

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

Viện công nghệ thông tin và truyền thông

Tài liệu thiết kế phần mềm

Phiên bản:1.2

Xây dựng ứng dụng EcoBikeRental

Môn: Thiết kế và xây dựng phần mềm

Giảng viên hướng dẫn: TS. Nguyễn Thị Thu Trang

Nhóm : 03

Thành viên nhóm:

Lê Văn Cường	20172987
Nguyễn Trọng Chinh	20172980
Bùi Tiến Đạt	20177024
Lã Văn Dân	20177023

*Hà Nội, ngày 3 tháng 1 năm 2021*

# Mục Lục

<b>Mục Lục .....</b>	<b>1</b>
<b>1 Giới thiệu .....</b>	<b>2</b>
1.1 Mục tiêu .....	2
1.2 Phạm vi .....	2
1.3 Từ điển thuật ngữ .....	2
1.4 Tài liệu tham khảo .....	3
<b>2 Kiến trúc hệ thống và thiết kế kiến trúc.....</b>	<b>4</b>
2.1 Các mẫu kiến trúc.....	4
2.2 Biểu đồ tương tác .....	4
2.3 Biểu đồ lớp phân tích.....	5
2.4 Biểu đồ lớp phân tích gộp.....	9
<b>3 Thiết kế chi tiết.....</b>	<b>10</b>
3.1 Thiết kế giao diện người dùng .....	10
3.1.1 Chuẩn hoá cấu hình màn hình .....	10
3.1.2 Sơ đồ chuyển đổi màn hình .....	11
3.1.3 Thông số kỹ thuật màn hình .....	12
3.2 Mô hình hoá dữ liệu .....	16
3.2.1 Mô tả chung.....	16
3.2.2 Thiết kế cơ sở dữ liệu.....	17
<b>4 Thiết kế lớp.....</b>	<b>20</b>
4.1.1 Sơ đồ chung .....	20
4.1.2 Sơ đồ lớp .....	20

# 1 Giới thiệu

## 1.1 Mục tiêu

Tài liệu này được thiết kế để đưa ra thiết kế phần mềm của ứng dụng EcoBikeRental. Nó bao gồm các mô tả tổng quan và trình bày chi tiết các thiết kế hệ thống và thiết kế kiến trúc, thiết kế giao diện và thiết kế lớp cho các chức năng của phần mềm, Thiết kế cơ sở dữ liệu cho phần mềm.

Tài liệu đưa ra một góc nhìn tổng quan cho các bên liên quan nhằm thiết kế và xây dựng một phần mềm đúng theo các yêu cầu. tài liệu phục vụ cho những nhà phát triển phần mềm, kiểm thử viên, nhà quản lý dự án cũng như các bên liên quan

## 1.2 Phạm vi

Phạm vi của phần mềm EcoBikerental nhằm tạo ra phân hệ quản lý các chức năng (function) mà người dùng có thể sử dụng tại thời điểm chạy. Ngay sau khi khởi chạy ứng dụng, người dùng có thể sử dụng các tính năng của hệ thống (thuê, trả xe) mà không cần phải thông qua bất cứ bước đăng nhập hay xác thực nào. Người quản trị có thể tạo các thay đổi về loại xe, giá xe. Mỗi khi có thêm một chức năng mới, người quản trị cần đưa các thông tin về chức năng này vào phần mềm để người dùng có thể sử dụng và dễ dàng quản lý.

## 1.3 Từ điển thuật ngữ

STT	Thuật ngữ	Giải thích	Ví dụ	Ghi chú
1	Token	Một phần dữ liệu được tạo ở phía server ra chứa thông tin về người dùng và mã <b>token</b> . Token được sử dụng để xác nhận người dùng khi muốn đăng nhập với token đã được cung cấp mà không phải sử dụng trực tiếp tài khoản và mật khẩu.	JWT	Được thiết kế nhỏ gọn và an toàn

2	API	Viết tắt của Application Programming Interface. API là một giao diện mà một hệ thống máy tính hay ứng dụng cung cấp để cho phép các yêu cầu dịch vụ có thể được tạo ra từ các chương trình máy tính khác, và/hoặc cho phép dữ liệu có thể được trao đổi qua lại giữa chúng.	API cộng trừ tiền được cung cấp từ phía ngân hàng	
---	-----	--	--	--

#### **1.4 Tài liệu tham khảo**

- EcoBikeRental-ProblemStatement-EN.pdf
- EcoBikeRental-ProblemStatement-VN.pdf
- TKXDPM.20201-Lab01. ArchitecturalDesign.pdf
- TKXDPM.20201-Lab02. Interfacedesign.pdf
- TKXDPM.20201-Lab03. ClassDesignAndDataModeling .pdf

## 2 Kiến trúc hệ thống và thiết kế kiến trúc

### 2.1 Các mẫu kiến trúc

Kiến trúc của phần mềm là: kiến trúc MVC (Model-View-Controller).

Mẫu kiến trúc Model-View-Controller là phương pháp chia nhỏ các thành phần dữ liệu (data), trình bày (output) và dữ liệu nhập từ người dùng (input) thành những thành phần riêng biệt.

- Các thành phần

Model chứa dữ liệu và các tính toán xử lý logic để giải quyết vấn đề mà phần mềm hướng tới (business logic). Thành phần model thường được trình bày ở dạng Domain Model.

View là thành phần đảm nhận trình bày từ những dữ liệu của Model. View bao gồm những gì thể hiện trên màn hình như các control, form, widget,...

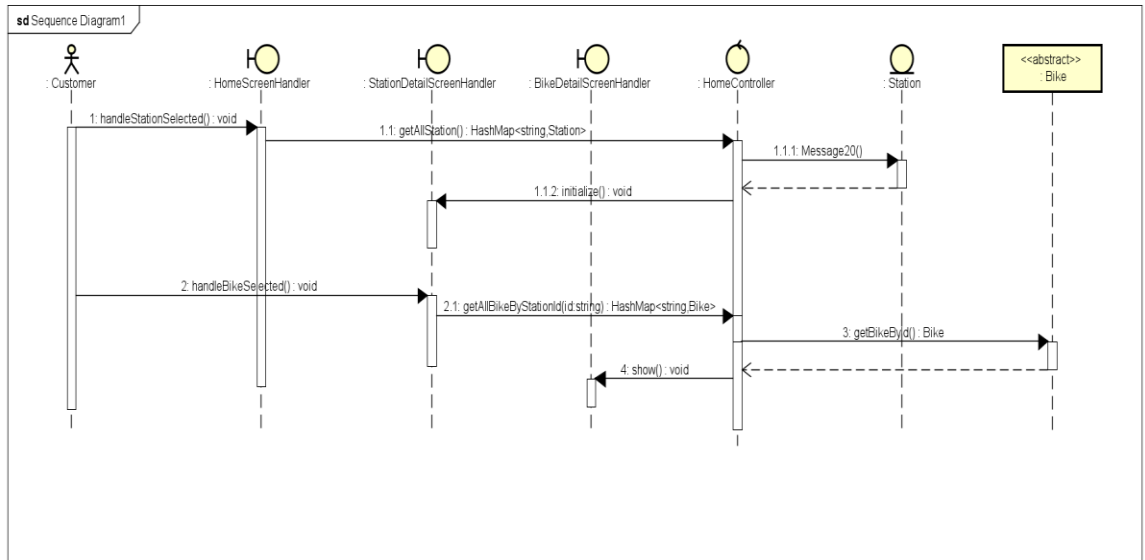
Controller là thành phần đảm nhận việc xử lý đáp trả lại các dữ liệu được đưa vào từ người dùng như các sự kiện chuột, bàn phím, các tương tác lên các control... Controller là cầu nối giữa người sử dụng và ứng dụng.

Lý do chọn kiến trúc MVC:

- Bảo trì mã dễ dàng, giúp mở rộng và phát triển
- Sự phát triển của các thành phần khác nhau có thể được thực hiện song song.
- Nó giúp bạn tránh sự phức tạp bằng cách chia một ứng dụng thành ba phần: Model, view và controller.
- Tất cả các layer (lớp) và các đối tượng là độc lập với nhau, có thể kiểm tra chúng một cách riêng biệt.

