

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

School of Information and communications technology

## Software Design Document

### Bike Rental Management

Subject: Thiết kế và xây dựng phần mềm

Nhóm 5

Nguyễn Hồng Quốc Khánh: 20170082

Nguyễn Văn Huy: 20170080

Trần Đình Hùng: 20170078

Trương Quang Khánh: 20170083



# Table of Contents

Table of Contents .....	1
1 Introduction .....	3
1.1 Objective.....	3
1.2 Scope .....	3
1.3 Glossary .....	3
1.4 References .....	3
2 Overall Description .....	5
2.1 General Overview.....	5
2.2 Assumptions/Constraints/Risks .....	8
2.2.1 Assumptions.....	8
2.2.2 Constraints .....	9
2.2.3 Risks.....	9
3 System Architecture and Architecture Design .....	10
3.1 Architectural Patterns .....	10
3.2 Interaction Diagrams .....	10
3.3 Analysis Class Diagrams .....	13
3.4 Unified Analysis Class Diagram .....	15
3.5 Security Software Architecture .....	15
4 Detailed Design .....	16
4.1 User Interface Design .....	16
4.1.1 Screen Configuration Standardization .....	16
4.1.2 Screen Transition Diagrams.....	18
4.1.3 Screen Specifications .....	19
4.2 Data Modeling .....	32
4.2.1 Conceptual Data Modeling .....	32
4.2.2 Database Design.....	32

4.3	Non-Database Management System Files .....	36
4.4	Class Design .....	36
4.4.1	General Class Diagram .....	36
4.4.2	Class Diagrams .....	38
4.4.3	Class Design.....	42
5	Design Considerations.....	54
5.1	Goals and Guidelines .....	54
5.2	Architectural Strategies .....	54
5.3	Coupling and Cohesion .....	54
5.4	Design Principles .....	56
5.5	Design Patterns .....	58

# 1 Introduction

## 1.1 Objective

Tài liệu này mô tả phần thiết kế phần mềm sau bước phân tích ở tài liệu srs. Tài liệu được sử dụng cho programmers, testers, maintainers, systems integrators, vv. Nó bao gồm việc thiết kế chi tiết cho kiến trúc, thiết kế giao diện và thiết kế lớp cho từng chức năng của hệ thống, cũng như việc thiết kế cơ sở dữ liệu của cả hệ thống để từ đó người đọc sẽ có cái nhìn rõ ràng hơn về phần mềm cần xây dựng và nó sẽ là tài liệu chính thức để từ đó những người xây dựng phần mềm có thể xây dựng nên phần mềm dựa vào tài liệu này.

## 1.2 Scope

Phần mềm có tên: Eco Bike Rental

Phần mềm sẽ cung cấp cho người dùng một giao diện để thực hiện các chức năng: xem thông chi tiết bãi xe, xem thông tin chi tiết xe trong bãi, thuê xe và trả xe.

## 1.3 Glossary

Bảng chú thích các thuật ngữ:

STT	Thuật ngữ	Giải thích	Ví dụ	Ghi chú
1	Button	Các nút bấm trong phần mềm		
2	Cơ sở dữ liệu	Là nơi lưu trữ các thông tin về dữ liệu cần thiết: như danh sách bãi xe, danh sách xe trong bãi, lịch sử các giao dịch		

3	List View	Một danh sách đọc ra từ cơ sở dữ liệu, có thể ấn chọn vào phần tử trong danh sách		
4	Text Field	Một ô trống, thường dùng cho người dùng có thể điền các thông tin cần thiết		

#### **1.4 References**

Tài liệu tham khảo bao gồm: javaFX doc, một số design pattern thông dụng, các tài liệu được cung cấp trong học phần thiết kế và xây dựng phần mềm.

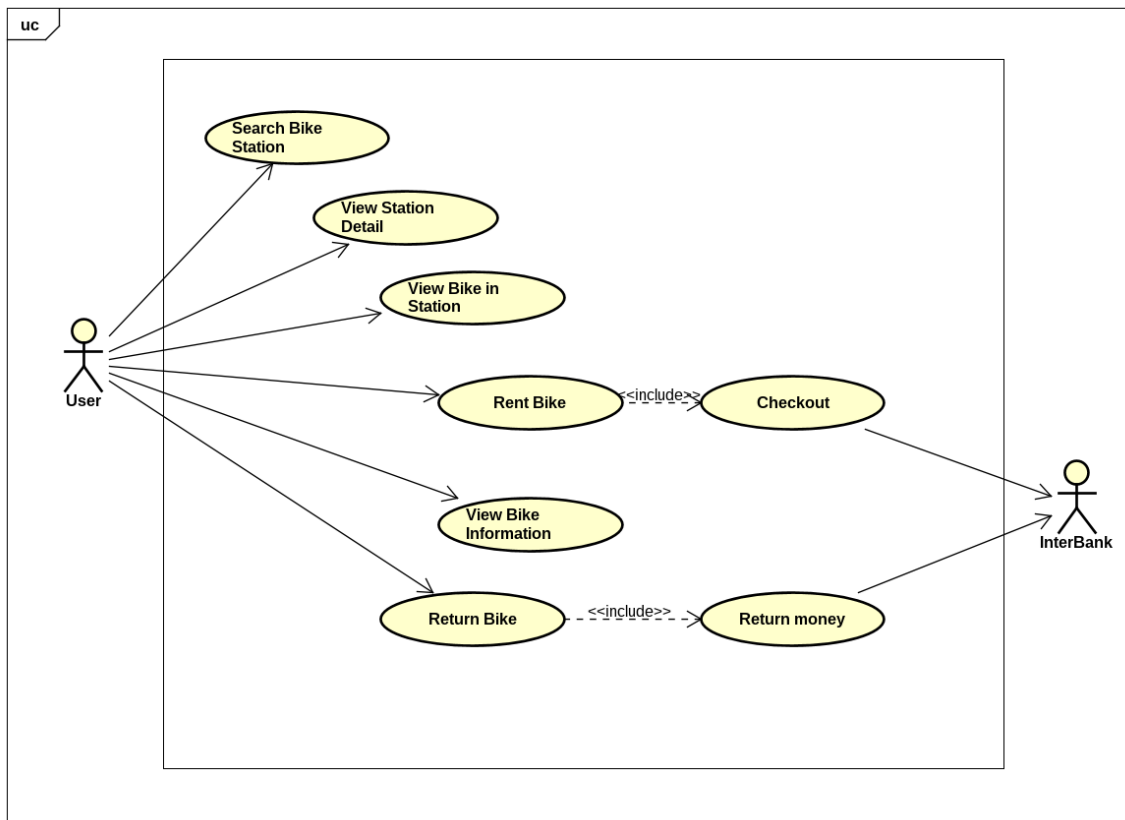
## 2 Overall Description

### 2.1 General Overview

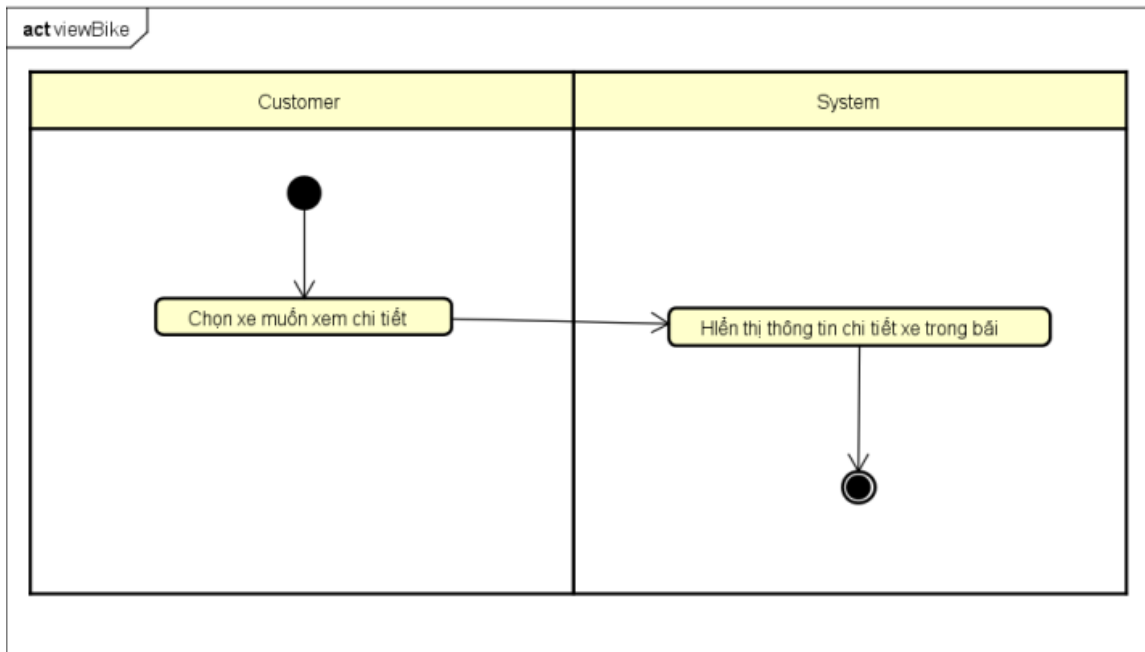
Ngày nay, nhu cầu tập thể dục thể thao nâng cao sức khỏe của con người ngày càng gia tăng. Một trong những hình thức tập luyện vừa hiệu quả vừa mang lại sự thư giãn tinh thần cao đó chính là đi xe đạp. Xuất phát từ nhu cầu thực tế này, phần mềm quản lý việc cho thuê trả xe ra đời.

Thông qua việc phân tích yêu cầu mà hệ thống đặt ra, nhóm đã thống nhất một số diagram tổng quan cho hệ thống bao gồm:

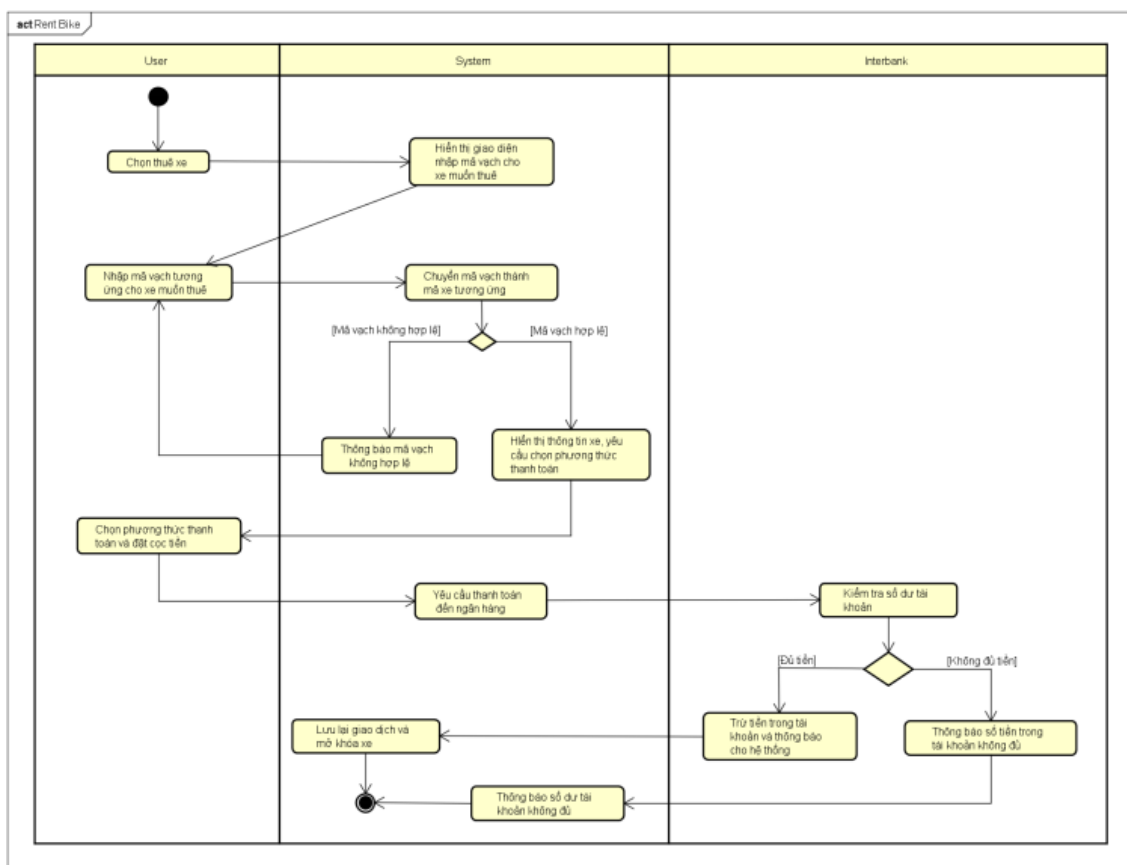
Sơ đồ UC tổng quan hệ thống:



Biểu đồ hoạt động usecase xem chi tiết xe trong bãi

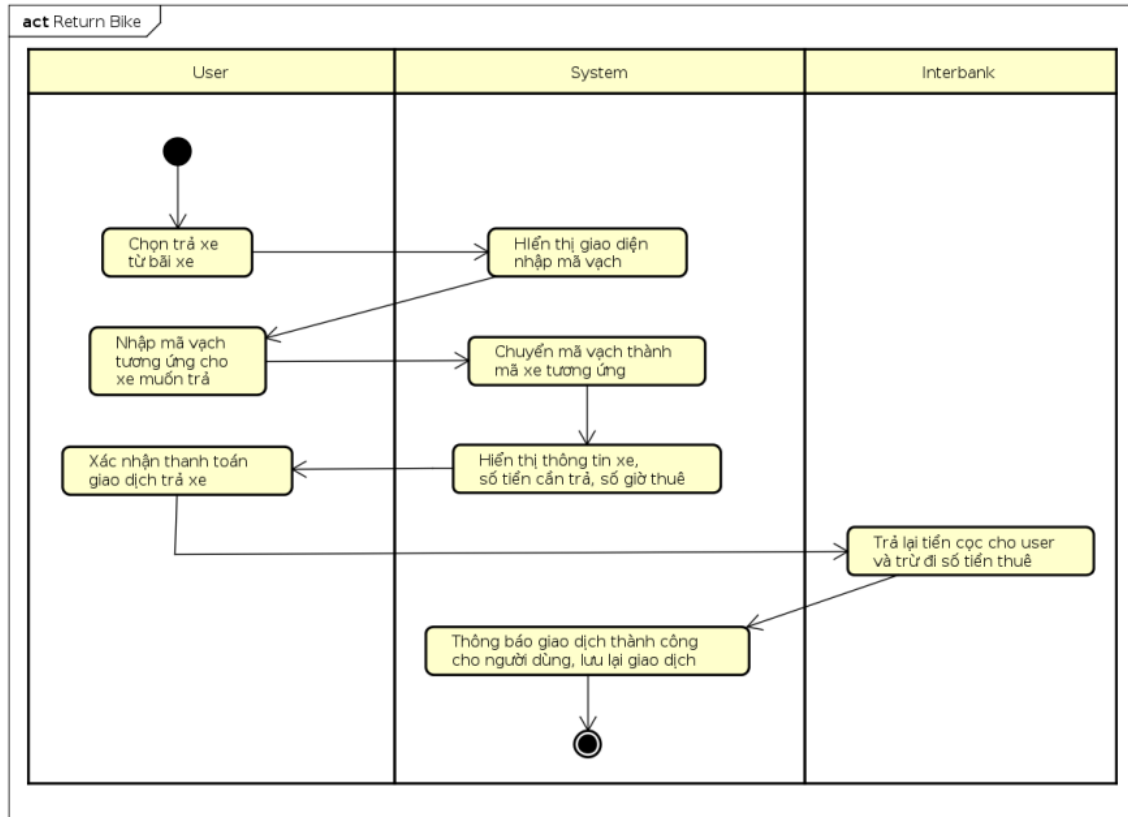


Biểu đồ hoạt động usecase thuê xe

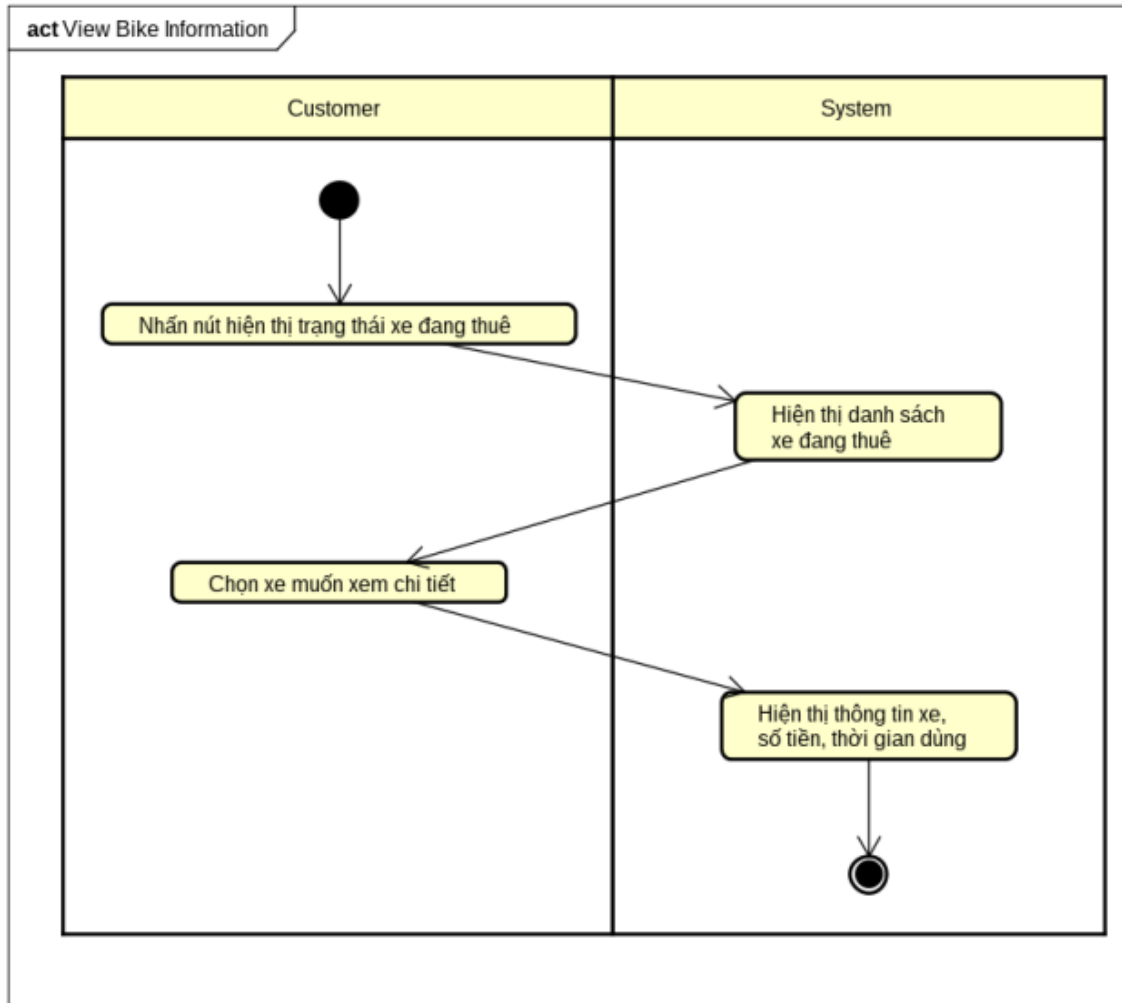




## Biểu đồ hoạt động usecase trả xe



## Biểu đồ hoạt động usecase xem xe đang thuê



## 2.2 Assumptions/Constraints/Risks

### 2.2.1 Assumptions

Một số yêu cầu phát sinh:

- *Thêm một loại xe mới:* Xe đạp đôi điện, giống xe đạp đôi thường nhưng có motor điện giúp đạp xe nhanh hơn. Cách tính giá xe đạp đôi điện đắt gấp 02 lần cách tính giá xe đạp đơn thường.
- *Thay đổi cách tính giá thuê xe*
- *Thêm một tính năng mới:* Cho phép khách hàng tạm dừng thời gian thuê xe. Khách hàng có thể tạm dừng thuê xe mọi lúc bằng cách khóa xe. Lúc này, hệ thống sẽ tạm dừng tính thời gian thuê xe. Hệ thống sẽ tự động tiếp tục tính thời gian thuê xen ngay khi khách hàng mở khóa xe

### 2.2.2 Constraints

Một số ràng buộc để có thể chạy được phần mềm bao gồm:

- Phần cứng: máy tính pc hoặc máy tính xách tay, hệ điều hành window hoặc ubuntu đều được
- Phần mềm:

1, Có máy ảo java

2, IDE intelli

3, JavaFX 11, Java 15, có config VM option.

4, Thư viện cần thiết: mysql-connector-java-5.1.49, json-20201115.jar, junit-5, jupyter:5.4.2, org.apache.httpcomponents:httpclient:4.3-alpha1

5, Database: mysql server, mysql workbench

### 2.2.3 Risks

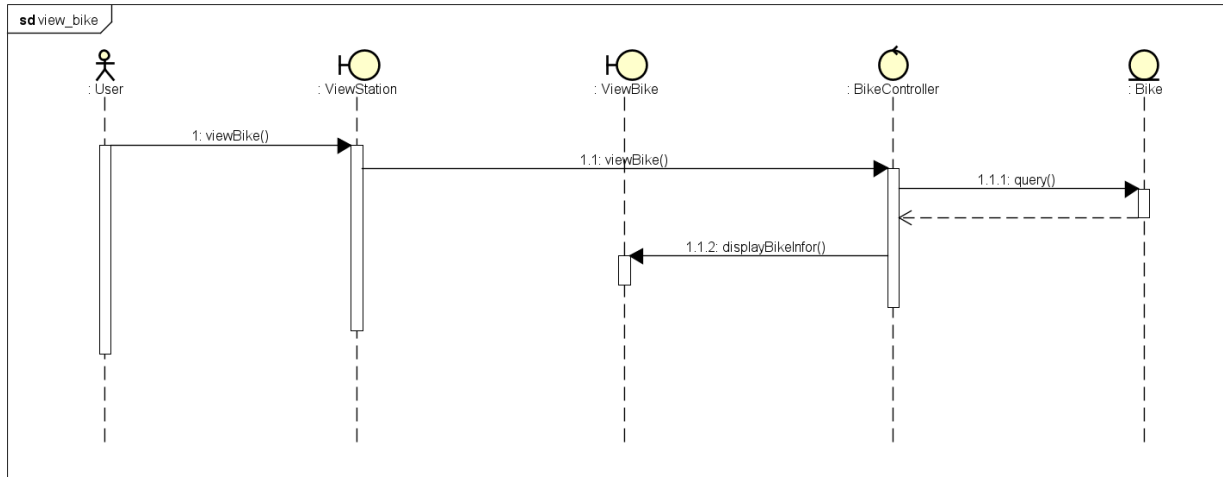
Rủi ro có thể có của phần mềm: bảo mật thấp nên dữ liệu trong database dễ bị lấy mất. Tuy nhiên do phần mềm mang tính chất demo là chính nên vấn đề này không mang lại thiệt hại quá lớn cho cá nhân hay tổ chức nào.

