

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

School of Information and communications technology



Software Design Document  
**CAPSTONE PROJECT**  
Subject: ITSS 20211

Group 1: Nguyễn Duy Khánh – 20184125 (L)

Trần Hoàng Việt - 20184232

Phùng Xuân Quân - 20184177

Trần Văn Tuấn - 20184223

Giảng viên : Nguyễn Thị Thu Trang

*<All notations inside the angle bracket are not part of this document, for its purpose is for extra instruction. When using this document, please erase all these notations and/or replace them with corresponding content as instructed>*

*<This document, written by Asst. Prof. NGUYEN Thi Thu Trang, is used as a case study for student with related courses. Any modifications and/or utilization without the consent of the author is strictly forbidden>*

# 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 .....	4
2.1 General Overview.....	4
2.2 Assumptions/Constraints/Risks .....	4
2.2.1 Assumptions.....	4
2.2.2 Constraints .....	4
2.2.3 Risks.....	5
3 System Architecture and Architecture Design .....	6
3.1 Architectural Patterns .....	6
3.2 Interaction Diagrams .....	6
3.3 Analysis Class Diagrams .....	9
3.4 Unified Analysis Class Diagram .....	12
3.5 Security Software Architecture .....	12
4 Detailed Design .....	13
4.1 User Interface Design .....	13
4.1.1 Screen Configuration Standardization .....	13
4.1.2 Screen Transition Diagrams.....	14
4.1.3 Screen Specifications .....	14
4.2 Data Modeling .....	21
4.2.1 Conceptual Data Modeling .....	21
4.2.2 Database Design.....	22

4.3	Non-Database Management System Files .....	28
4.4	Class Design .....	28
4.4.1	General Class Diagram .....	28
4.4.2	Class Diagrams .....	29
4.4.3	Class Design.....	30
5	Design Considerations.....	37
5.1	Goals and Guidelines .....	37
5.2	Architectural Strategies .....	37
5.3	Coupling and Cohesion .....	38
5.4	Design Principles .....	38
5.5	Design Patterns .....	38

## List of Figures

No table of figures entries found.

## List of Tables

No table of figures entries found.

# 1 Introduction

## 1.1 Objective

Tài liệu thiết kế phần mềm này cung cấp bản thiết kế chi tiết về hệ thống giả lập thuê / trả xe đạp EcoBikeRental

Người dùng mong đợi của tài liệu là những lập trình viên, những người tham gia vào triển khai, bảo trì, đánh giá chất lượng, cho hệ thống giả lập EcoBikeRental.

## 1.2 Scope

Hệ thống giả lập EcoBikeRental hướng đến mục tiêu đơn giản cho phép người sử dụng làm quen dễ dàng.

Thời gian đáp ứng tối đa của hệ thống là 1 giây lúc bình thường hoặc 2 giây lúc cao điểm. Giúp người dùng có trải nghiệm tốt nhất khi sử dụng. Tuy nhiên hệ thống giả lập này chỉ cho phép thanh toán thông qua thẻ tín dụng.

## 1.3 Glossary

## 1.4 References

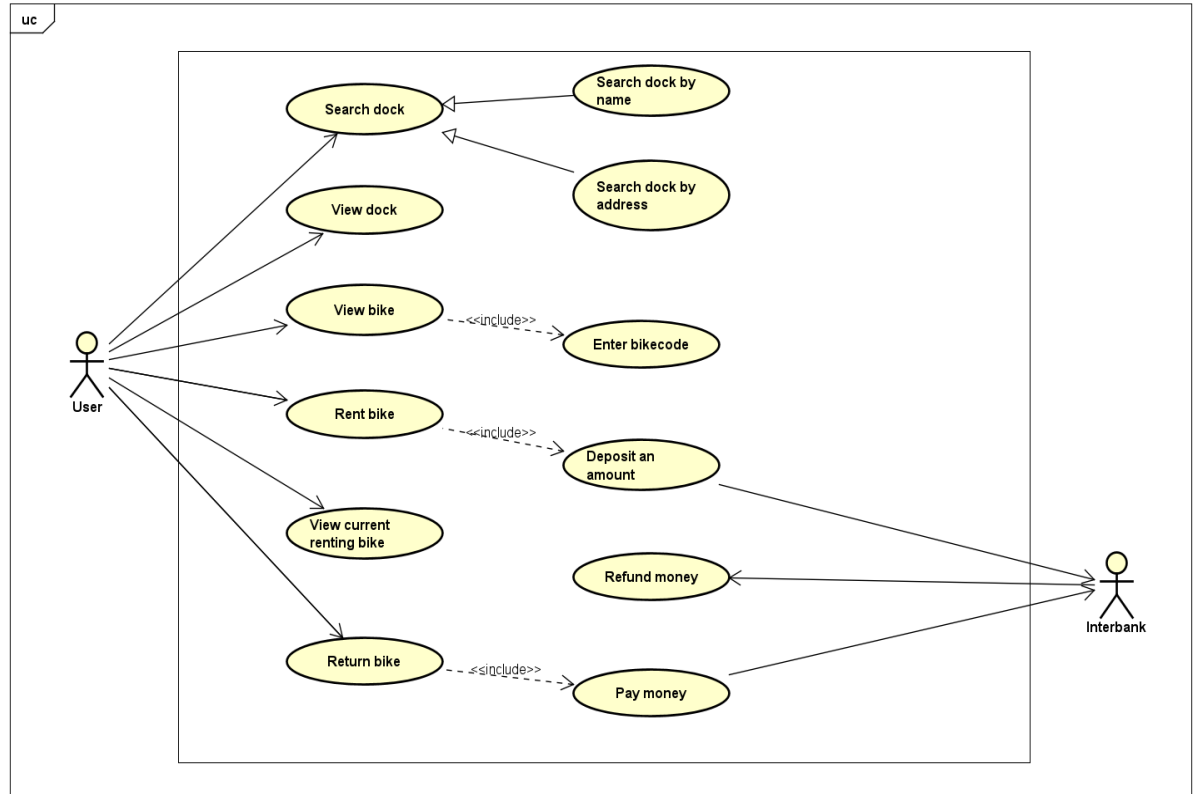
Centers for Medicare & Medicaid Services. (n.d.). *System Design Document Template*. Retrieved from Centers for Medicare & Medicaid Services:  
<https://www.cms.gov/Research-Statistics-Data-and-Systems/CMS-Information-Technology/XLC/Downloads/SystemDesignDocument.docx>

## 2 Overall Description

### 2.1 General Overview

Hệ thống được tổ chức, thiết kế dựa trên mô hình MVC và có áp dụng các nguyên lý SOLID trong thiết kế phần mềm. Điều này giúp cho hệ thống EcoBikeRental trở lên dễ bảo trì và thay đổi theo yêu cầu, phản hồi từ phía người dùng.

Biểu đồ use case tổng quan của hệ thống:



### 2.2 Assumptions/Constraints/Risks

#### 2.2.1 Assumptions

Hệ điều hành : Windows, Ubuntu

#### 2.2.2 Constraints

- Ngôn ngữ lập trình Java
- DBMS : PostgreSQL

### **2.2.3 Risks**

Hệ thống thường xuyên thay đổi, thêm mới chức năng sẽ khiến kiến trúc bị tác động và ảnh hưởng tới những chức năng khác nếu hệ thống thiết kế không

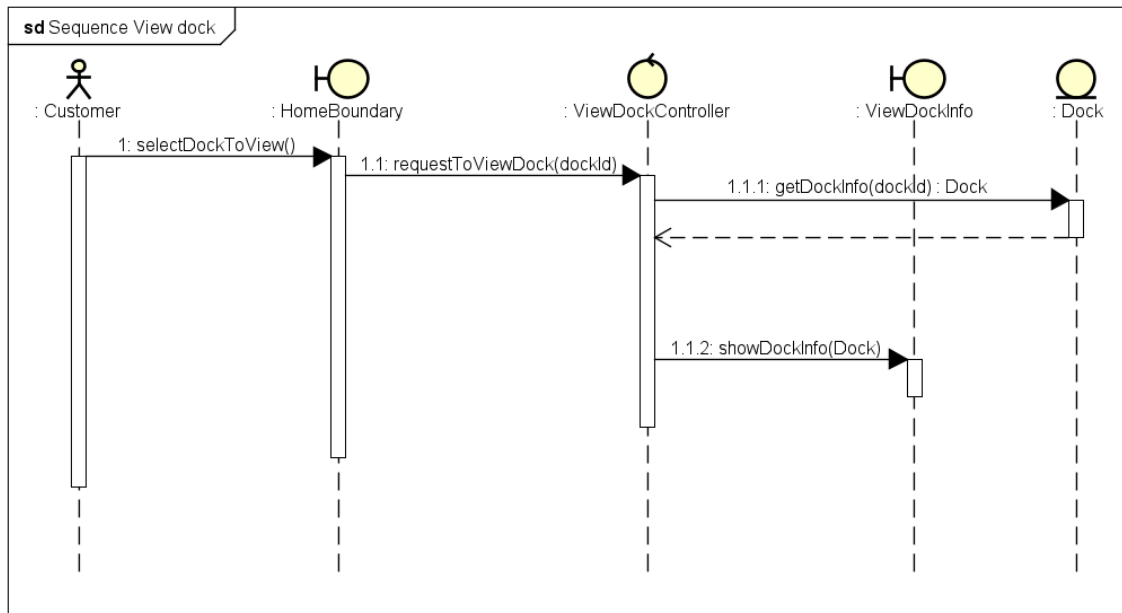
## 3 System Architecture and Architecture Design

### 3.1 Architectural Patterns

*<Specify and briefly describe the chosen architectural patterns and the reasons why they were chosen>*

