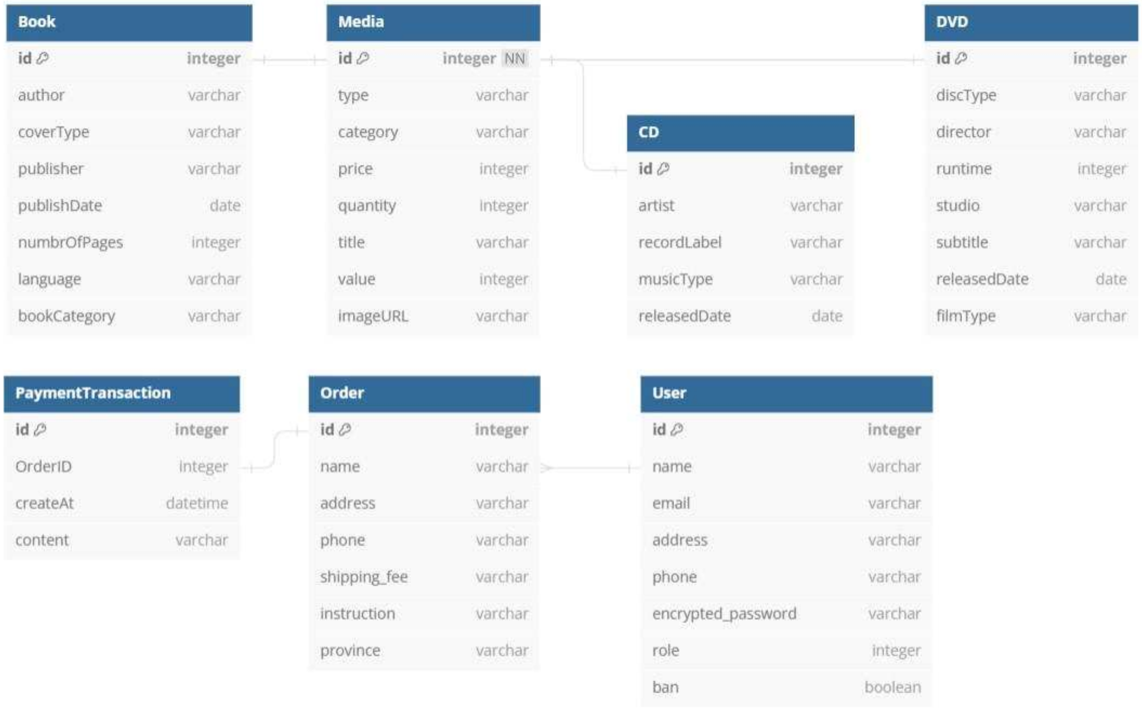


6.1. Conceptual Data Model


numberOfPages
coverType
bookCategory
language
publisher
author
Book
id
publishDate
1
id
artist
recordLabel
CD
1
musicType
releasedDate
is a
is a
category
value
type
1
1
Media
1
is a
id
discType
director
id
DVD
1
runtime
imageURL
title
price
quantity
filmType
releasedDate
subtitle
studio


id
role
encrypted_password
User
email
ban
1
name
address
phone
has
N
Order
intruction
shipping_fee
province
1
has
order_id
1
id
createdAt
PaymentTransaction
content

6.2. Logical Data Model



6.3. Physical Data Model

Media

STT	PK	FK	Trường	Kiểu dữ liệu	Bắt buộc	Mô tả
1	*		id	Integer	Có	ID, auto increment
2			type	Varchar(45)	Có	Loại sản phẩm
3			category	Varchar(45)	Có	Loại nội dung sản phẩm
4			price	Integer	Có	Giá sản phẩm
5			quantity	Integer	Có	Số lượng sản phẩm
6			title	Varchar(45)	Có	Tên sản phẩm
7			value	Integer	Có	
8			imageURL	Varchar(45)	Có	Đường dẫn hình ảnh sản phẩm

Book

STT	PK	FK	Trường	Kiểu dữ liệu	Bắt buộc	Mô tả
1	*		id	Integer	Có	ID, auto increment
2			author	Varchar(45)	Có	Tên tác giả
3			coverType	Varchar(45)	Có	Kiểu bìa sách
4			publisher	Varchar(45)	Có	Tên nhà xuất bản
5			publishDate	Date	Có	Ngày xuất bản

6			numberOfPages	Integer	Có	Số trang
7			languge	Varchar(45)	Có	Ngôn ngữ
8			bookCategory	Varchar(45)	Có	Thể loại nội dung

CD

STT	PK	FK	Trường	Kiểu dữ liệu	Bắt buộc	Mô tả
1	*		id	Integer	Có	ID, auto increment
2			artist	Varchar(45)	Có	Tên nghệ sĩ
3			recordLabel	Varchar(45)	Có	Tên hãng sản xuất
4			musicType	Varchar(45)	Có	Thể loại nhạc
5			releasedDate	Date	Có	Ngày phát hành

DVD

STT	PK	FK	Trường	Kiểu dữ liệu	Bắt buộc	Mô tả
1	*		id	Integer	Có	ID, auto increment
2			discType	Varchar(45)	Có	Kiểu đĩa
3			director	Varchar(45)	Có	Đạo diễn
4			runtime	Integer	Có	Thời lượng
5			studio	Varchar(45)	Có	Hãng sản xuất
6			subtitle	Varchar(45)	Có	Phụ đề
7			releasedDate	Date	Có	Ngày phát hành
8			filmType	Varchar(45)	Có	Thể loại nội dung

PaymentTransaction

STT	PK	FK	Trường	Kiểu dữ liệu	Bắt buộc	Mô tả
1	*		id	Integer	Có	ID, auto increment
2		*	orderID	Integer	Có	ID đơn hàng
3			createAt	DateTime	Có	Thời gian giao dịch
4			content	Varchar(45)	Có	Nội dung

