

**HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY**  
**School of Information and communications technology**



Software Design Document

Version 1.3

**Phân tích và thiết kế hệ thống**

**EcoBikeRental**

**Subject: Thiết kế và xây dựng phần mềm**

Nhóm 3:

Trần Trọng Hiệp – MSSV: 20190051

Trần Lê Hiệp – MSSV: 20190050

Lê Huy Hoàng – MSSV: 20190053

Giảng viên hướng dẫn: TS. Nguyễn Thị Thu Trang

Hà Nội, ngày 5 tháng 2 năm 2023

# Mục lục

<b>1 Introduction</b>	<b>4</b>
1.1 Objective	4
1.2 Scope	4
1.3 Glossary	4
1.4 References	4
<b>2 Overall Description</b>	<b>4</b>
2.1 General Overview	4
2.2 Assumptions/Constraints/Risks	4
<b>3 System Architecture and Architecture Design</b>	<b>4</b>
3.1 Architectural Patterns	4
3.2 Interaction Diagrams	5
3.3 Analysis Class Diagrams	10
3.4 Unified Analysis Class Diagram	14
3.5 Security Software Architecture	14
<b>4 Detailed Design</b>	<b>14</b>
4.1 User Interface Design	14
4.1.1 Screen Configuration Standardization	14
4.1.2 Screen Transition Diagrams	16
4.1.3 Screen Specifications	16
4.2 Data Modeling	21
4.2.1 Conceptual Data Modeling	21
4.2.2 Database Design	21
4.3 Non-Database Management System Files	23
4.4 Class Design	28
4.4.1 General Class Diagram	28
4.4.2 Class Diagrams	28
4.4.3 Class Design	32
<b>5 Design Considerations</b>	<b>35</b>
5.1 Goals and Guidelines	35
5.2 Architectural Strategies	35
5.3 Coupling and Cohesion	35
5.4 Design Principles	36

<b>5.4.1 Single Responsibility Principle .....</b>	<b>36</b>
<b>5.4.2 Open/Closed Principle .....</b>	<b>36</b>
<b>5.4.3 Liskov substitution principle .....</b>	<b>37</b>
<b>5.4.4 Interface segregation principle .....</b>	<b>38</b>
<b>5.4.5 Dependency Inversion principle .....</b>	<b>38</b>
<b>5.5 Design Patterns .....</b>	<b>38</b>
<b>5.5.1 Singleton .....</b>	<b>38</b>
<b>5.5.2 DAO - Data Access Object pattern .....</b>	<b>40</b>
<b>5.5.3 Strategy Pattern .....</b>	<b>41</b>

## **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**

Ứng dụng EcoBike Rental giả lập cho việc khách hàng có thể thuê và trả xe theo mô hình như trên mà không xét đến các chức năng như xác thực người dùng, chỉ quan tâm đến các chức năng liên quan đến thuê xe và trả xe.

### **1.3 Glossary**

Bảng chú thích các thuật ngữ:

### **1.4 References**

Tài liệu tham khảo bao gồm: javaFX doc, tài liệu được cung cấp trong học phần thiết kế và xây dựng phần mềm.

## **2 Overall Description**

### **2.1 General Overview**

Phần mềm được thiết kế theo thiết kế 3 tầng, dùng trên desktop. Thiết kế này không thích hợp với người dùng trong thực tế vì không ai mang PC hay laptop đi để thuê xe đạp. Bên cạnh đó, laptop hay PC cũng không có chức năng quét barcode hay có mạng liên tục. Tuy nhiên, với khuôn khổ của bài tập, phần mềm được thiết kế phù hợp để demo.

### **2.2 Assumptions/Constraints/Risks**

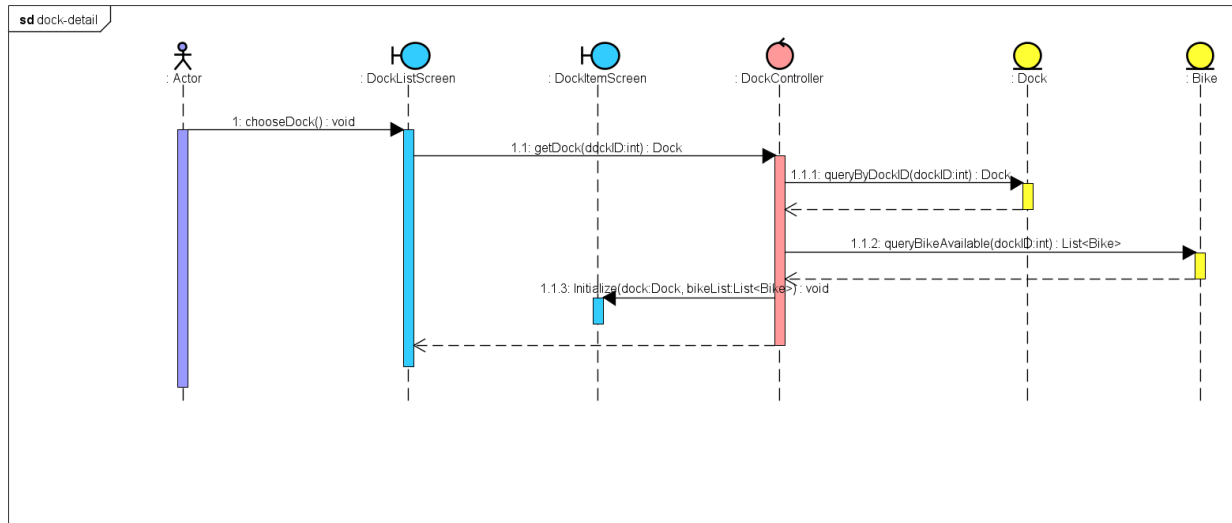
Không có.