### 3 System Architecture and Architecture Design

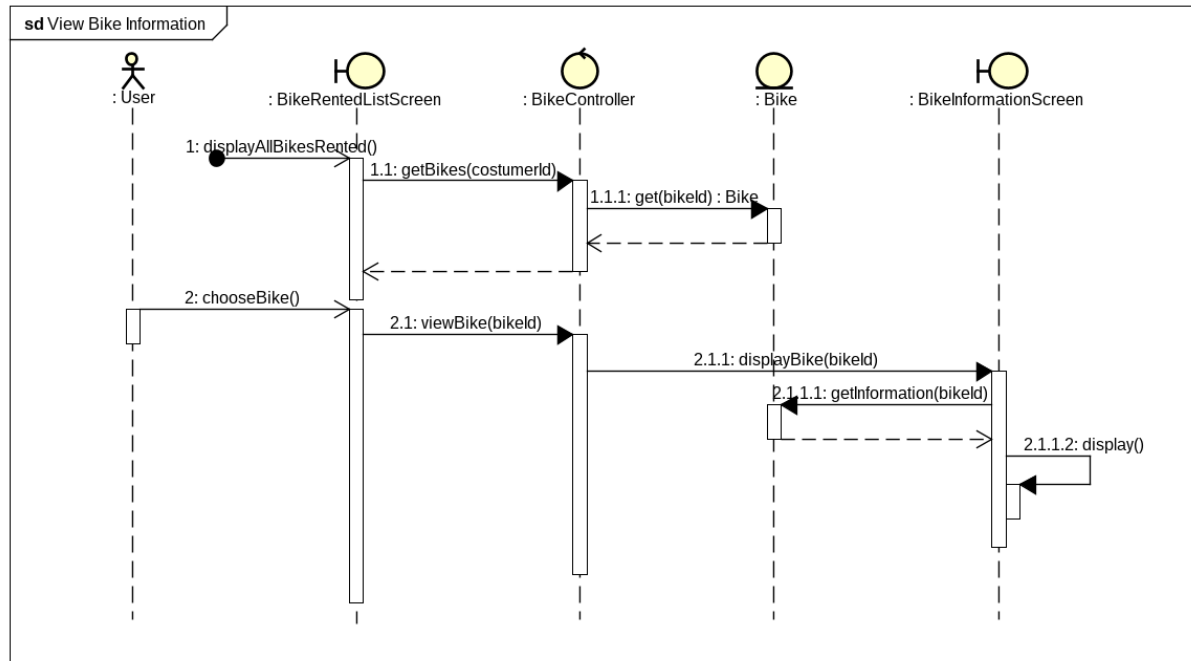
#### 3.1 Architectural Patterns

#### 3.2 Interaction Diagrams

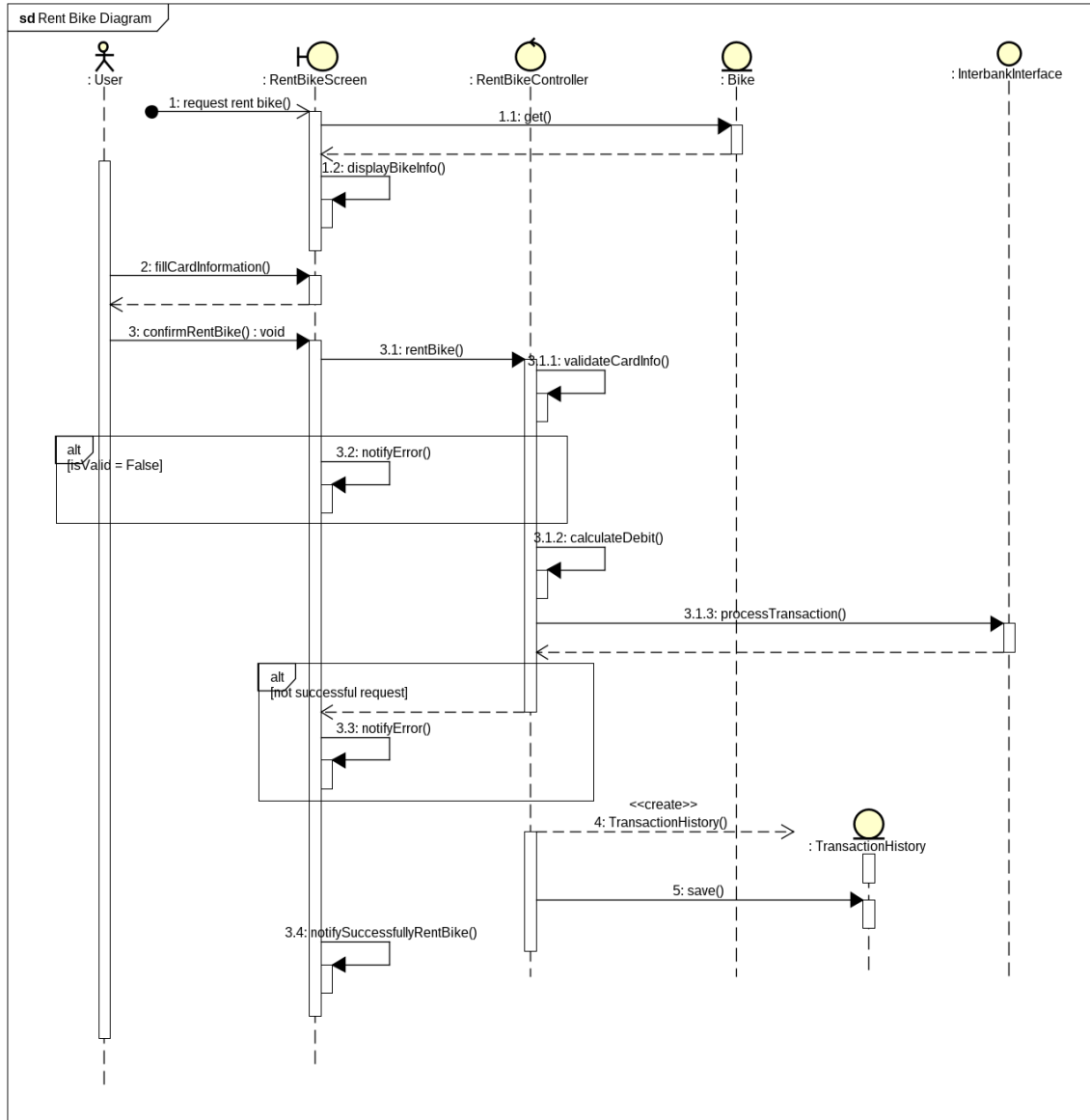
Đây là biểu đồ cho use case View Bike



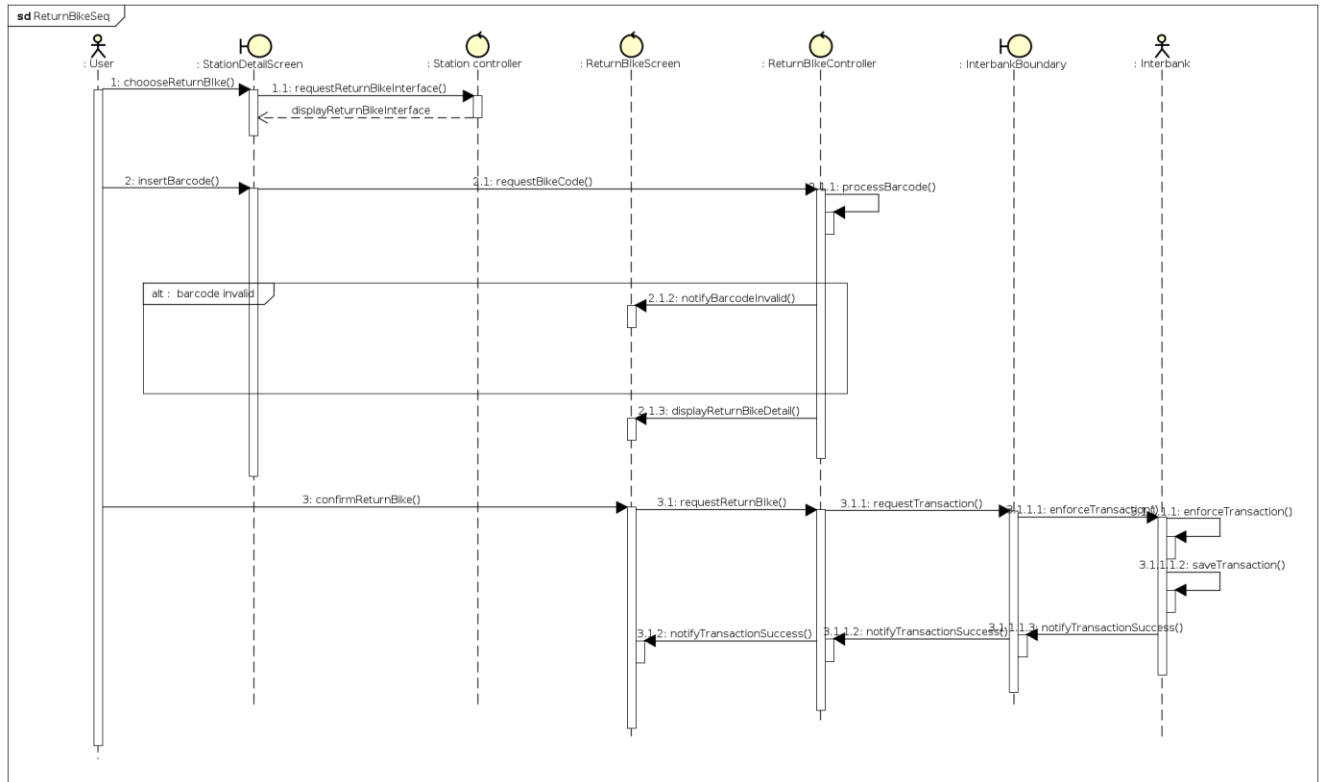
Đây là biểu đồ cho use case View Rented List Bike



Dưới đây là biểu đồ sequence cho use case rent bike.

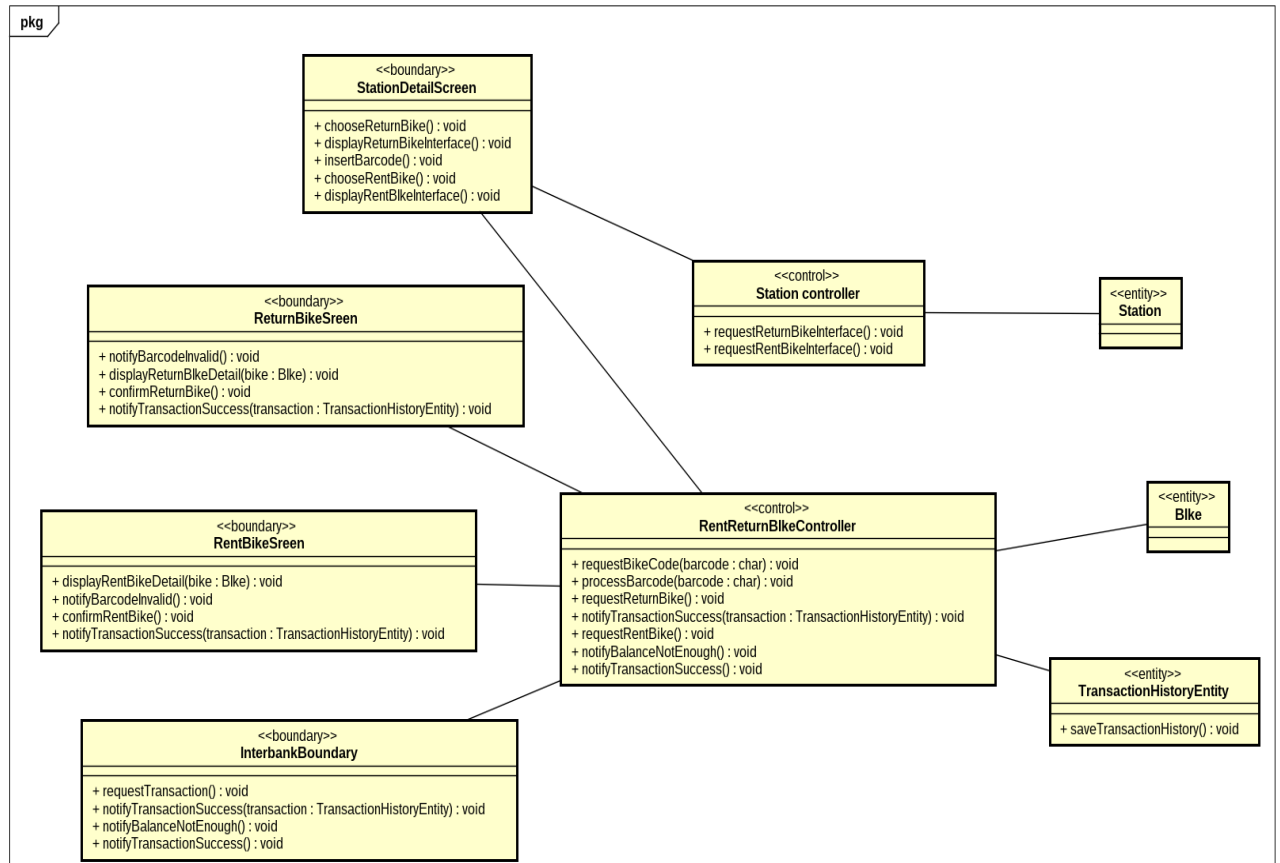


Đây là biểu đồ cho use case return bike

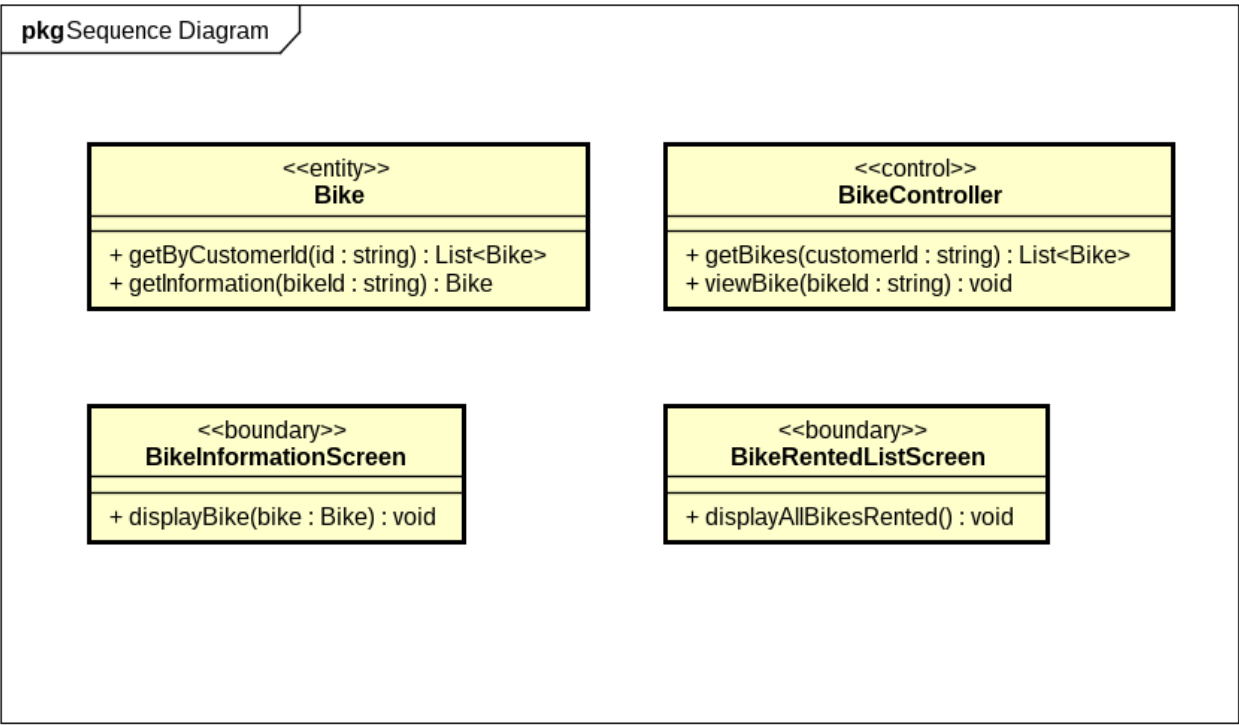


### 3.3 Analysis Class Diagrams

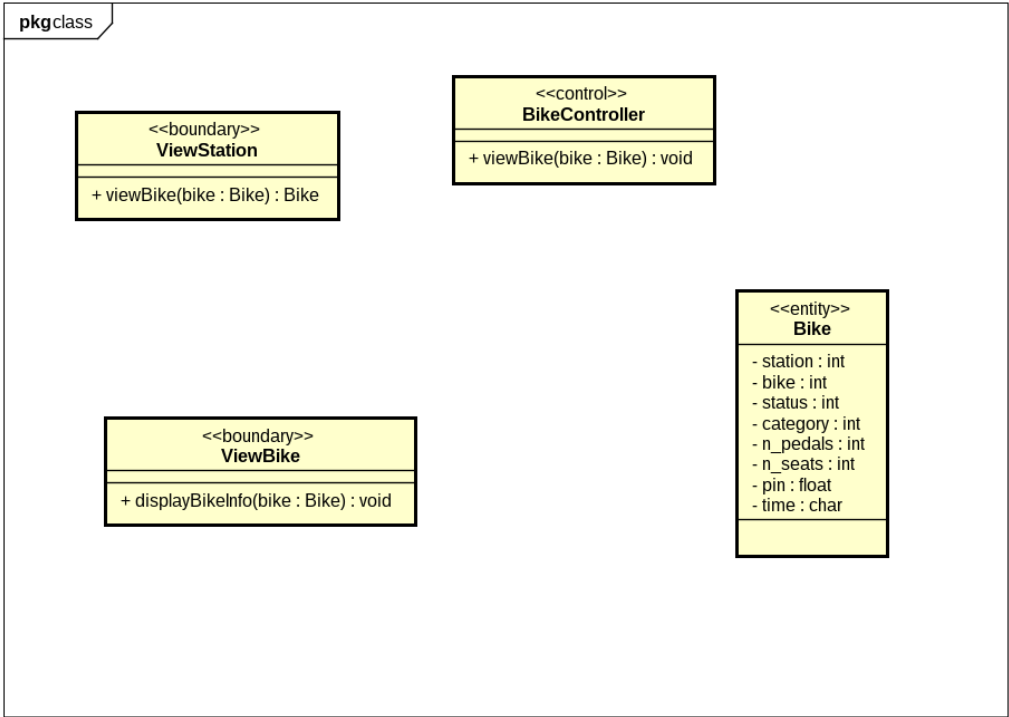
#### 3.3.1 Biểu đồ lớp cho use case rent bike và return bike



3.3.2 Biểu đồ lớp cho use case "view rented bike"



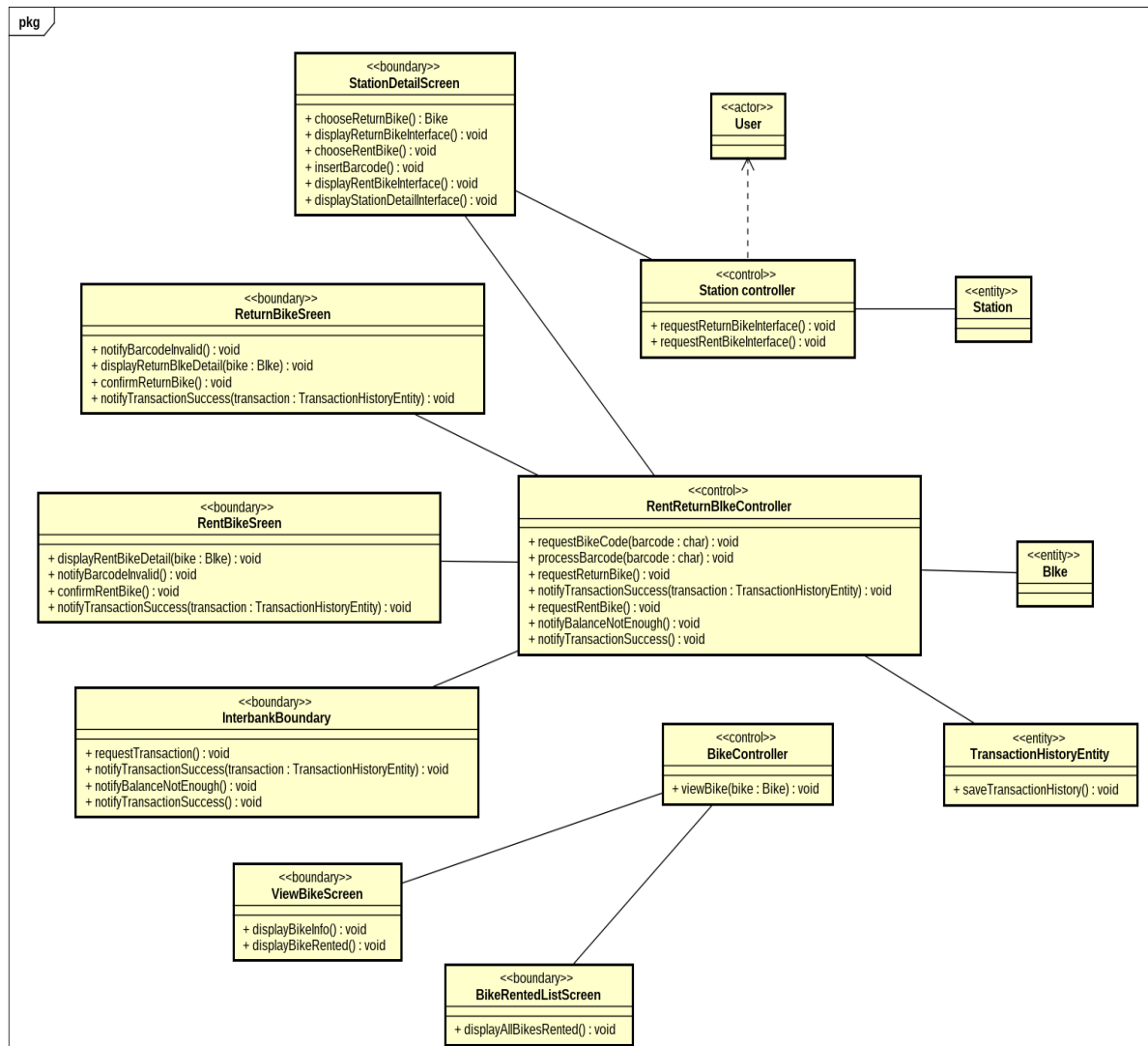
3.3.3 Biểu đồ lớp cho use case "View Bike Information"





### 3.4 Unified Analysis Class Diagram

## Biểu đồ lớp phân tích cho hệ thống



### 3.5 Security Software Architecture

Nếu sau này hệ thống mở rộng thêm chức năng đăng nhập cho người dùng thì người dùng phải nhập mật khẩu với mỗi role tương ứng (như khách hàng hay quản lý bãi xe).