Order

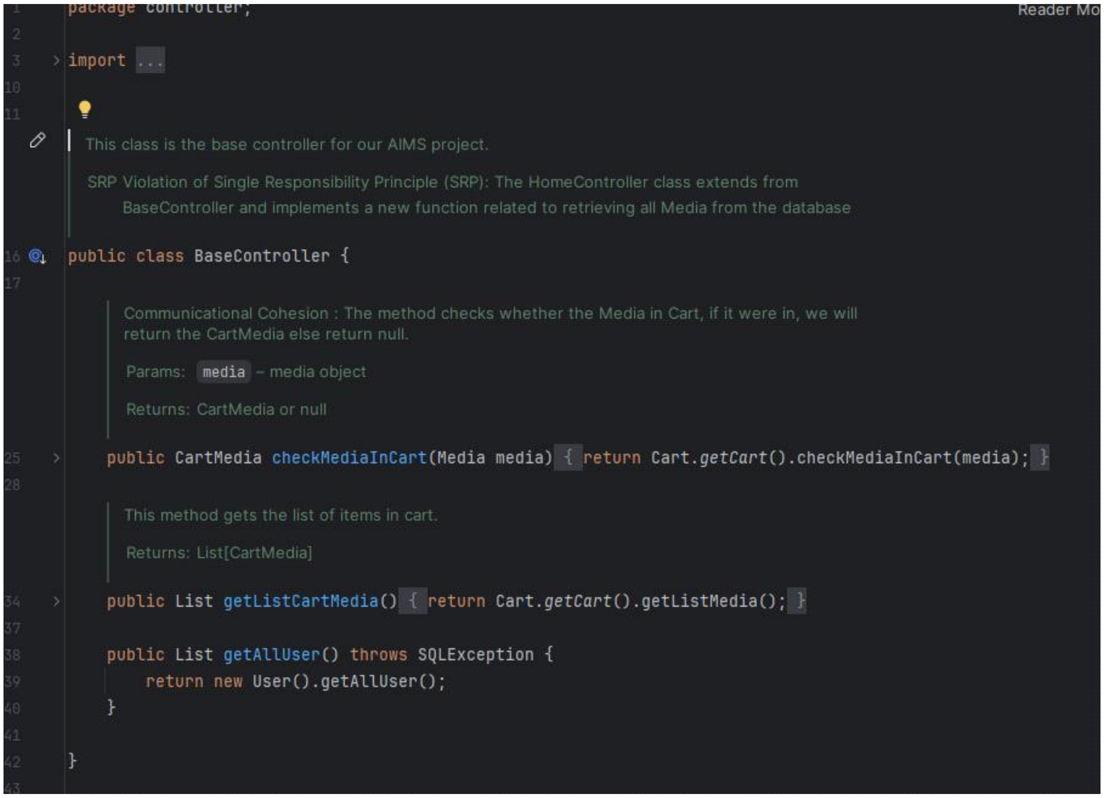
STT	PK	FK	Trường	Kiểu dữ liệu	Bắt buộc	Mô tả
1	*		id	Integer	Có	ID, auto increment
2			name	Varchar(45)	Có	Tên người đặt
3			address	Varchar(45)	Có	Địa chỉ nhận hàng
4			phone	Varchar(45)	Có	Số điện thoại người đặt
5			shipping_fee	Integer	Có	Phí vận chuyển
6			instruction	Varchar(255)	Có	Yêu cầu
7			province	Varchar(255)	Có	Tỉnh thành

User

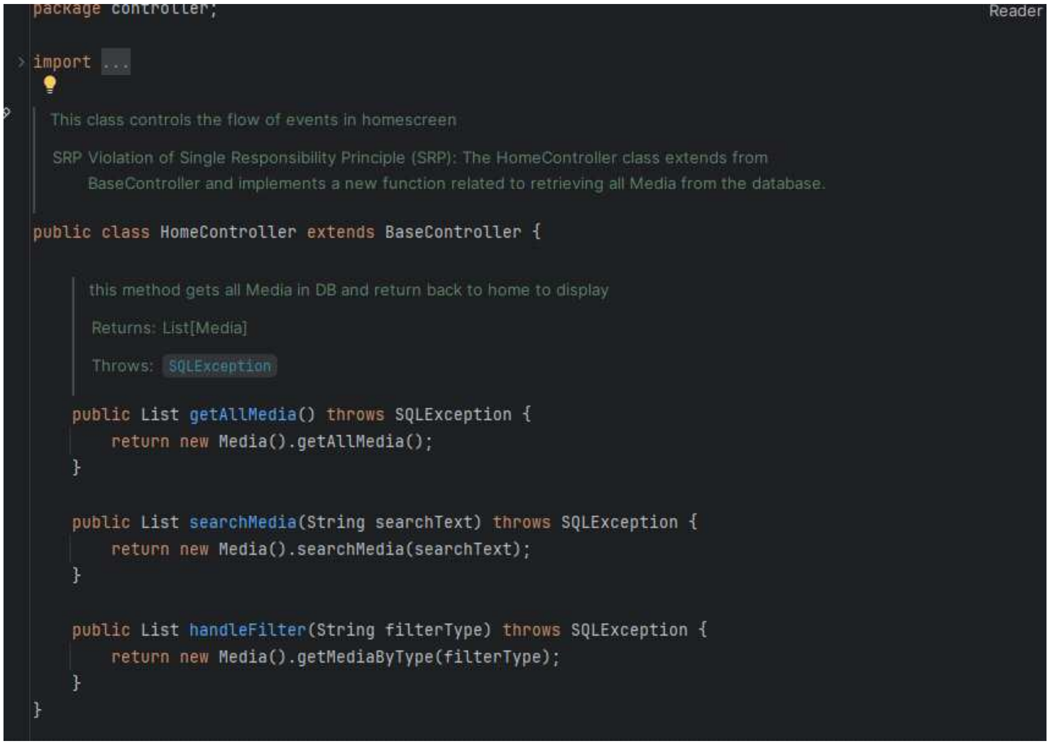
STT	PK	FK	Trường	Kiểu dữ liệu	Bắt buộc	Mô tả
1	*		id	Integer	Có	ID, auto increment
2			name	Varchar(45)	Có	Tên người dùng
3			email	Varchar(45)		Email người dùng
4			address	Varchar(45)	Có	Địa chỉ người dùng
5			phone	Varchar(45)	Có	Số điện thoại người dùng
6			encrypted_password	Varchar(45)	Có	Mật khẩu mã hóa người dùng
7			role	Integer	Có	Vai trò
8			ban	Boolean	Có	Tình trạng ban