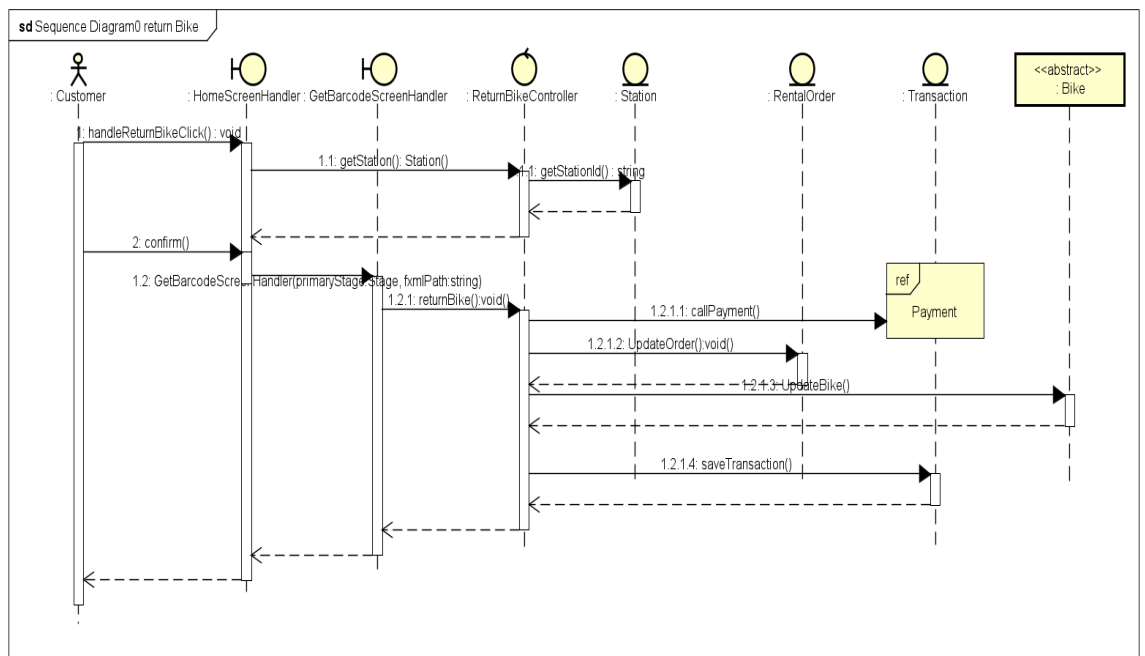
### 2.2 Biểu đồ tương tác

View bike



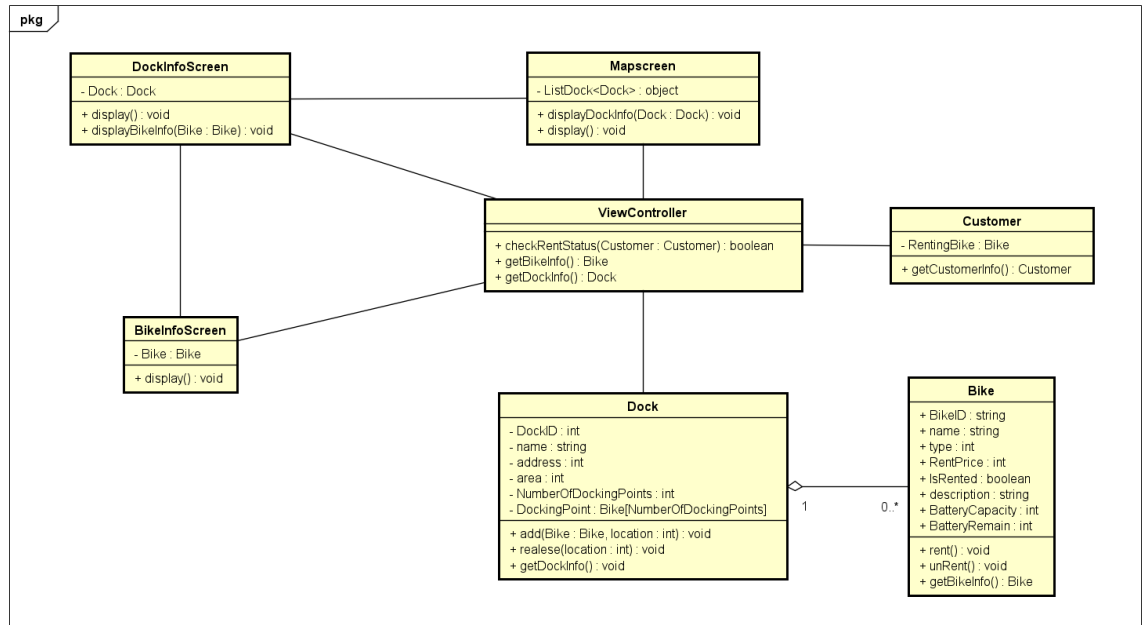
Rent bike

ReturnBike

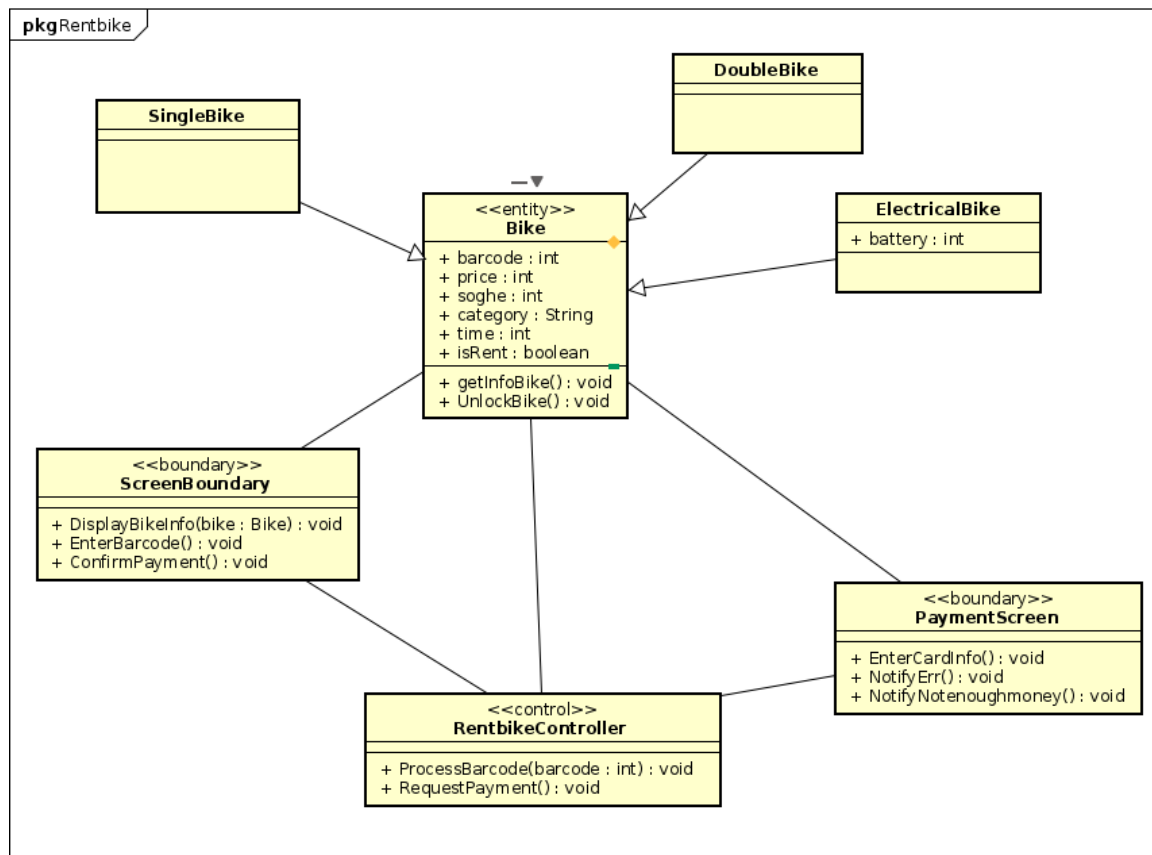


## 2.3 Biểu đồ lớp phân tích

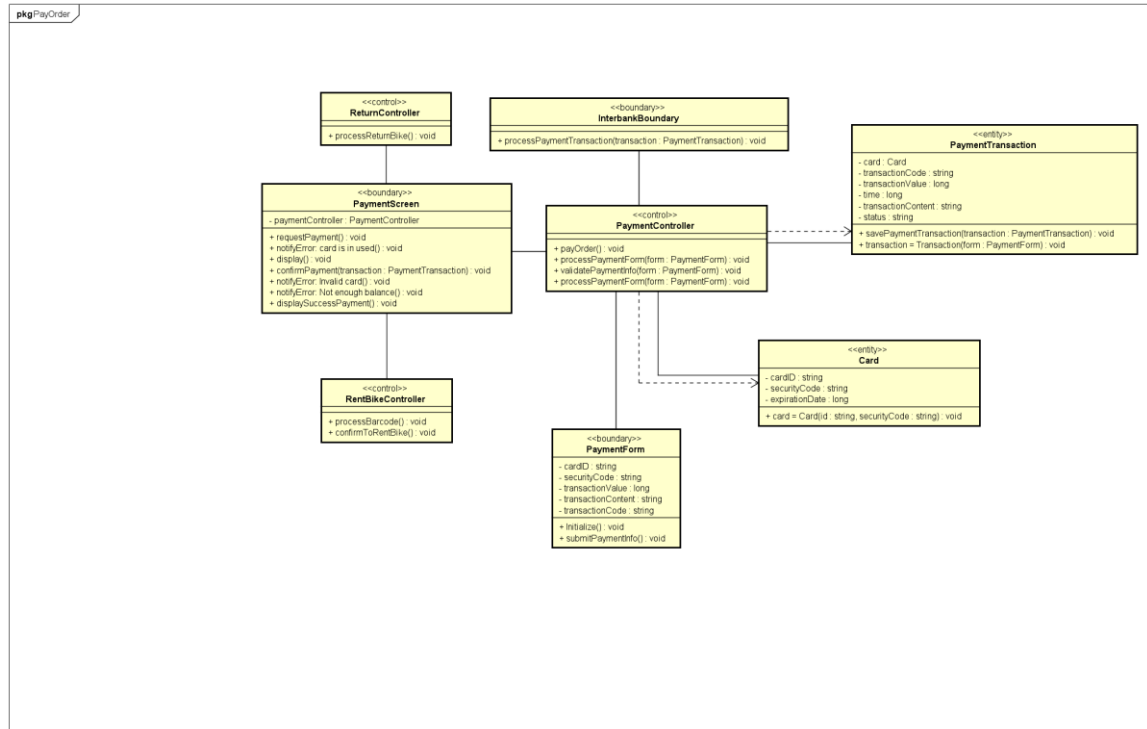
ViewBike



## RentBike

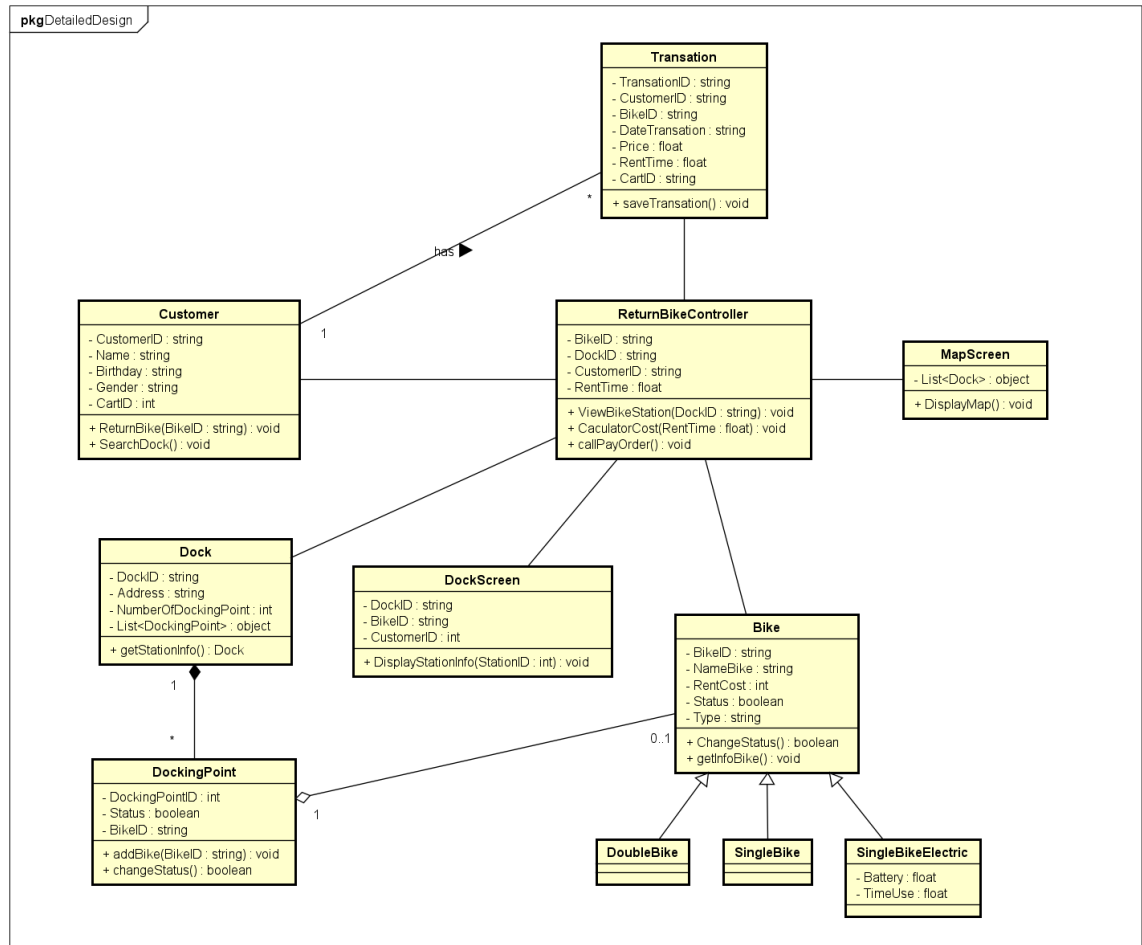


## Payment

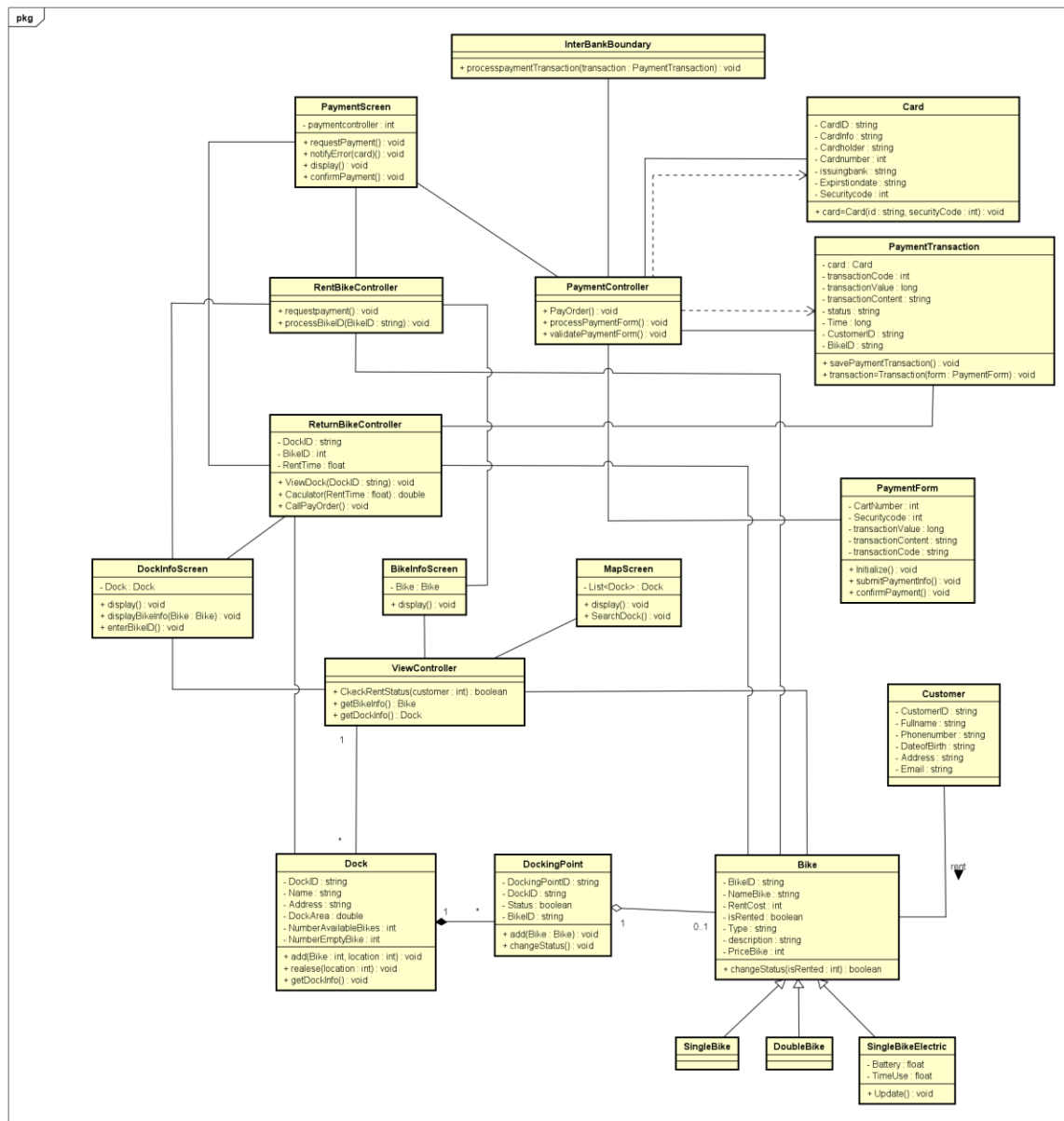


Return Bike





## 2.4 Biểu đồ lớp phân tích gộp



## 3 Thiết kế chi tiết

### 3.1 Thiết kế giao diện người dùng

#### 3.1.1 Chuẩn hoá cấu hình màn hình

##### Display

Số lượng màu được hỗ trợ: 16,777,216 màu

Độ phân giải: 674 x 445 pixels

##### Screen

Vị trí của của button: Ở dưới cùng (theo chiều dọc) và ở giữa (theo chiều ngang)

của khung.

Vị trí của message: Ở giữa trung tâm khung màn hình

Vị trí của screen title: Title đặt ở góc trên bên trái của màn hình.