## **3 System Architecture and Architecture Design**

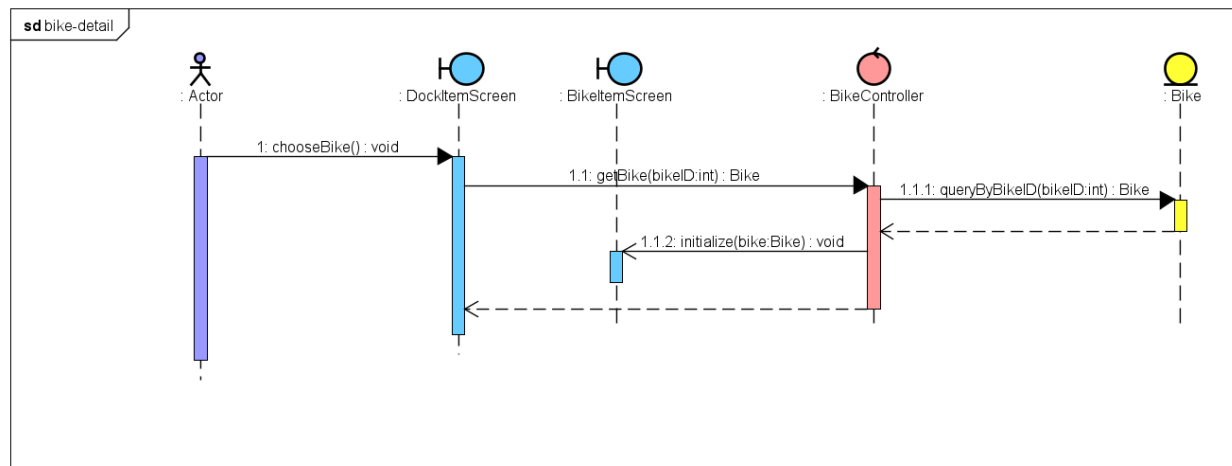
### **3.1 Architectural Patterns**

Nhóm chọn thiết kế theo kiến trúc 3 tầng

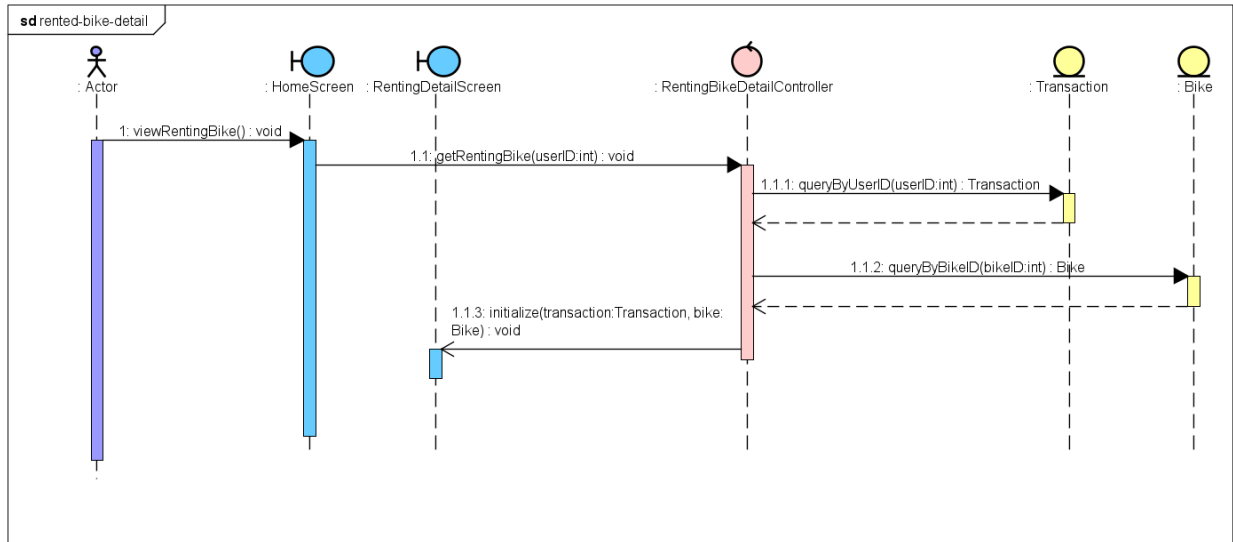
## 3.2 Interaction Diagrams



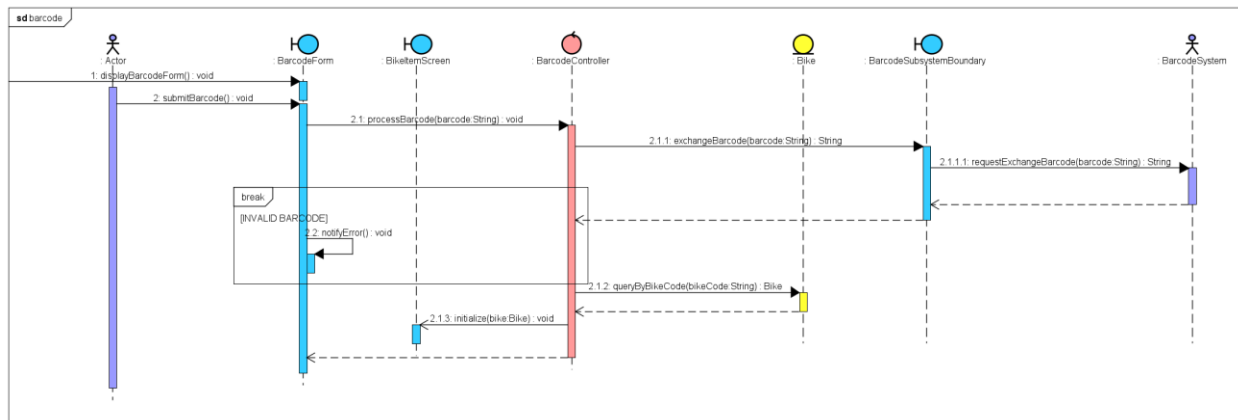
Hình 1 Xem chi tiết bãi xe



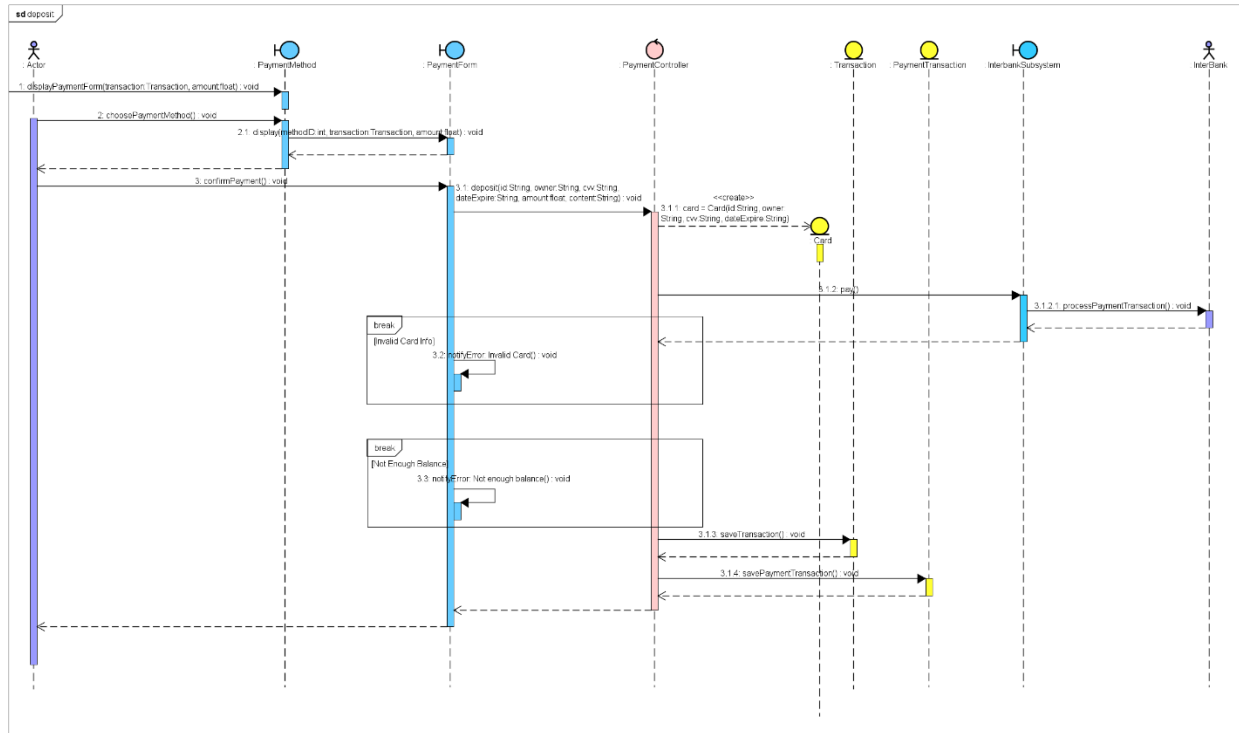
Hình 2 Xem chi tiết xe



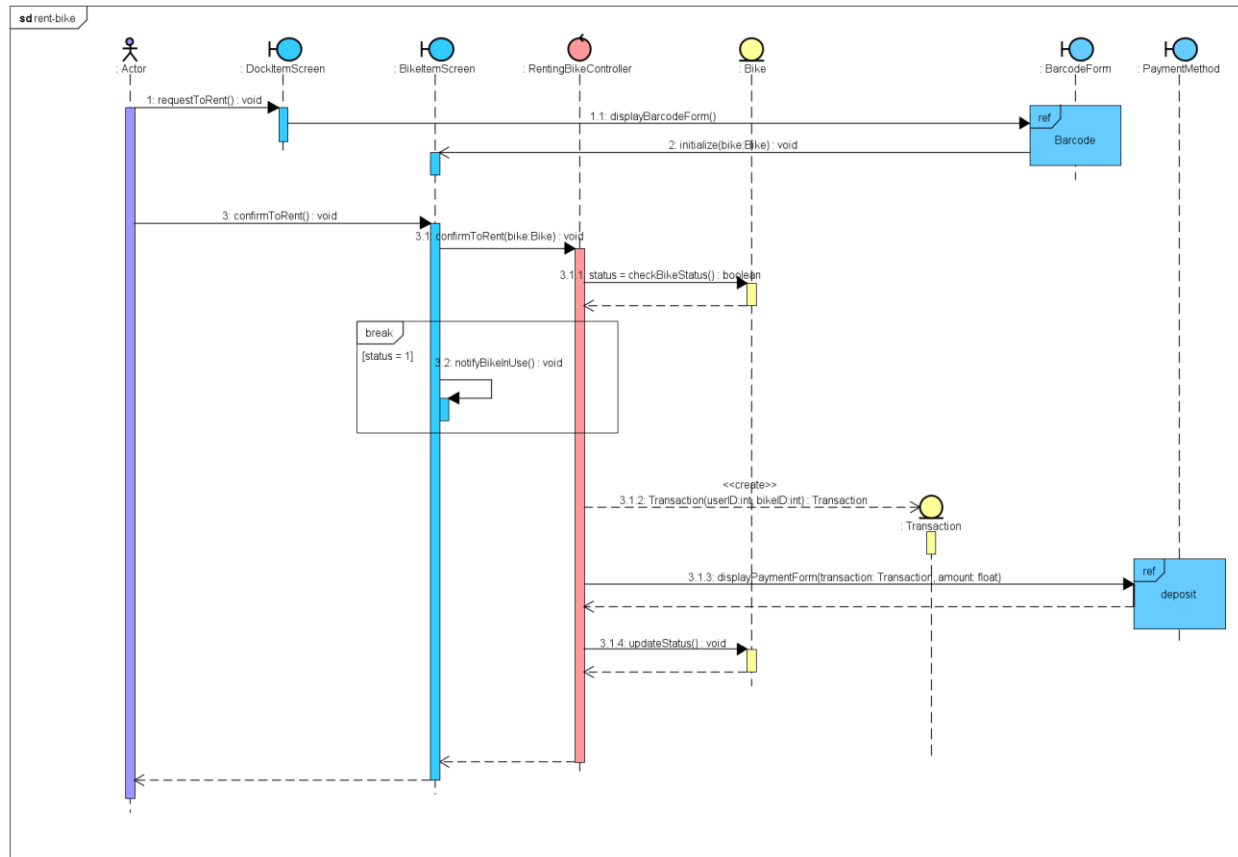
Hình 3 Xem xe đang thuê



Hình 4 Chuyển đổi barcode

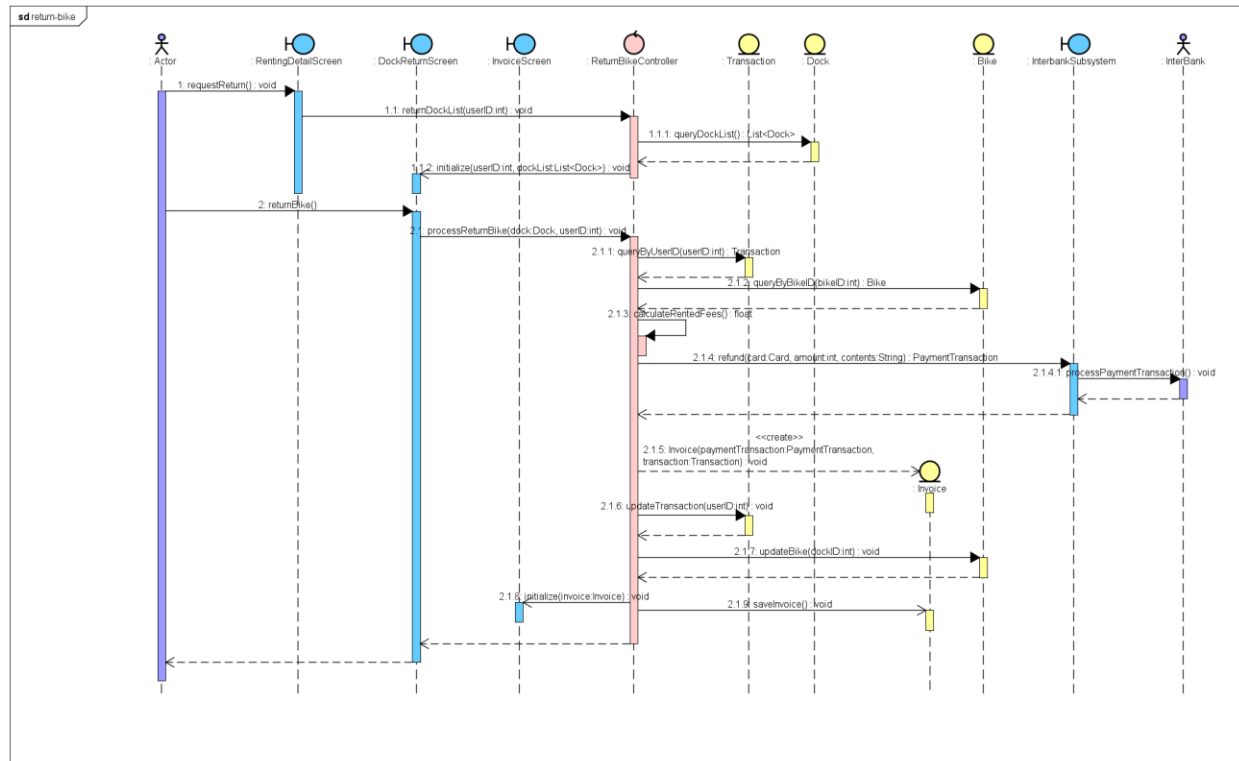


Hình 5 Đặt cọc