- # Good Design

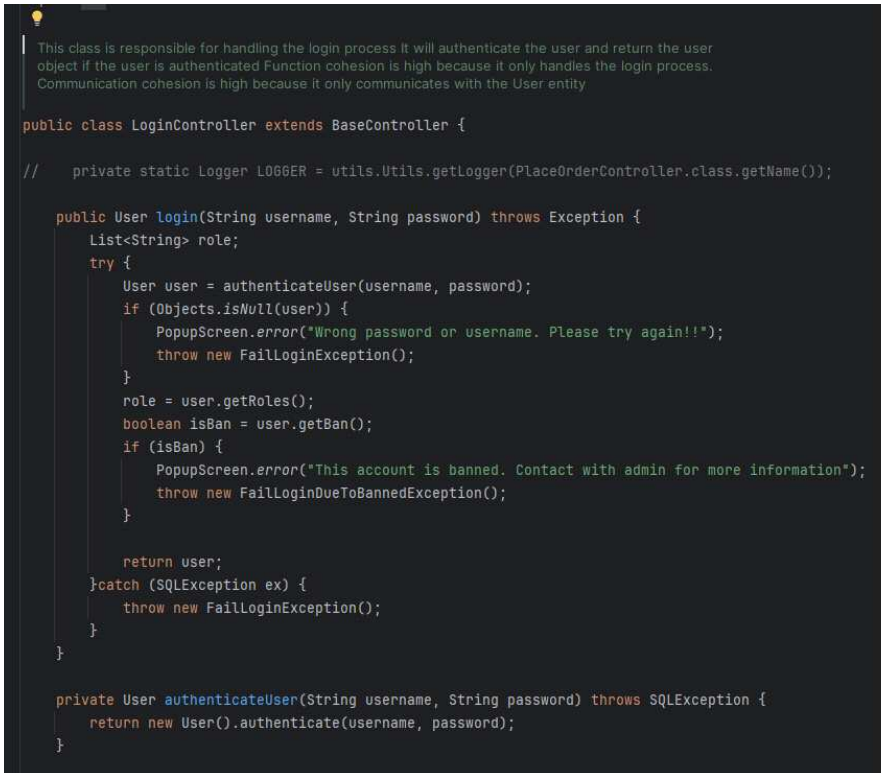
7.1. Cohesion



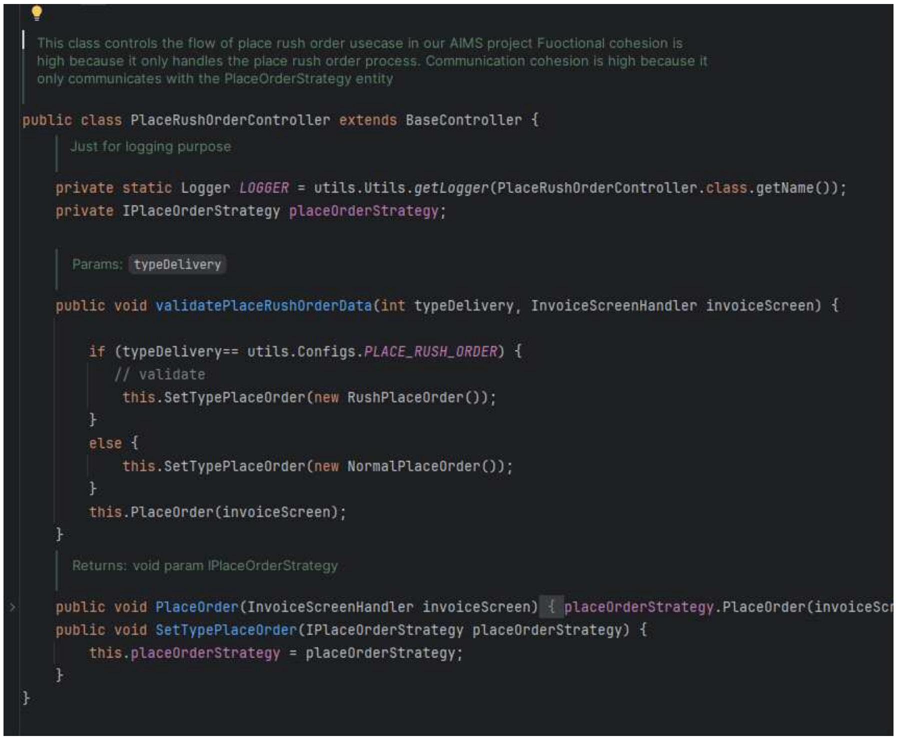
```
package controller;

import ...

This class is the base controller for our AIMS project.
SRP Violation of Single Responsibility Principle (SRP): The HomeController class extends from
    BaseController and implements a new function related to retrieving all Media from the database
public class BaseController {

    Communicational Cohesion : The method checks whether the Media in Cart, if it were in, we will
    return the CartMedia else return null.
    Params: media – media object
    Returns: CartMedia or null
    public CartMedia checkMediaInCart(Media media) { return Cart.getCart().checkMediaInCart(media); }

    This method gets the list of items in cart.
    Returns: List[CartMedia]
    public List getListCartMedia() { return Cart.getCart().getListMedia(); }

    public List getAllUser() throws SQLException {
        return new User().getAllUser();
    }

}
```



```java
package controller;

import ...

    This class controls the flow of events in homescreen
    SRP Violation of Single Responsibility Principle (SRP): The HomeController class extends from
        BaseController and implements a new function related to retrieving all Media from the database.

public class HomeController extends BaseController {

        this method gets all Media in DB and return back to home to display
        Returns: List[Media]
        Throws: SQLException

    public List getAllMedia() throws SQLException {
        return new Media().getAllMedia();
    }

    public List searchMedia(String searchText) throws SQLException {
        return new Media().searchMedia(searchText);
    }

    public List handleFilter(String filterType) throws SQLException {
        return new Media().getMediaByType(filterType);
    }
}
```