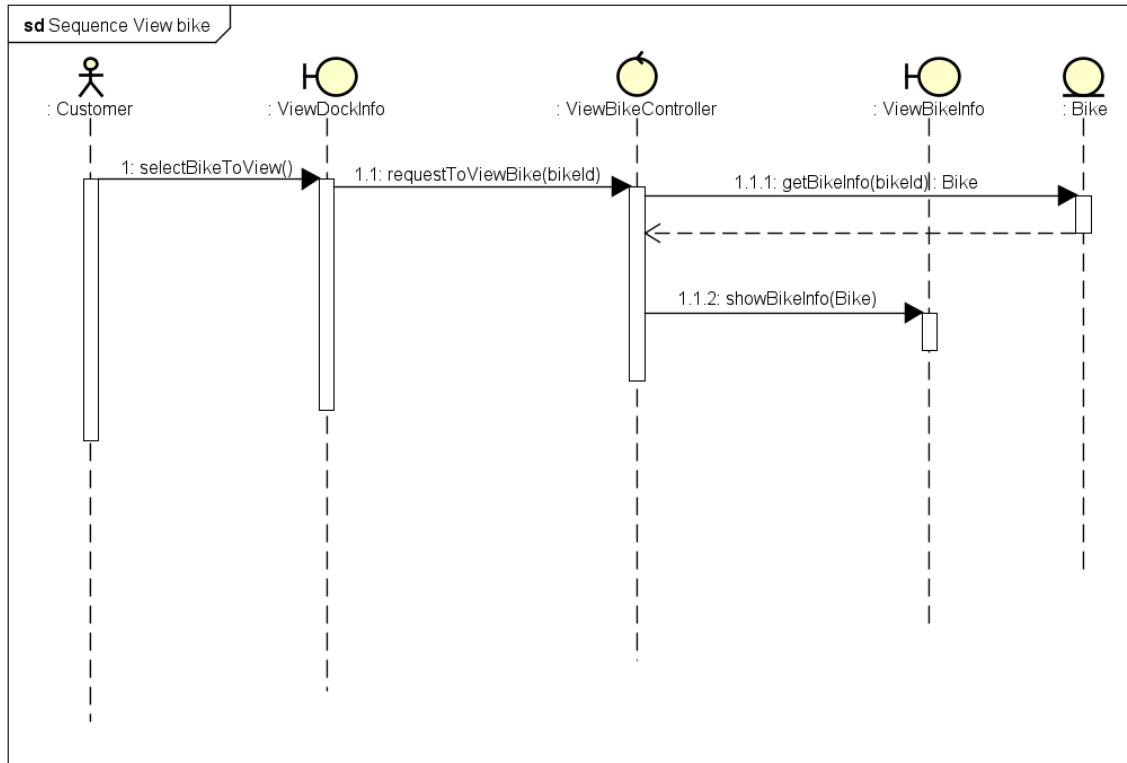
### 3.2 Interaction Diagrams

#### 3.2.1. Use case “View dock”

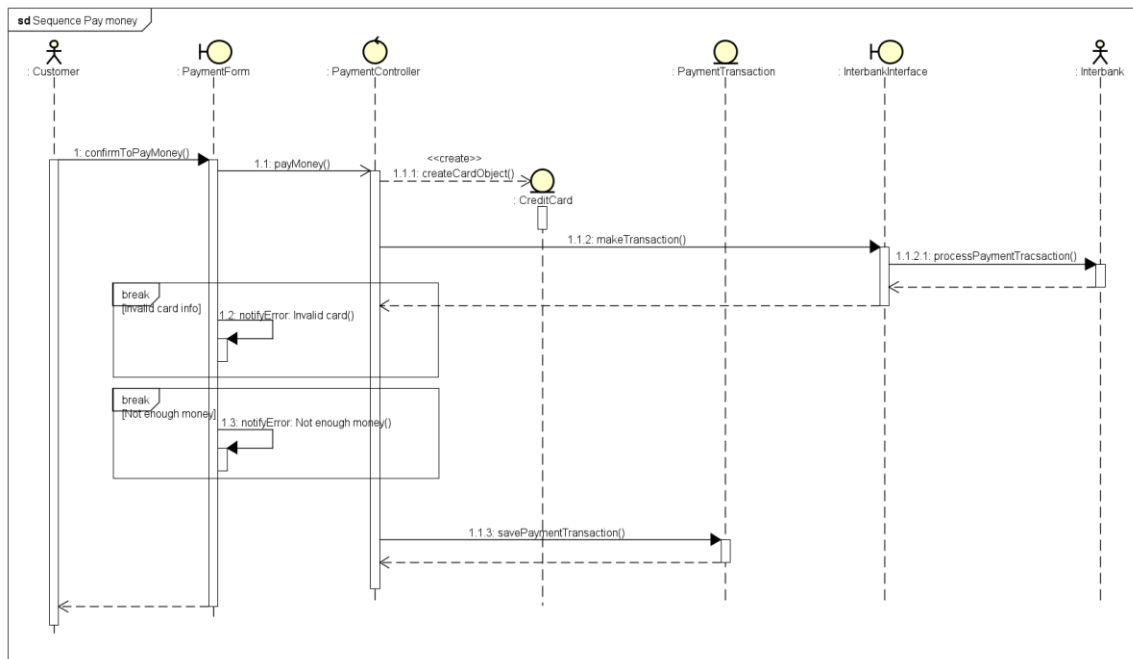


#### 3.2.2. Use case “View bike”

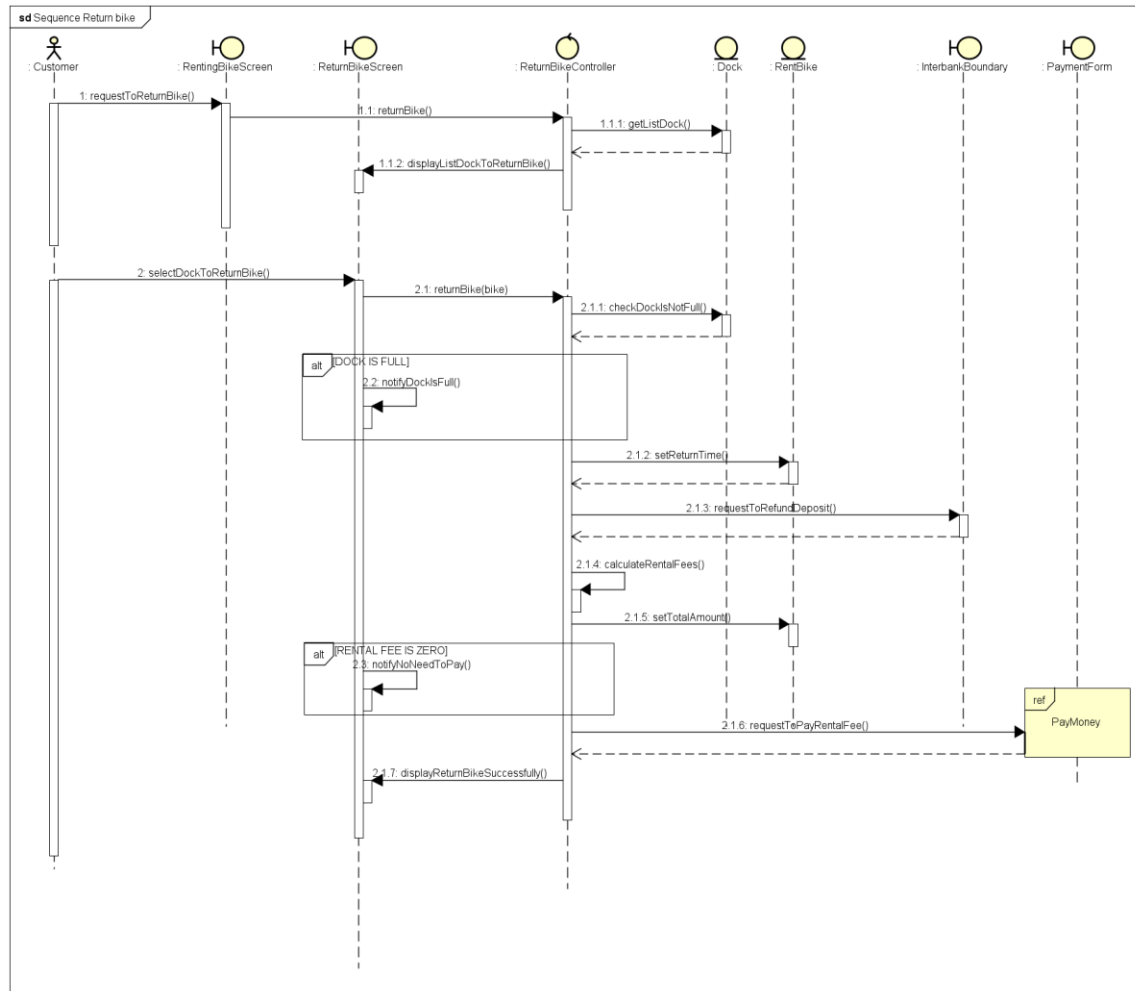




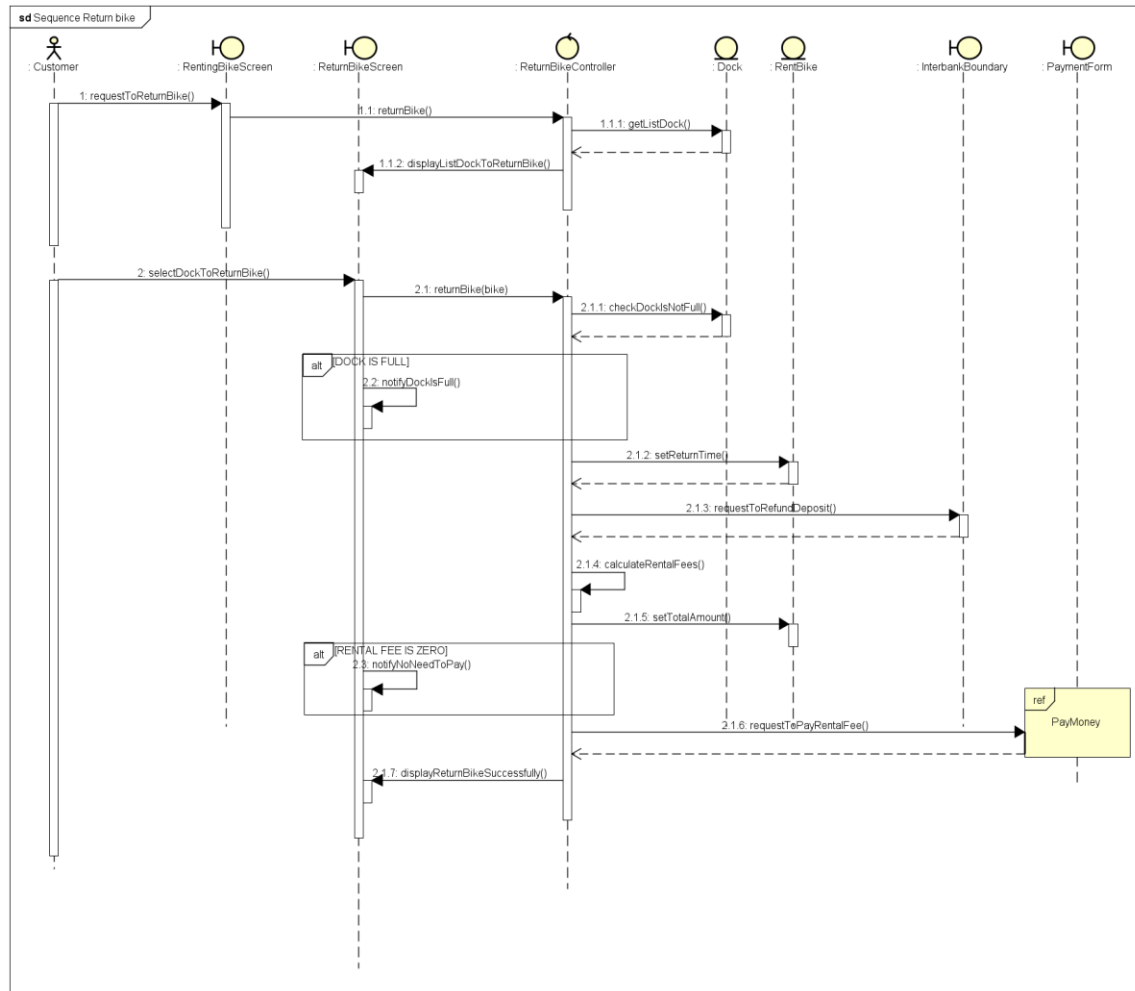
### 3.2.3. Use case “Pay money”