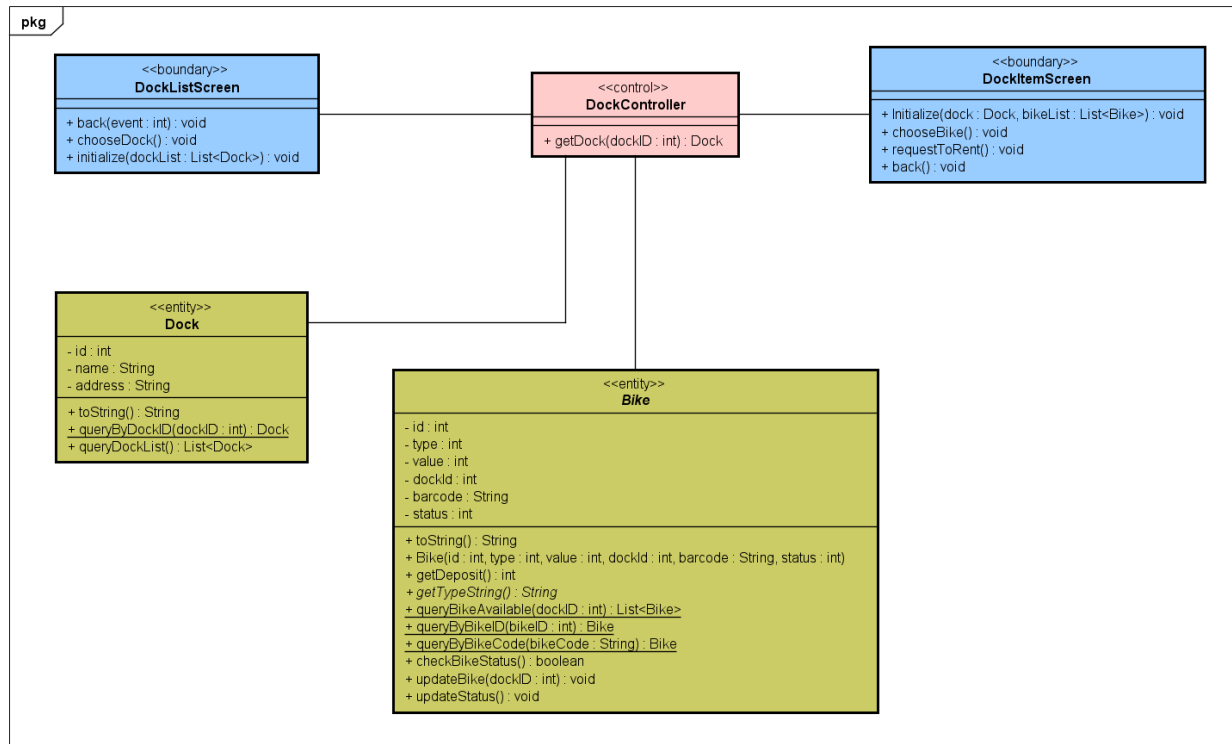
Hình 6 Thuê xe



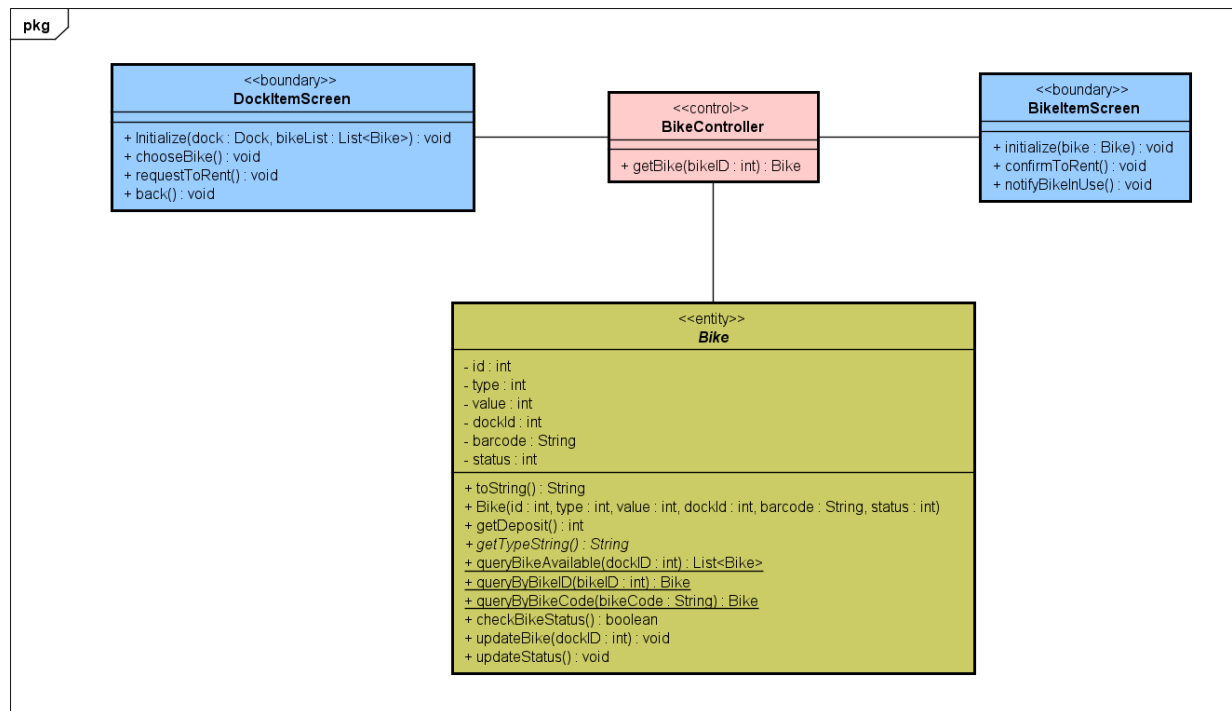


Hình 7 Trả xe

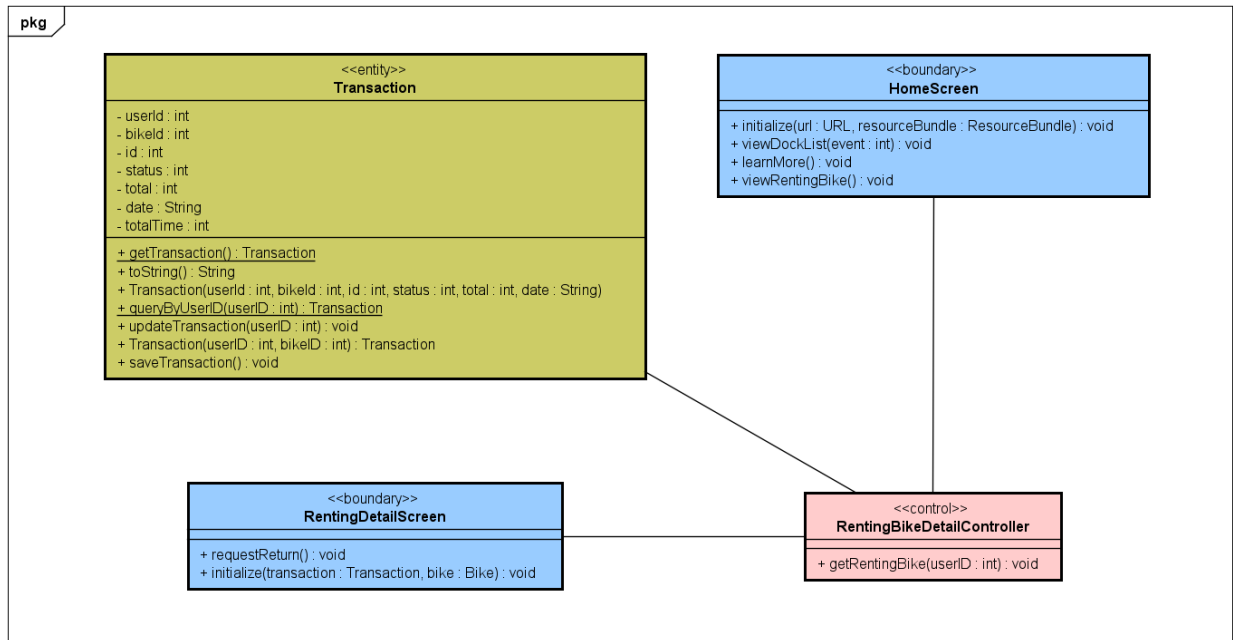
### 3.3 Analysis Class Diagrams



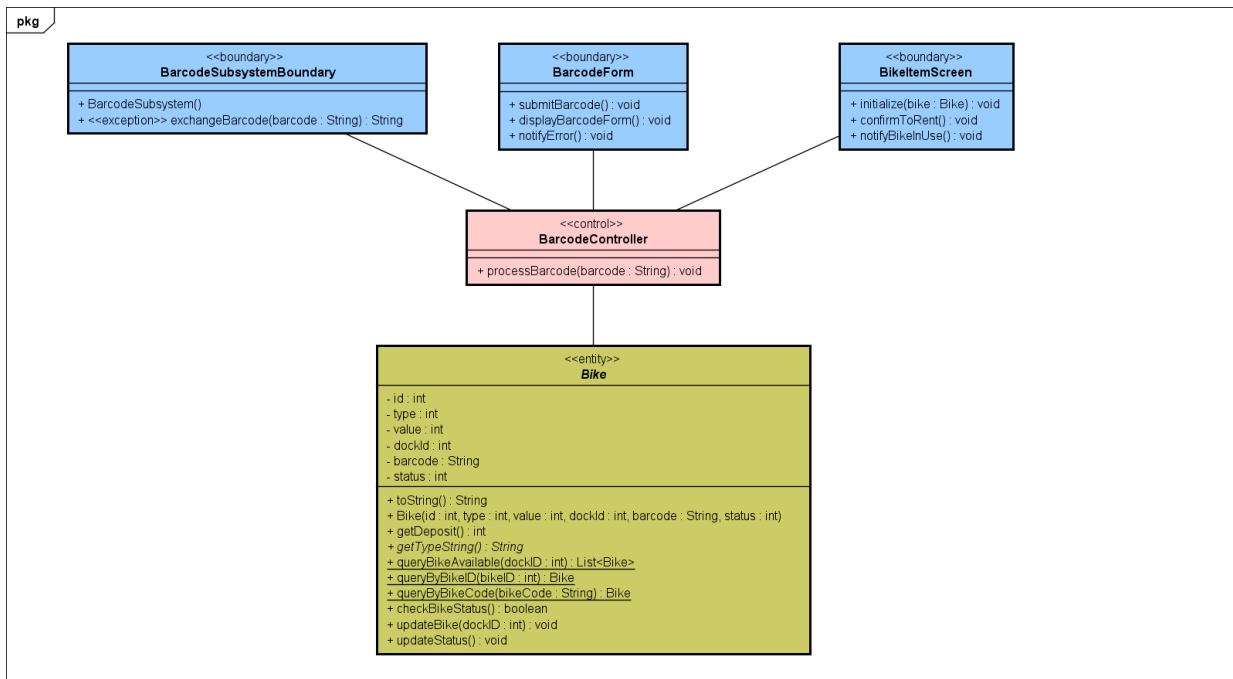
Hình 8 Xem chi tiết bãi xe



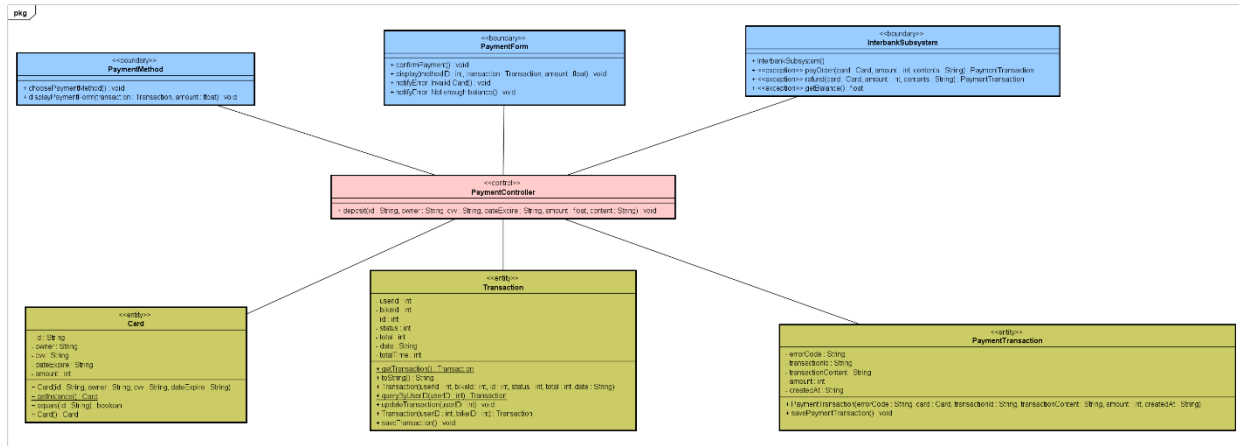
Hình 9 Xem chi tiết xe



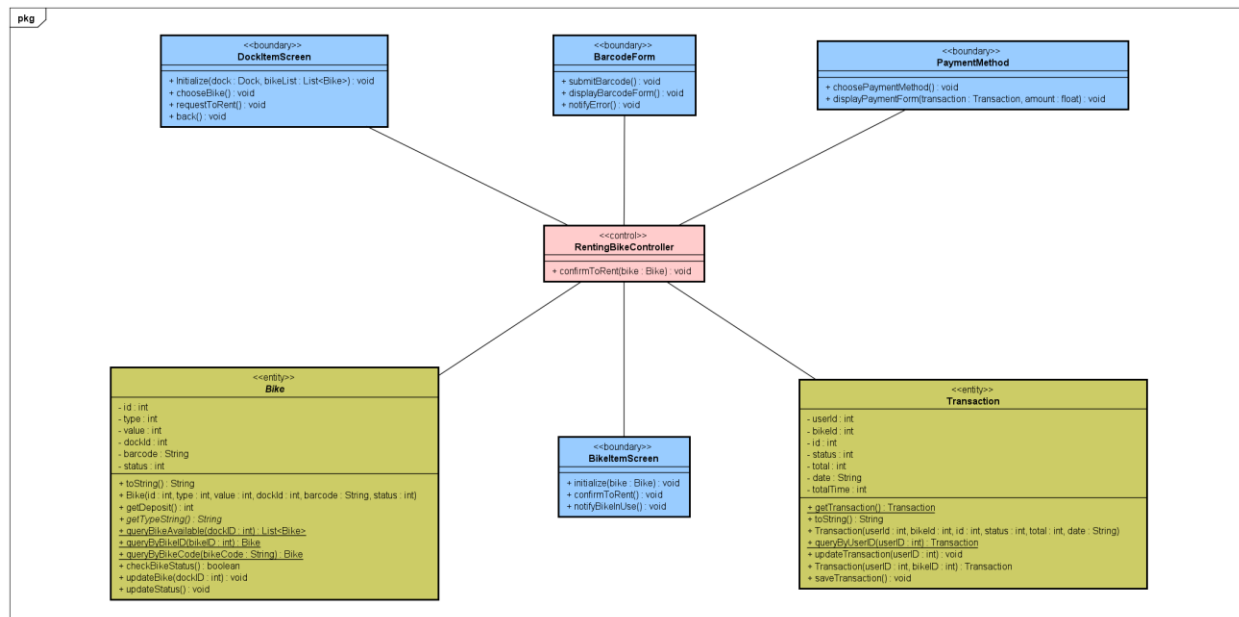
Hình 10 Xem chi tiết xe đang thuê



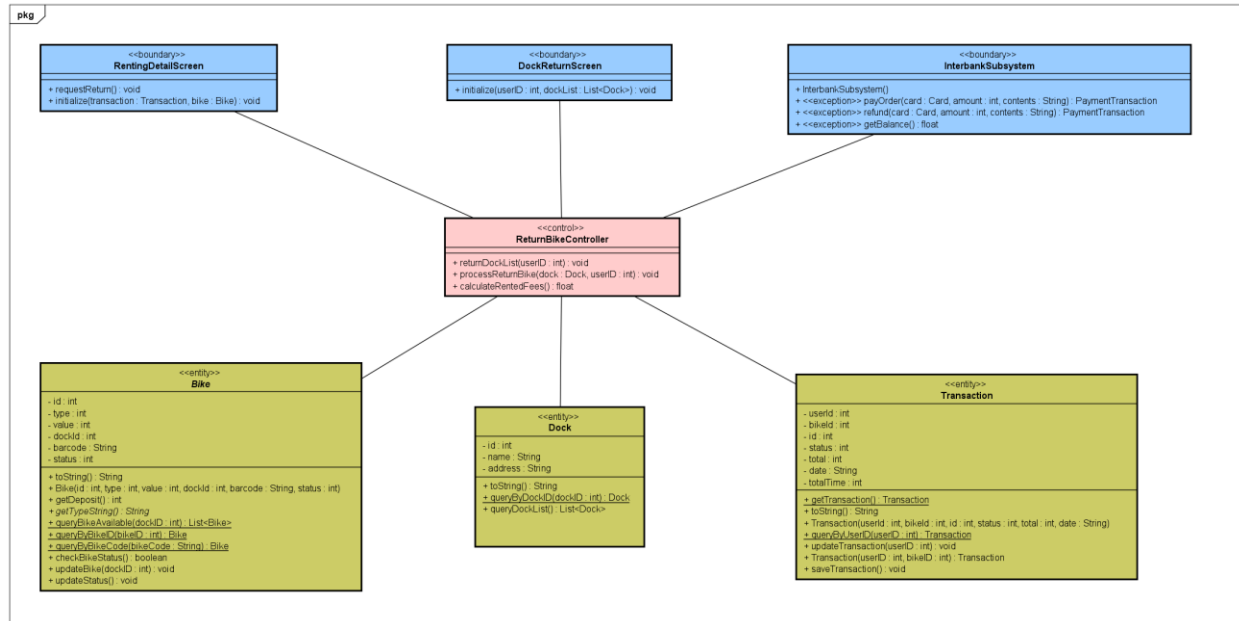
Hình 11 Chuyển đổi barcode



Hình 12 Đặt cọc

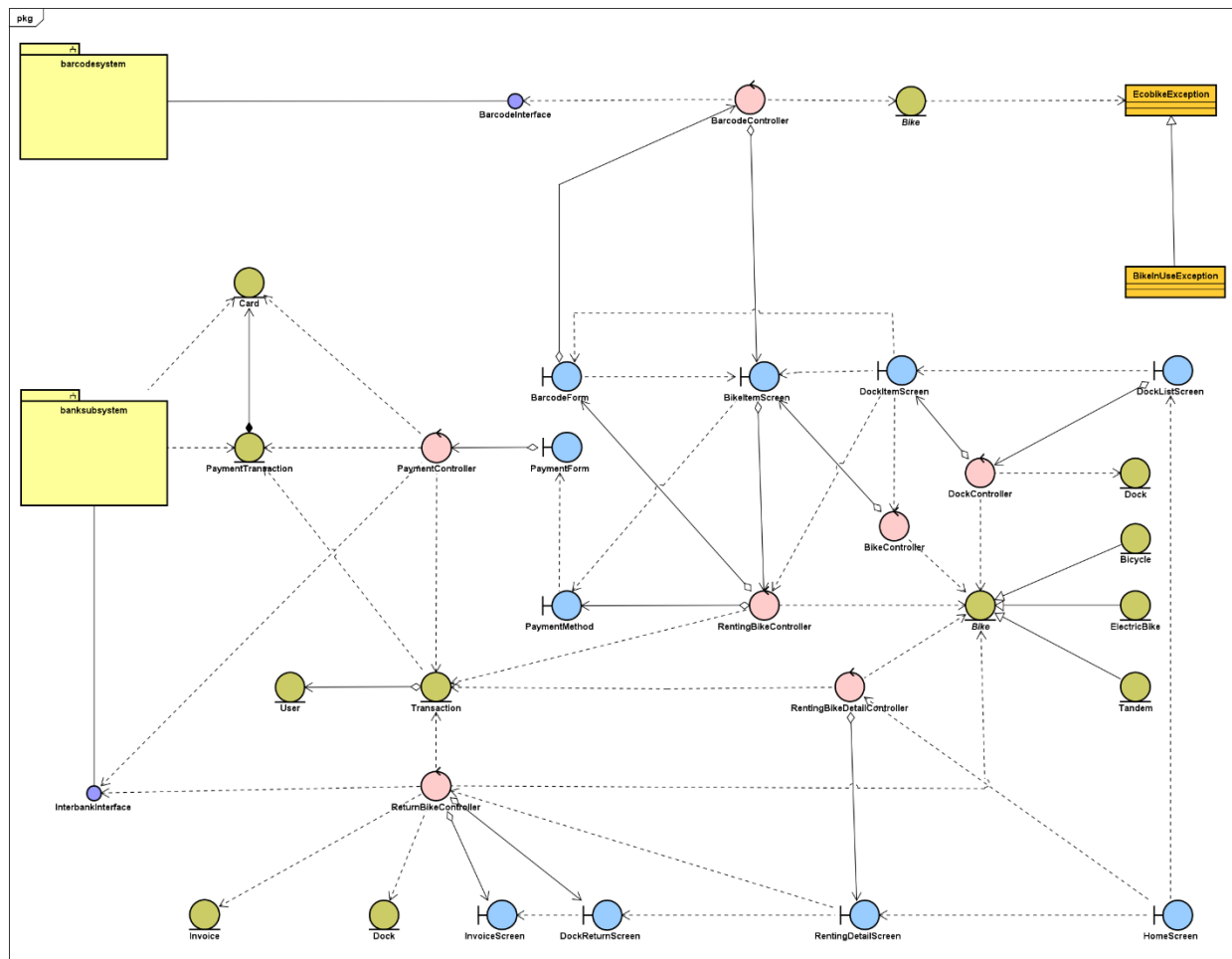


Hình 13 Thuê xe



Hình 14 Trả xe

### 3.4 Unified Analysis Class Diagram



Hình 15 Biểu đồ lớp phân tích kết hợp

### 3.5 Security Software Architecture

Không có

## 4 Detailed Design

## 4.1 User Interface Design

#### 4.1.1 Screen Configuration Standardization

#### 4.1.1.1 Display

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

Độ phân giải: 1366 x 768 pixels

#### 4.1.1.2 Screen

Vị trí của của button: Ở giữa (theo chiều dọc) và ở bên phải (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.