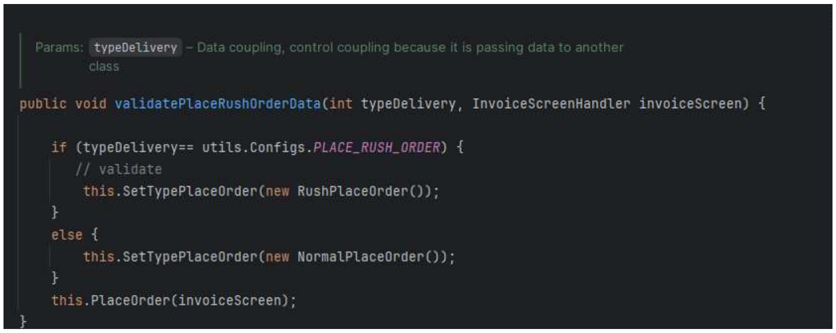


```java
    This class is responsible for handling the login process It will authenticate the user and return the user
    object if the user is authenticated Function cohesion is high because it only handles the login process.
    Communication cohesion is high because it only communicates with the User entity

public class LoginController extends BaseController {

//    private static Logger LOGGER = utils.Utils.getLogger(PlaceOrderController.class.getName());

    public User login(String username, String password) throws Exception {
        List<String> role;
        try {
            User user = authenticateUser(username, password);
            if (Objects.isNull(user)) {
                PopupScreen.error("Wrong password or username. Please try again!!");
                throw new FailLoginException();
            }
            role = user.getRoles();
            boolean isBan = user.getBan();
            if (isBan) {
                PopupScreen.error("This account is banned. Contact with admin for more information");
                throw new FailLoginDueToBannedException();
            }

            return user;
        }catch (SQLException ex) {
            throw new FailLoginException();
        }
    }

    private User authenticateUser(String username, String password) throws SQLException {
        return new User().authenticate(username, password);
    }
```

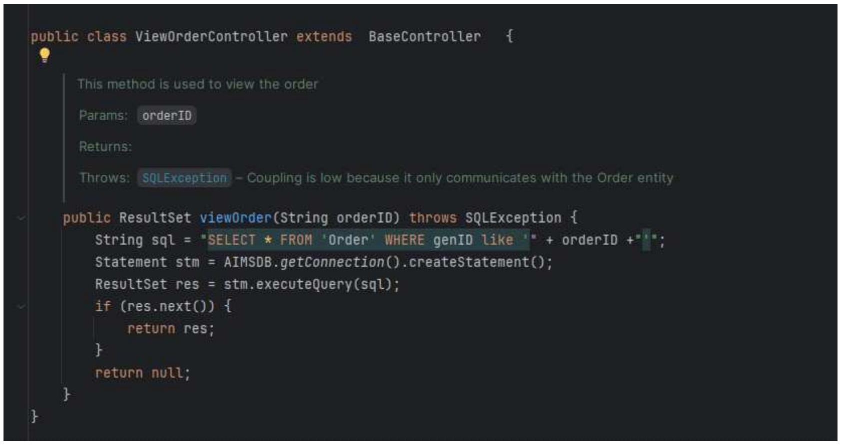


```
This class controls the flow of place rush order usecase in our AIMS project Fuoctional cohesion is
high because it only handles the place rush order process. Communication cohesion is high because it
only communicates with the PlaceOrderStrategy entity

public class PlaceRushOrderController extends BaseController {
    Just for logging purpose

    private static Logger LOGGER = utils.Utils.getLogger(PlaceRushOrderController.class.getName());
    private IPlaceOrderStrategy placeOrderStrategy;


     Params: typeDelivery

    public void validatePlaceRushOrderData(int typeDelivery, InvoiceScreenHandler invoiceScreen) {

        if (typeDelivery== utils.Configs.PLACE_RUSH_ORDER) {
           // validate
            this.SetTypePlaceOrder(new RushPlaceOrder());
        }
        else {
            this.SetTypePlaceOrder(new NormalPlaceOrder());
        }
        this.PlaceOrder(invoiceScreen);
    }
     Returns: void param IPlaceOrderStrategy

    public void PlaceOrder(InvoiceScreenHandler invoiceScreen) { placeOrderStrategy.PlaceOrder(invoiceScr
    public void SetTypePlaceOrder(IPlaceOrderStrategy placeOrderStrategy) {
        this.placeOrderStrategy = placeOrderStrategy;
    }
}
```