### 3.2.4. Use case “Rent bike”

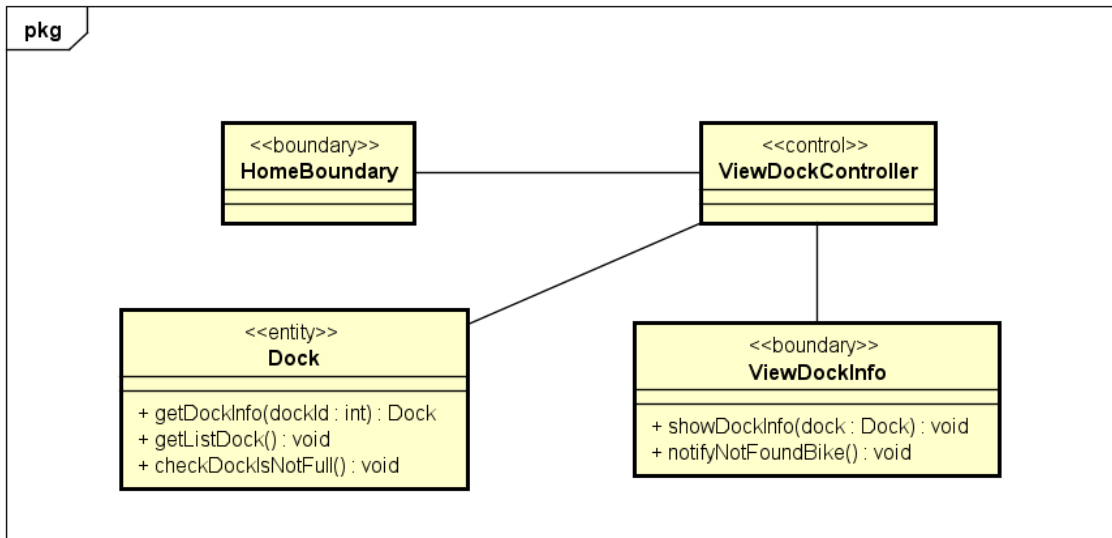


### 3.2.5. Use case “Return bike”

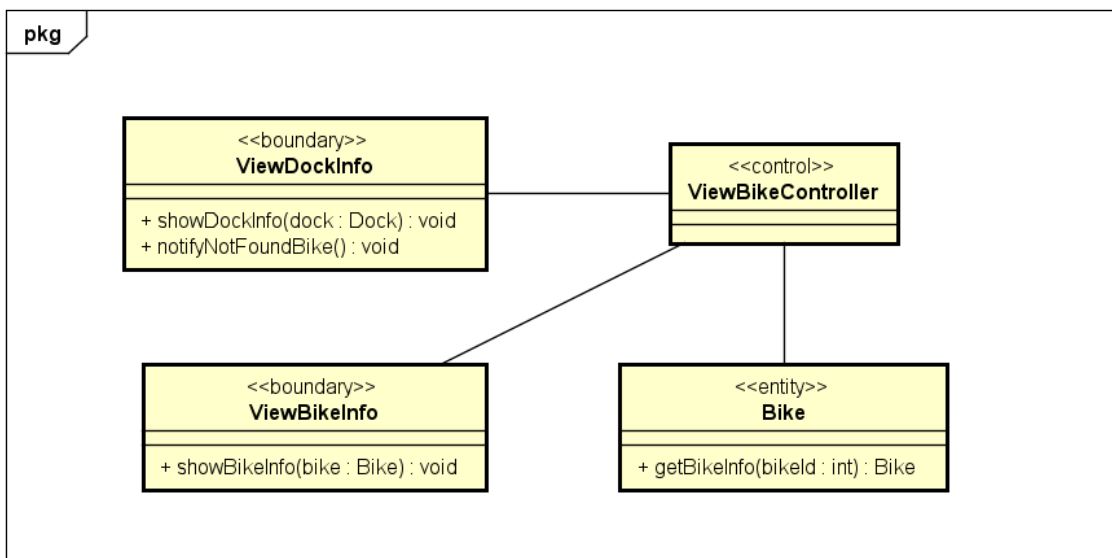


### 3.3 Analysis Class Diagrams

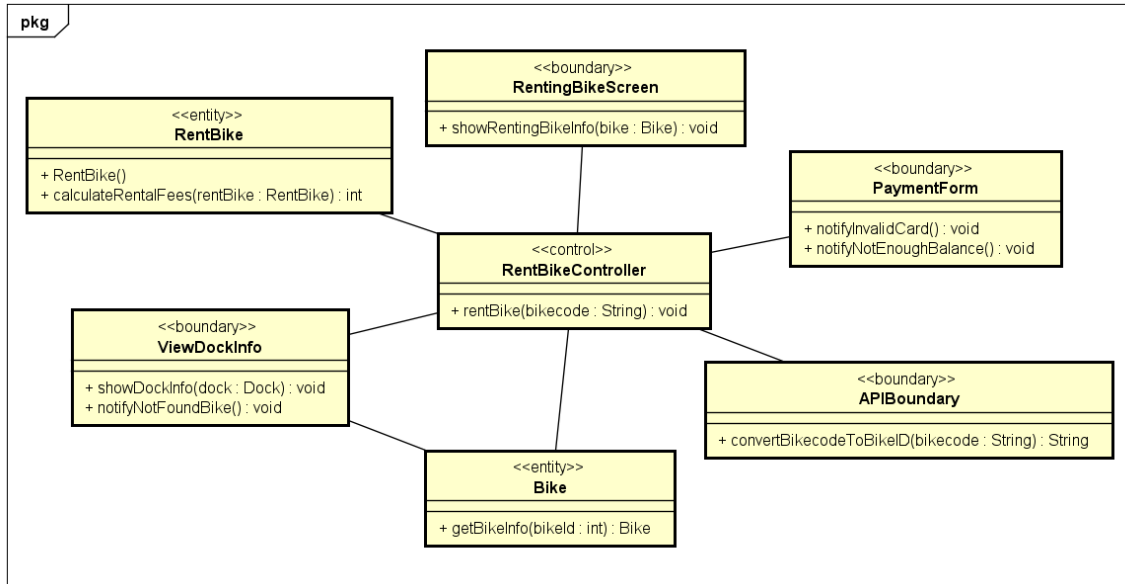
#### 3.3.1. Use case “View dock”



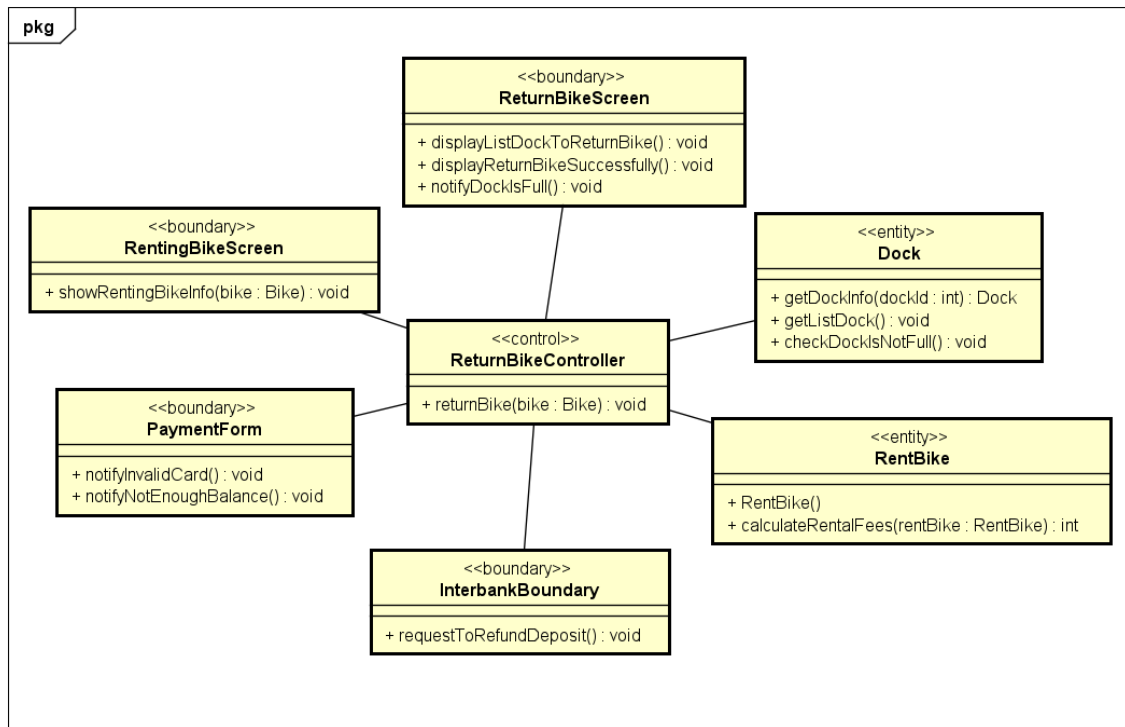
### 3.3.2. Use case “View bike”