#### **4.1.1.3 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 (first screen)
2. home screen - màn hình chính
3. dock list – xem danh sách các bãi xe
4. bike list - xem chi tiết danh sách xe trong bãi xe
5. bike detail - xem thông tin chi tiết xe trong bãi
6. bike form renting – điền thông tin thuê xe
7. rented detail - xem thông tin chi tiết xe vừa thuê thành công

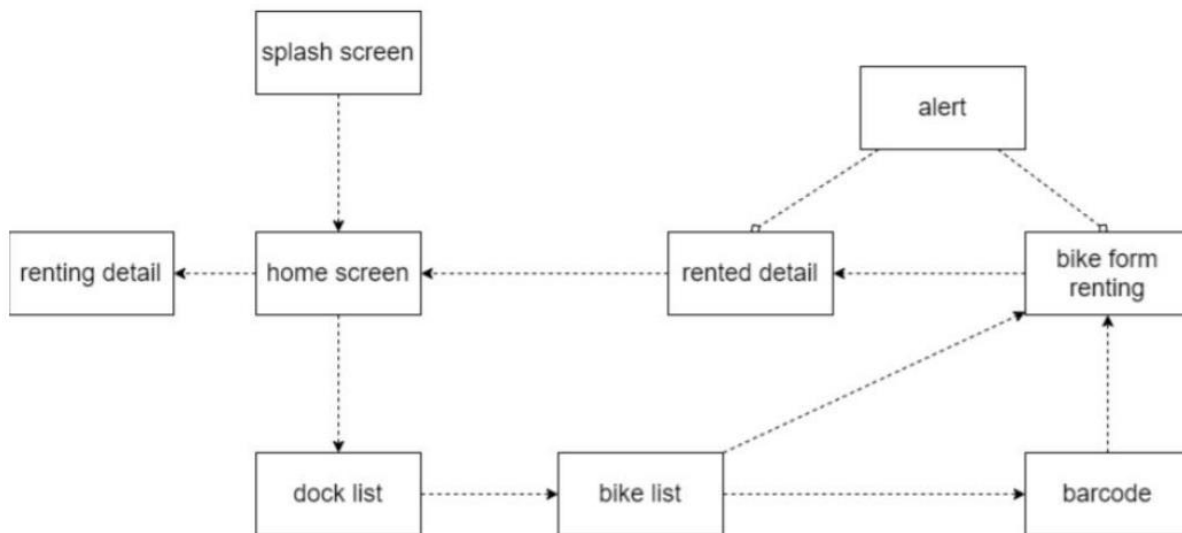
8. renting detail - xem thông tin xe đang thuê

9. alert - pop up xác nhận thuê xe 10. barcode - pop up điền mã xe muốn thuê để thuê xe