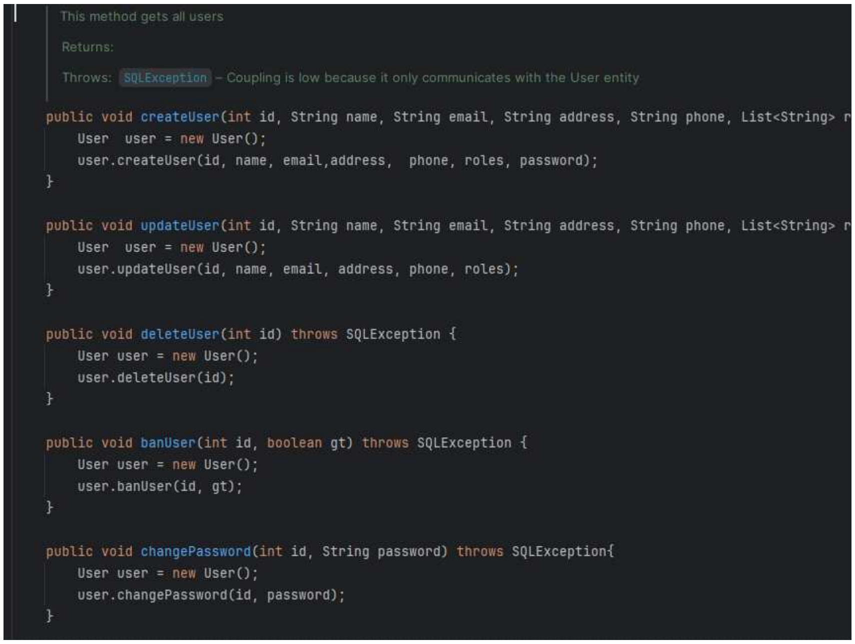
7.2. Coupling



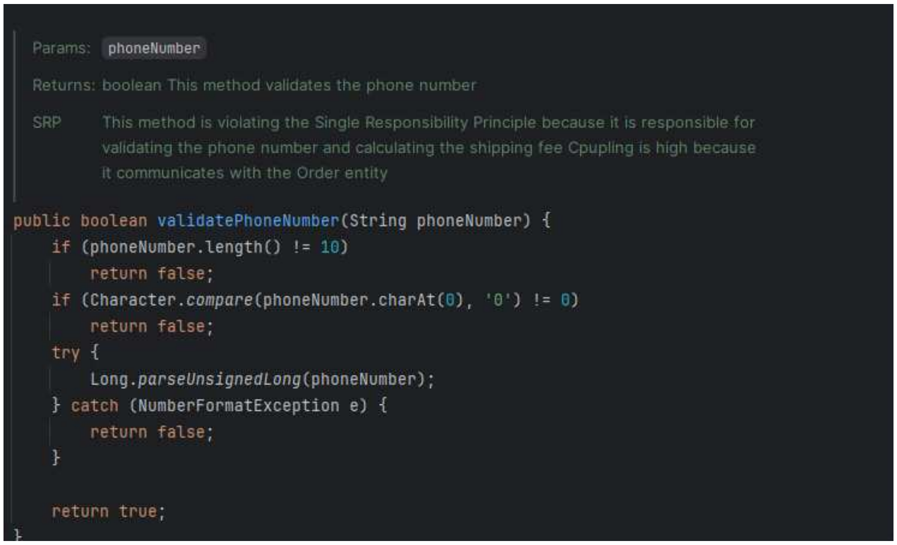
```
Params: typeDelivery – Data coupling, control coupling because it is passing data to another
        class
public void validatePlaceRushOrderData(int typeDelivery, InvoiceScreenHandler invoiceScreen) {

    if (typeDelivery== utils.Configs.PLACE_RUSH_ORDER) {
       // validate
        this.SetTypePlaceOrder(new RushPlaceOrder());
    }
    else {
        this.SetTypePlaceOrder(new NormalPlaceOrder());
    }
    this.PlaceOrder(invoiceScreen);
}
```



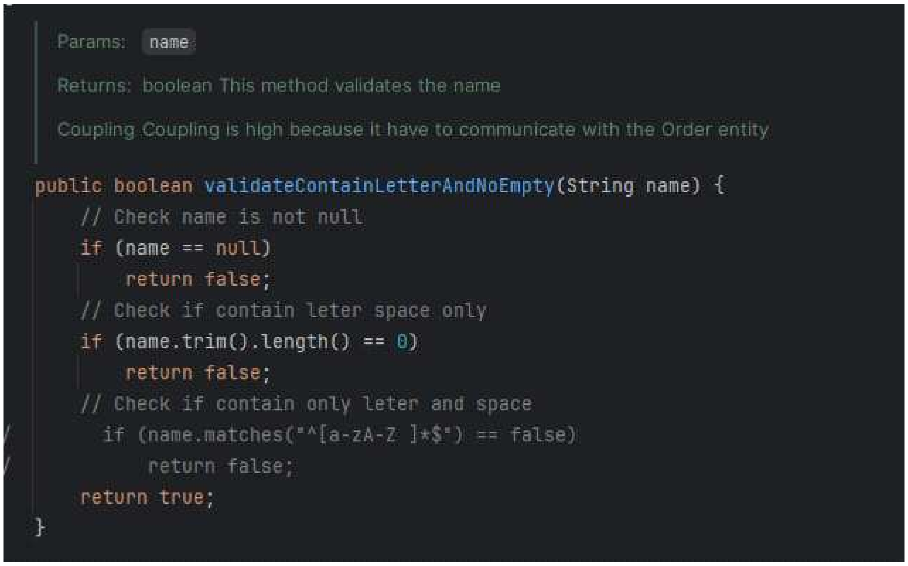
```
public class ViewOrderController extends  BaseController   {

    This method is used to view the order
    Params: orderID
    Returns:
    Throws: SQLException – Coupling is low because it only communicates with the Order entity
    public ResultSet viewOrder(String orderID) throws SQLException {
        String sql = "SELECT * FROM 'Order' WHERE genID like '" + orderID +"'";
        Statement stm = AIMSDB.getConnection().createStatement();
        ResultSet res = stm.executeQuery(sql);
        if (res.next()) {
            return res;
        }
        return null;
    }
}
```



```
    This method gets all users
    Returns:
    Throws: SQLException – Coupling is low because it only communicates with the User entity

public void createUser(int id, String name, String email, String address, String phone, List<String> r
    User  user = new User();
    user.createUser(id, name, email,address,  phone, roles, password);
}

public void updateUser(int id, String name, String email, String address, String phone, List<String> r
    User  user = new User();
    user.updateUser(id, name, email, address, phone, roles);
}

public void deleteUser(int id) throws SQLException {
    User user = new User();
    user.deleteUser(id);
}

public void banUser(int id, boolean gt) throws SQLException {
    User user = new User();
    user.banUser(id, gt);
}

public void changePassword(int id, String password) throws SQLException{
    User user = new User();
    user.changePassword(id, password);
}
```