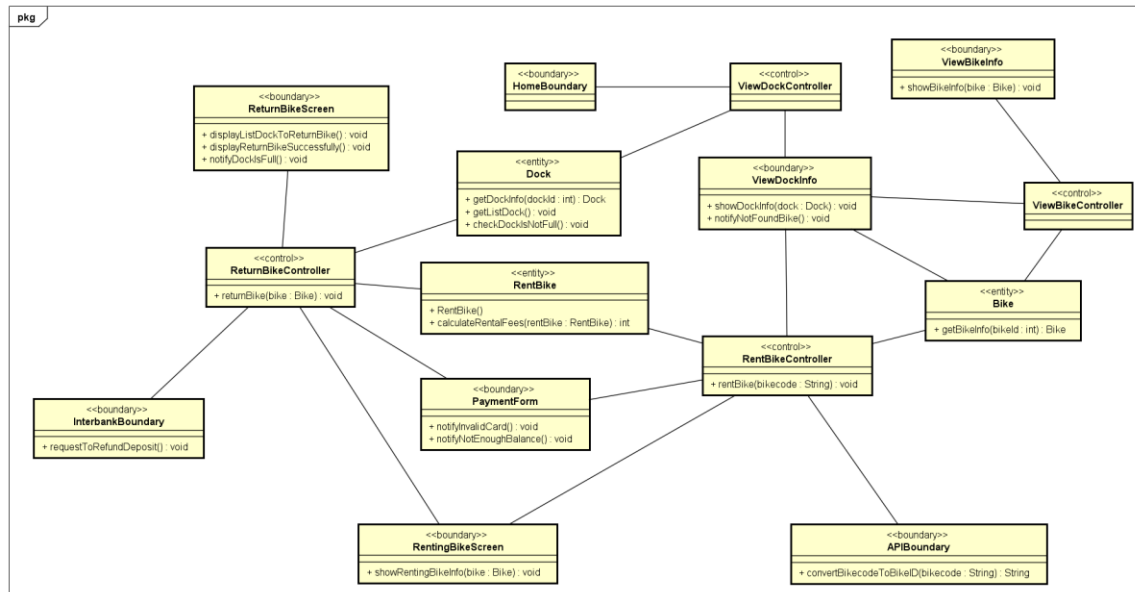
### 3.3.3. Use case “Rent bike”



### 3.3.4. Use case "Return bike"



### 3.4 Unified Analysis Class Diagram



### 3.5 Security Software Architecture

## 4 Detailed Design

### 4.1 User Interface Design

#### 4.1.1 Screen Configuration Standardization

##### *Screen*

Location of standard buttons: At the bottom (vertically) and in the middle (horizontally) of the frame

Location of the messages: Starting from the top vertically and in the middle horizontally of the frame down to the bottom.

Display of the screen title: The title is located at the top of the frame in the middle.

Consistency in expression of alphanumeric numbers: comma for separator of thousand while strings only consist of characters, digits, commas, dots, spaces, underscores, and hyphen symbol.

##### *Control*

Size of the text: medium size (mostly 24px). Font: Segoe UI. Color: #000000

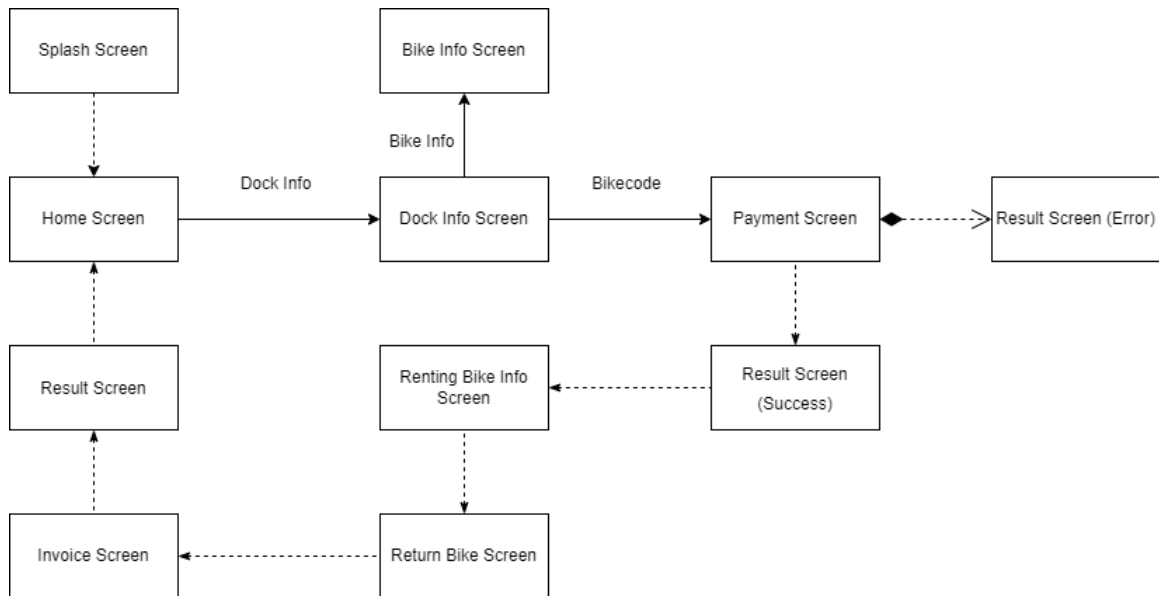
Input check process: Should check if it is empty or not. Next, check if the input is in the correct format or not

Sequence of moving the focus: There will be no stack frames. Each screen will be separated. However, the manual is considered a popup message, as the main screen cannot be operated while the manual screen is shown. After the opening screen, the app will start with splash screen, and then the first screen (home screen) will appear.

Sequences of the system screens:


1. Splash screen(first screen)
2. Home Screen
3. Dock Info Screen
4. Bike Info Screen
5. Payment Screen
6. Result Screen
7. Renting Bike Info Screen
8. Return Bike Screen
9. Invoice Screen
10. Result Screen

## 4.1.2 Screen Transition Diagrams



## 4.1.3 Screen Specifications

### 1. Home Screen

EcoBikeRental Software		Date of creation	Approved by	Reviewed by	Person in charge
Screen specification	Home screen	07/11/2021			Group1
		Control	Operation	Function	
		Area for input the dock information	Initial	Display the form (name,address)	
		Find button	Click	Find the dock info in database	
		Board for dock info	output	List dock info and list dock info result when click find button	




	Choose button	click	Go to dock infoscreen

### Defining the field attributes

Screen name	Home screen			
Item name	Number of digits (bytes)	Type	Field attribute	Remarks
Address	100	String	black	Right -justified
Name	50	String	black	Right -justified

## 2. Dock Info Screen

EcoBikeRental Software		Date of creation	Approved by	Reviewed by	Person in charge
Screen specification	Dock Info screen	07/11/2021			Group1
		Control	Operation	Function	
		Area for the dock information	Initial	Display the Dock info	
		Rent Button	Click	go to Payment Screen	
		Back button	Click	Back to home Screen	

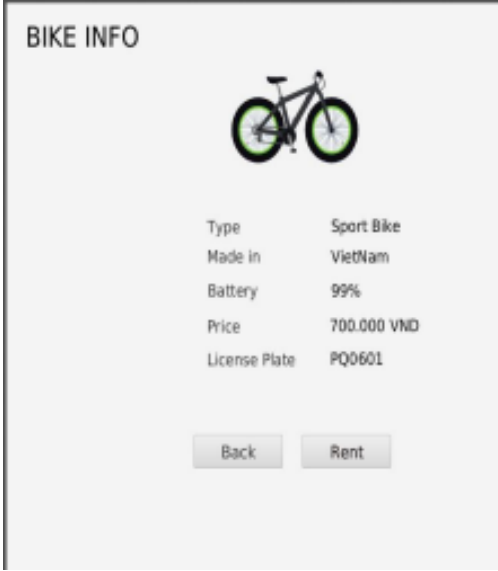
	Choose button	click	Go to bike infoscreen

### Defining the field attributes

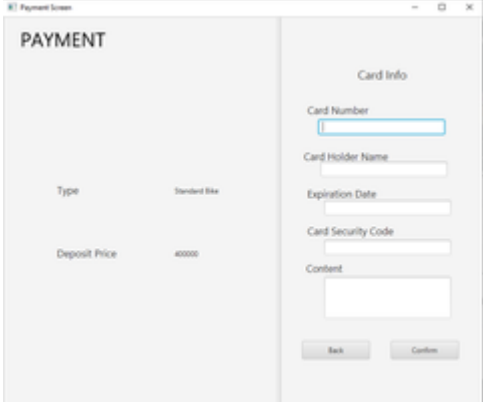
Screen name	Home screen			
Item name	Number of digits (bytes)	Type	Field attribute	Remarks
Barcode	100	String	black	Right -justified

### 3. Bike Info Screen

EcoBikeRental Software		Date of creation	Approved by	Reviewed by	Person in charge
Screen specification	Bike Info screen	07/11/2021			Group1
		Control	Operation	Function	
		Screen for displaying	Initial	Display bike info	
		button back	click	back to dock info screen	
		Display model of bike	Initial	Display bike info	

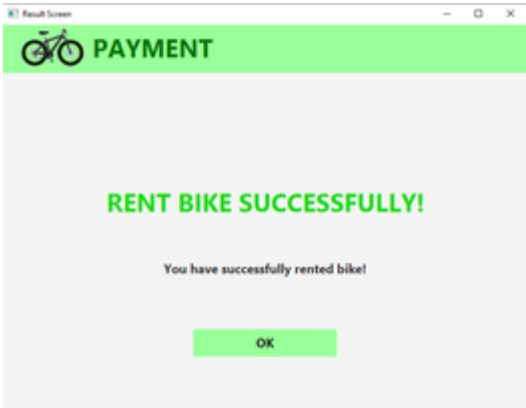
#### 4. Payment Screen

EcoBikeRental Software		Date of creation	Approved by	Reviewed by	Person in charge
Screen specification	Payment Screen	07/11/2021			Group1
		Control	Operation	Function	
		Area for the bike information	Initial	Display the bike information	
		Back button	Click	Back to bike info screen	
		Confirm button	click	Go to Result Screen	