Sự nhất quán trong hiển thị chữ số: dấu phẩy để phân cách hàng nghìn và chuỗi chỉ bao gồm các ký tự, chữ số, dấu phẩy, dấu chấm, dấu cách, dấu gạch dưới và ký hiệu gạch nối.

##### Control

Kích thước text: medium size (24px). Font: Segoe UI. Color: #000000

Xử lý check input: Nên kiểm tra xem input có empty hay không. Tiếp theo, kiểm tra

xem input có đúng format hay không.

Dịch chuyển màn hình: Không có các khung chồng lên nhau. Các màn hình được

tách biệt. Tuy nhiên, hướng dẫn sử dụng được xem như là 1 popup message vì màn hình chính ở dưới sẽ không thể thao tác trong khi màn hình hướng dẫn sử dụng đang được hiển thị. Ban đầu khi app khởi chạy thì màn hình splash screen (màn hình chớp) sẽ được hiện lên và sau đó màn hình đầu tiên(Home Screen) sẽ xuất hiện

**Thứ tự các màn hình trong hệ thống:**

1. Splash Screen
2. Home screen (first screen)
3. Enterbarcode Screen/Station Screen
4. Bike detail Screen
5. Confirm rental order
6. PaymentForm Screen
7. Payment connfirm screen
8. Paymeny result Screen