#### 4.1.1.4 Error

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

#### 4.1.2 Screen Transition Diagrams



Hình 16 Sơ đồ chuyển màn hình

#### 4.1.3 Screen Specifications

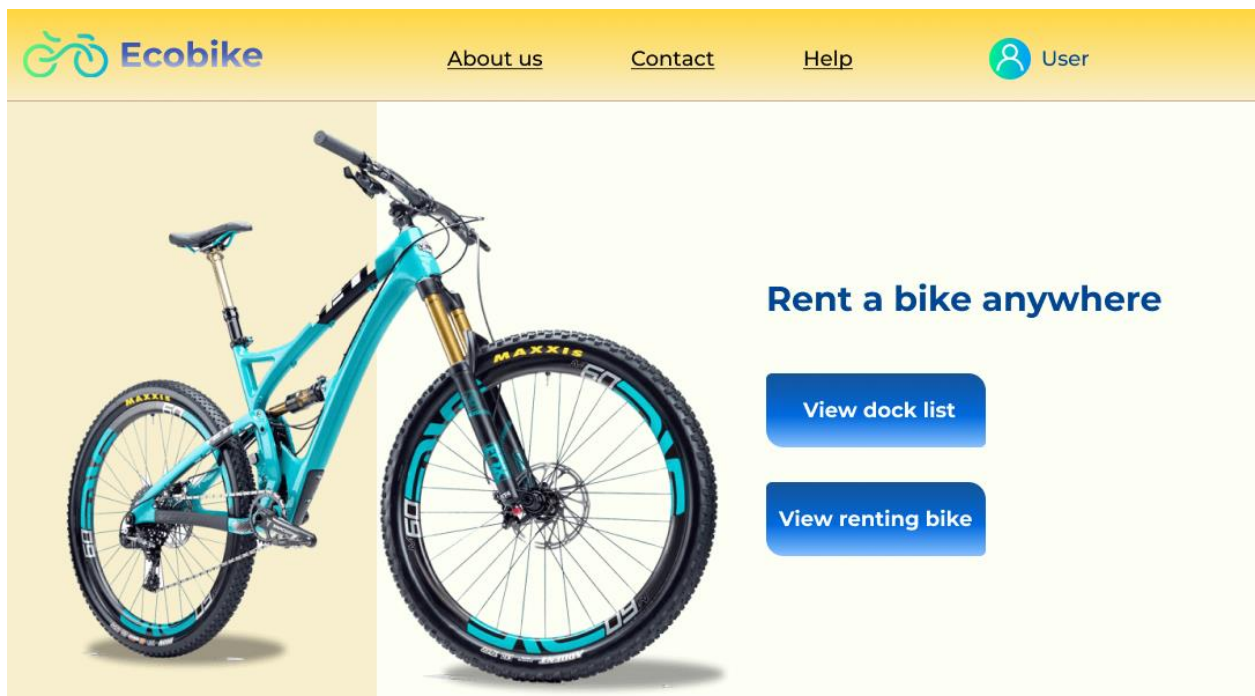
##### 4.1.3.1 splash screen





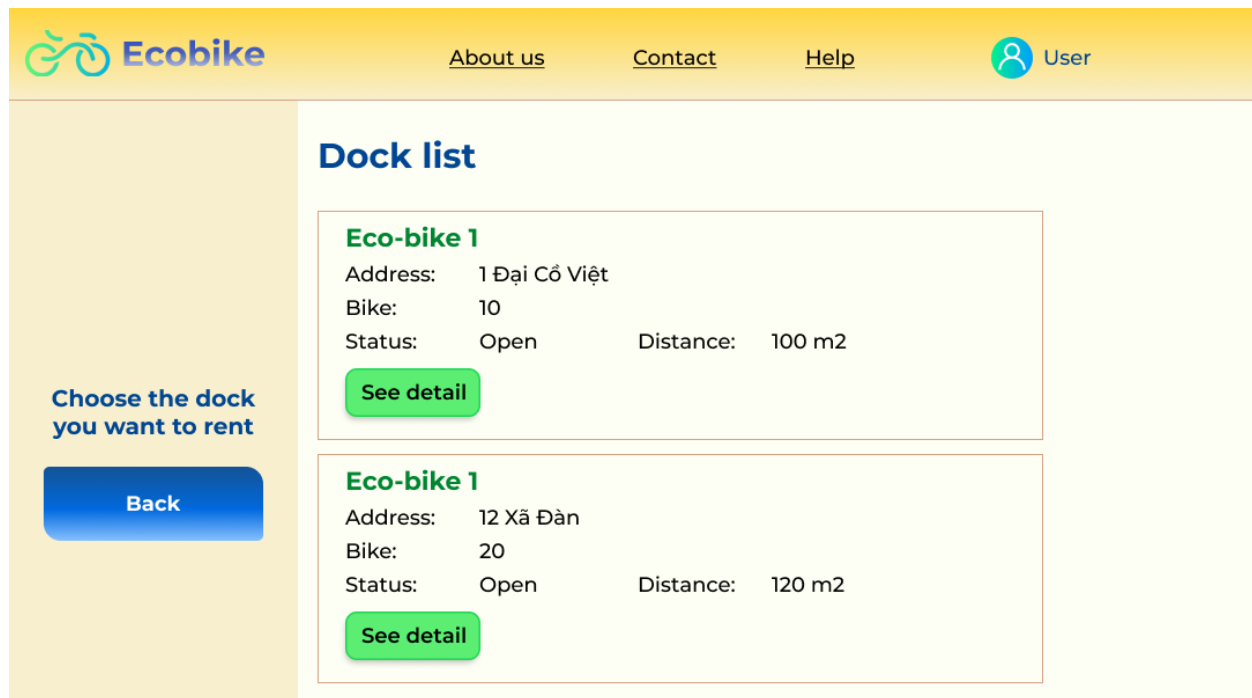
*Hình 17 splash screen*

#### 4.1.3.2 home screen



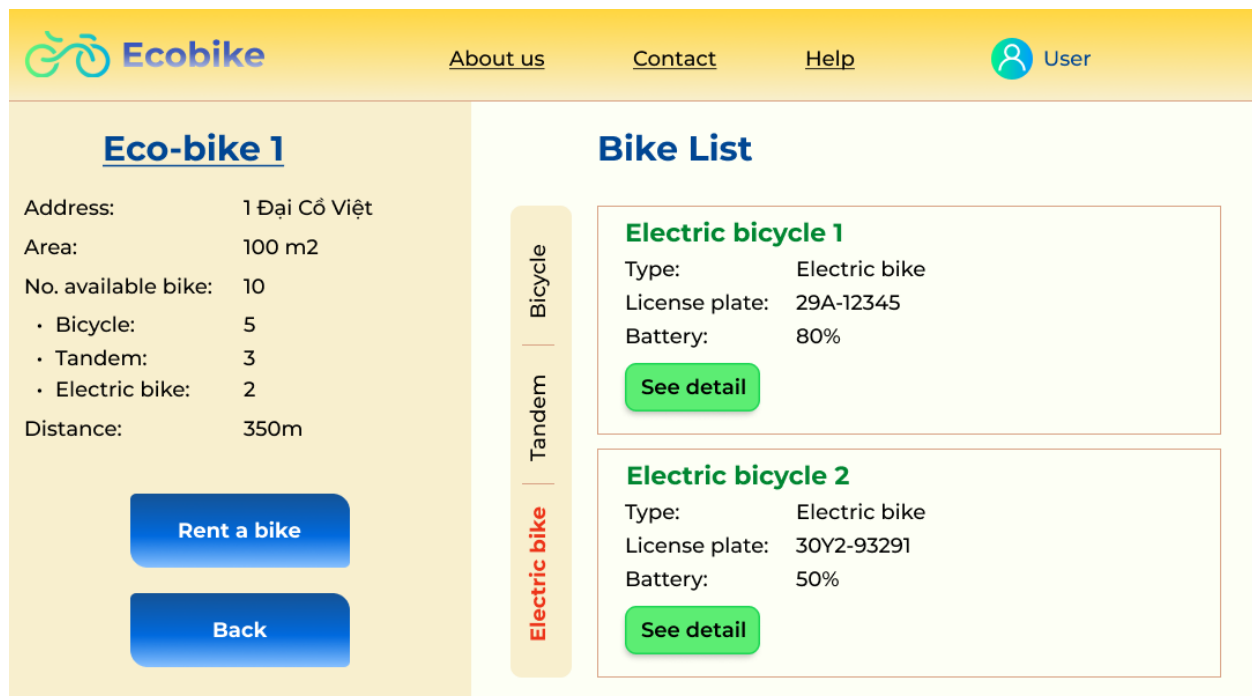
*Hình 18 home screen*

#### 4.1.3.3 dock list



Hình 19 dock list

#### 4.1.3.4 bike list



Hình 20 bike list

#### 4.1.3.5 bike detail

## Electric bicycle 1

*This bike is available to rent !*

Bike ID: **M8UI92TH**  
Type: **Electric bike**  
License plate: **29A-12345**  
Battery: **80%**  
Barcode: **81276371232**

Value: **VND 700,000**

Deposit: **VND 280,000**  
(40% Value)

**Confirm to rent**

**Back**

*Hình 21 bike detail*

## Payment form

Cardholder name

\*

Card number

\*

Expiration date



\*

Issuing bank

\*

Security code

\*

Transaction content

Amount

280,000



VND

**Confirm**

**Back**

*Hình 22 Payment form*

#### 4.1.3.7 renting detail

[About us](#)[Contact](#)[Help](#) User

### Bike rental information

Bike ID:	M8UI92TH	Rental time:	0h 40m
Bike name:	Electric bicycle 1	Amount:	VND 19,500
Bike type:	Electric bike		
License plate:	29A-12345		
Remaining battery:	80%		

Return bike

Back

Hình 23 bike renting detail

#### 4.1.3.8 barcode - pop up

## Barcode Converter

Enter barcode here:

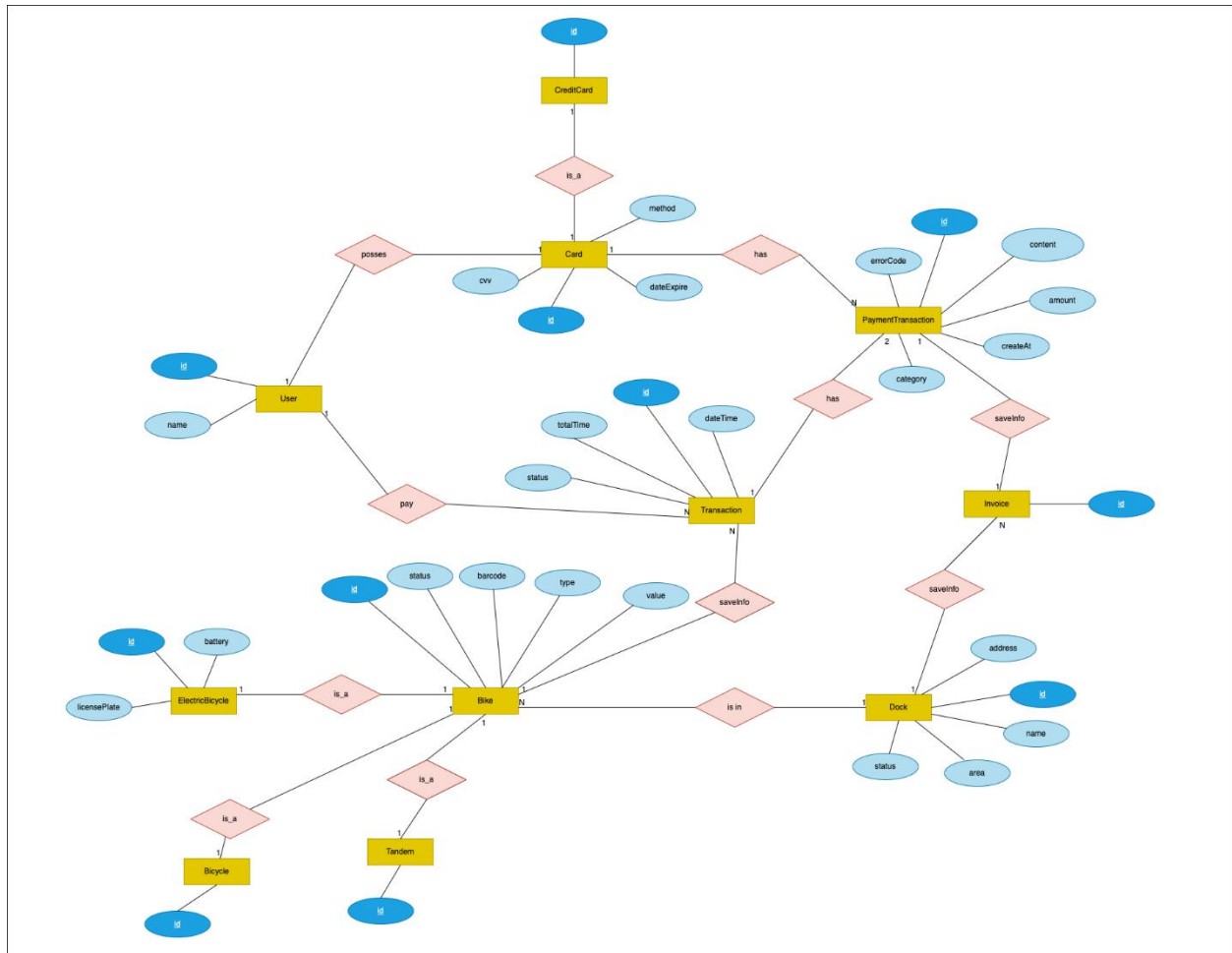
Back

Confirm

Hình 24 barcode

## 4.2 Data Modeling

### 4.2.1 Conceptual Data Modeling



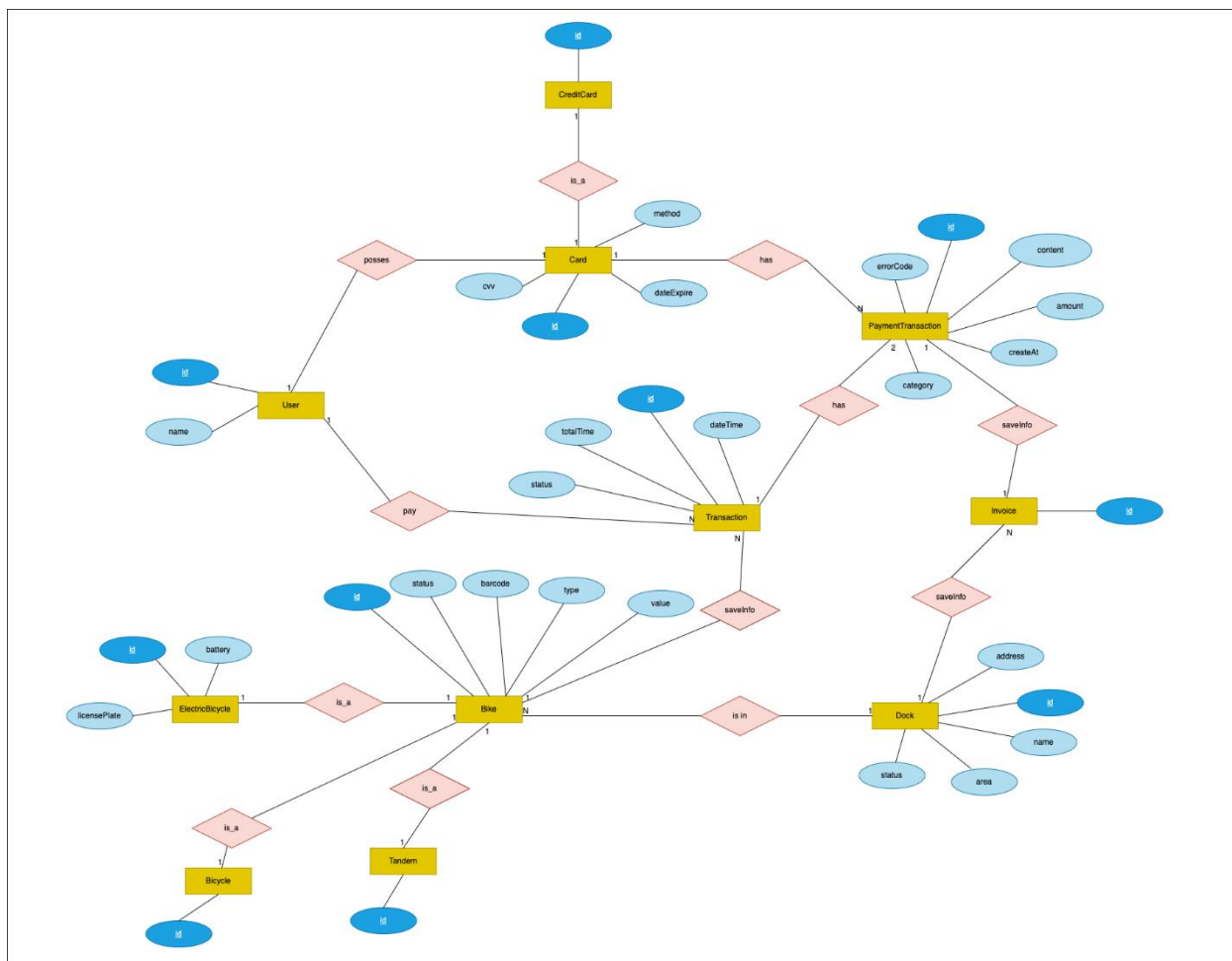
Hình 25 ER diagram

### 4.2.2 Database Design

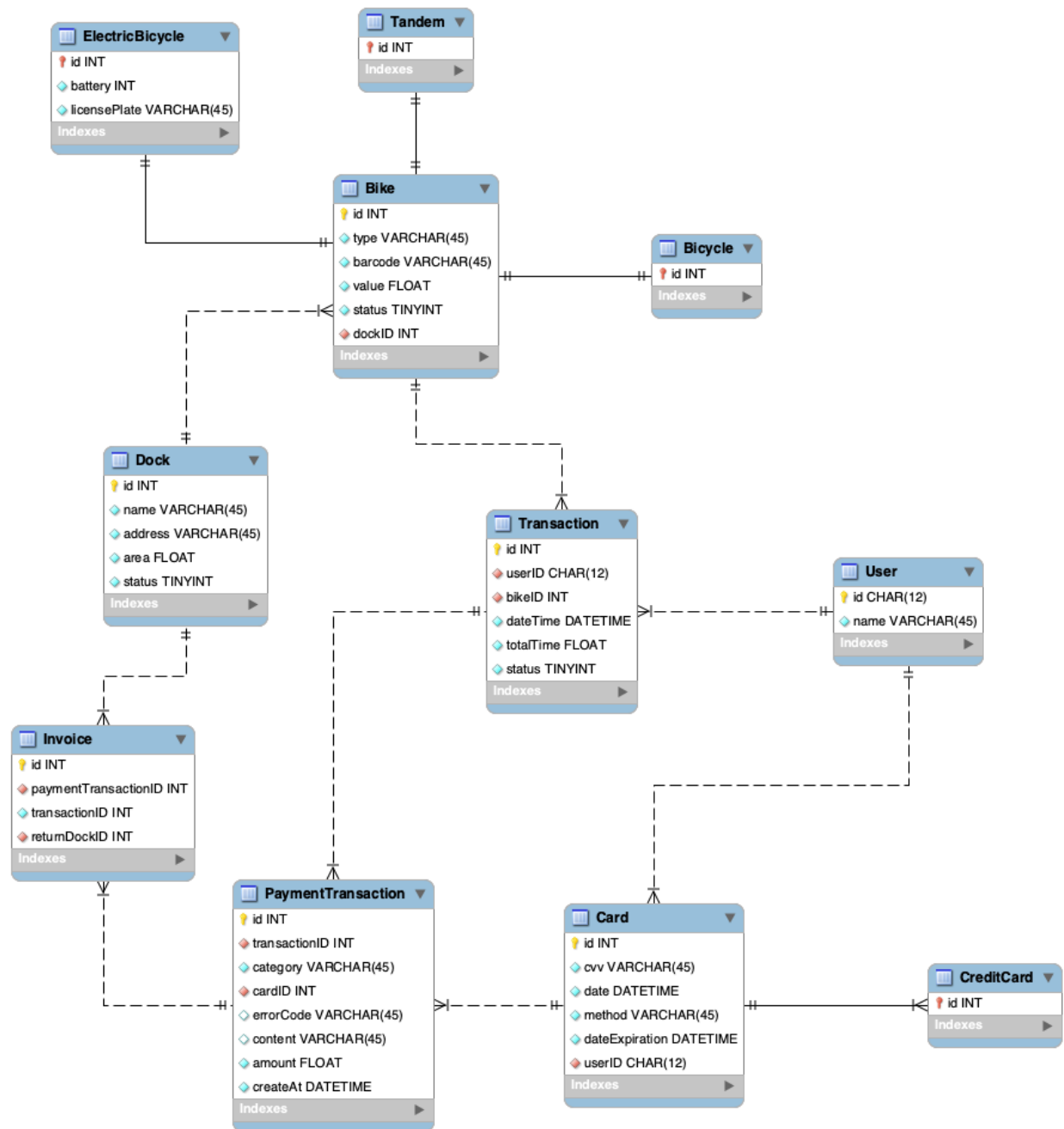
#### 4.2.2.1 Database Management System

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 Database Diagram



Hình 26 ER diagram



Hình 27 Data modeling

#### 4.2.2.3 Database Detail Design

##### ▪ Bike

#	PK	FK	Column name	Data type	Mandatory	Description
---	----	----	-------------	-----------	-----------	-------------

1.	x		id	INT	Yes	ID, auto increment
2.			type	VARCHAR(45)	Yes	Bike type, e.g., bicycle, tandem, e-bicycle
3.			barcode	VARCHAR(45)	Yes	Bike barcode
4.			value	FLOAT	Yes	Price of bike
5.			status	TINYINT	Yes	bike in use or not (status = false mean bike is in use)
6.		x	dockID	INT	Yes	

#### ▪ Bicycle

#	PK	FK	Column name	Data type	Mandatory	Description
1.	x		id	INT	Yes	ID, same as ID of Bike of which type is bicycle

#### ▪ Tandem

#	PK	FK	Column name	Data type	Mandatory	Description
1.	x		id	INT	Yes	ID, same as ID of Bike of which type is Tandem

#### ▪ ElectricBicycle

#	PK	FK	Column name	Data type	Mandatory	Description
1.	x		id	INT	Yes	ID, same as ID of Bike of which type is Electric bicycle
2.			battery	INT	Yes	Current battery
3.			licensePlate	VARCHAR(45)	Yes	

#### ▪ Dock



#	PK	FK	Column name	Data type	Mandatory	Description
1.	x		id	INT	Yes	ID, auto increment
2.			name	VARCHAR(45)	Yes	Dock name
3.			address	VARCHAR(45)	Yes	Dock address
4.			area	FLOAT	Yes	Price of bike
5.			status	TINYINT	Yes	Dock is active or not (status = true mean dock is active)

▪ User

#	PK	FK	Column name	Data type	Mandatory	Description
1.	x		id	CHAR(12)	Yes	User ID
2.			name	VARCHAR(45)	Yes	User name

▪ Card

#	PK	FK	Column name	Data type	Mandatory	Description
1.	x		id	INT	Yes	Card ID
2.			cvv	VARCHAR(45)	Yes	cvv code
3.			method	VARCHAR(45)	Yes	Card method, e.g., credit card
4.			dateExpiration	DATETIME	Yes	Expiration date
5.		x	userID	CHAR(12)	Yes	owner id of card

▪ Credit card

#	PK	FK	Column name	Data type	Mandatory	Description
1.	x		id	INT	Yes	ID, same as ID of Card of which type is Credit

▪ **Transaction**

#	PK	FK	Column name	Data type	Mandatory	Description
1.	x		id	INT	Yes	ID, auto increment
2.		x	userID	INT	Yes	User ID
3.		x	bikeID	INT	Yes	Bike ID
4.			dateTime	DATETIME	Yes	Rental start time
5.			totalTime	FLOAT	No	Total rental time (= null if status = true)
6.			status	TINYINT	Yes	End of renting or not (status = true mean end of renting)

▪ **PaymentTransaction**

#	PK	FK	Column name	Data type	Mandatory	Description
1.	x		id	INT	Yes	ID
2.		x	transactionID	INT	Yes	Transaction ID
3.			category	VARCHAR(45)	Yes	Payment category, e.g., pay, refund
4.		x	cardID	INT	Yes	Card ID
5.			errorCode	VARCHAR(45)	Yes	Error code
6.		x	content	VARCHAR(45)	No	Transaction content
7.			amount	FLOAT	Yes	Transaction amount
8.			creatAt	DATETIME	Yes	Date of creation

▪ **Invoice**

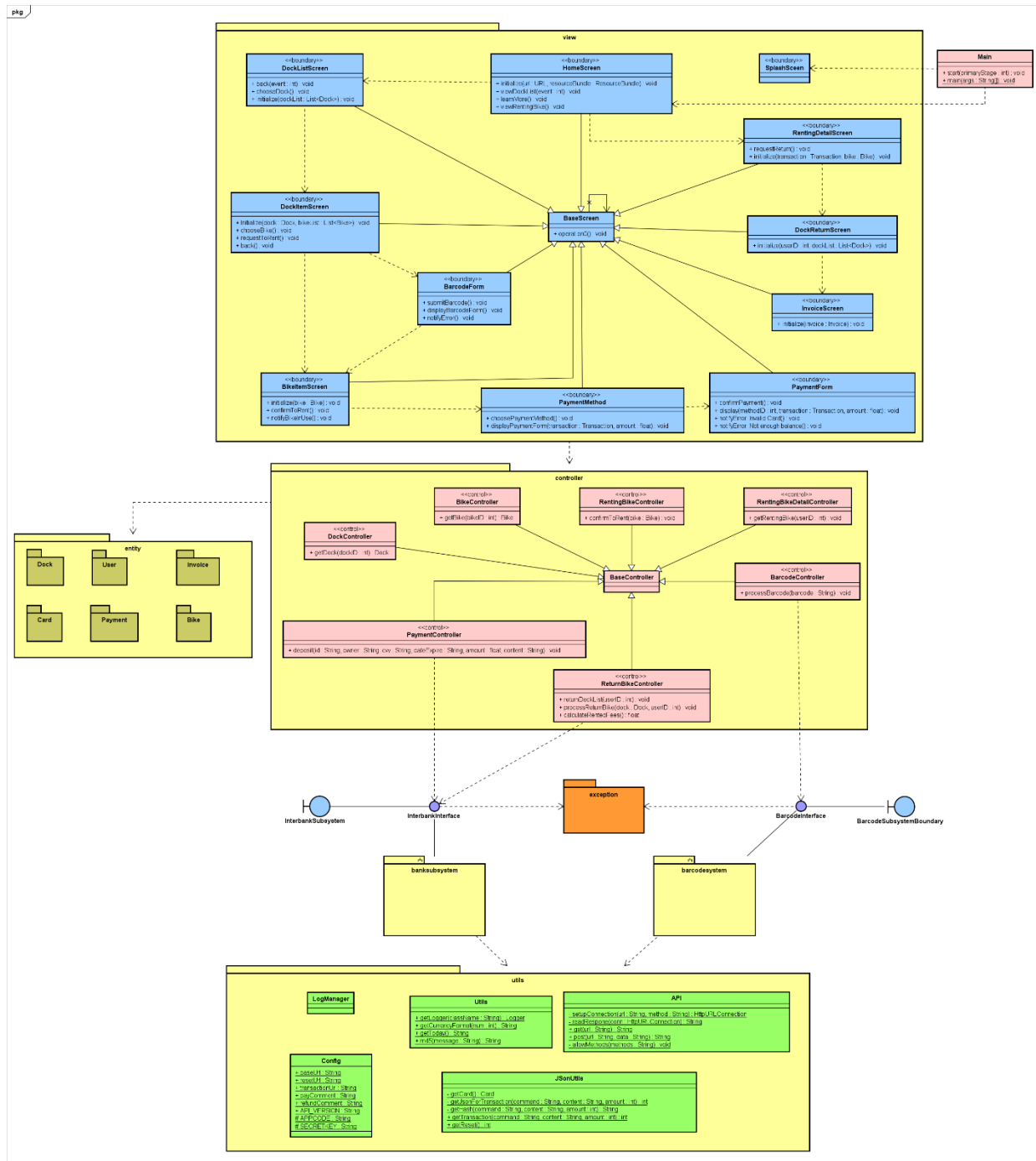
#	PK	FK	Column name	Data type	Mandatory	Description
1.	x		id	INT	Yes	ID, auto increment
2.		x	paymentTransactionID	INT	Yes	Payment transaction ID

3.		x	returnDockID	INT	Yes	Dock ID user return bike
----	--	---	--------------	-----	-----	-----------------------------

### 4.3 Non-Database Management System Files

Nếu trong trường hợp cần back up database thì nhóm sẽ export 1 file sql để có thể tái tạo database trên máy khác.