### Defining the field attributes

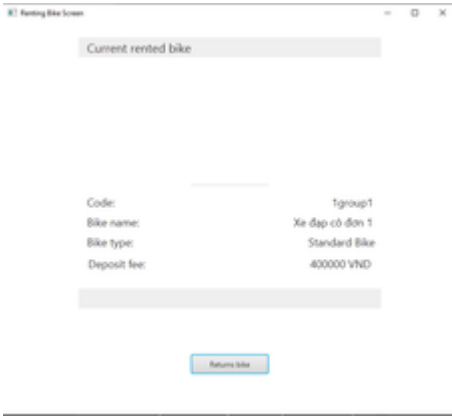
Screen name	Home screen			
Item name	Number of digits (bytes)	Type	Field attribute	Remarks
Card holder name	100	String	black	Right - justified
Card number	50	String	black	Right - justified
EX date	15	String	black	Right - justified
CVV code	6	int	black	Right - justified
Content	1000	String	black	Right - justified

### 5. Result Screen

EcoBikeRental Software		Date of creation	Approved by	Reviewed by	Person in charge
Screen specification	Result Screen	07/11/2021			Group1
		Control	Operation	Function	
		Area for payment information	Initial	Display the payment info	
		OK button	Click	Back to the payment screen or go to renting bike info screen	

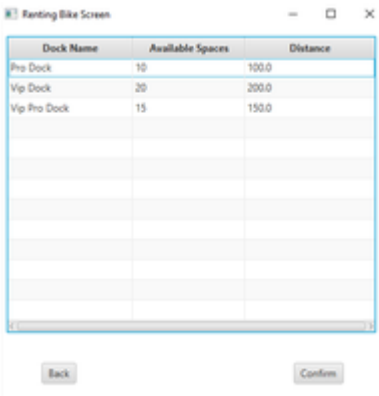
--	--	--	--

## 6. Renting Bike Info Screen

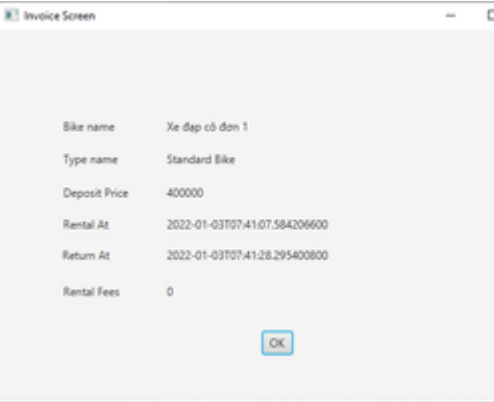
EcoBikeRental Software		Date of creation	Approved by	Reviewed by	Person in charge
Screen specification	Renting bike info screen	07/11/2021			Trần Hoàng Việt
		Control	Operation	Function	
		Area for the Renting bike Info	Initial	Display the bike info	
		return bike button	Click	Go to return bike screen	

## 7. Return Bike Screen

EcoBikeRental Software		Date of creation	Approved by	Reviewed by	Person in charge
Screen specification	Return bike screen	07/11/2021			Trần Hoàng Việt
		Control	Operation	Function	
		back button	Click	go to renting bike info screen	

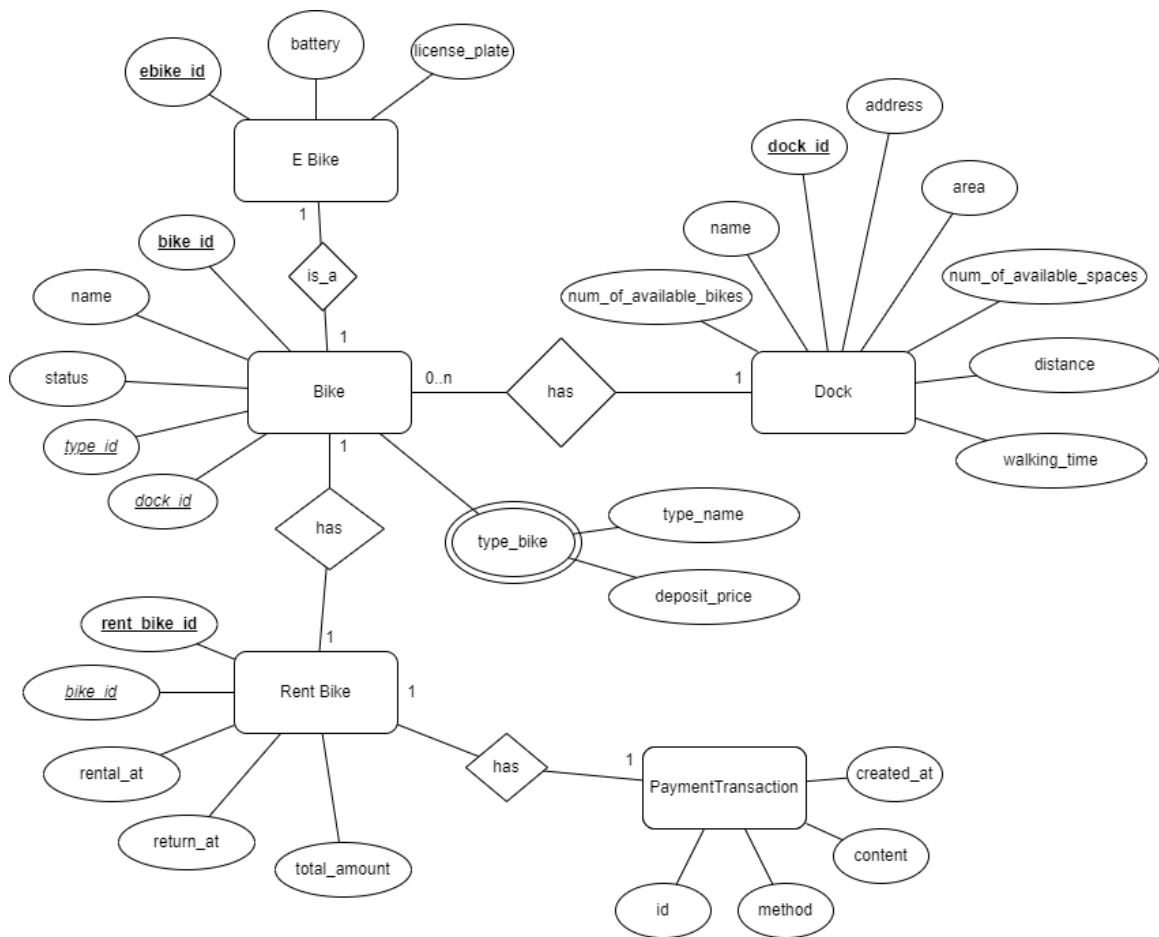
	Board for dock info	output	List dock info
	Confirm	click	Go to invoice screen

## 8. Invoice Screen

EcoBikeRental Software		Date of creation	Approved by	Reviewed by	Person in charge
Screen specification	Invoice Screen	07/11/2021			Trần Hoàng Việt
		Control	Operation	Function	
		Area for Invoice information	Initial	Display invoice info	
		Ok button	Click	Go to home screen	

## 4.2 Data Modeling

### 4.2.1 Conceptual Data Modeling



- Bảng dock(bãi xe) bao gồm các thuộc tính : tên, địa chỉ, diện tích, số xe đang ở trong bãi, số vị trí trống trong bãi (sử dụng khi khách trả xe), khoảng cách hiện tại của khách đến dock và kèm theo thời gian đi bộ. Một dock sẽ có thể có 0 hoặc nhiều xe.
- Bảng type\_bike(loại xe) bao gồm thuộc tính : tên loại xe và giá đặt cọc của từng loại. Một loại xe có thể bao gồm nhiều xe.
- Bảng bike bao gồm các thuộc tính : tên, trạng thái (đang được thuê hay không), id của dock mà bike đang ở đó, thuộc tính phức type\_bike như mô tả ở trên. Một xe chỉ thuộc về một loại xe và chỉ nằm ở một dock.

- Bảng ebike(xe điện) kế thừa bảng bike nhưng có thêm thuộc tính khác : biển số xe và lượng pin của xe. Một ebike ứng với một bike.
- Bảng rent\_bike dùng để lưu giao dịch thuê xe, bao gồm : id của xe đã thuê, thời gian bắt đầu thuê, thời gian trả xe và số tiền thuê. Khách hàng chỉ được dùng một thẻ để thuê một xe.
- Bảng payment\_transaction lưu thông tin giao dịch với ngân hàng. Một payment\_transaction sẽ ứng với một rent\_bike.

## 4.2.2 Database Design

### 4.2.2.1 Database Management Systems

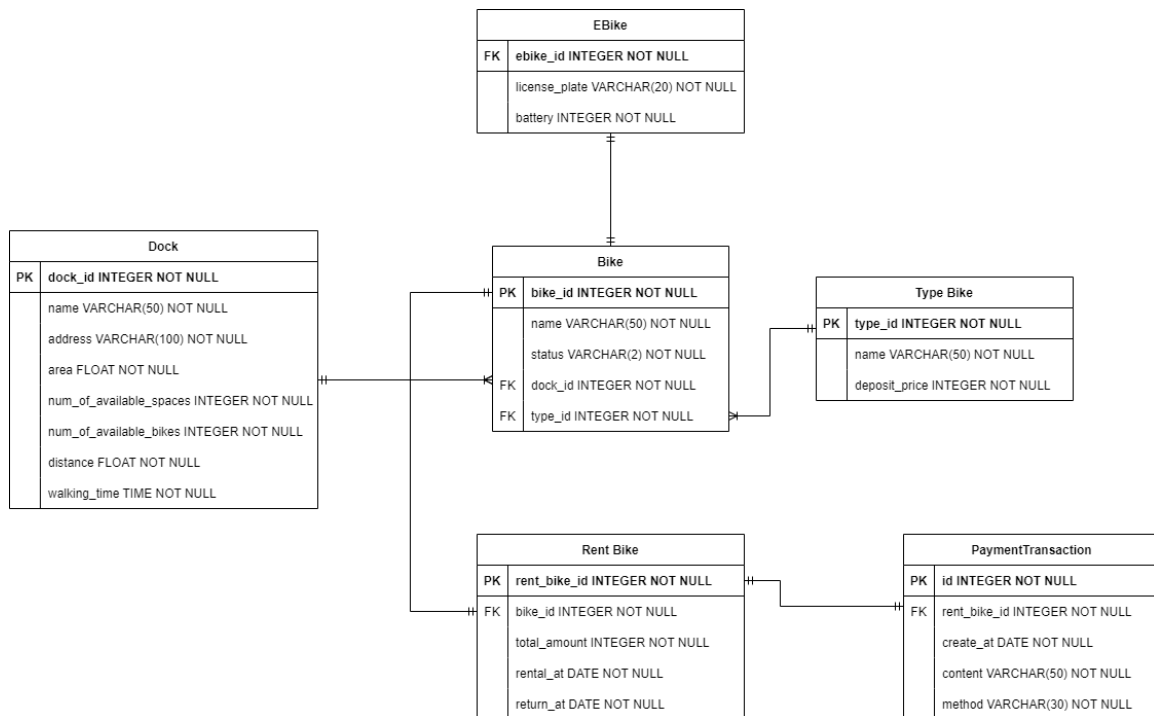
Group1 sử dụng postgresql làm DBMS trong dự án này vì:

Postgresql là hệ quản trị cơ sở dữ liệu mã nguồn mở

Postgresql tương thích với nhiều ngôn ngữ lập trình phổ biến hiện nay trong đó có Java.