**Nhập input từ bàn phím**

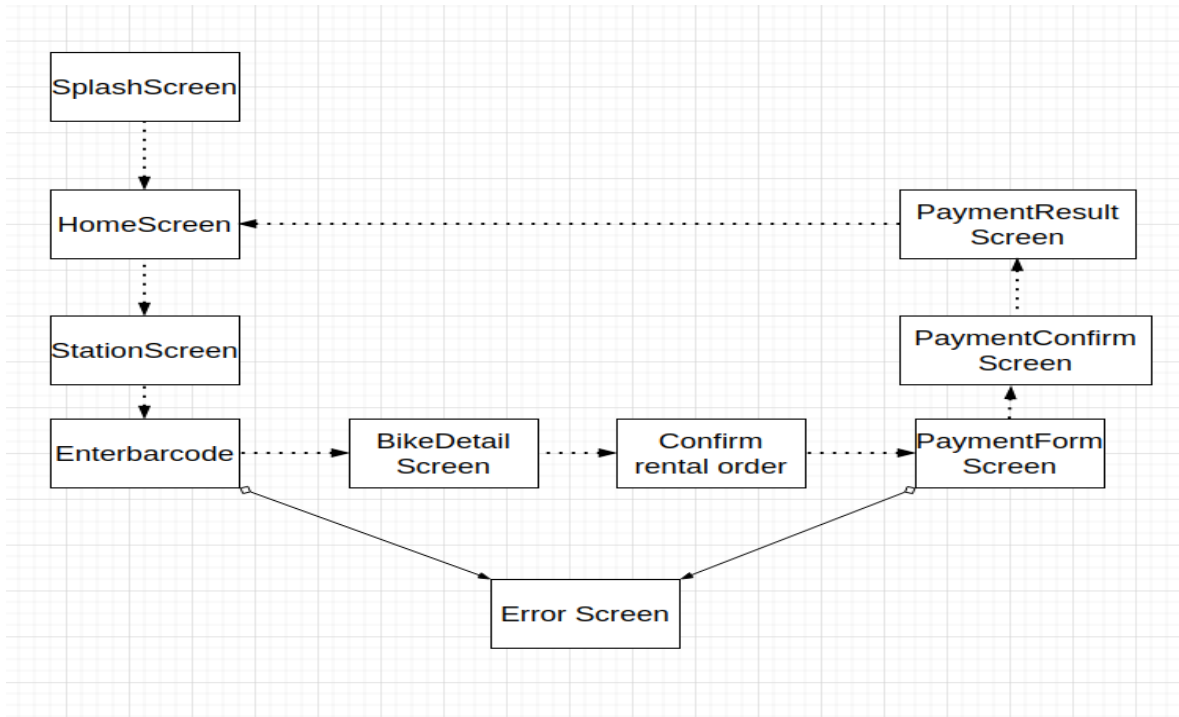
Sẽ không có phím tắt. Có các button quay lại để quay lại các màn hình trước đó.

Ngoài ra button “X” nằm ở thanh tiêu đề bên phải để đóng screen

**Error**

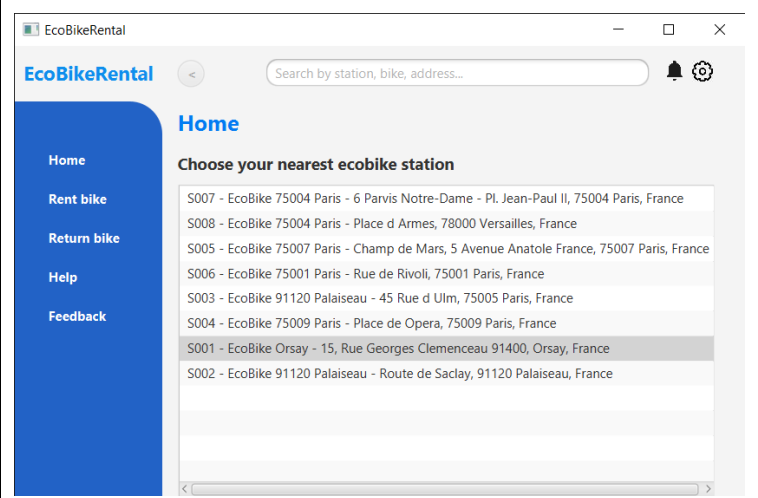
Một thông điệp sẽ được hiện lên để thông báo cho người dùng biết vấn đề đang gặp phải là gì.

**3.1.2 Sơ đồ chuyển đổi màn hình**



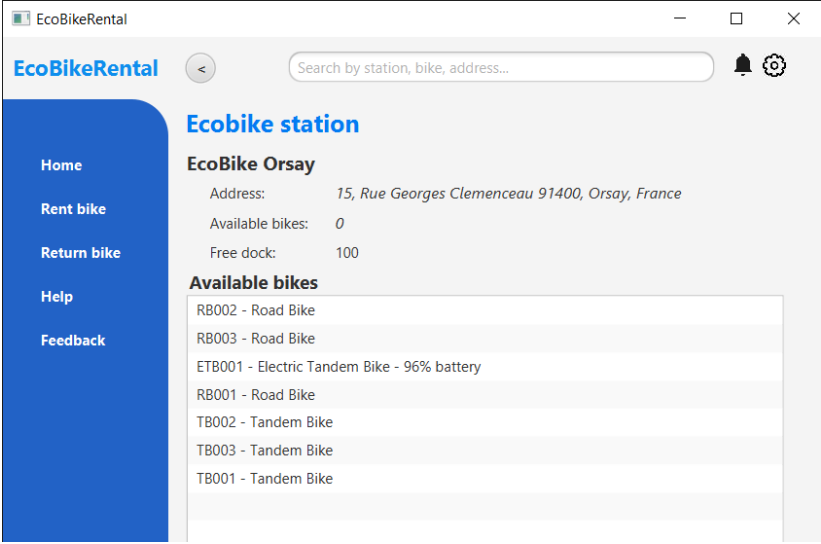
### 3.1.3 Thông số kỹ thuật màn hình

#### Home

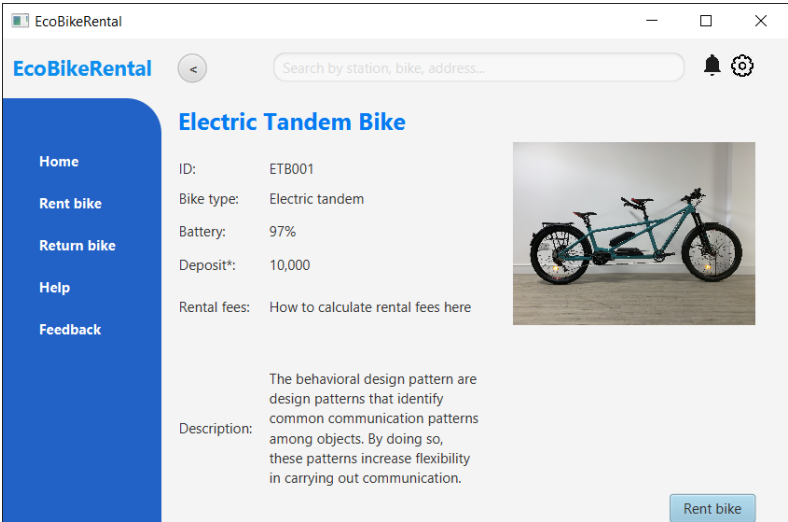
EcoBikeRental	Control	Operation	Function
	Search	Fill and enter	Tìm kiếm xe và bãi xe theo code
	Home	click	
	RentBike	click	Chuyển đến màn hình thuê xe
	ReturnBike	click	Chuyển đến màn hình trả xe

	Back	Click	Quay lại màn hình trước
--	------	-------	-------------------------

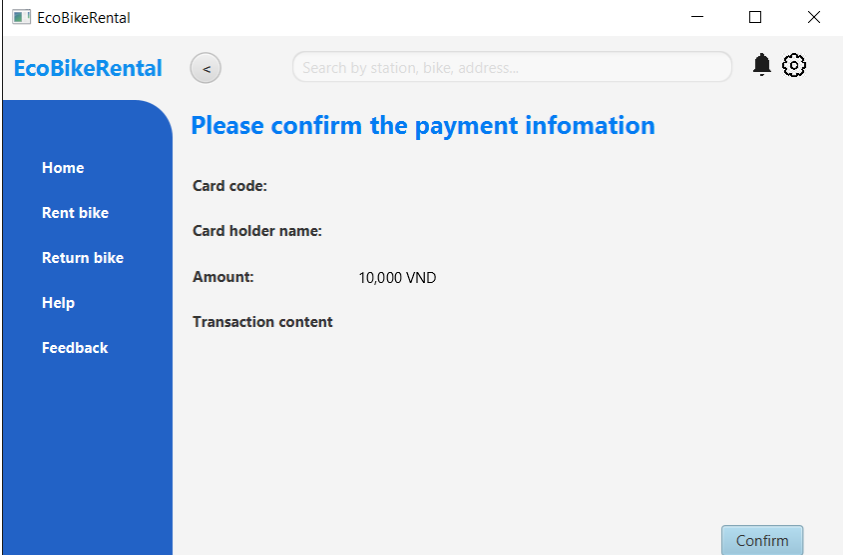
## ViewDock

	Control	Operation	Function
	List bike	select	Xem Xe

## ViewBike

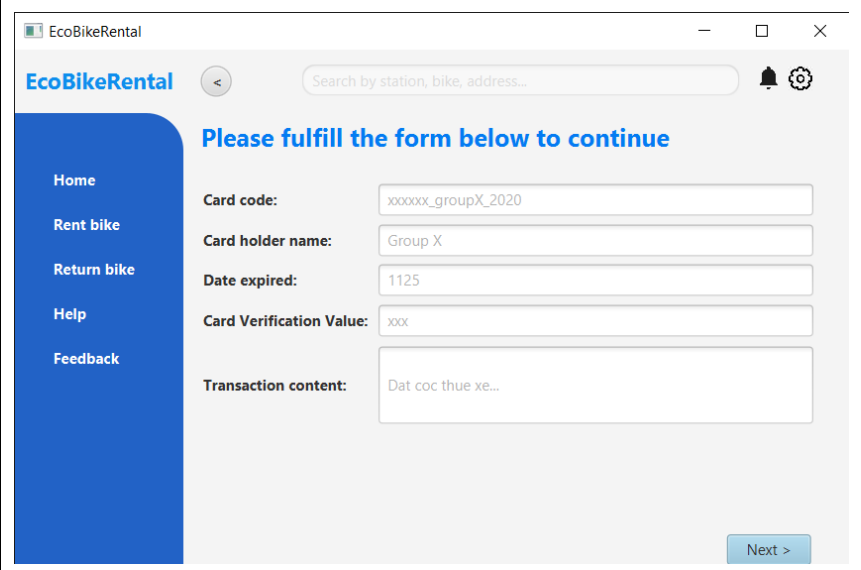
	Control	Operation	Function
	Rent bike	click	Chấp nhận thuê xe

## Comfirm rental bike

EcoBikeRental				Control	Operation	Function
				pay	click	Thanh toán xe

## PaymentForm

				Control	Operation	Function
				Card code	fill	Nhập mã thẻ
				Card holder Name	fill	Nhập tên thẻ
				Date expired	fill	Nhập ngày hết

		hạn của thẻ
Card Verification Value	fill	Nhập mã PIN
Transaction content	fill	Nhập nội dung giao dịch
Next	click	Tiếp tục để giao dịch.