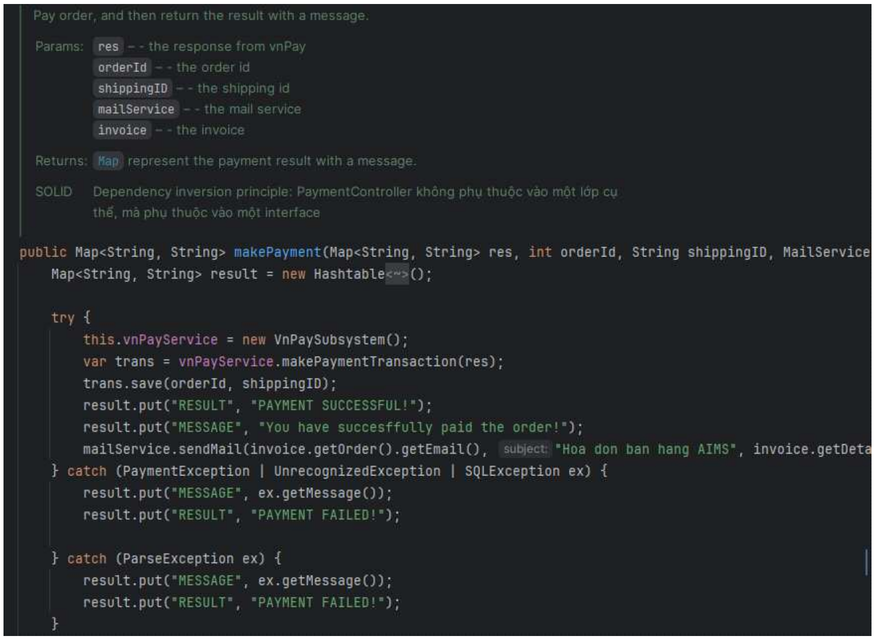


```
Params:  phoneNumber
Returns: boolean This method validates the phone number
SRP      This method is violating the Single Responsibility Principle because it is responsible for
         validating the phone number and calculating the shipping fee Cpupling is high because
         it communicates with the Order entity
public boolean validatePhoneNumber(String phoneNumber) {
    if (phoneNumber.length() != 10)
        return false;
    if (Character.compare(phoneNumber.charAt(0), '0') != 0)
        return false;
    try {
        Long.parseUnsignedLong(phoneNumber);
    } catch (NumberFormatException e) {
        return false;
    }

    return true;
}
```

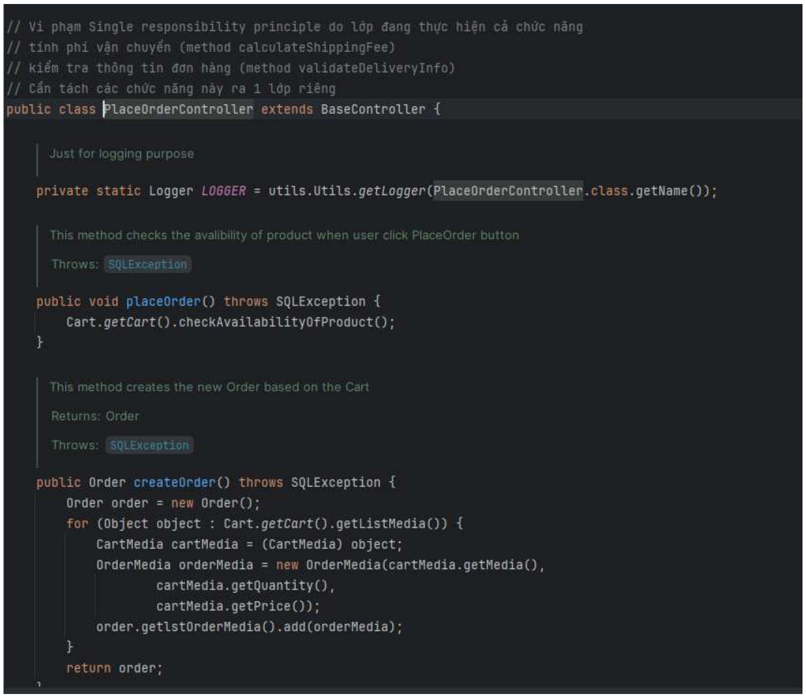


```
Params:  name

Returns: boolean This method validates the name

Coupling Coupling is high because it have to communicate with the Order entity

public boolean validateContainLetterAndNoEmpty(String name) {
    // Check name is not null
    if (name == null)
        return false;
    // Check if contain leter space only
    if (name.trim().length() == 0)
        return false;
    // Check if contain only leter and space
      if (name.matches("^[a-zA-Z ]*$") == false)
          return false;
    return true;
}
```