Postgresql có thể hoạt động như một cơ sở dữ liệu phía máy chủ.

### 4.2.2.2 Logical Data Model





Trình tự tạo bảng : dock -> type\_bike -> bike -> ebike -> rent\_bike -> invoice -> payment\_transaction

#### 4.2.2.3 Physical Data Model

##### 1. Dock

#	PK	FK	Column name	Data type	Mandatory	Description
1	x		Dock_id	Integer	Yes	The dock's id
2			Name	Character Varying(50)	Yes	The dock's name
3			Address	Character Varying(100)	Yes	The dock's address
4			Area	Float	Yes	The dock's area
5			Num_of_available_spaces	Integer	Yes	Num of available spaces in dock in currently
6			Num_of_available_bikes	Integer	Yes	Num of available bikes in dock in currently
7			Distance	Float	Yes	Distance from customer's position to dock
8			Walking_time	Time	Yes	Walking time to the dock

##### 2. Type Bike

#	PK	FK	Column name	Data type	Mandatory	Description
1	x		Type_id	Integer	Yes	The type's id
2			Name	Character Varying(50)	Yes	The type's name
3			Deposit_price	Integer	Yes	Bike's deposit price

### 3. Bike

#	PK	FK	Column name	Data type	Mandatory	Description
1	x		Bike_id	Integer	Yes	The bike's id
2			Name	Character Varying(50)	Yes	The bike's name
3			Status	Character Varying(2)	Yes	The bike's status (R-reting or NR-not renting )
4		x	Dock_id	Integer	Yes	The id of dock that has bike
5		x	Type_id	Integer	Yes	The type's id of bike

### 4. EBike

#	PK	FK	Column name	Data type	Mandatory	Description
1		x	ebike_id	Integer	Yes	The ebike's id
2			Battery	Integer	Yes	The battery of ebike now
3			License_plate	Character Varying(20)	Yes	The ebike's license plate

### 5. Rent Bike

#	PK	FK	Column name	Data type	Mandatory	Description
1	x		Rent_bike_id	Integer	Yes	The rent bike's id
2		x	Bike_id	Integer	Yes	The bike's id
3			Rental_at	DATETIME	Yes	The time that started rent bike
4			Return_at	DATETIME	Yes	The time that return bike
5			Total_amount	Integer	Yes	Rental fees

## 6. Payment Transaction

#	PK	FK	Column name	Data type	Mandatory	Description
1.	x		Id	Integer	Yes	ID
2.			Method	VARCHAR(45)	Yes	Payment method
3.			Created_at	DATETIME	Yes	Date of creation
4.			Content	VARCHAR(45)	Yes	Transaction content
5.		x	Rent_bike_id	Integer	Yes	Rent bike ID

### DATABASE SCRIPT :

create table dock (

dock\_id integer NOT NULL,

```

    dock_name char(50) NOT NULL,
    dock_address char(100) NOT NULL,
    area float NOT NULL,
    num_of_available_spaces integer NOT NULL,
    num_of_available_bikes integer NOT NULL,
    distance float NOT NULL,
    walking_time time NOT NULL,
    Constraint dock_pk primary key(dock_id)
);

create table type_bike (
    type_id integer NOT NULL,
    type_name char(50) NOT NULL,
    deposit_price integer NOT NULL,
    Constraint type_pk primary key(type_id)
);

create table bike (
    bike_id integer NOT NULL,
    dock_id integer NOT NULL,
    name char(50) NOT NULL,
    status char(2) NOT NULL,
    type_id integer NOT NULL,
    Constraint bike_pk primary key(bike_id),
    Constraint bike_fk_dock foreign key(dock_id) references dock(dock_id),
    Constraint bike_fk_type foreign key(type_id) references type_bike(type_id)
);

create table ebike (
    ebike_id integer NOT NULL,

```

```

        battery integer NOT NULL,
        license_plate char(20) NOT NULL,
        Constraint ebike_pk primary key(ebike_id),
        Constraint ebike_fk foreign key(ebike_id) references bike(bike_id)
    );

create table rent_bike (
    rent_bike_id serial primary key NOT NULL,
    bike_id integer NOT NULL,
    rental_at DATETIME NOT NULL,
    return_at DATETIME NOT NULL,
    total_amount integer NOT NULL,
    Constraint rent_bike_fk foreign key(bike_id) references bike(bike_id)
);

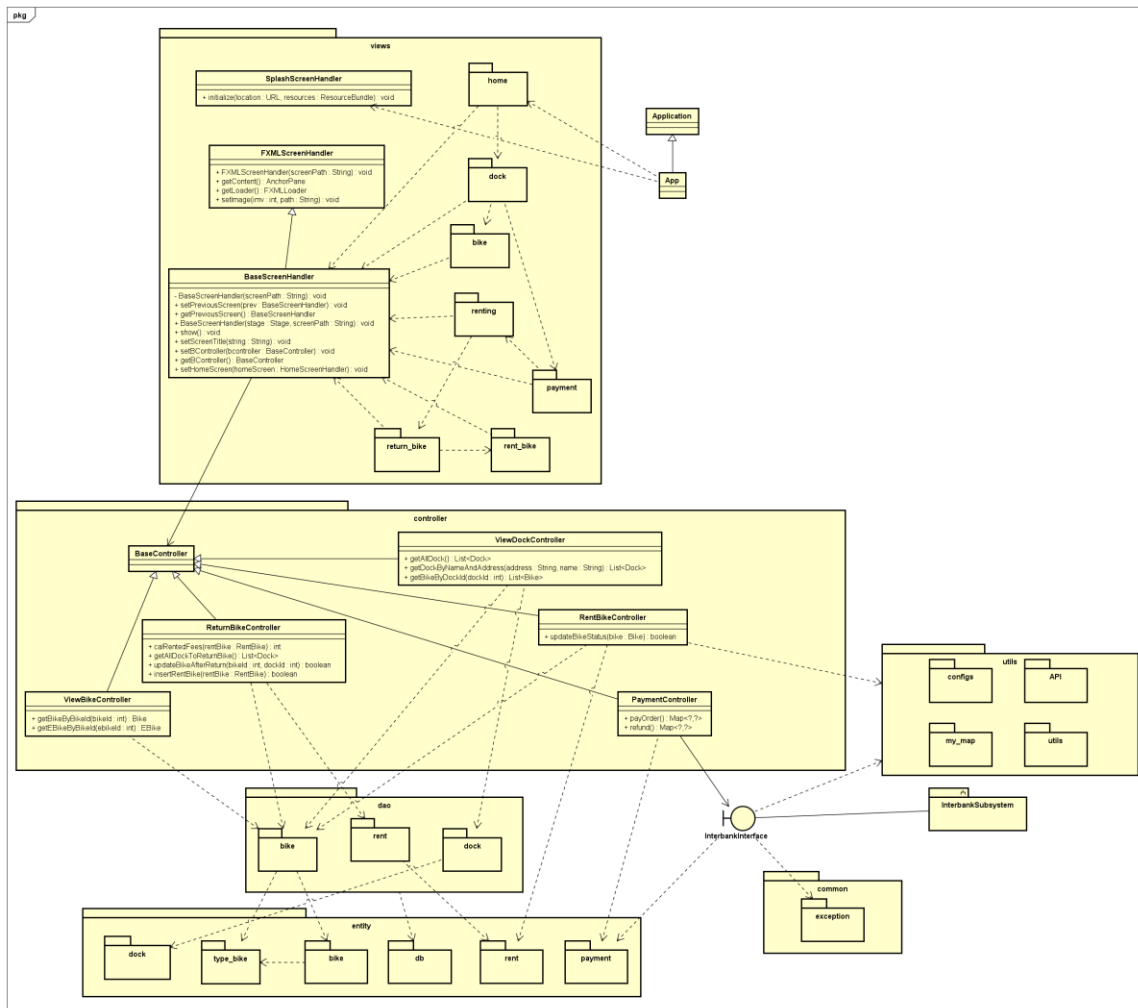
create table payment_transaction (
    id serial primary key NOT NULL,
    content varchar(45) NOT NULL,
    create_at DATETIME NOT NULL,
    method varchar(45) NOT NULL,
    rent_bike_id integer NOT NULL,
    Constraint payment_transaction_id foreign key(rent_bike_id) references
rent_bike(rent_bike_id)
);