## Return Bike

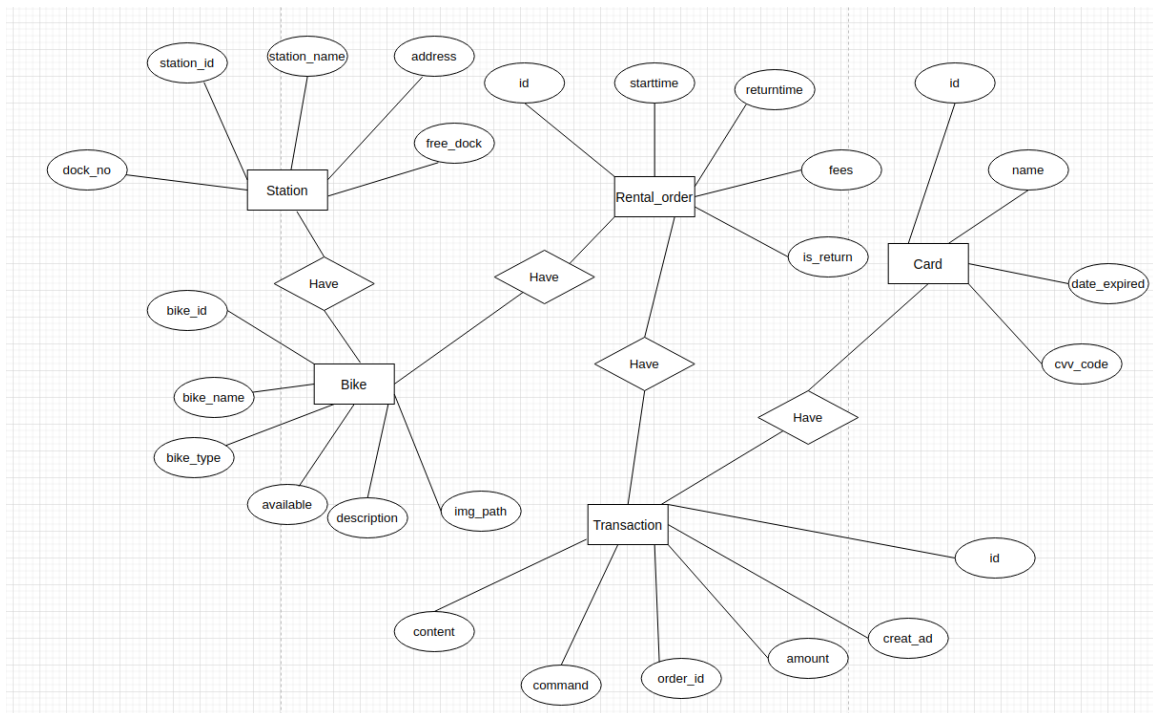
	Control	Operation	Function
	BarCode	fill	Nhập mã xe dx thuê
	Card Code	fill	Nhập mã thẻ đã thuê



<div> <div>EcoBikeRental</div> <div> <div>&lt;</div> <div>Search by station, bike, address...</div> <div> <div>🔔</div> <div>⚙️</div> </div> </div> <div> <div>Home</div> <div>Rent bike</div> <div>Return bike</div> <div>Help</div> <div>Feedback</div> </div> <div> <div>Provide infomation to return bike</div> <div> <div>Please enter the barcode of the bike you want to rent:</div> <div></div> </div> <div> <div>Enter the credit card code that was used to rent the bike:</div> <div></div> </div> <div>Return bike</div> </div> </div>	Return Bike	click	Chấp nhận trả xe
---	-------------	-------	------------------

## 3.2 Mô hình hoá dữ liệu

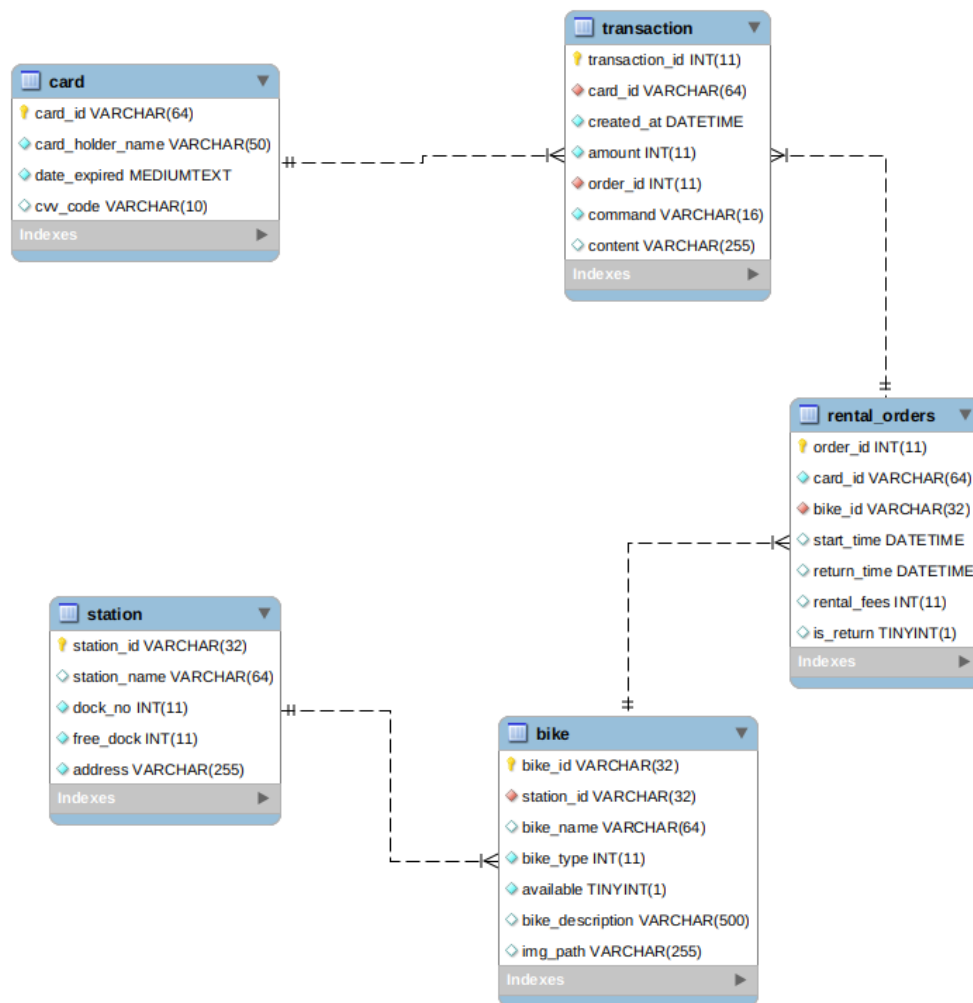
### 3.2.1 Mô tả chung



### 3.2.2 Thiết kế cơ sở dữ liệu

#### 3.2.2.1 Hệ thống quản lý dữ liệu

Hệ thống quản lý dữ liệu trên MySQL



#### 3.2.2.2 Mô hình dữ liệu logic

#### 3.2.2.3 Mô hình dữ liệu vật lý

*Station*

<i>stt</i>	<i>PK</i>	<i>FK</i>	<i>Column name</i>	<i>Data type</i>	<i>Default value</i>	<i>Mandatory</i>	<i>Description</i>
1	x		station_id	varchar(32)		Yes	id
2			station_name	varchar(64)		Yes	name of station
3			dock_no	int		Yes	
4			free_dock	int		Yes	
5			address	varchar(255)		Yes	

### *Bike*

<i>stt</i>	<i>PK</i>	<i>FK</i>	<i>Column name</i>	<i>Data type</i>	<i>Default value</i>	<i>Mandatory</i>	<i>Description</i>
1	x		bike_id	varchar(32)		Yes	id
2		x	station_id	varchar(32)		Yes	id_station
3			bike_name	varchar(64)		Yes	name of bike
4			bike_type	int		Yes	
5			available	int		Yes	
6			bike_description	varchar(500)		Yes	
7			img_path	Varchar(255)		No	bike image path

### *Card*

<i>stt</i>	<i>PK</i>	<i>FK</i>	<i>Column name</i>	<i>Data type</i>	<i>Default value</i>	<i>Mandatory</i>	<i>Description</i>
1	x		card_id	varchar(64)		Yes	id card
2			card_holder_name	varchar(32)		Yes	
3			date_expired	mediumtext		Yes	Expiration date

4			cvv_code	varchar(10)		Yes	cvv code
---	--	--	----------	-------------	--	-----	----------

### *Transaction*

<i>stt</i>	<i>PK</i>	<i>FK</i>	<i>Column name</i>	<i>Data type</i>	<i>Default value</i>	<i>Mandatory</i>	<i>Description</i>
1	x		transaction_id	int		Yes	id transaction
2		x	card_id	varchar(64)		Yes	id card
3			created_at	date		Yes	
4			amount	int		Yes	amount
5		x	order_id	int		Yes	order id
6			command	varchar(16)		Yes	command
7			content	varchar(255)		Yes	content