## 4 Detailed Design

### 4.1 User Interface Design

#### 4.1.1 Screen Configuration Standardization

##### *Display*

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

Độ phân giải: 800 x 600 pixels

##### *Screen*

Vị trí của cửa button: ở trên thanh header phía trên hoặc ở phần trung tâm màn hình, tùy thuộc vào cấu hình màn hình. Riêng button Back luôn nằm ở góc trên bên phải màn hình.

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.

Template: tất cả các màn hình đều có một header và footer màu xanh, một ảnh logo và một ảnh minh họa bãi xe ở phía bên trái.

##### *Control*

Kích thước text: Color: tùy thuộc vào button chứa nó và tính chất của text.

Xử lý check input: Trước tiên 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 đầu tiên(Home Screen) sẽ xuất hiện.

Thứ tự xuất hiện các màn hình trong hệ thống: (Do tùy thuộc vào người dùng sử dụng chức năng nào của app và click vào button tương ứng nào nên thứ tự này chỉ là tương đối).

1. Home screen
2. Station detail screen – Màn hình xem chi tiết 1 bãi xe
3. View bike screen – Màn hình xem chi tiết xe trong bãi
4. Rent bike screen – Màn hình hiển thị thông tin xe muốn thuê và các trường thông tin để nhập tài khoản thẻ ngân hàng của người dùng
5. Notification screen – Màn hình hiển thị kết quả của việc mượn và trả xe
6. Bike rented list screen – Màn hình hiển thị danh sách các xe đang thuê của người dùng
7. Rented bike screen – Màn hình hiển thị thông tin chi tiết của xe đang thuê, bao gồm thông tin xe và giờ mượn, giá thuê tính đến thời điểm hiện tại
8. Return bike screen – Màn hình hiển thị các trường thông tin để nhập tài khoản thẻ ngân hàng của người dùng.

### ***Nhập input từ bàn phím***

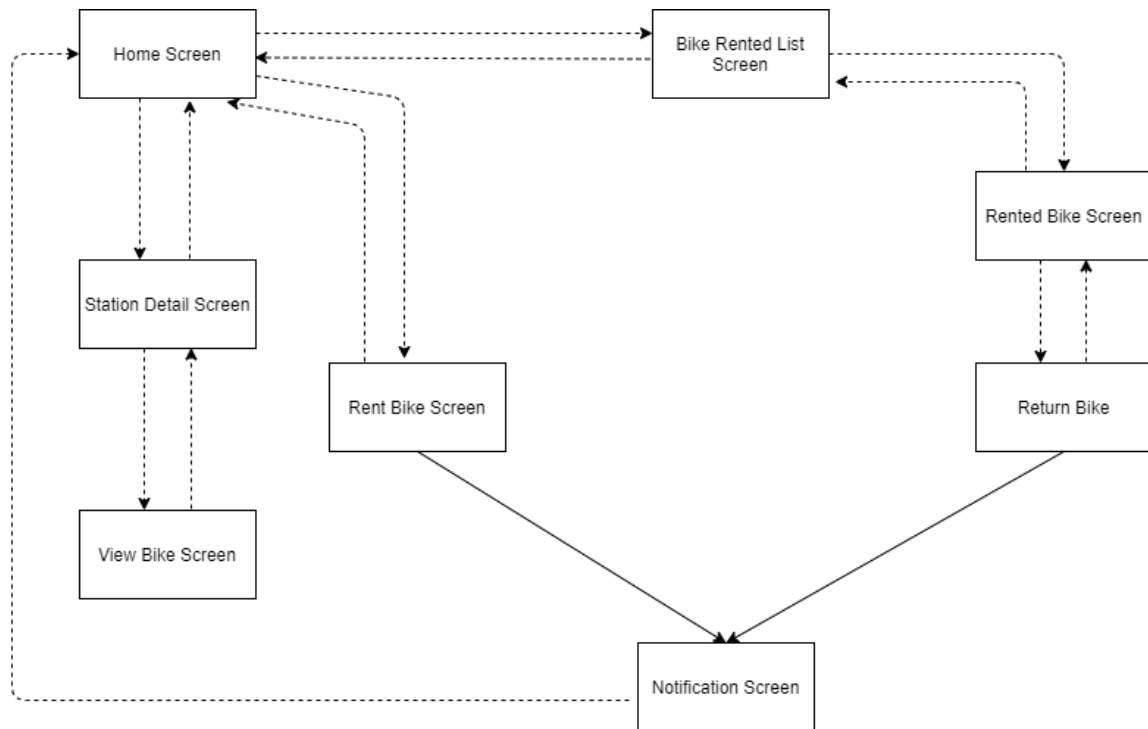
Sẽ không có phím tắt. Có các button quay lại để quay lại các màn hình trước đó. Ngoài ra button “X” nằm ở thanh tiêu đề bên phải để đóng screen như thông thường.

### ***Error***

Một thông điệp sẽ được hiện lên để thông báo cho người dùng biết vấn đề đang gặp phải là gì. Thông điệp có dạng text màu đỏ.

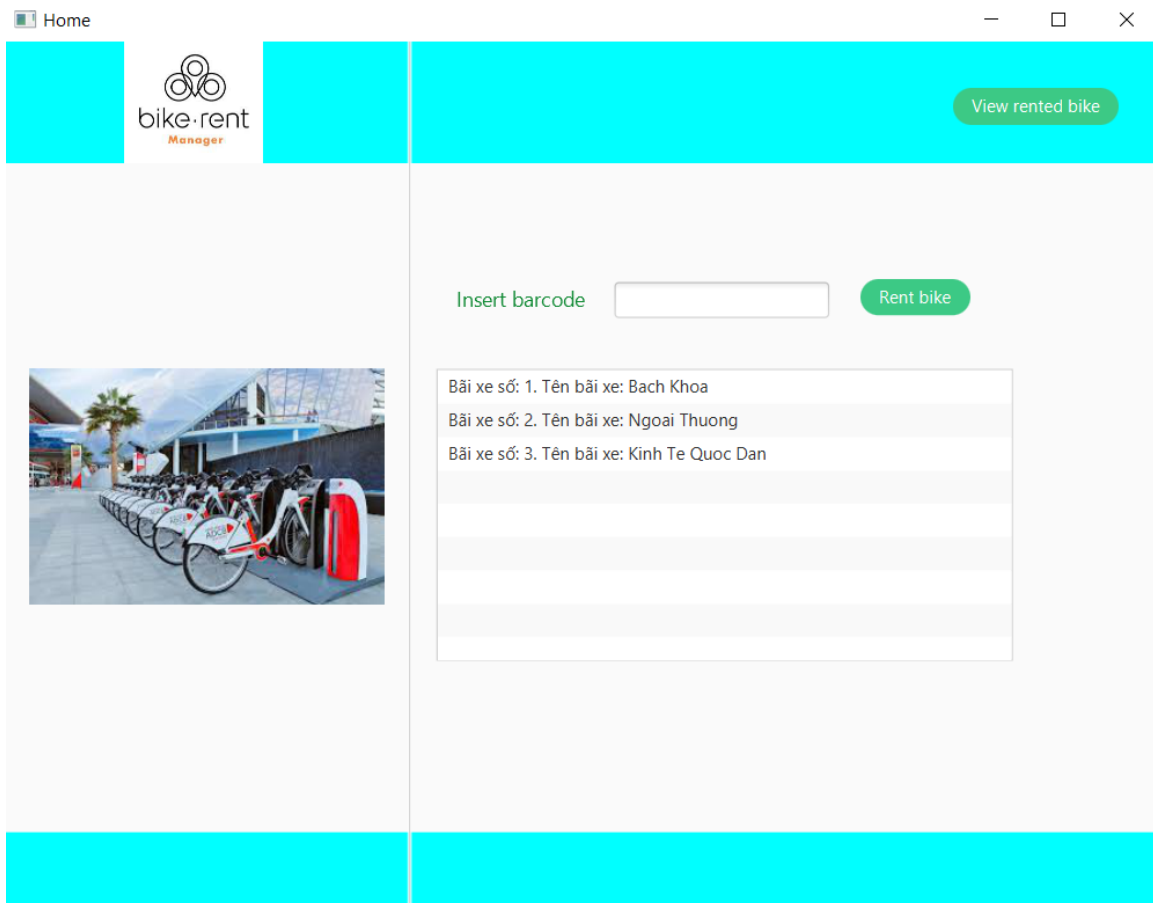
#### 4.1.2 Screen Transition Diagrams

Sơ đồ dịch chuyển giữa các màn hình:



### 4.1.3 Screen Specifications

#### 4.1.3.1 Home Screen:



Đặc tả màn hình:

Điều khiển	Thao tác	Chức năng
Rent bike button	Click	Nhận barcode được nhập vào, xử lý và di chuyển đến rent bike screen(nếu barcode hợp lệ) hoặc thông báo barcode không hợp lệ
Text field	Điền barcode của xe	Nhận vào barcode của xe
View Rented Bike button	Click	Di chuyển đến bike rented list screen
List view	Click	Hiển thị danh sách bãi xe, click vào bãi xe nào thì di chuyển đến Station Detail


		Screen của bãi xe tương ứng
--	--	-----------------------------

Định nghĩa các trường thuộc tính:


Tên	Kích thước (bytes)	Loại	Thuộc tính	Lưu ý
Insert barcode		Text		Text theo chuẩn đã quy định
List view				Gồm list các bãi xe đọc ra từ cơ sở dữ liệu

#### 4.1.3.2 Station Detail Screen:

Station detail



Back



Xe đạp số 1.Tên xe: No\_001. Barcode: 1

Xe đạp số 2.Tên xe: No\_002. Barcode: 2

Xe đạp số 3.Tên xe: No\_003. Barcode: 3

Xe đạp số 4.Tên xe: No\_004. Barcode: 4

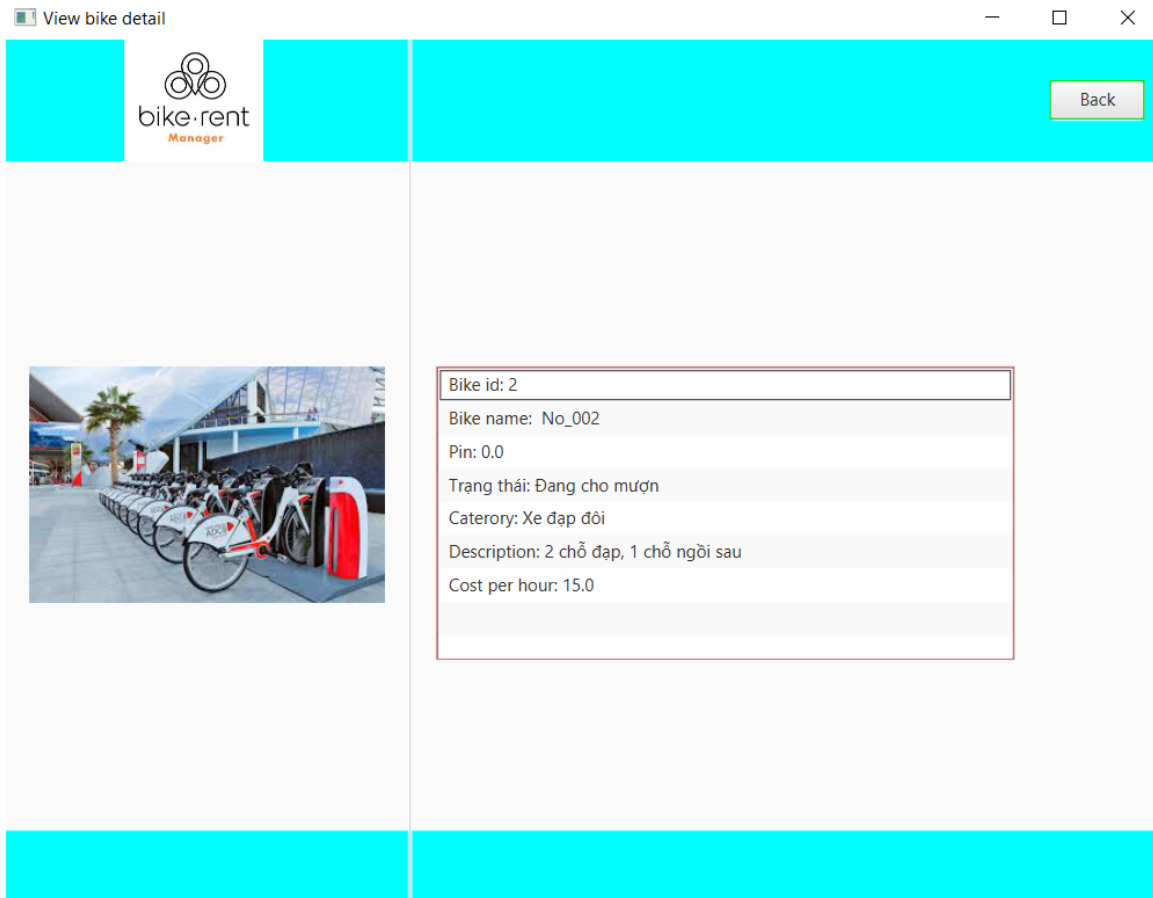
Đặc tả màn hình:

Điều khiển	Thao tác	Chức năng
List view	Click	Hiển thị danh sách xe trong bãi , click vào xe nào thì di chuyển đến View Bike Screen của xe tương ứng
Back button	Click	Di chuyển trở lại Home Screen

Định nghĩa các trường thuộc tính:

Tên	Kích thước (bytes)	Loại	Thuộc tính	Lưu ý
stationID		int		Trường thuộc tính ở trong code, không hiển thị lên màn hình
List view				Gồm list các xe đọc ra từ cơ sở dữ liệu

#### 4.1.3.3 View Bike Screen:



Đặc tả màn hình:

Điều khiển	Thao tác	Chức năng
List view	Initial	Hiển thị các thuộc tính của xe đọc được từ cơ sở dữ liệu
Back button	Click	Di chuyển trở lại Station Detail Screen




Định nghĩa các trường thuộc tính:


Tên	Kích thước (bytes)	Loại	Thuộc tính	Lưu ý
stationID		int		Trường thuộc tính ở trong code, không hiển thị lên màn hình
bikeID		int		Trường thuộc tính ở trong code, không hiển thị lên màn hình