```

## 4.3 Non-Database Management System Files

## 4.4 Class Design

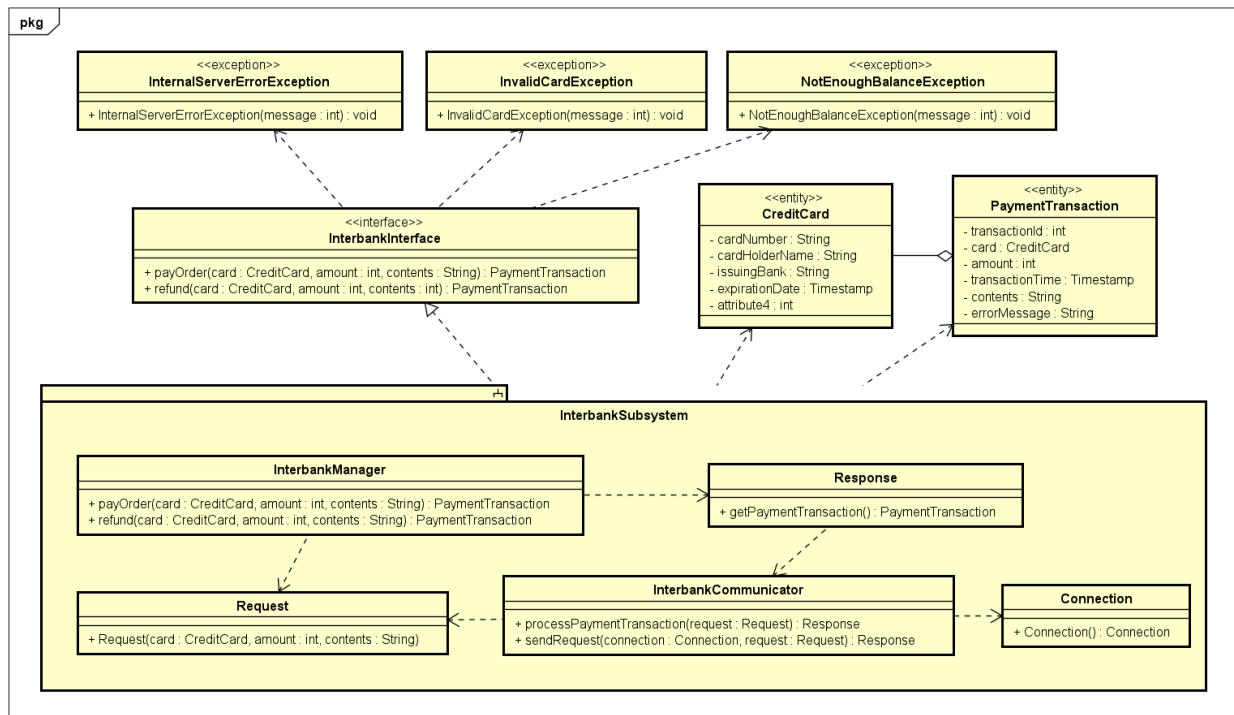
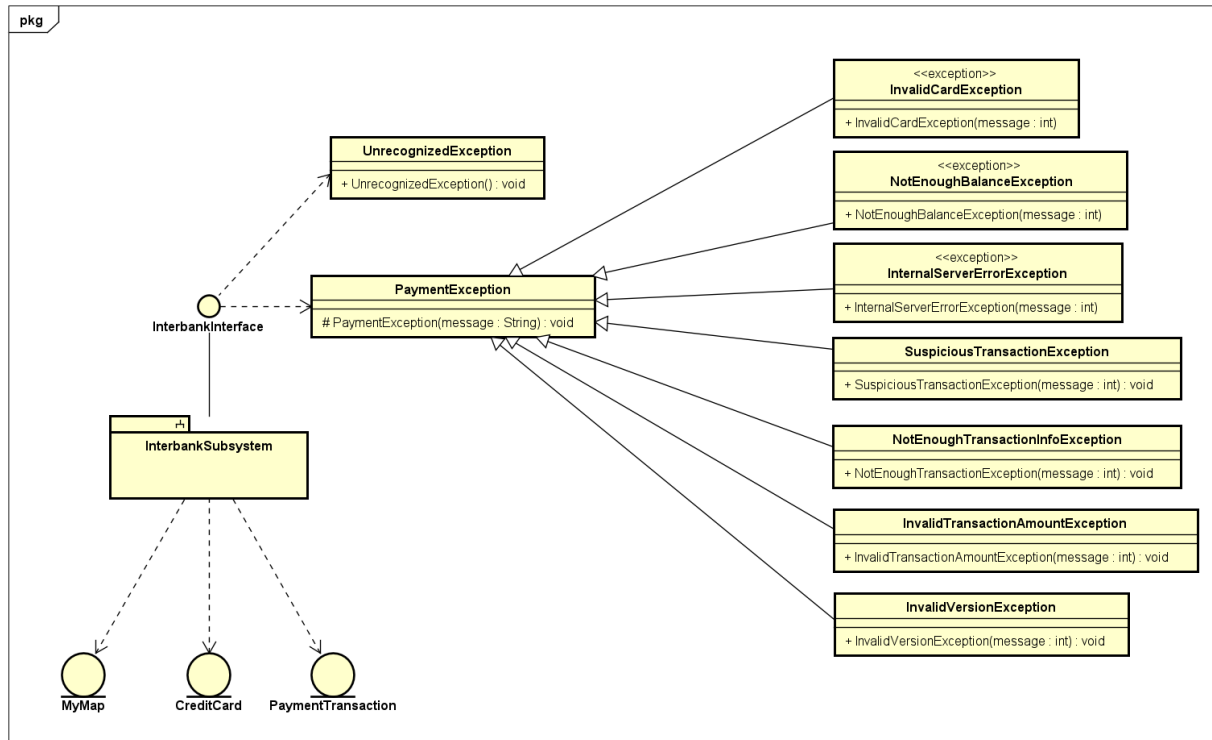
### 4.4.1 General Class Diagram



## 4.4.2 Class Diagrams

### 4.4.2.1 Class Diagram for Package A

### 4.4.2.2 Class Diagram for Subsystem B



### 4.4.3 Class Design

#### 4.4.3.1 Class “ViewDockController”



#### Attribute

None

#### Operation

#	Name	Return type	Description (purpose)
1	getDockInfo	Dock	Return result of dock when user chooses a dock in screen

#### Parameter:

- dockId - id of dock

#### Exception:

None

#### Method

None

#### State

None



#### 4.4.3.2 Class “ViewBikeController”



##### Attribute

None

##### Operation

#	Name	Return type	Description
1	getBikeInfo	Bike	Return result of bike when user chooses a bike in screen

##### Parameter:

- bikeCode - system's bike code

##### Exception:

None

##### Method

- getBikeId: Use user-entered code, convert to system's bike code.