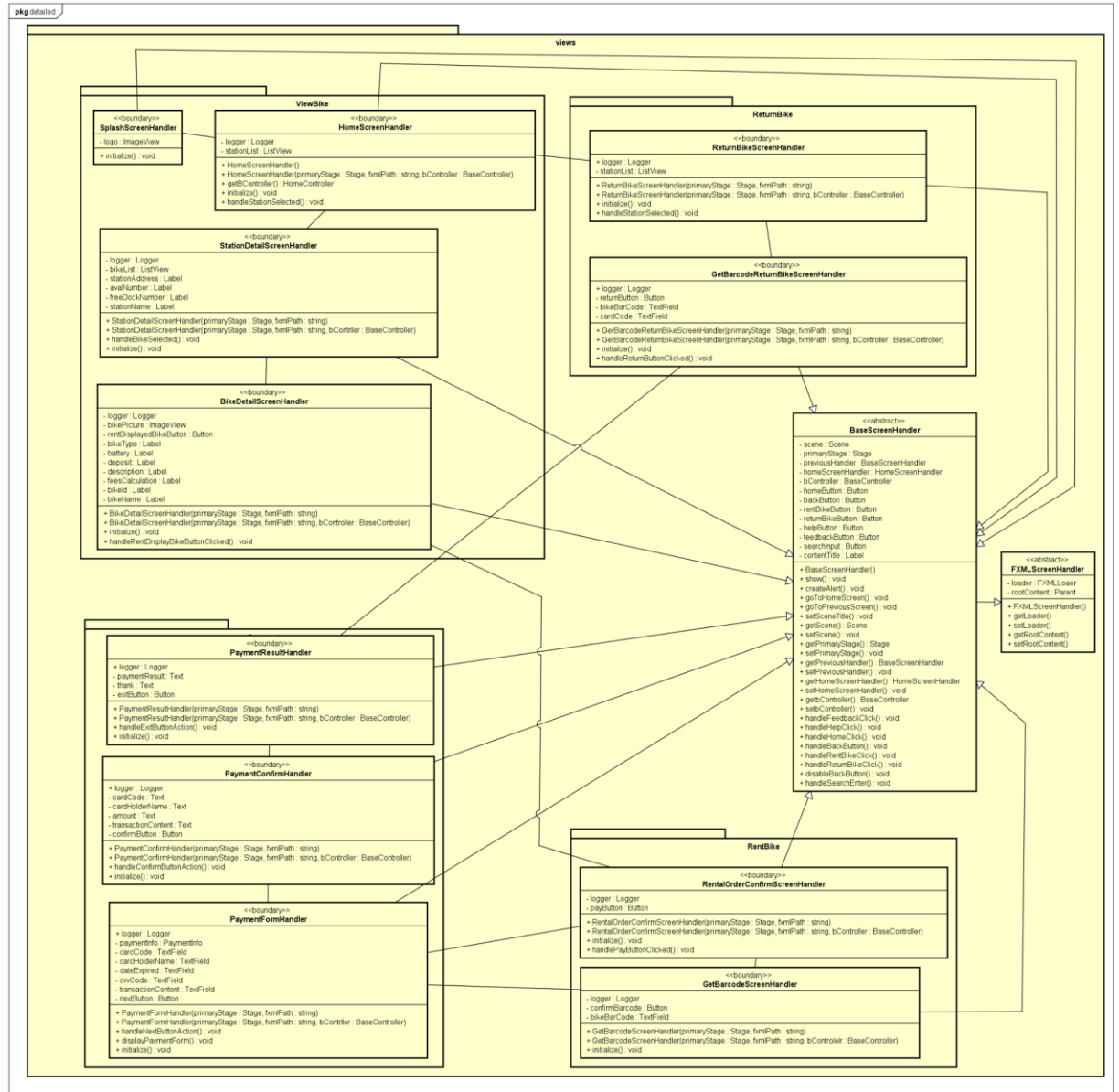
### *Rental\_orders*

<i>stt</i>	<i>PK</i>	<i>FK</i>	<i>Column name</i>	<i>Data type</i>	<i>Default value</i>	<i>Mandatory</i>	<i>Description</i>
1	x		order_id	int		Yes	order_id
2		x	card_id	varchar(64)		Yes	id card
3		x	bike_id	varchar(32)		Yes	bike_id
4			start_time	datetime		Yes	start time
5			return_time	datetime		Yes	return time
6			rental_fees	int		Yes	fee
7			is_return	tinyint		Yes	status