#### 4.1.3.4 Rent Bike Screen:

Rent bike



Back



Bike id: 1

Bike name: No\_001

Pin: 0.0

Trạng thái: Đang cho mượn

Caterory: Xe đạp đơn

Description: 1 chỗ đạp, 1 chỗ ngồi sau

Cost per hour: 10.0

Card code118131\_group5\_2020

Owner296

Cvv CodeGroup 5

Date Expired1125

Confirm rent bike

Valid infomation

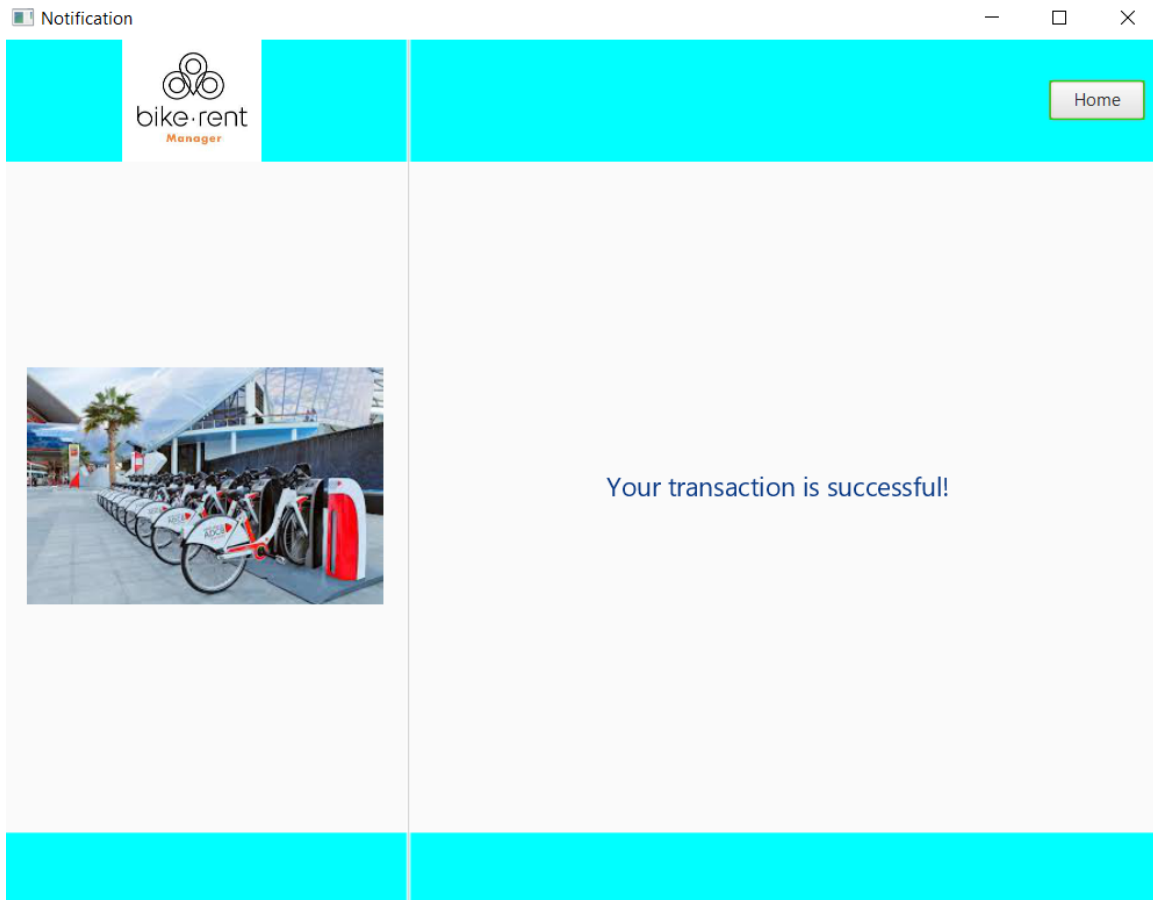
Đặc tả màn hình:

Điều khiển	Thao tác	Chức năng
List view	Initial	Hiển thị các thuộc tính của xe người dùng muốn thuê đọc được từ cơ sở dữ liệu
Back button	Click	Di chuyển trở lại Home Screen
Valid information button	Click	Thông thường là để check xem thông tin đã điền có đúng không. Tuy nhiên, hiện tại thì button này đang có chức năng điền thông tin mặc định (do mỗi nhóm chỉ có 1 card để dùng cho API)
Confirm rent bike button	Click	Đọc thông tin card điền vào và thực hiện giao dịch thuê xe, di chuyển đến màn hình thông báo kết quả giao dịch Notification Screen

Định nghĩa các trường thuộc tính:

Tên	Kích thước (bytes)	Loại	Thuộc tính	Lưu ý
Các text field				Các thuộc tính của javafx để nhận dữ liệu người dùng điền vào
Text				Text hiển thị theo quy chuẩn đã đề ra

#### 4.1.3.5 Notification Screen:



Đặc tả màn hình:


Điều khiển	Thao tác	Chức năng
Text	Initial	Hiển thị kết quả của transaction thuê hoặc trả xe.
Home button	Click	Di chuyển trở lại Home Screen

Định nghĩa các trường thuộc tính:


Tên	Kích thước (bytes)	Loại	Thuộc tính	Lưu ý
Text				Text hiển thị theo quy chuẩn đã đề ra

#### 4.1.3.6 Bike Rented List Screen:

Bike rented list



Back



Xe số: 2 thuê ở bãi xe số 1

Xe số: 1 thuê ở bãi xe số 1

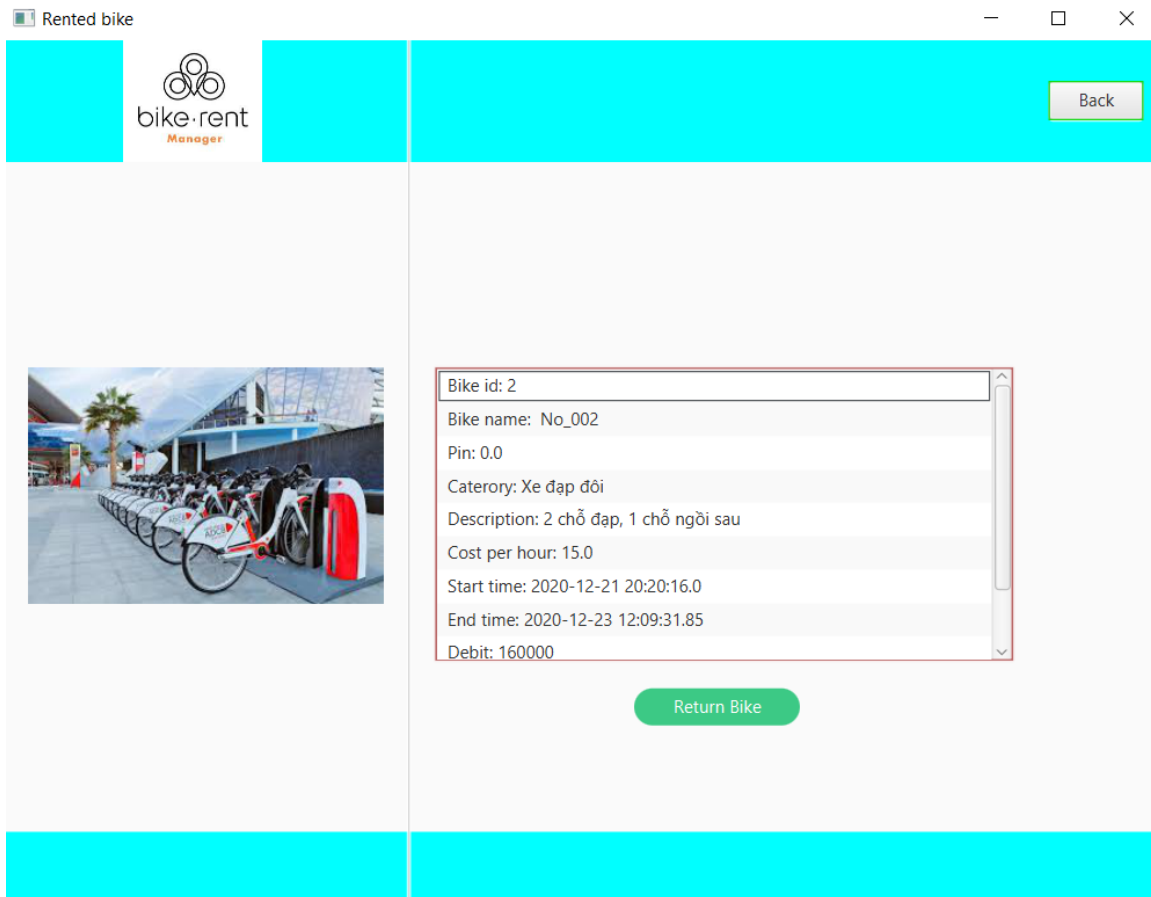
Đặc tả màn hình:

Điều khiển	Thao tác	Chức năng
List view	Click	Hiển thị danh sách xe đang thuê của người dùng, click vào xe nào thì di chuyển đến Rented Bike Screen của xe tương ứng
Back button	Click	Di chuyển trở lại Home Screen

Định nghĩa các trường thuộc tính:

Tên	Kích thước (bytes)	Loại	Thuộc tính	Lưu ý
userID		int		Trường thuộc tính ở trong code, không hiển thị lên màn hình
List view				Gồm list các xe đang thuê của người dùng đọc ra từ cơ sở dữ liệu

#### 4.1.3.7 Rented Bike Screen:



Đặc tả màn hình:


Điều khiển	Thao tác	Chức năng
List view	Initial	Hiển thị thông tin chi tiết của xe đang thuê, bao gồm cả thời gian bắt đầu và thời gian kết thúc mượn(hiện tại), tiền cọc cũng như giá thuê xe tính đến hiện tại
Back button	Click	Di chuyển trở lại Bike Rented List Screen
Return Bike button	Click	Di chuyển đến Return Bike Screen

Định nghĩa các trường thuộc tính:


Tên	Kích thước (bytes)	Loại	Thuộc tính	Lưu ý
List view				Gồm list các thuộc tính của xe đang cho thuê

#### 4.1.3.8 Return Bike Screen:

Return bike



Back



Card code

118131\_group5\_2020

Owner

296

Cw Code

Group 5

Date Expired

1125

Confirm return bike

Valid infomation



Đặc tả màn hình:

Điều khiển	Thao tác	Chức năng
Back button	Click	Di chuyển trở lại Rented Bike Screen
Valid information button	Click	Thông thường là để check xem thông tin đã điền có đúng không. Tuy nhiên, hiện tại thì button này đang có chức năng điền thông tin mặc định (do mỗi nhóm chỉ có 1 card để dùng cho API)
Confirm Return Bike button	Click	Xác nhận trả xe, di chuyển đến màn hình thông báo kết quả transaction Notification Screen

Định nghĩa các trường thuộc tính:

Tên	Kích thước (bytes)	Loại	Thuộc tính	Lưu ý
Text				Hiển thị các text theo quy chuẩn có sẵn
Text field				Các trường để nhập thông tin thẻ khách hàng

## 4.2 Data Modeling

### 4.2.1 Conceptual Data Modeling

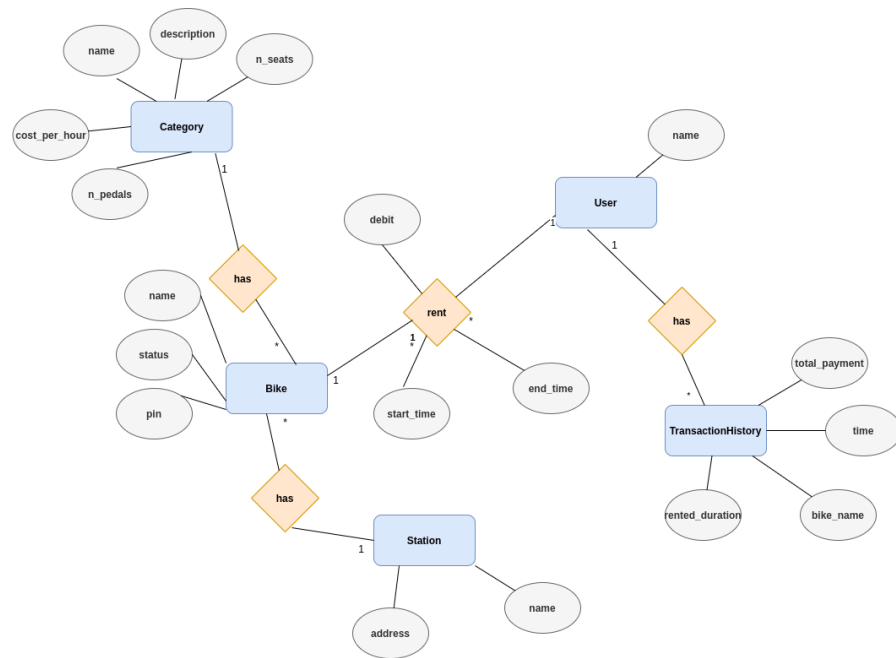


Figure 1: E-R Diagram

### 4.2.2 Database Design

#### 4.2.2.1 Database Management Systems

Nhóm em sử dụng MySQL cho đề tài này, bởi vì có những sự liên kết giữa các thực thể với nhau.

#### 4.2.2.2 Logical Data Model

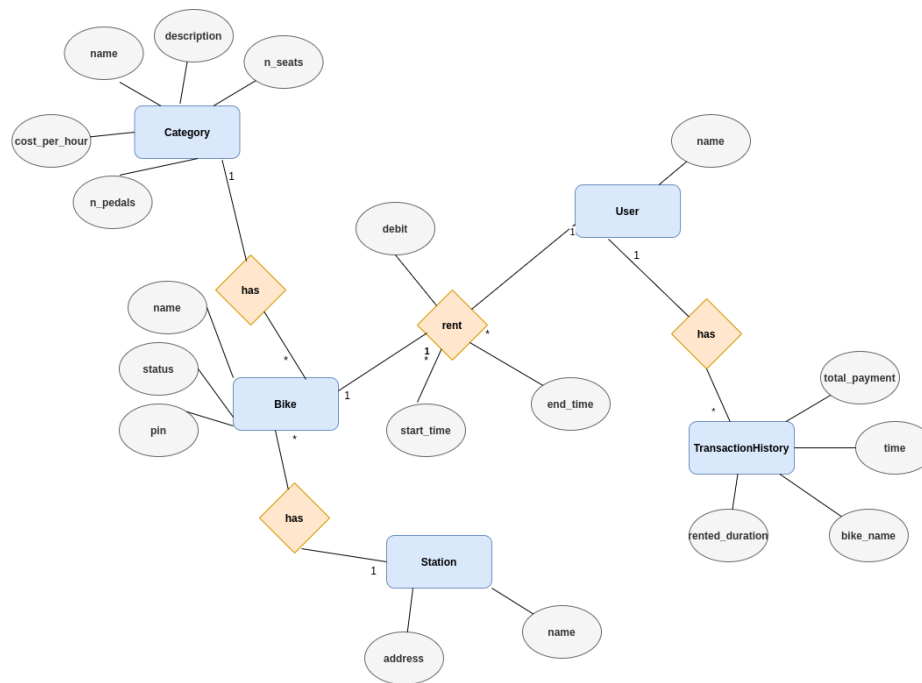


Figure 2: E-R Diagram

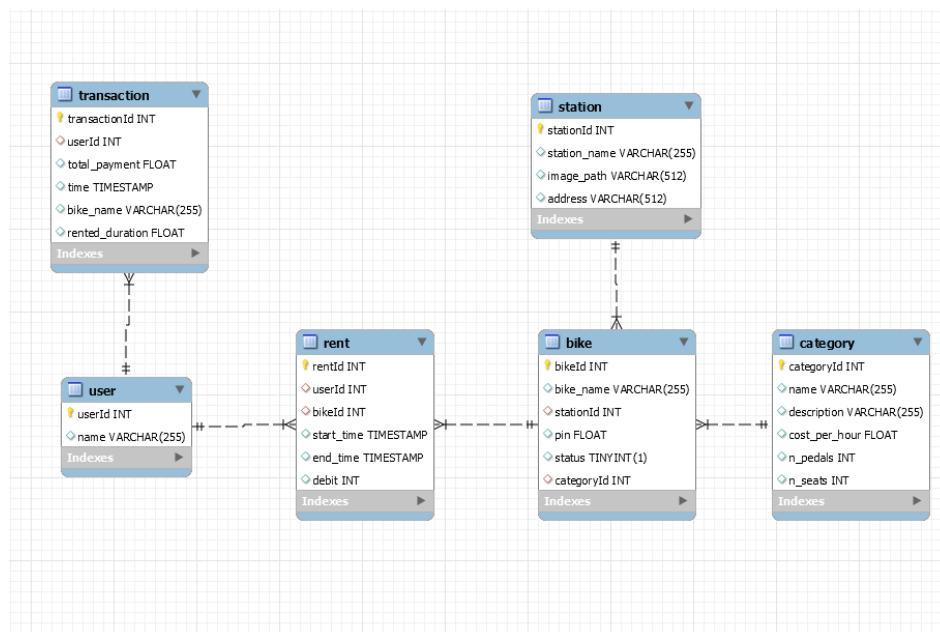


Figure 3: Data modeling

#### 4.2.2.3 Physical Data Model

station							
#	PK	FK	Column name	Data type	Default value	Mandatory	Description
1	x		stationId	int	auto increment	x	
2			station_name	varchar(255)			
3			image_path	varchar(255)			
4			address	varchar(255)			

category							
#	PK	FK	Column name	Data type	Default value	Mandatory	Description
1	x		categoryId	int	auto increment	x	
2			name	varchar(255)			
3			description	varchar(255)			
4			cost_per_hour	float			
5			n_pedals	int			
6			n_seats	int			