#### 4.4.3.3. Class “RentBikeController”



##### Attribute

None

##### Operation

#	Name	Return type	Description (purpose)
---	------	-------------	-----------------------

1	rentBike	void	User rent bike
---	----------	------	----------------

*Parameter:*

- bikecode – code of bike that the user wants to rent

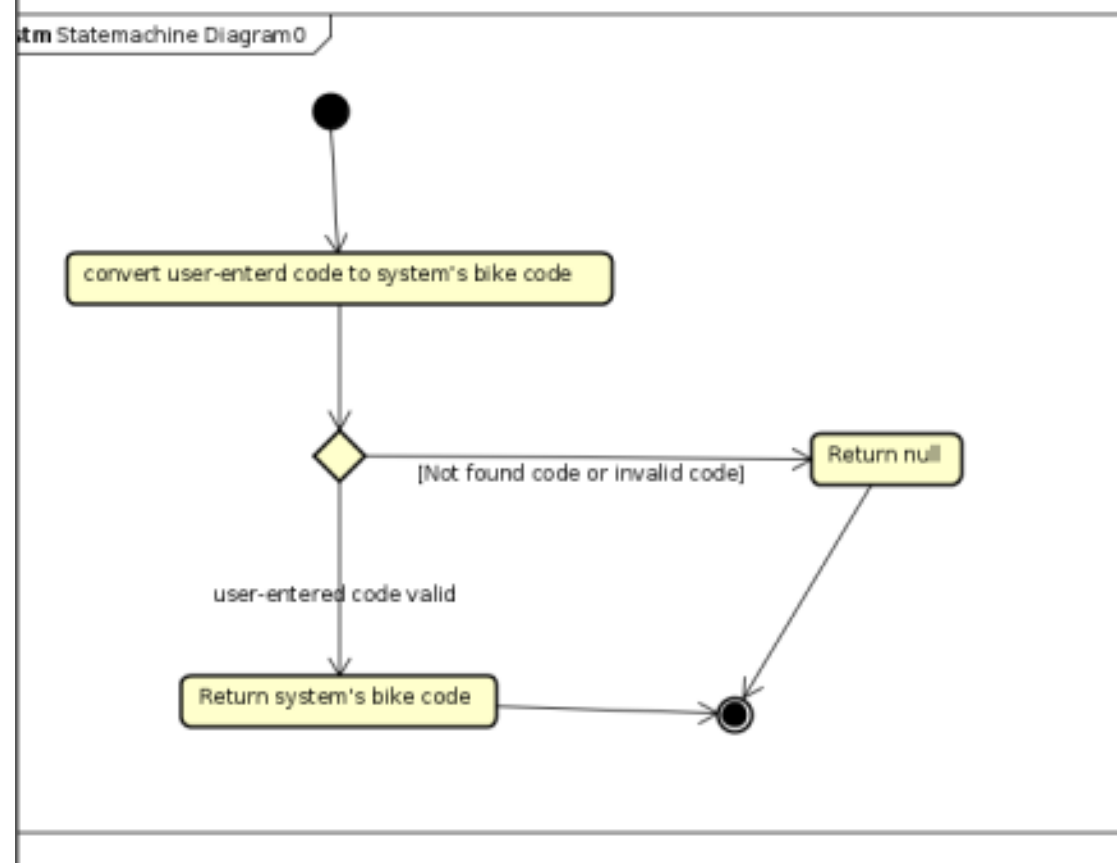
*Exception:*

None

**Method**

None

**State Machine**



#### 4.4.3.4. Class “ReturnBikeController”



**Attribute**

None

**Operation**

#	Name	Return type	Description (purpose)
1	returnBike	void	User return bike

*Parameter:*

- bike - the rented bike information

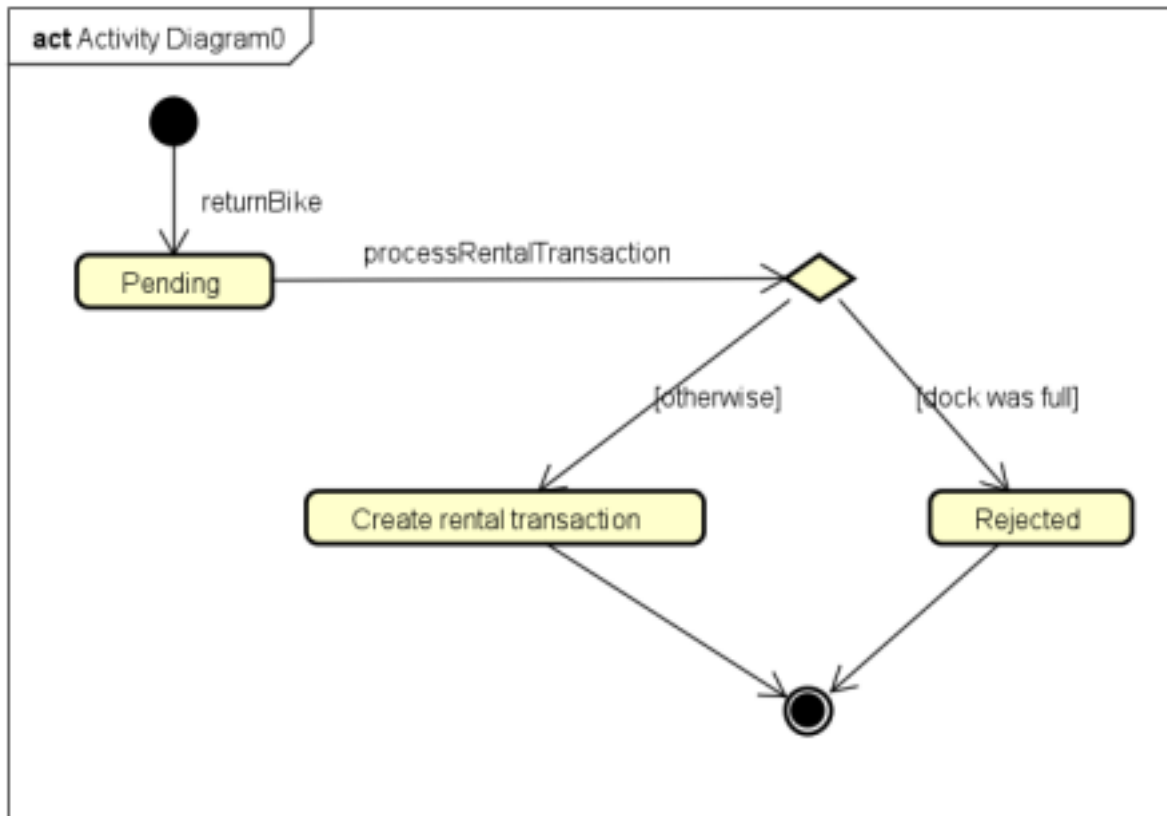
*Exception:*

None

**Method**

None

**State**



#### 4.4.3.5. Class “InterbankInterface”

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

#### Operation

#	Name	Return type	Description (purpose)
1	payOrder	PaymentTransaction	Pay order, and then return the paymenttransaction
2	refund	PaymentTransaction	Refund, and then return the payment transaction

#### Parameter:

- card – the credit card used for payment/refund

- amount – the amount to pay/refund
- contents – the transaction contents

*Exception:*

- PaymentException – if responded with a pre-defined error code
- UnrecognizedException – if responded with an unknown error code or something goes wrong

## Method

None

## State

None

### 4.4.3.6. Class “PaymentController”

<<control>> PaymentController	
- card : CreditCard	
- interbank : InterbankInterface	
+ payOrder(amount : int, contents : String, cardNumber : String, cardHolderName : String, expirationDate : String, securityCode : String) : Map<String,String>	
- getExpirationDate(date : String) : String	

## Attribute

#	Name	Data type	Default value	Description
1	card	CreditCard	NULL	Represent the card used for payment
2	interbank	InterbankInterface	NULL	Represent the Interbank subsystem

## Operation

#	Name	Return type	Description (purpose)
1	payOrder	Map<String,String>	Pay order, and then return the result with a message

*Parameter:*

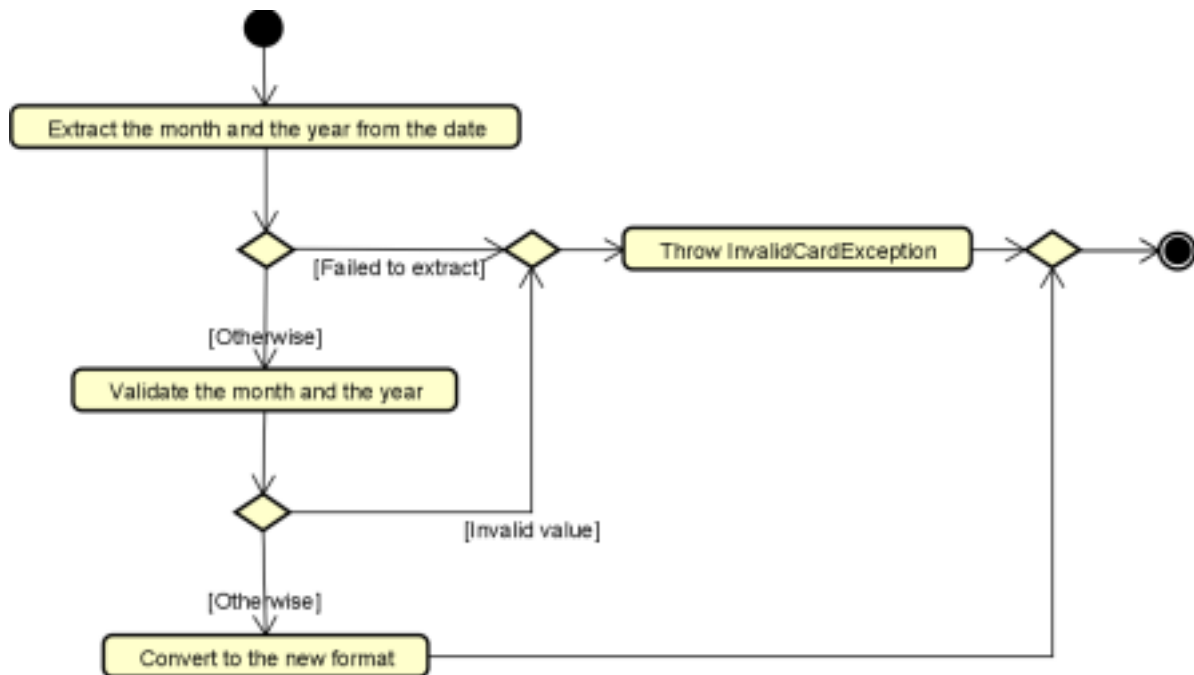
- amount - the amount to pay
- contents - the transaction contents
- cardNumber - the card number
- cardHolderName - the card holder name
- expirationDate - the expiration date in the format "mm/yy"
- securityCode - the cvv/cvc code of the credit card

*Exception:*

None

## Method

- getExpirationDate: Given the String "date" representing the expiration date in the format "mm/yy", this method convert it into the required format "mmyy." The algorithm is illustrated as follows.



**State**

None

## 5 Design Considerations

*<Describe issues which need to be addressed or resolved before attempting to devise a complete design solution>*

### 5.1 Goals and Guidelines

*<Describe any goals, guidelines, principles, or priorities which dominate or embody the design of the system and its software.*

*Examples of such goals might be: an emphasis on speed versus memory use; or working, looking, or “feeling” like an existing product.*

*Guidelines include coding guidelines and conventions.*

*For each such goal or guideline, describe the reason for its desirability unless it is implicitly obvious.*

*Describe any design policies and/or tactics that do not have sweeping architectural implications (meaning they would not significantly affect the overall organization of the system and its high-level structures), but which nonetheless affect the details of the interface and/or implementation of various aspects of the system (e.g., choice of which specific product to use)>*

### 5.2 Architectural Strategies

*<Describe any design decisions and/or strategies that affect the overall organization of the system and its higher-level structures. These strategies should provide insight into the key abstractions and mechanisms used in the system architecture. Describe the reasoning employed for each decision and/or strategy (possibly referring to previously stated design goals and principles) and how any design goals or priorities were balanced or traded-off.*

*Examples of design decisions might concern (but are not limited to) things like the following:*

- *Use of a particular type of product (programming language, database, library, commercial off-the-shelf (COTS) product, etc.)*
- *Reuse of existing software components to implement various parts/features of the system*
- *Future plans for extending or enhancing the software*
- *User interface paradigms (or system input and output models)*

- *Hardware and/or software interface paradigms*
- *Error detection and recovery*
- *Memory management policies*
- *External databases and/or data storage management and persistence*
- *Distributed data or control over a network*
- *Generalized approaches to control*
- *Concurrency and synchronization*
- *Communication mechanisms*
- *Management of other resources*

>

### **5.3 Coupling and Cohesion**

Coupling :

Stamp Coupling :Nhóm đã cố gắng thiết kế các module để sao cho việc giao tiếp giữa các module trong hệ thống qua việc truyền các tham số dữ liệu nguyên thủy. Việc CRUD các đối tượng cũng thực hiện qua việc lấy tham số là dữ liệu nguyên thủy từ các thành phần khác

Cohesion :

Functional Cohesion: Việc thực hiện theo nguyên lý SOLID trong đó nguyên lý S được nhóm áp dụng trong project , đảm bảo cho các thành phần trong chương trình đảm nhiệm những vai trò cụ thể duy nhất. Từ đó các thành phần trong mỗi nhỏ trong đó tập trung xây dựng các chức năng để thực hiện vai trò cụ thể của module đó, làm tăng sự gắn kết giữa các thành phần hơn

### **5.4 Design Principles**

*Single Responsibility*

*Nguyên lí này được nhóm chú trọng triển khai nhiều nhất. Việc áp dụng kiến trúc MVC cùng với đó là việc phân tích bài toán theo từng đối tượng, chức năng cụ thể. Nên thiết kế phần nào thỏa mãn SRP*

### **5.5 Design Patterns**

*Trong project này, nhóm sử dụng mẫu thiết kế Singleton cho một số thành phần.*



Việc tương tác giữa hệ thống với cơ sở dữ liệu khá là thường xuyên, vì thế không cứ mỗi lần kết nối với DBMS ta lại tạo ra một thực thể mới để làm việc này.

Một điều nữa mà nhóm sử dụng Singleton là nhóm xây dựng một lớp DAO (Data access object) những lớp này có nhiệm vụ tương tác với dữ liệu . Chức năng chủ yếu là CRUD các đối tượng với DB. Vì chức năng hoàn toàn giống nhau, nên khi cần CRUD ta không cần tạo ra một thể hiện mới của lớp DAO này, mà chỉ cần một thể hiện duy nhất để làm việc này.

Việc áp dụng mẫu thiết kế Singleton:

- Giúp cho chương trình trở lên đơn giản hơn
- Giảm thiểu chi phí bộ nhớ cho chương trình.
- Khi code ta cũng không cần quan tâm nhiều đến việc cần phải khởi tạo một thực thể mới.
- Việc thiết kế Singleton, giúp ta có thể áp dụng Dependency Injection vào trong chương trình

Khi áp dụng vào code : nhóm sử dụng Lazy Singleton để tạo một Singleton cho lớp

```
public class LazyInitializedSingleton {  
  
    private static LazyInitializedSingleton instance;  
  
    private LazyInitializedSingleton(){}  
  
    public static LazyInitializedSingleton getInstance(){  
        if(instance == null){  
            instance = new LazyInitializedSingleton();  
        }  
        return instance;  
    }  
}
```