## 4 Thiết kế lớp

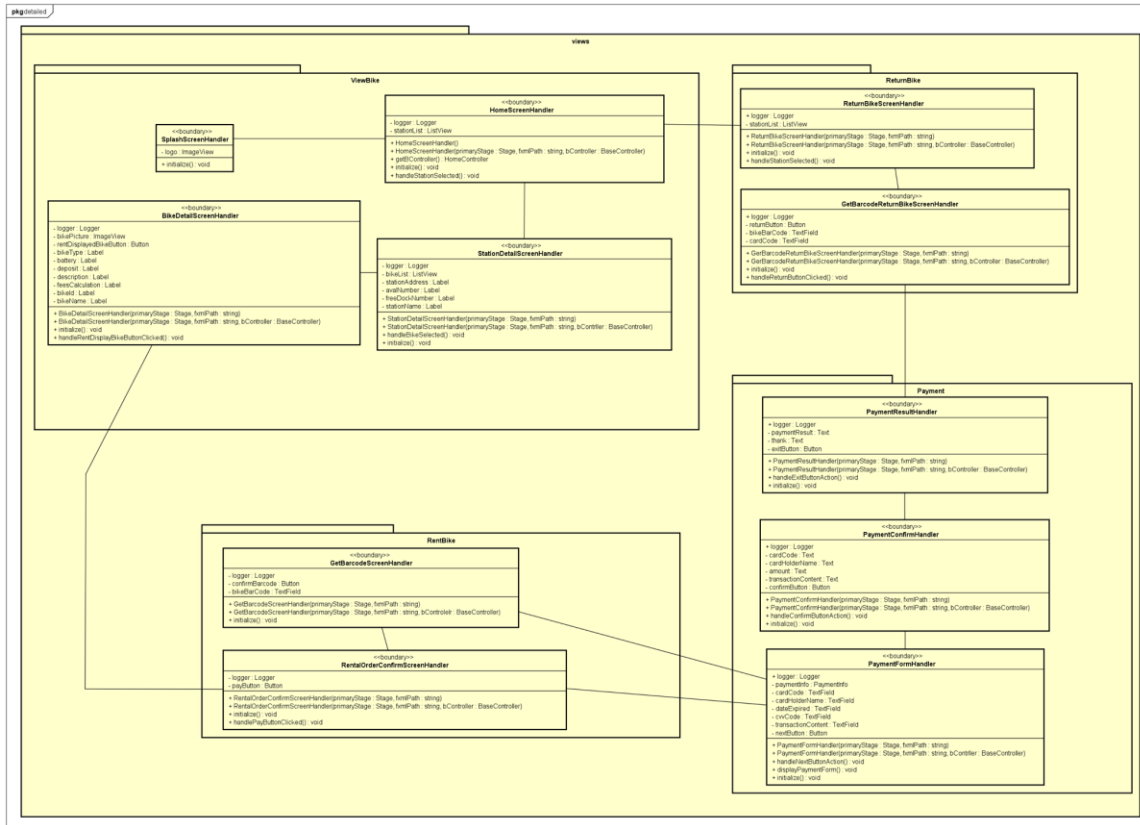
### 4.1.1 Sơ đồ chung

### 4.1.2 Sơ đồ lớp

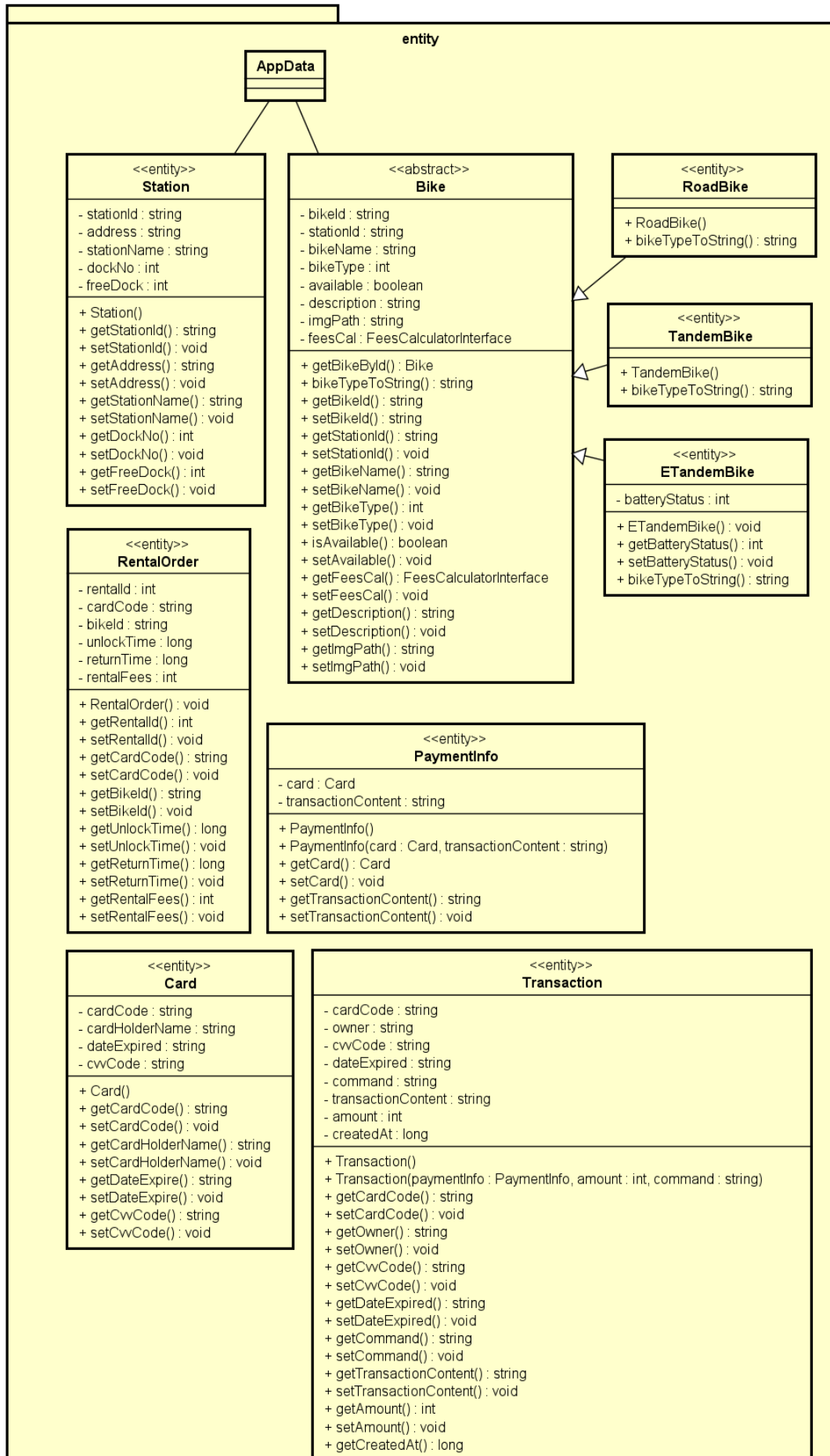


#### 4.1.2.1 Class Diagram for Package

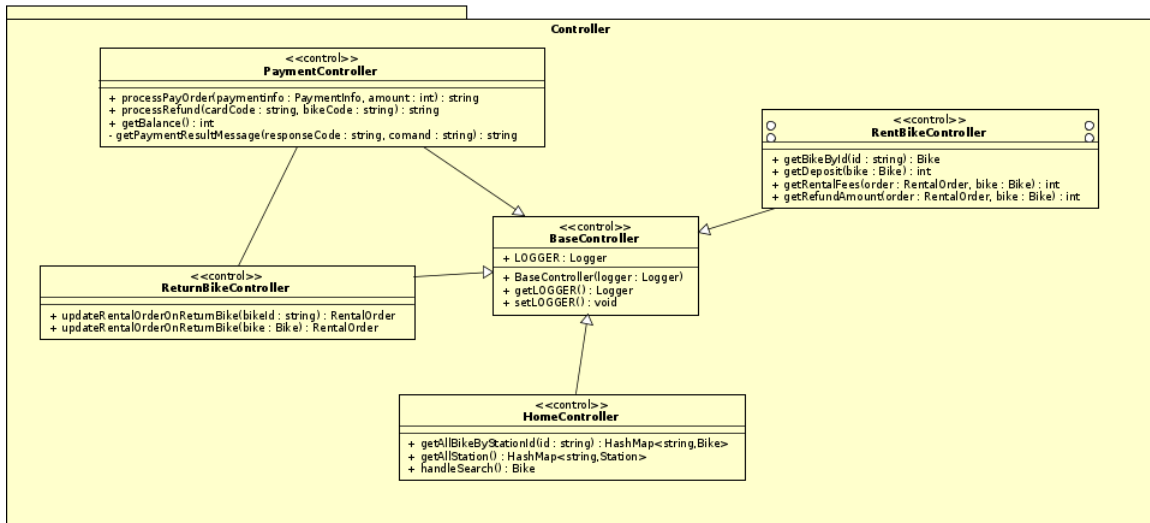
View



Entity



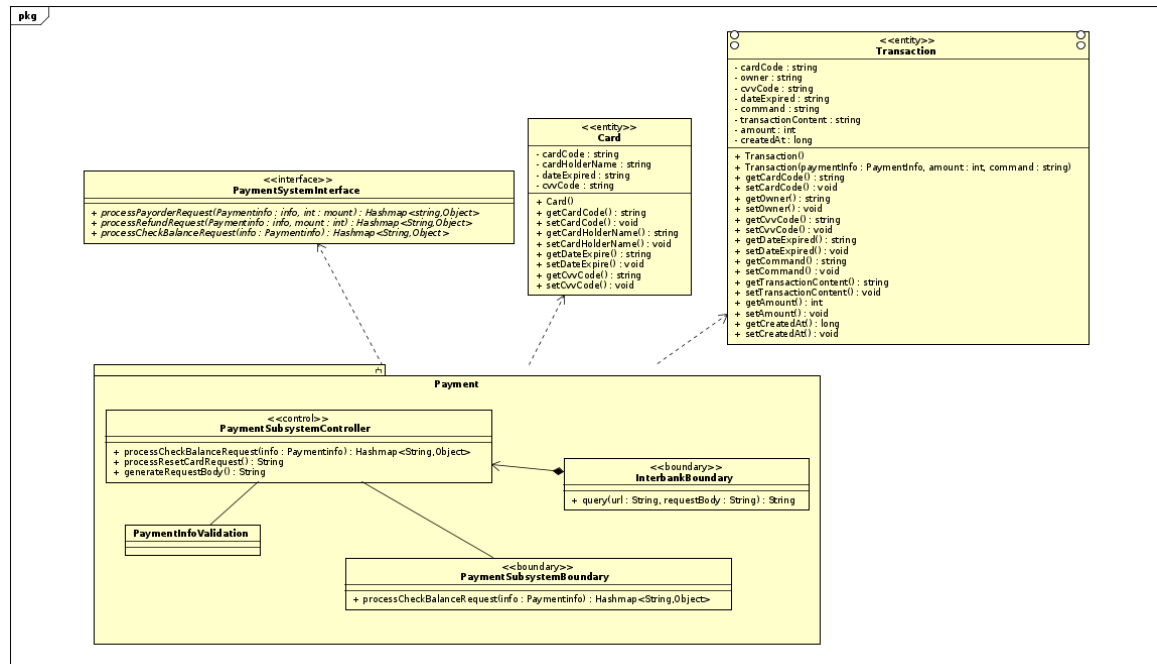
## Controller





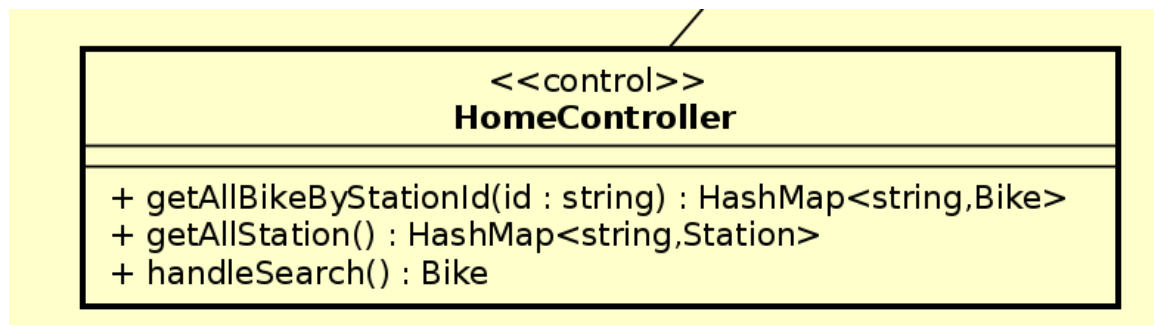
### 4.1.2.2 Class Diagram for Subsystem

#### Payment



## 4.2 Class design

### 4.2.1 Class “Homecontroller”



## Attribute

Không

## Operation

STT	Tên	Kiểu dữ liệu trả về	Mô tả
1	getAllBikeBytationId	HashMap<string, Bike>	Lấy danh sách các xe trong một bãi
2	GetAllStation	HashMap<string,Station>	lấy danh sách bãi xe
3	handleSearch	Bike	Tìm xe theo từ khóa

### Parameter

id : mã bãi xe

### Exception:

- PaymentException – nếu mã lỗi trả về đã biết
- UnrecognizedException – nếu không tìm thấy mã lỗi trả về hoặc có lỗi hệ thống

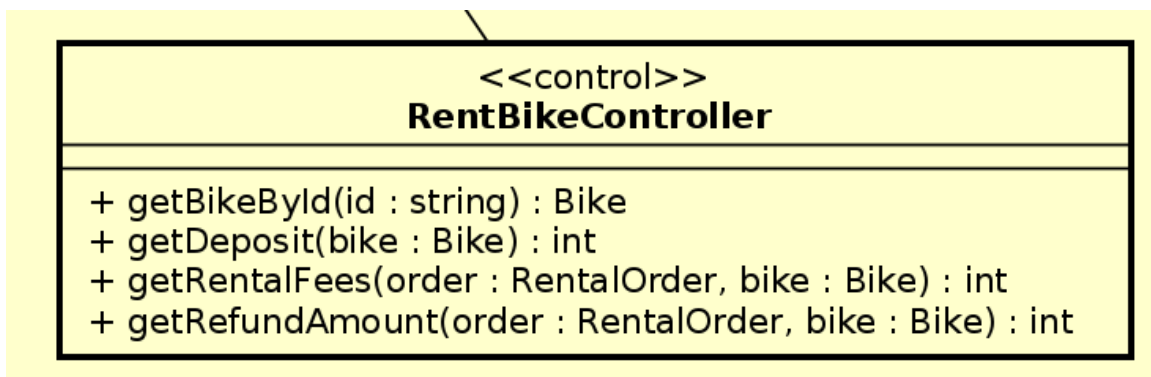
### Method

Không

### State

Không

#### 4.2.2 Class “RentbikeController”



## Attribute

Không

## Operation

STT	Tên	Kiểu dữ liệu trả về	Mô tả
1	GetBikeById	Bike	chọn xe từ mã xe
2	GetDeposit	Int	lấy tiền đặt cọc xe
3	GetRentalFee	int	Tính phí thuê
4	GetRefundAmount	Int	tính tiền còn lại

## Parameter

id : mã id xe

bike: Đối tượng xe thuê

order: hóa đơn thuê xe

## Exception:

- PaymentException – nếu mã lỗi trả về đã biết
- UnrecognizedException – nếu không tìm thấy mã lỗi trả về hoặc có lỗi hệ thống

## Method

Không

## State

Không

### 4.2.3 Class “ReturnBikeController”

<<control>> <b>ReturnBikeController</b>	
+ updateRentalOrderOnReturnBike(bikeId : string) : RentalOrder + updateRentalOrderOnReturnBike(bike : Bike) : RentalOrder	