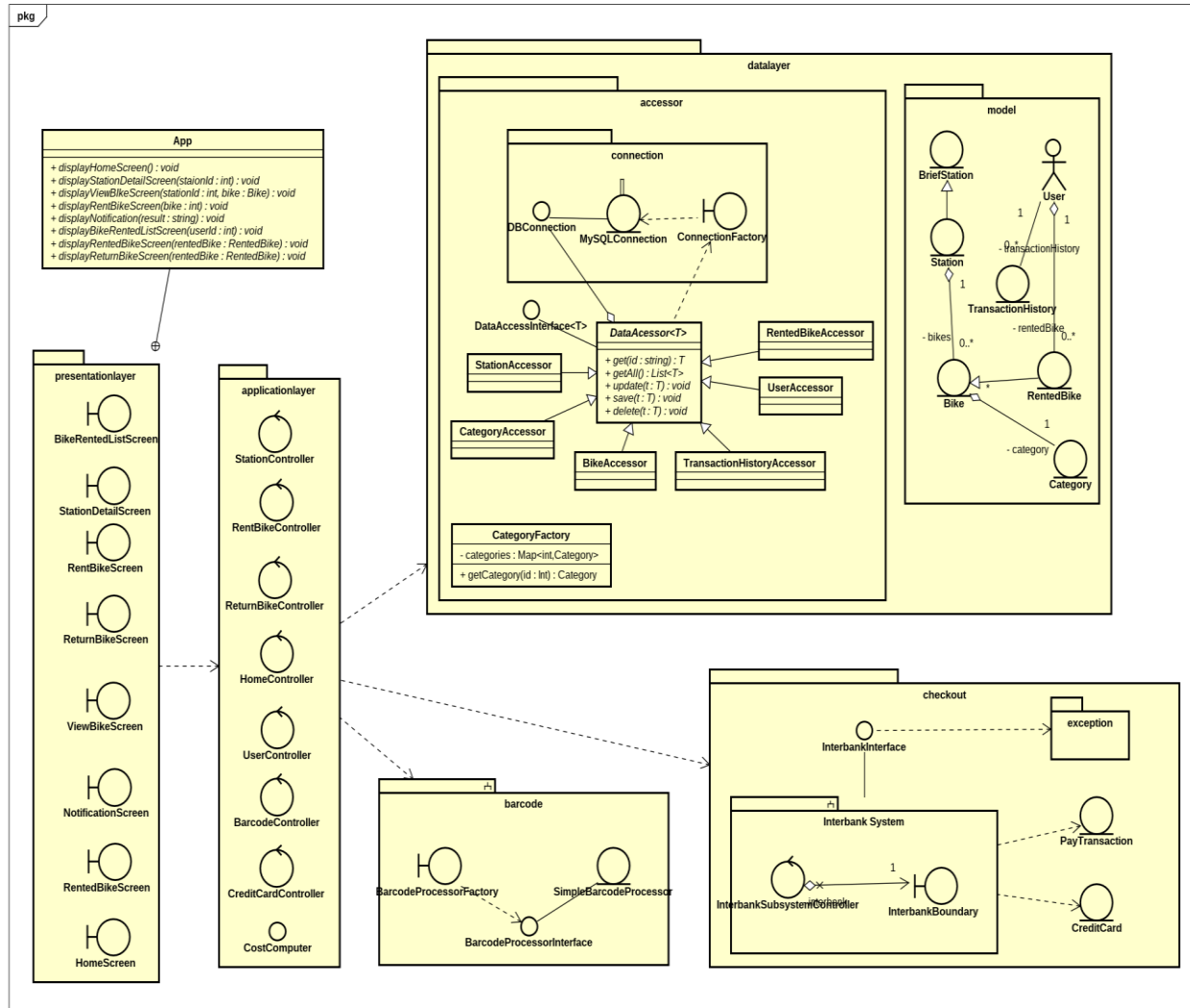
bike							
#	PK	FK	Column name	Data type	Default value	Mandatory	Description
1	x		bikeId	int	auto increment	x	
2			bike_name	varchar(255)			
3		x	stationId	int		x	foreign key of station.stationId
4			pin	float			
5			status	Boolean			
6		x	categoryId	int		x	foreign key of category.categoryId

user							
#	PK	FK	Column name	Data type	Default value	Mandatory	Description
1	x		userId	int	auto increment	x	
2			name	varchar(255)			

rent							
#	PK	FK	Column name	Data type	Default value	Mandatory	Description
1	x		rentInt	int	auto increment	x	
2		x	userId	int		x	foreign key of user.userId
3		x	bikeId	int		x	foreign key of bike.bikeId
4			start_time	timestamp			
5			end_time	timestamp			
6			debit	int			

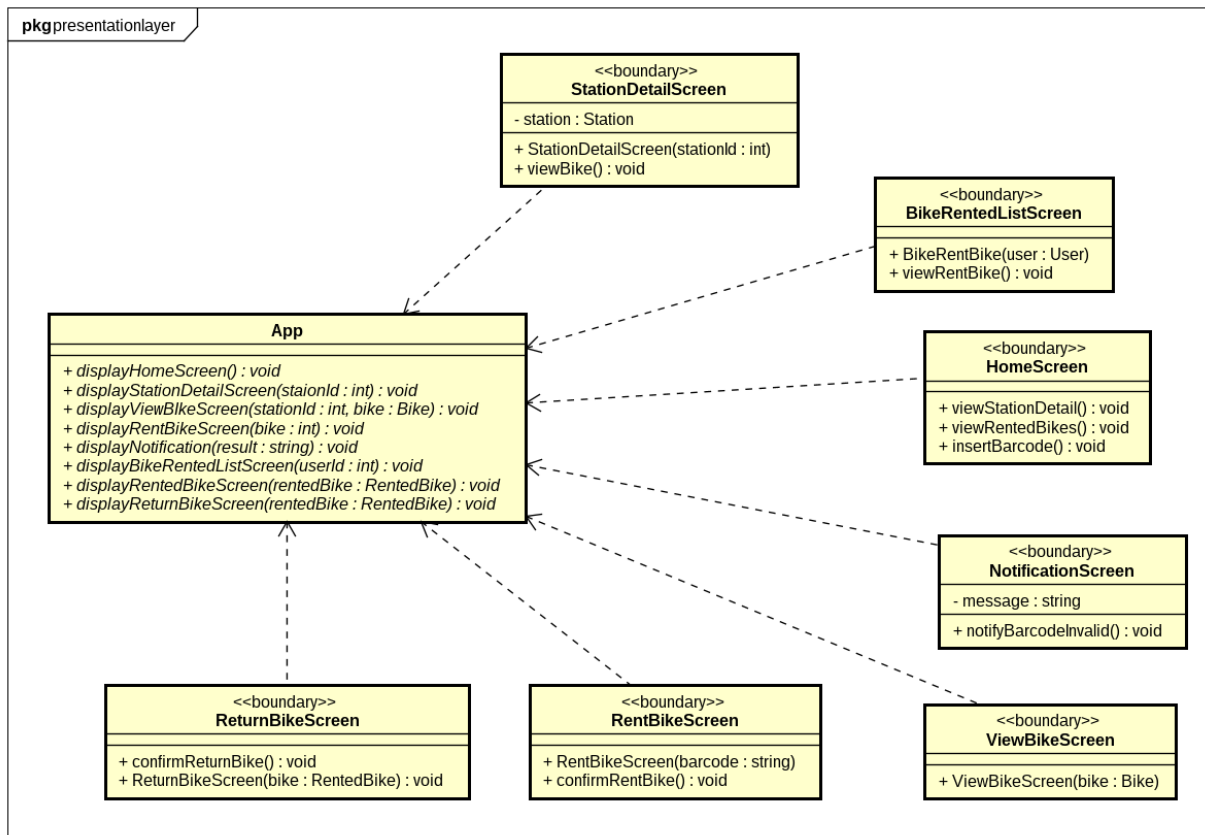
transaction							
#	PK	FK	Column name	Data type	Default value	Mandatory	Description
1	x		transactionId	int	auto increment	x	
2		x	userId	int		x	foreign key of user.userId
3			total_payment	float			
4			time	timestamp			
5			bike_name	varchar(255)			