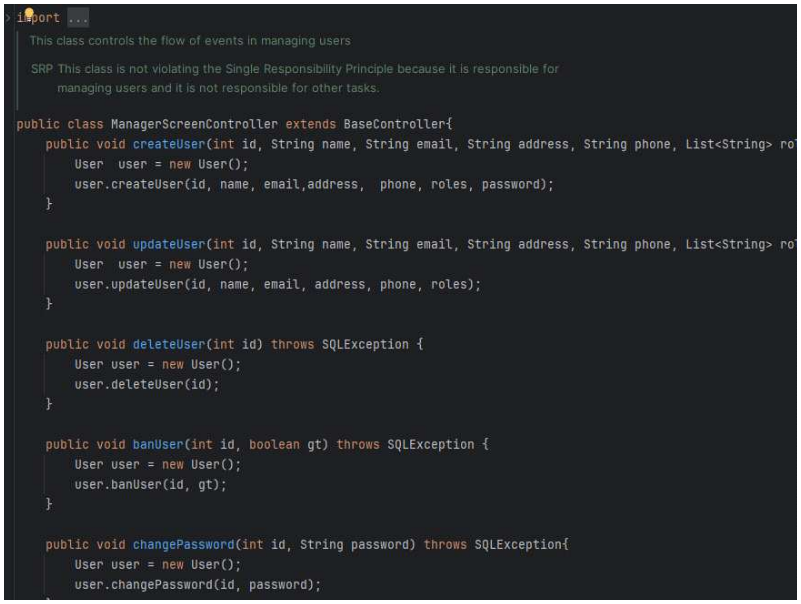
7.3. SOLID



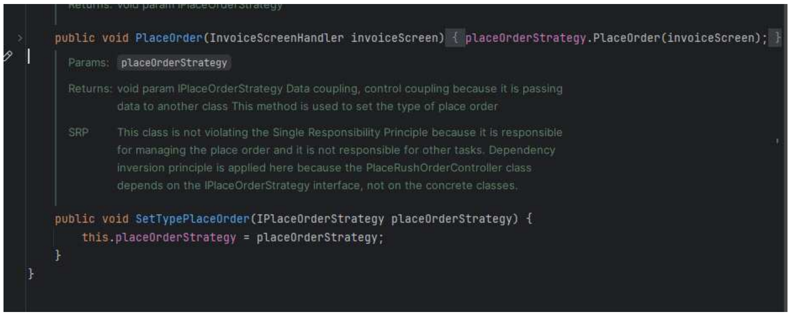
```
Pay order, and then return the result with a message.

Params:  res - - the response from vnPay
         orderId - - the order id
         shippingID - - the shipping id
         mailService - - the mail service
         invoice - - the invoice

Returns: Map represent the payment result with a message.

SOLID    Dependency inversion principle: PaymentController không phụ thuộc vào một lớp cụ
         thể, mà phụ thuộc vào một interface

public Map<String, String> makePayment(Map<String, String> res, int orderId, String shippingID, MailService
    Map<String, String> result = new Hashtable<~>();

    try {
        this.vnPayService = new VnPaySubsystem();
        var trans = vnPayService.makePaymentTransaction(res);
        trans.save(orderId, shippingID);
        result.put("RESULT", "PAYMENT SUCCESSFUL!");
        result.put("MESSAGE", "You have succesffully paid the order!");
        mailService.sendMail(invoice.getOrder().getEmail(), subject: "Hoa don ban hang AIMS", invoice.getDeta
    } catch (PaymentException | UnrecognizedException | SQLException ex) {
        result.put("MESSAGE", ex.getMessage());
        result.put("RESULT", "PAYMENT FAILED!");

    } catch (ParseException ex) {
        result.put("MESSAGE", ex.getMessage());
        result.put("RESULT", "PAYMENT FAILED!");
    }
```



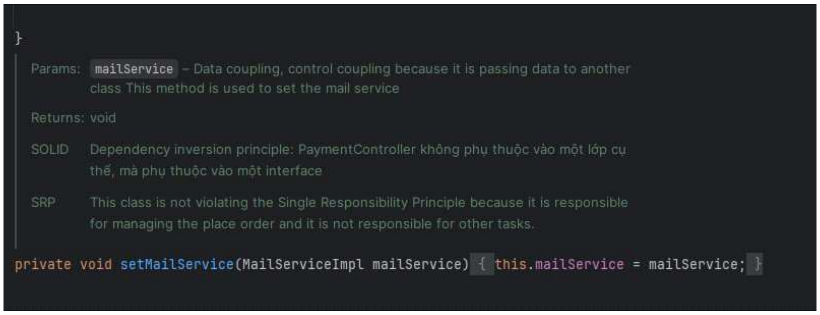
```
// Vi phạm Single responsibility principle do lớp đang thực hiện cả chức năng
// tính phí vận chuyển (method calculateShippingFee)
// kiểm tra thông tin đơn hàng (method validateDeliveryInfo)
// Cần tách các chức năng này ra 1 lớp riêng
public class PlaceOrderController extends BaseController {

    Just for logging purpose

    private static Logger LOGGER = utils.Utils.getLogger(PlaceOrderController.class.getName());

    This method checks the avalibility of product when user click PlaceOrder button
    Throws: SQLException

    public void placeOrder() throws SQLException {
        Cart.getCart().checkAvailabilityOfProduct();
    }

    This method creates the new Order based on the Cart
    Returns: Order
    Throws: SQLException

    public Order createOrder() throws SQLException {
        Order order = new Order();
        for (Object object : Cart.getCart().getListMedia()) {
            CartMedia cartMedia = (CartMedia) object;
            OrderMedia orderMedia = new OrderMedia(cartMedia.getMedia(),
                    cartMedia.getQuantity(),
                    cartMedia.getPrice());
            order.getlstOrderMedia().add(orderMedia);
        }
        return order;
    }
```



```
import ...
This class controls the flow of events in managing users
SRP This class is not violating the Single Responsibility Principle because it is responsible for
    managing users and it is not responsible for other tasks.

public class ManagerScreenController extends BaseController{
    public void createUser(int id, String name, String email, String address, String phone, List<String> ro
        User  user = new User();
        user.createUser(id, name, email,address,  phone, roles, password);
    }

    public void updateUser(int id, String name, String email, String address, String phone, List<String> ro
        User  user = new User();
        user.updateUser(id, name, email, address, phone, roles);
    }

    public void deleteUser(int id) throws SQLException {
        User user = new User();
        user.deleteUser(id);
    }

    public void banUser(int id, boolean gt) throws SQLException {
        User user = new User();
        user.banUser(id, gt);
    }

    public void changePassword(int id, String password) throws SQLException{
        User user = new User();
        user.changePassword(id, password);
    }
```



```
    Returns: void param IPlaceOrderStrategy

    public void PlaceOrder(InvoiceScreenHandler invoiceScreen) { placeOrderStrategy.PlaceOrder(invoiceScreen); }
      Params:  placeOrderStrategy
      Returns: void param IPlaceOrderStrategy Data coupling, control coupling because it is passing
               data to another class This method is used to set the type of place order
      SRP      This class is not violating the Single Responsibility Principle because it is responsible
               for managing the place order and it is not responsible for other tasks. Dependency
               inversion principle is applied here because the PlaceRushOrderController class
               depends on the IPlaceOrderStrategy interface, not on the concrete classes.
    public void SetTypePlaceOrder(IPlaceOrderStrategy placeOrderStrategy) {
        this.placeOrderStrategy = placeOrderStrategy;
    }
}
```