### Attribute

Không

### Operation

STT	Tên	Kiểu dữ liệu trả về	Mô tả
1	updateRentalOrderOnReturnBike	RentalOrder	Cập nhật thông tin thuê xe

### Parameter

bikeid : mã id xe

bike: Đối tượng xe thuê

### Exception:

- PaymentException – nếu mã lỗi trả về đã biết
- UnrecognizedException – nếu không tìm thấy mã lỗi trả về hoặc có lỗi hệ thống

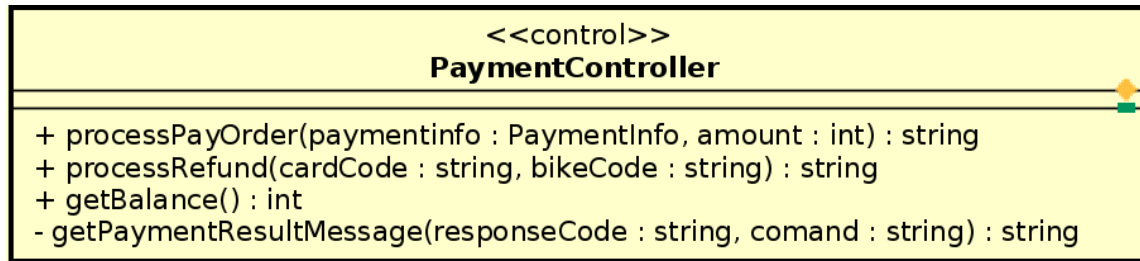
### Method

Không

### State

Không

#### 4.2.4 Class “PaymentController”



## Attribute

Không

## Operation

STT	Tên	Kiểu dữ liệu trả về	Mô tả
1	processPayOrder	String	xử lý thanh toán
2	ProcessRefund	String	xử lý hoàn lại tiền
3	getBalance	Int	
4	getPaymentResultMessage	String	trả lại kết quả thanh toán

## Parameter

amount: số tiền thanh toán

paymentInfo: thông tin thanh toán

cardCode: mã thẻ thanh toán

bikeCode: mã xe

responseCode: mã kết quả thanh toán

command: Hành động thanh toán

## Exception:

- PaymentException – nếu mã lỗi trả về đã biết
- UnrecognizedException – nếu không tìm thấy mã lỗi trả về hoặc có lỗi hệ thống

## **Method**

Không

## **State**

Không

## 5 Design Considerations

### 5.1 Coupling and Cohesion

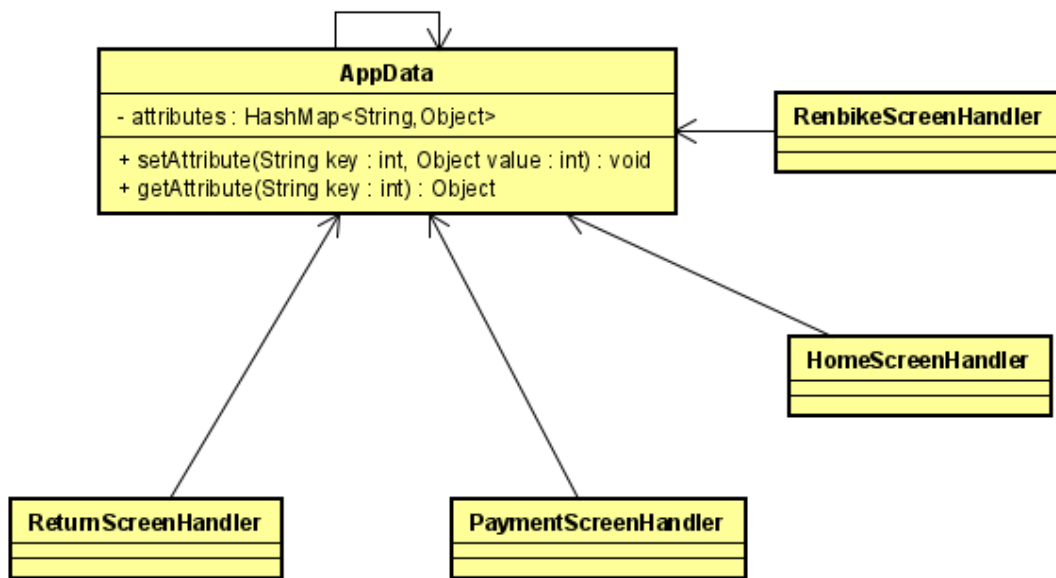
#### 5.1.1 Coupling

##### 5.1.1.1 Content coupling

Thiết kế hiện tại không vi phạm content coupling.

##### 5.1.1.2 Common coupling

Trong thiết kế cũ, use case chia sẻ dữ liệu với nhau thông qua lớp *AppData*. Lớp *AppData* này thực chất có một thuộc tính là đối tượng *HashMap*, các lớp khác có thể truy cập vào thông qua một getter public, sau đó có thể đẩy dữ liệu vào hoặc lấy dữ liệu ra thông qua key.



Thiết kế trên vi phạm common coupling vì tất cả các thành phần trong chương trình đều có thể truy cập, chỉnh sửa thông tin trong class *AppData*.

Với thiết kế mới, lớp *AppData* được bỏ đi. Thay vào đó, tất cả dữ liệu đều thuộc về một class nào đó. Dữ liệu được truyền giữa các class khi khởi tạo đối tượng của class đó.

Dưới đây là một ví dụ trong class `views.handler.payment.PaymentFormScreenHandler`. Ở thiết kế cũ, sau khi lấy thông tin thanh toán từ người dùng, thông tin này được đẩy vào class `AppData` chứa dữ liệu dùng chung. Ở thiết kế mới, dữ liệu này được đóng gói trong class `PaymentConfirmScreenHandler` và được truyền thông qua phương thức khởi tạo.

```

100 // validate user input information
101 // if information is not valid, create an alert to notify to the user
102 // in contrast, switch to the confirm screen
103 if (!PaymentInfoValidation.validatePaymentInfor(paymentInfo)) {
104     logger.info("Invalid payment info");
105     BaseScreenHandler.createAlert(AlertType.ERROR, "Invalid input",
106         "Check the payment information and try again!");
107 } else {
108     logger.info("payment info is valid");
109     AppData.setAttribute("payment info", paymentInfo);
110
111     PaymentConfirmHandler confirmPaymentHandler;
112     try {
113         confirmPaymentHandler = new PaymentConfirmHandler(this.getPrimaryStage(),
114             Configs.PAYMENT_CONFIRM_SCREEN);
115         confirmPaymentHandler.setHomeScreenHandler(this.getHomeScreenHandler());
116         confirmPaymentHandler.setPreviousHandler(this);
117         confirmPaymentHandler.show();
118     } catch (IOException ex) {
119         // TODO Auto-generated catch block
120         logger.info("Error occurred! " + ex.getMessage());
121     }
122 }

```

**Figure 1 Thiết kế cũ vi phạm common coupling**

```

88 // validate user input information
89 // if information is not valid, create an alert to notify to the user
90 // in contrast, switch to the confirm screen
91 if (!PaymentInfoValidation.validatePaymentInfor(paymentInfo)) {
92     logger.info("Invalid payment info");
93     BaseScreenHandler.createAlert(AlertType.ERROR, "Invalid input",
94         "Check the payment information and try again!");
95 } else {
96     logger.info("payment info is valid");
97     Invoice invoice = new Invoice(order, paymentInfo);
98
99     PaymentConfirmHandler confirmPaymentHandler;
100     try {
101         confirmPaymentHandler = new PaymentConfirmHandler(this.getPrimaryStage(),
102             Configs.PAYMENT_CONFIRM_SCREEN, invoice);
103         confirmPaymentHandler.setHomeScreenHandler(this.getHomeScreenHandler());
104         confirmPaymentHandler.setPreviousHandler(this);
105         confirmPaymentHandler.show();
106     } catch (IOException ex) {
107         // TODO Auto-generated catch block
108         logger.info("Error occurred! " + ex.getMessage());
109     }
110 }

```

**Figure 2 Thiết kế mới đã bỏ lớp AppData**

### 5.1.1.3 Control coupling



#### **5.1.1.4 Stamp coupling**

#### **5.1.1.5 Data coupling**

### **5.1.2 Cohesion**

## **5.2 Design Principles**

### **5.2.1 S – single responsibility**

Các lớp đã thỏa mãn mỗi lớp chỉ có một chức năng duy nhất.

### **5.2.2 O – open for extend, close for modify**

- Tính phí thuê xe được thực hiện bởi class *RentalFeesCalculator01*, class này implement *RentalFeesCalculator*. Với cách thiết kế này, nếu có một cách tính phí thuê mới, ta chỉ cần implement lại interface *RentalFeesCalculator*.
- Use case thanh toán được thiết kế thành một subsystem, implement cho interface *PaymentSystemInterface*. Với cách thiết kế này, khi ứng dụng cho phép một cách thanh toán khác, người lập trình chỉ cần xây dựng thêm một subsystem và implement các phương thức *pay order*, *refund...* của interface *PaymentSystemInterface*.

### **5.2.3 Liskov substitution**

Nguyên tắc này đảm bảo các instance của lớp con có thể thay thế lớp cha ở mọi trường hợp.

Trong thiết kế của project, các controller đc extends từ *BaseController* và các handler được extends từ *BaseScreenHandler* đều tuân thủ nguyên tắc này.

### **5.2.4 Interface segregation**

### **5.2.5 Dependency inversion**

## **5.3 Design Patterns**

### **5.3.1 Singleton**

Mẫu thiết Singleton được áp dụng để đảm bảo một lớp chỉ có duy nhất một thể hiện (instance). Cụ thể, có 1 lớp được áp dụng mẫu thiết kế singleton: lớp *DBConnection* có

nhiệm vụ kết nối và duy trì kết nối tới database. Cần áp dụng singleton cho lớp này vì chỉ cần duy nhất một kết nối tới cơ sở dữ liệu khi chạy ứng dụng.