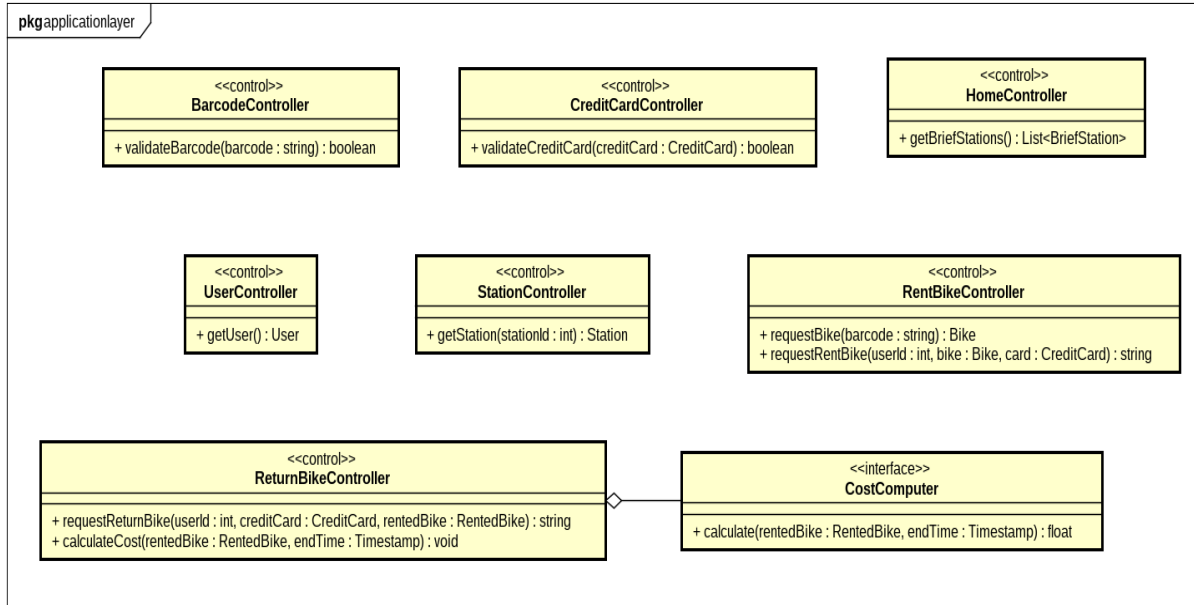
## 4.4.2 Class Diagrams

### 4.4.2.1 Class Diagram for presentationlayer package

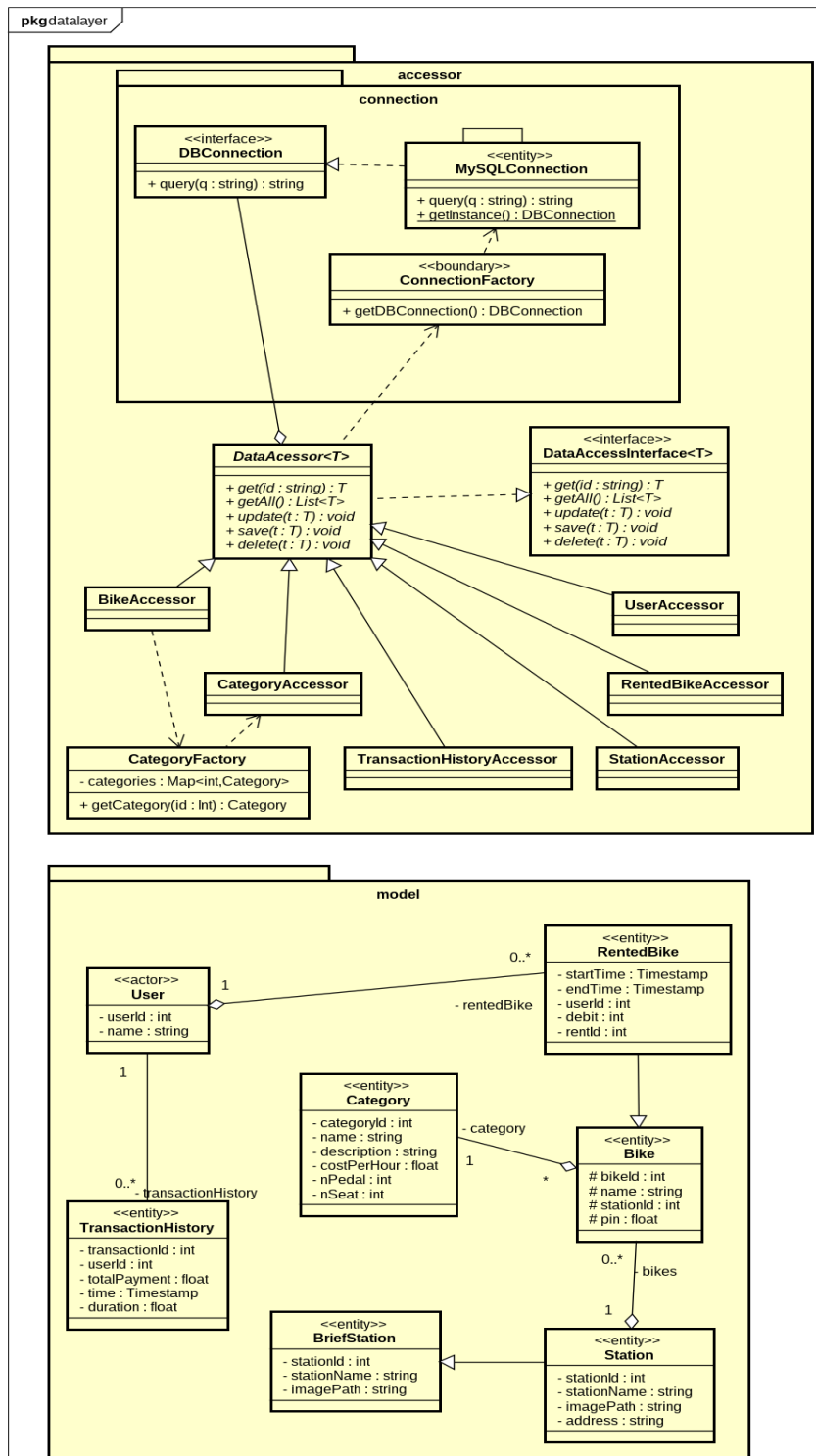




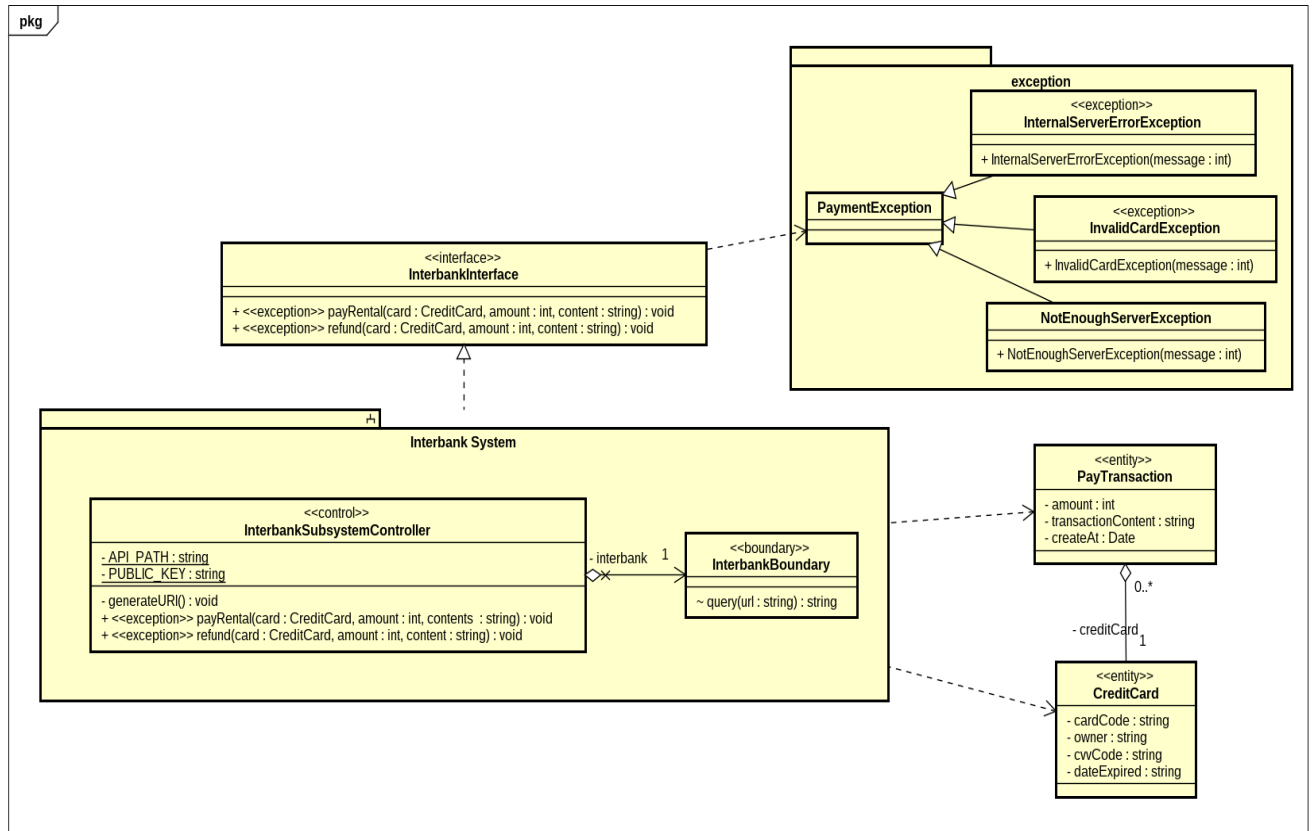
#### 4.4.2.2 Class Diagram for applicationlayer package



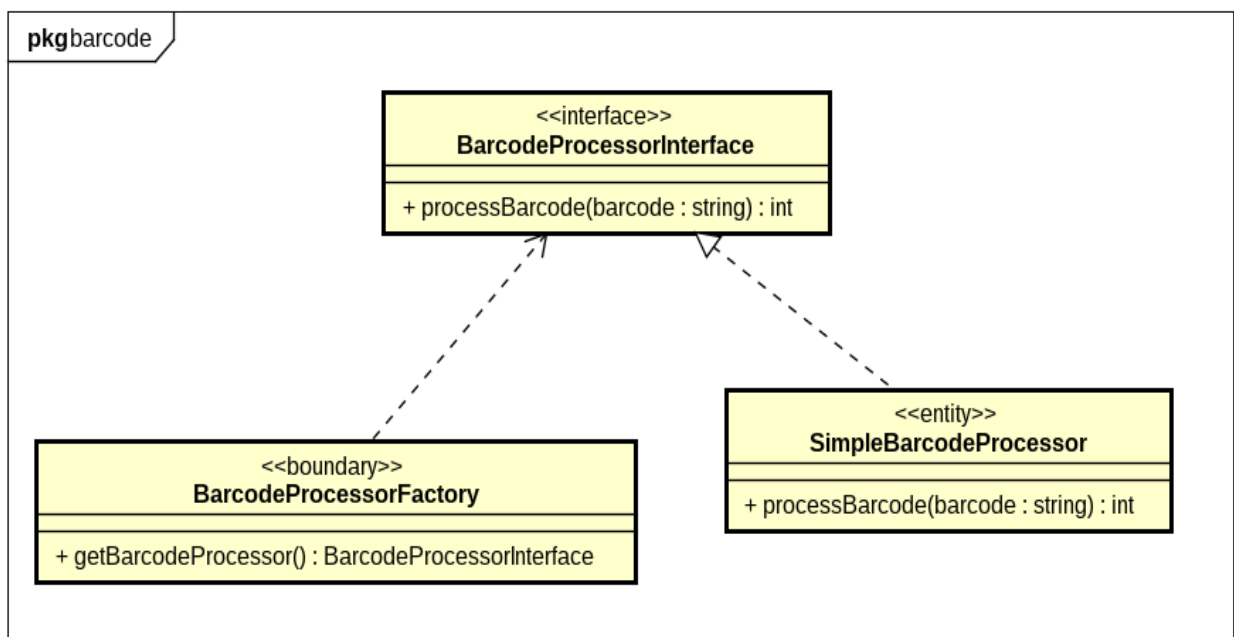
#### 4.4.2.3 Class Diagram for datalayer package



#### 4.4.2.4 Class Diagram for checkout package

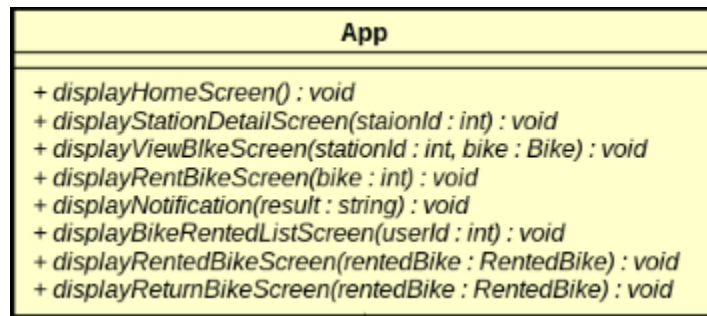


#### 4.4.2.5 Class Diagram for applicationlayer package



### 4.4.3 Class Design

#### 4.4.3.1 Class “App”



#### Attribute

Không

#### Operation

#	Name	Return type	Description (purpose)
1	displayHomeScreen	void	Hiển thị màn hình chính

Parameter:

- Không

Exception:

- Không

#	Name	Return type	Description (purpose)
2	displayStationDetailScreen	void	Hiển thị màn hình chính

Parameter:

- stationId: ID bãi xe cần hiển thị

Exception:

- Không

#	Name	Return type	Description (purpose)
3	displayViewBikeScreen	void	Hiển thị thông tin chi tiết xe

Parameter:

- stationId: ID bãi xe
- bike: xe cần hiển thị thông tin

Exception:

- Không

#	Name	Return type	Description (purpose)
4	displayRentBikeScreen	void	Hiển thị màn hình thuê xe

Parameter:

- stationId: ID bãi xe
- bike: xe cần hiển thị thông tin để thuê

Exception:

- Không

#	Name	Return type	Description (purpose)
5	displayNotification	void	Hiển thị thông báo thành công/thất bại

Parameter:

- result: thông tin cần hiển thị (thành công/thất bại)

Exception:

- Không

#	Name	Return type	Description (purpose)
6	displayBikeRentedListScreen	void	Hiển thị các xe đã thuê

Parameter:

- userId: ID người dùng đã thuê xe

Exception:

- Không

#	Name	Return type	Description (purpose)
7	displayRentedBikeScreen	void	Hiển thị chi tiết xe đã thuê

Parameter:

- rentedBike: xe đã thuê

Exception:

- Không

#	Name	Return type	Description (purpose)
8	displayReturnBikeScreen	void	Hiển thị màn hình trả xe

Parameter:

- rentedBike: xe đã thuê

*Exception:*

- Không

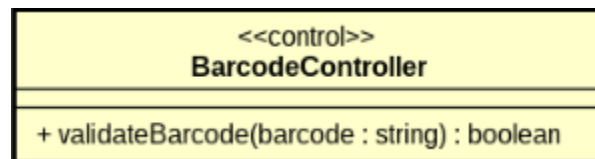
### Method

Không

### State

Không

#### 4.4.3.2 Class “BarcodeController”



### Attribute

Không

### Operation

#	Name	Return type	Description (purpose)
1	validateBarcode	boolean	Kiểm tra barcode nhập vào đúng định dạng hay không

*Parameter:*

- barcode: mã barcode nhập vào để kiểm tra

*Exception:*

- Không

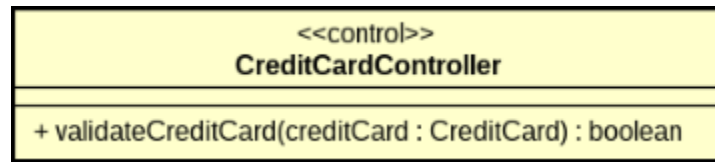
### Method

Không

### State

Không

#### 4.4.3.3 Class “CreditCardController”



##### Attribute

Không

##### Operation

#	Name	Return type	Description (purpose)
1	validateCreditCard	boolean	Xác minh thông tin thẻ tín dụng

Parameter:

- creditCard: thẻ tín dụng cần kiểm tra

Exception:

- Không

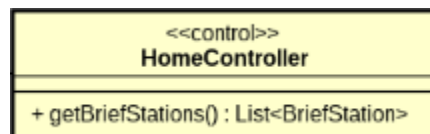
##### Method

Không

##### State

Không

#### 4.4.3.4 Class “HomeController”



##### Attribute

Không

##### Operation

#	Name	Return type	Description (purpose)
1	getBriefStations	List<BriefStation>	Trả về danh sách các bãi xe

*Parameter:*

- Không

*Exception:*

- Không

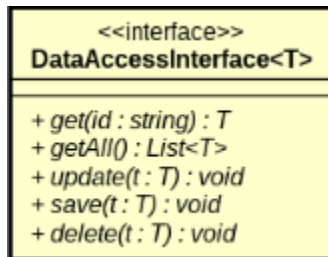
## Method

Không

## State

Không

### 4.4.3.5 Class “*DataAccessInterface<T>*”



## Attribute

Không

## Operation

#	Name	Return type	Description (purpose)
1	get	T	Truy xuất cơ sở dữ liệu và lấy ra đối tượng tương ứng với T theo id

*Parameter:*

- id: id đối tượng cần lấy ra

*Exception:*

- Không

#	Name	Return type	Description (purpose)
2	getAll	List<T>	Truy xuất cơ sở dữ liệu và lấy ra toàn bộ đối tượng ứng với T

*Parameter:*



- Không

*Exception:*

- Không

#	Name	Return type	Description (purpose)
3	update	void	Cập nhật đối tượng trong cơ sở dữ liệu
4	save	void	Lưu đối tượng vào cơ sở dữ liệu
5	delete	void	Xoá đối tượng khỏi cơ sở dữ liệu

*Parameter:*

- t: đối tượng cần thực hiện thay đổi

*Exception:*

- Không

## Method

Không

## State

Không

### 4.4.3.6 Class “InterbankInterface”

<<interface>> <b>InterbankInterface</b>	
+ <<exception>> payRental(card : CreditCard, amount : int, contents : String) : PaymentTransaction + <<exception>> refund(card : CreditCard, amount : int, contents : String) : PaymentTransaction	

## Attribute

Không

## Operation

#	Name	Return type	Description (purpose)
1	payRental	PaymentTransaction	Thanh toán và trả về thông tin giao dịch
2	refund	PaymentTransaction	Hoàn tiền và trả về thông tin giao dịch

*Parameter:*

- card: thẻ tín dụng để giao dịch
- amount: số tiền giao dịch
- contents: nội dung giao dịch

*Exception:*

- PaymentException: nếu mã lỗi trả về đã biết
- UnrecognizedException: nếu không có mã lỗi trả về hoặc lỗi hệ thống

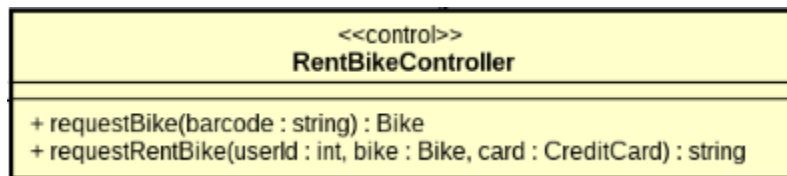
## Method

Không

## State

Không

### 4.4.3.7 Class “RentBikeController”



## Attribute

Không

## Operation

#	Name	Return type	Description (purpose)
1	requestBike	Bike	Trả về xe người dùng yêu cầu

*Parameter:*

- barcode: barcode tương ứng với xe muốn thuê

*Exception:*

- Không

#	Name	Return type	Description (purpose)
2	requestRentBike	Bike	Xử lý yêu cầu thuê xe và trả về thông báo thành công/thất bại

*Parameter:*

- userId: id người dùng
- bike: xe mà người dùng muốn thuê
- card: thẻ tín dụng của người dùng