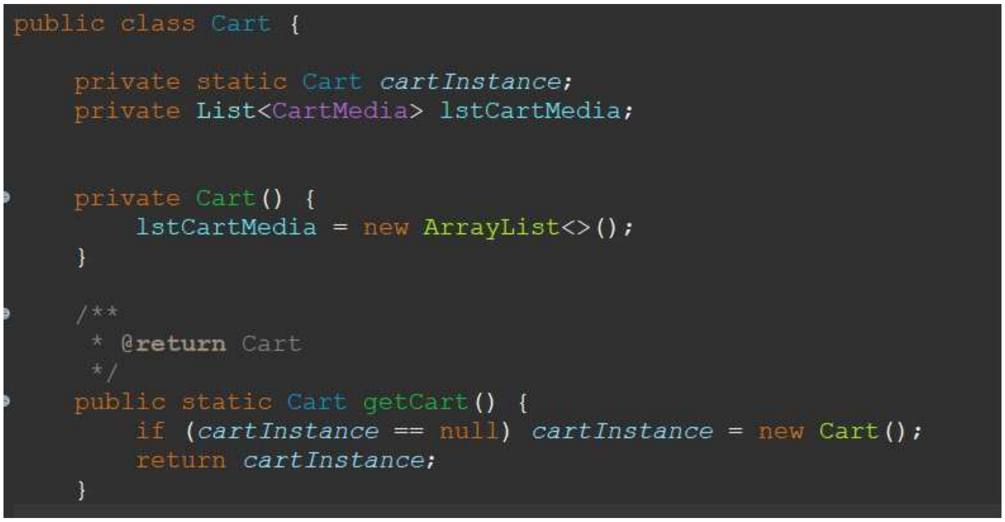


```
}
  Params:  mailService – Data coupling, control coupling because it is passing data to another
           class This method is used to set the mail service
  Returns: void
  SOLID    Dependency inversion principle: PaymentController không phụ thuộc vào một lớp cụ
           thế, mà phụ thuộc vào một interface
  SRP      This class is not violating the Single Responsibility Principle because it is responsible
           for managing the place order and it is not responsible for other tasks.
private void setMailService(MailServiceImpl mailService) { this.mailService = mailService; }
```

7.4. Design Pattern

7.4.1 Singleton

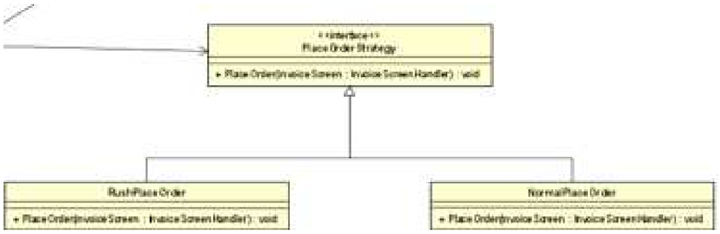
- Singleton pattern is used in the Cart class. Cart has a private static field called cartInstance, which holds the single instance of the Cart class. The lstCartMedia field is a list that stores instances of CartMedia. It represents the items present in the shopping cart. The constructor of Cart is declared as private, preventing direct instantiation of the class from outside. The getCart method is a static public method that provides access to the single instance of the Cart class. It follows the Singleton pattern by ensuring that only one instance of Cart is created.
- By using the Singleton pattern, the Cart class ensures that there is only one instance of the class throughout the application, allowing centralized access to the shopping cart from different parts of the code. It is used in this scenarios to ensure there is only one cart per software session.



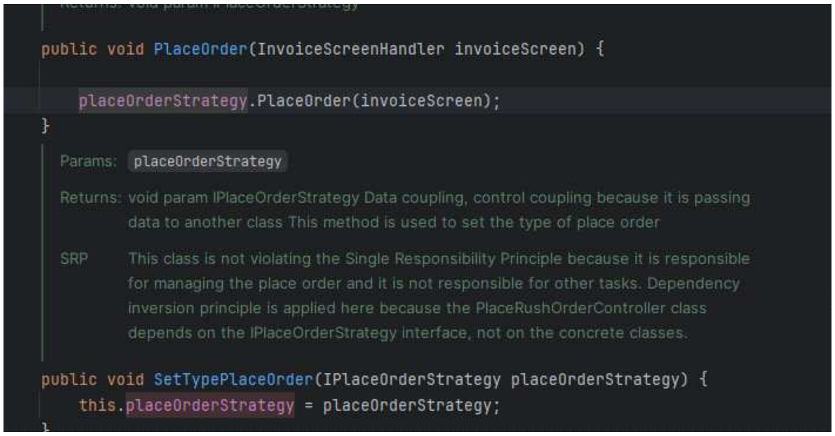
```
public class Cart {

    private static Cart cartInstance;
    private List<CartMedia> lstCartMedia;


    private Cart() {
        lstCartMedia = new ArrayList<>();
    }

    /**
     * @return Cart
     */
    public static Cart getCart() {
        if (cartInstance == null) cartInstance = new Cart();
        return cartInstance;
    }
```

7.4.2 Strategy

- Strategy Pattern is a behavioral design pattern that allows you to define a family of algorithms, encapsulate each one, and make them interchangeable. The strategy pattern lets the algorithm vary independently from the clients that use it.
- Using the Strategy Pattern for handling the delivery stage (calculating shipping fees, displaying the invoice), we can define different strategies for calculating shipping costs and displaying invoices.
- Using the Strategy Pattern allows you to flexibly choose and change the algorithms for calculating shipping fees and displaying invoices without modifying the Order class's code. This design pattern helps make the code more extensible, maintainable, and clear



<<interface>>
Place Order Strategy
+ PlaceOrder(invoiceScreen : InvoiceScreenHandler) : void
RushPlaceOrder
+ PlaceOrder(invoiceScreen : InvoiceScreenHandler) : void
NormalPlaceOrder
+ PlaceOrder(invoiceScreen : InvoiceScreenHandler) : void



```
    Returns: void param IPlaceOrderStrategy

public void PlaceOrder(InvoiceScreenHandler invoiceScreen) {

    placeOrderStrategy.PlaceOrder(invoiceScreen);
}
    Params:  placeOrderStrategy
    Returns: void param IPlaceOrderStrategy Data coupling, control coupling because it is passing
             data to another class This method is used to set the type of place order
    SRP      This class is not violating the Single Responsibility Principle because it is responsible
             for managing the place order and it is not responsible for other tasks. Dependency
             inversion principle is applied here because the PlaceRushOrderController class
             depends on the IPlaceOrderStrategy interface, not on the concrete classes.
public void SetTypePlaceOrder(IPlaceOrderStrategy placeOrderStrategy) {
    this.placeOrderStrategy = placeOrderStrategy;
}
```