*Exception:*

- Không

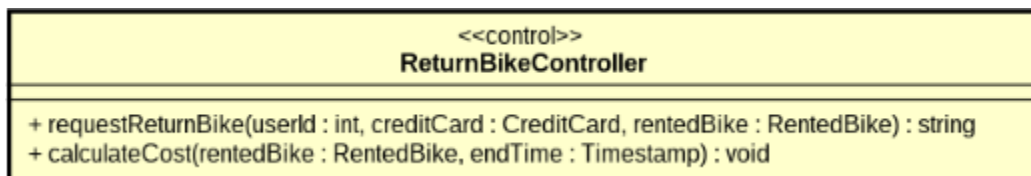
## Method

Không

## State

Không

### 4.4.3.8 Class “ReturnBikeController”



## Attribute

Không

## Operation

#	Name	Return type	Description (purpose)
1	requestReturnBike	String	Xử lý yêu cầu trả xe của người dùng và thông báo tương ứng (thành công/thất bại)

*Parameter:*

- userId: id người dùng
- rentedBike: xe mà người dùng muốn trả
- creditCard: thẻ tín dụng của người dùng

*Exception:*

- Không

#	Name	Return type	Description (purpose)
2	calculateCost	void	Tính số tiền người dùng cần trả

*Parameter:*

- rentedBike: xe mà người dùng muốn trả
- endTime: thời điểm trả xe

*Exception:*

- Không

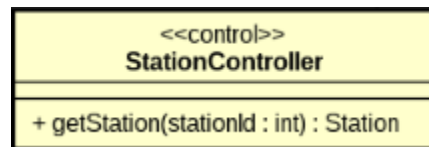
#### **Method**

Không

#### **State**

Không

#### **4.4.3.9 Class “StationController”**



#### **Attribute**

Không

#### **Operation**

#	Name	Return type	Description (purpose)
1	getStation	Station	Trả về bãi xe tương ứng id

*Parameter:*

- stationId: id bãi xe cần lấy thông tin

*Exception:*

- Không

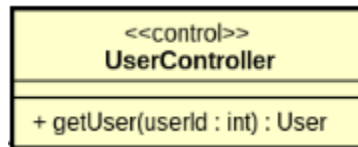
#### **Method**

Không

#### **State**

Không

#### 4.4.3.10 Class “UserController”



##### Attribute

Không

##### Operation

#	Name	Return type	Description (purpose)
1	getUser	User	Trả về thông tin người dùng tương ứng id

Parameter:

- userId: id người dùng cần lấy thông tin

Exception:

- Không

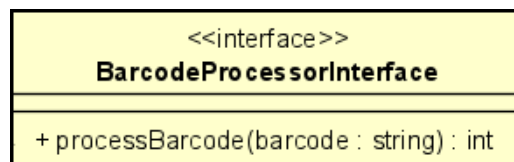
##### Method

Không

##### State

Không

#### 4.4.3.11 Class “BarcodeProcessorInterface”



##### Attribute

Không

##### Operation

#	Name	Return type	Description (purpose)
1	processBarcode	int	Xử lý barcode thành mã xe

Parameter:

- barcode: barcode cần xử lý

*Exception:*

- Không

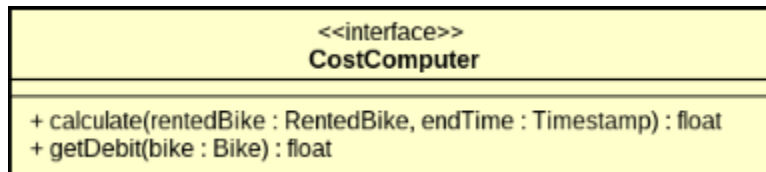
## Method

Không

## State

Không

### 4.4.3.12 Class “CostComputer”



## Attribute

Không

## Operation

#	Name	Return type	Description (purpose)
1	calculate	float	Tính số tiền người dùng phải trả sau khi thuê xe

*Parameter:*

- rentedBike: xe người dùng đã thuê
- endTime: thời điểm trả xe

*Exception:*

- Không

#	Name	Return type	Description (purpose)
1	getDebit	float	Tính số tiền người dùng phải đặt cọc với loại xe tương ứng

*Parameter:*

- bike: xe cần tính tiền đặt cọc

*Exception:*

- Không

**Method**

Không

**State**

Không

**4.4.3.13 Class “DBConnection”**

<<interface>> DBConnection	
+ <i>query</i> ( <i>q : string</i> ) : <i>ResultSet</i> + <i>execute</i> ( <i>q : string</i> ) : <i>void</i> + <i>executeUpdate</i> ( <i>q : string</i> ) : <i>void</i>	

**Attribute**

Không

**Operation**

#	Name	Return type	Description (purpose)
1	query	ResultSet	Thực hiện select trong cơ sở dữ liệu
2	execute	void	Thực hiện delete trong cơ sở dữ liệu
3	executeUpdate	void	Thực hiện update và save trong cơ sở dữ liệu

*Parameter:*

- q: câu lệnh cần chạy trong cơ sở dữ liệu

*Exception:*

- Không

**Method**

Không

**State**

Không

## 5 Design Considerations

### 5.1 Goals and Guidelines

#### 1. Mục tiêu

Phần mềm mô phỏng quá trình quản lý bãi xe của khu đô thị hoặc ở công cộng và cung cấp tiện ích thuê xe cho người dùng, với ứng dụng mô phỏng có thể chạy được đa nền tảng, thuận tiện cho người dùng trong việc tìm kiếm xe và bãi xe. Thiết kế mô hình với các thành phần độc lập với nhau, tức là tại một thời điểm không cần quá nhiều đối tượng phải tồn tại trong hệ thống, giúp tiết kiệm bộ nhớ và tăng thời gian xử lý.

#### 2. Cách dùng

Không nên thuê xe quá lâu vì như vậy số tiền ảo mô phỏng sẽ không đáp ứng được. Không nên thuê quá nhiều xe vì phải đặt cọc quá nhiều tiền, cũng có thể gây ra lỗi không đủ số dư trong tài khoản.

### 5.2 Architectural Strategies

1. Hệ thống sử dụng ngôn ngữ lập trình Java, với thư viện JavaFx hỗ trợ đa nền tảng, có thể chạy trên nhiều hệ điều hành khác nhau.

2. Sử dụng hệ quản trị cơ sở dữ liệu MySQL, vì hệ thống yêu cầu tính đồng bộ cao giữa các bãi xe, trạng thái các xe, các giao dịch của người dùng, và dữ liệu của hệ thống cũng không phát sinh trở thành dữ liệu lớn trong tương lai đủ xa. (Ví dụ: Khoảng 10 năm)

3. Trong tương lai sẽ dự định phát triển thêm cả nền tảng web, do các controller độc lập với boundary nên có thể tận dụng lại được ở backend server.

4. Để khắc phục lỗi xảy ra ở phía người dùng (ví dụ: hệ thống người dùng bị hỏng trong quá trình đang thuê xe), mọi thông tin của xe đều được lưu trên server và sẽ phát triển một nền tảng web để người dùng có thể trả xe dù điện thoại hay máy tính của họ bị hỏng.

5. Tại nhiều người cùng truy cập đến hệ thống, database sẽ có nhiều slave để đảm nhiệm chức năng đọc, còn việc ghi thì sẽ ghi vào một cụm database được kết nối bằng thông với độ trễ thấp, thích hợp cho việc đồng bộ dữ liệu thời gian thực.

6. Các phương thức giao tiếp giữa người dùng và server là giao thức https với việc liên quan đến thông tin cá nhân user và http với việc truy nhập đến thông tin bike hoặc station.

### 5.3 Coupling and Cohesion

Về tổng quan, em đánh giá hệ thống là loose coupling và high cohesion.



### **5.3.1 Cohesion**

#### **5.3.1.1 *representationlayer package***

Trong lớp HomeScreen và App có nhiều hơn 1 phương thức khác khởi tạo, các phương thức này ở đây bởi vì nó đều là các chức năng chính hệ thống cung cấp cho người dùng. Vì vậy, Home Screen có mức cohesion là Logical.

Trong các lớp khác thì chỉ có một phương thức public duy nhất tương ứng với trách nhiệm của lớp ấy, nên nó có mức cohesion là Functional

#### **5.3.1.2 *applicationlayer package***

Trong lớp ReturnBikeController và RentBikeController đều có 2 phương thức. Hai lớp này điều khiển luồng giao tiếp giữa người dùng, hệ thống và ngân hàng, và 2 phương thức của từng lớp là một phần của chuỗi giao tiếp đó, nên hai lớp này có mức độ cohesion là Communicational

Trong lớp CostComputer có 2 phương thức là getDebit và calculate, đều là tính toán giá tiền nhưng mà cho hai trường hợp khác nhau, có tham số đầu vào khác nhau, nên mức độ Cohesion của lớp này là Logical.

Trong các lớp còn lại của package này, chúng đều chỉ có duy nhất một phương thức public thực hiện nghĩa vụ của chúng nên mức độ cohesion là Functional.

#### **5.3.1.3 *datalayer package***

Các lớp ở trong accessor subpackge của datalayer triển khai theo interface DataAccessorInterface có thiết kế theo mẫu Data Access Object, với 5 chức năng cơ bản của mẫu này. Do đó, mức cohesion của lớp này là Procedural

#### **5.3.1.4 *checkout package***

Interbank Interface cung cấp cho người dùng 2 phương thức xử lý cùng một kiểu dữ liệu, cùng giao tiếp với interbank boundary nhưng theo 2 cách khác nhau, và có entry point cho mỗi phương thức là khác nhau, nên chung có mức độ cohesion là Informational.

### 5.3.2 Coupling

#### 5.3.2.1 *presentationlayer package*

Các lớp màn hình trong package này chỉ phụ thuộc vào lớp App, nên giữa các thành phần màn hình sẽ độc lập với nhau. Còn giữa lớp App và lớp \*Screen thì lớp App có trách nhiệm di chuyển các màn hình và chúng đồng nhất về kiểu dữ liệu primitive và các đối tượng entity nên mức độ coupling sẽ là data coupling.

#### 5.3.2.2 *applicationlayer package*

Các lớp controller trong package này không giao tiếp với nhau nên chúng là độc lập. Lớp RentBikeController và ReturnBikeController giao tiếp với SimpleCostComputer thông qua interface CostComputer nên mức độ coupling là data coupling, tức là controller chỉ phụ thuộc vào SimpleCostComputer ở kiểu dữ liệu cho tham số các phương thức trong interface.

#### 5.3.2.3 *datalayer package*

Trong accessor subpackage, các accessor class không phụ thuộc vào nhau. Chúng phụ thuộc vào các lớp entity ở model subpackage. Do đó chúng có mức độ coupling là data coupling.

#### 5.3.2.4 *checkout package*

Lớp interbank subsystem controller chỉ sử dụng phương thức của interbank boundary và giữa chúng không có tương tác làm thay đổi trạng thái lẫn nhau hay là việc thừa dữ liệu trong tham số trong message, nên mức độ coupling là data coupling.

## 5.4 Design Principles

Về tổng quan, các lớp trong hệ thống được thiết kế tuân theo nguyên lý SOLID.

## **Single Responsibility**

Các lớp trong hệ thống phần lớn đều có 1 phương thức duy nhất, trừ các lớp thực hiện việc chuyển màn hình như App, các phương thức này chính là chức năng duy nhất của các lớp đó.

## **Open for extension, close for modification**

Các lớp thực hiện thuật toán hay lớp thực hiện quy trình lấy dữ liệu đều là abstraction, nên khi muốn thay đổi thì ta chỉ cần thêm một lớp dẫn suất mới.

## **Liskov substitution**

Lớp RentedBike là lớp con của Bike, lớp Station là lớp con của BriefStation và các lớp con này đều có thể thay thế cho lớp cha ở bất kỳ nơi nào lớp cha xuất hiện. Mục đích tạo ra quan hệ cha con này vì muốn tạo ra một đối tượng Station thì cần lấy nhiều thông tin, trong khi đó đôi khi ta không cần nhiều như vậy, nên BriefStation là lớp cơ bản chứa ít thông tin nhưng đủ cho việc hiện thị overview, còn muốn xem chi tiết sẽ tạo ra lớp Station sau. Tương tự, RentedBike sẽ có đầy đủ thành phần của Bike, nhưng sẽ thêm các thuộc tính mà có ở BikeRented như thời gian thuê, ...

## **Interface segregation**

Không đáp ứng được ở CostComputer Interface do nó có 2 phương thức là calculate và getDebit là độc lập với nhau nhưng lại để ở một interface. Ta có thể tách nó ra làm 2 interface để đảm bảo nguyên lý này, nhưng trong quá trình thiết kế, nhóm nhận thấy cả hai phương thức này đều cần thiết và được sử dụng trong controller class, và chúng ta có thể tạo ra lớp mới mà giữ nguyên phương thức calculate ở lớp cha và override lại phương thức getDebit ở lớp con, nên nhóm quyết định để 2 phương thức này vào cùng một interface để khi người dùng muốn tạo một CostComputerImplementer mới thì phải định nghĩa hoặc kế thừa cả phương thức trên.

## **Dependency inversion**

Trong các sự phụ thuộc giữa package controller và các package khác đều là thông qua các interface. Các thành phần trong subsystem hay package cung cấp chức năng như datalayer, các lớp cụ thể sẽ triển khai (hay là phụ phục) và lớp trừu tượng (interface). Như vậy ta có thể dễ dàng thay đổi cách cài đặt của từng chức năng mà không ảnh hưởng đến các thành phần mà dùng nó.

## **5.5 Design Patterns**

*Trong thiết kế, nhóm có sử dụng một số design pattern sau.*

### **5.5.1 Factory**

Việc khởi tạo lớp BarcodeProcessing có thể rất phức tạp vì trong thực tế barcode sẽ có dạng ảnh và chúng ta phải có một hệ thống con để xử lý ảnh. Vì vậy, nhóm sử dụng Factory pattern cho việc khởi tạo.

### **5.5.2 Data Access Object**

Các lớp lấy dữ liệu từ nguồn dữ liệu đều có một số đặc điểm chung, nên với Data Access Object pattern thì các lớp sử dụng sẽ không cần quan tâm đến từng lớp lấy dữ liệu cụ thể sẽ có tên là gì mà chỉ cần biết đến interface DAO với kiểu dữ liệu entity tương ứng với nó.

### **5.5.3 Fly Weight**

Nhận thấy lớp Category sẽ không bao giờ thay đổi trạng thái, cũng như là rất nhiều bike chỉ cần dùng một trạng thái thuộc tính của category nên dùng mẫu này để tối ưu bộ nhớ lưu trữ, chi phí cho connection lấy dữ liệu từ database cũng như thời gian tạo mới một category.

### **5.5.4 Visitor (Đã thiết kế nhưng do điều kiện nên không thực hiện)**

Nhận thấy lớp App trong package presentaitonlayer nhận nhiệm vụ thực hiện display màn hình, và phụ thuộc vào từng màn hình thì sẽ có một kiểu display riêng. Như vậy có thể thiết kế App là một visitor với phương thức visit là display còn các màn hình là lớp dẫn suất của interface Screen tương ứng là acceptor. Nhưng do đặc điểm của javaFX nên nhóm đã không triển khai mẫu dữ liệu này.