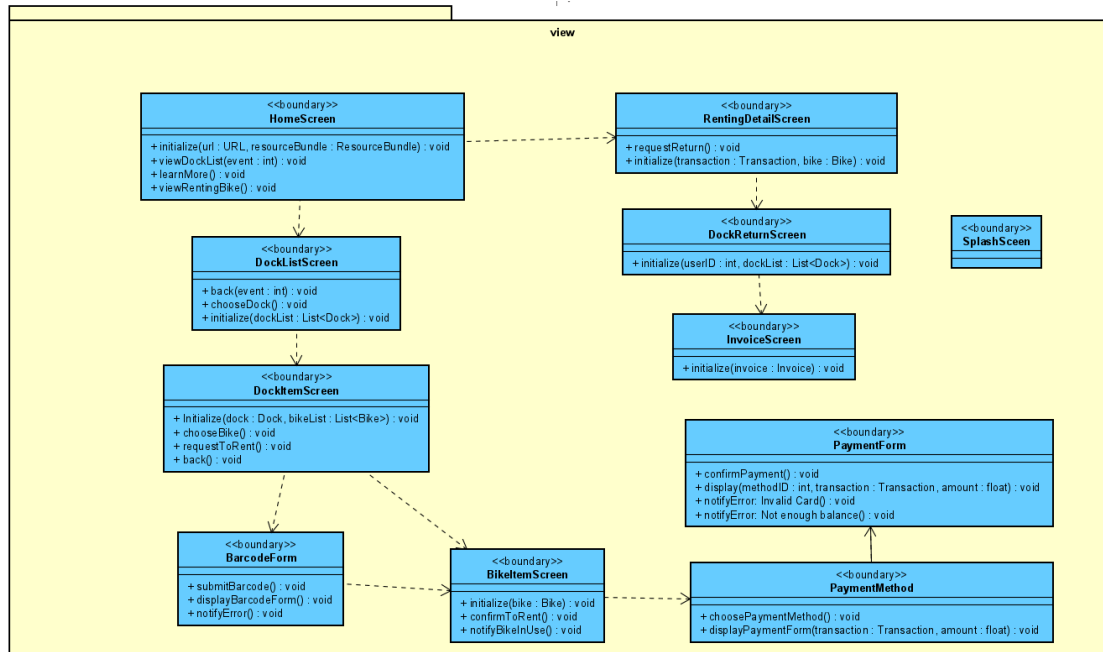
#### 4.4.1 General Class Diagram



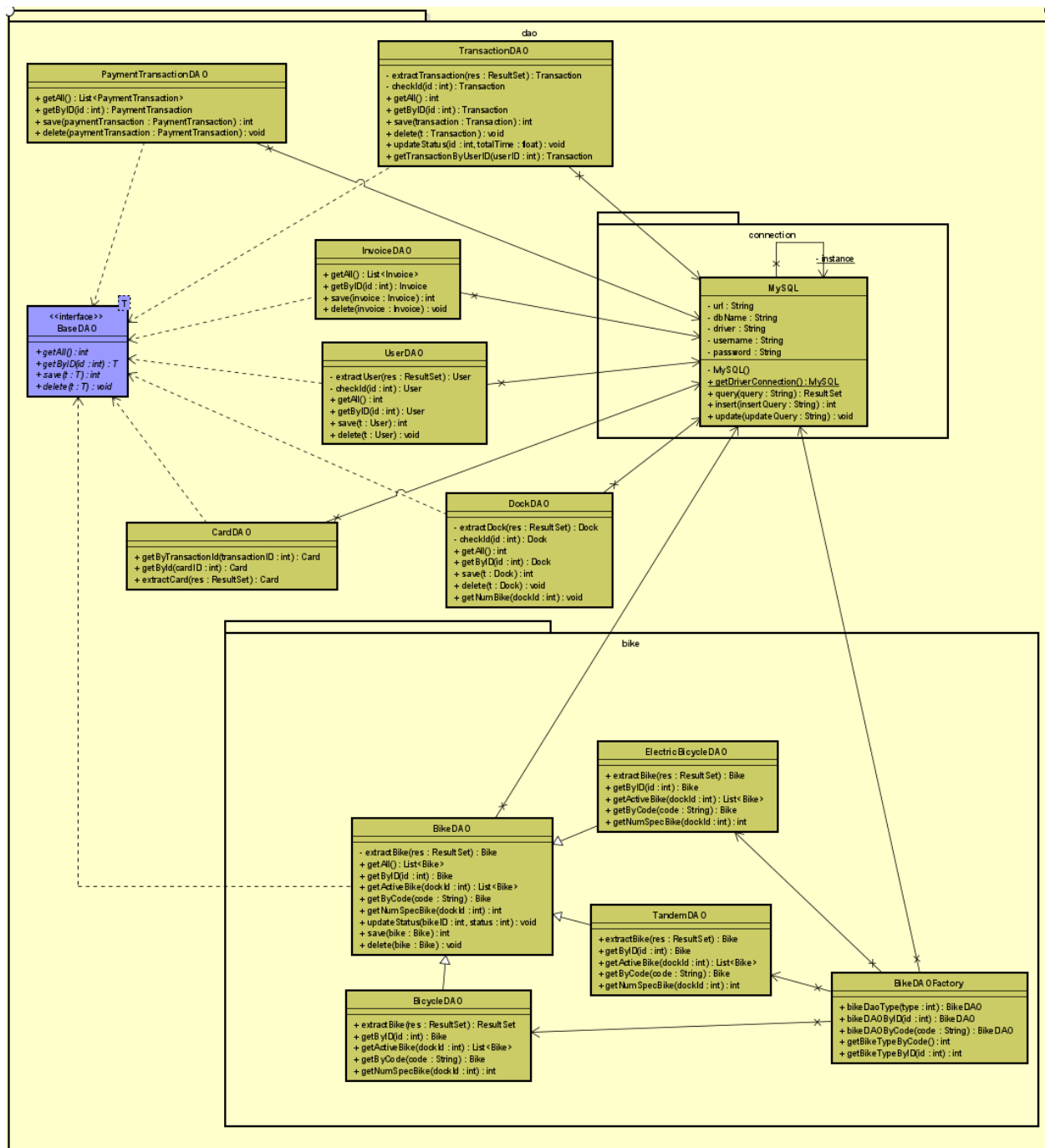
Hình 28 Biểu đồ lớp

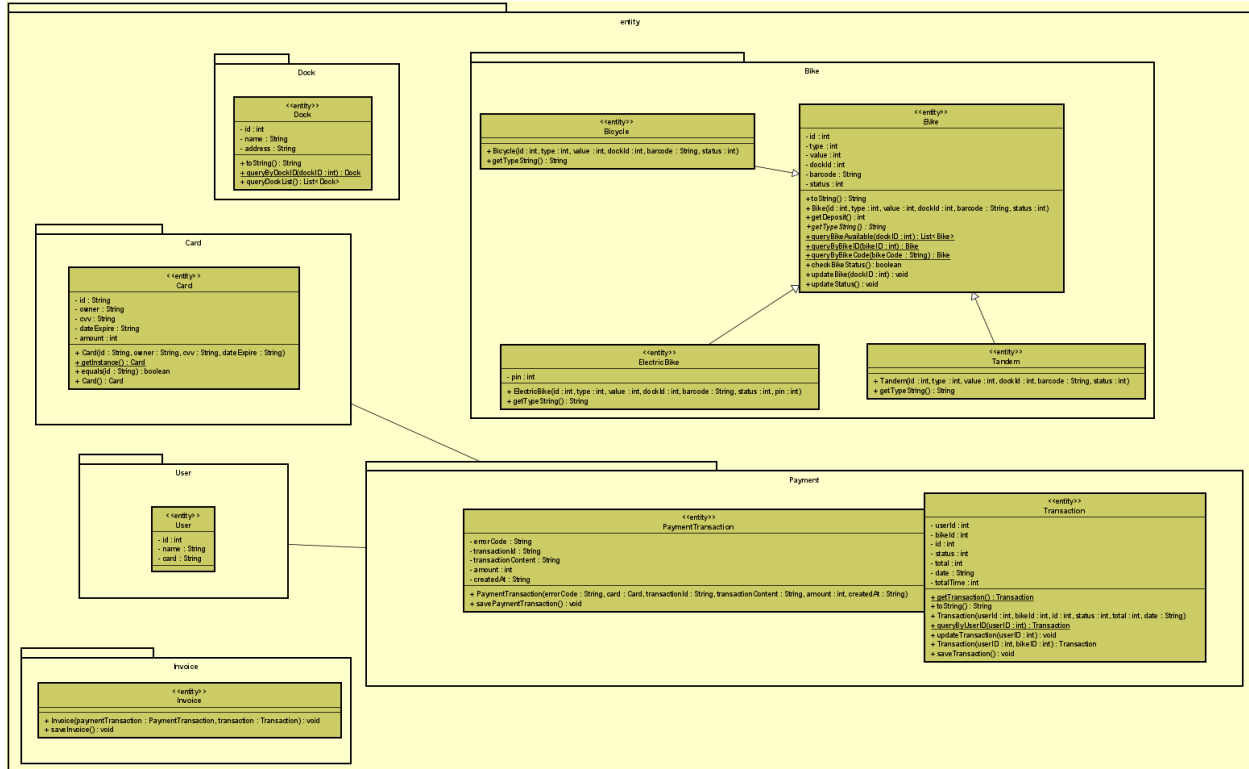
### 4.4.2 Class Diagrams

#### 4.4.2.1 Class Diagram for view package

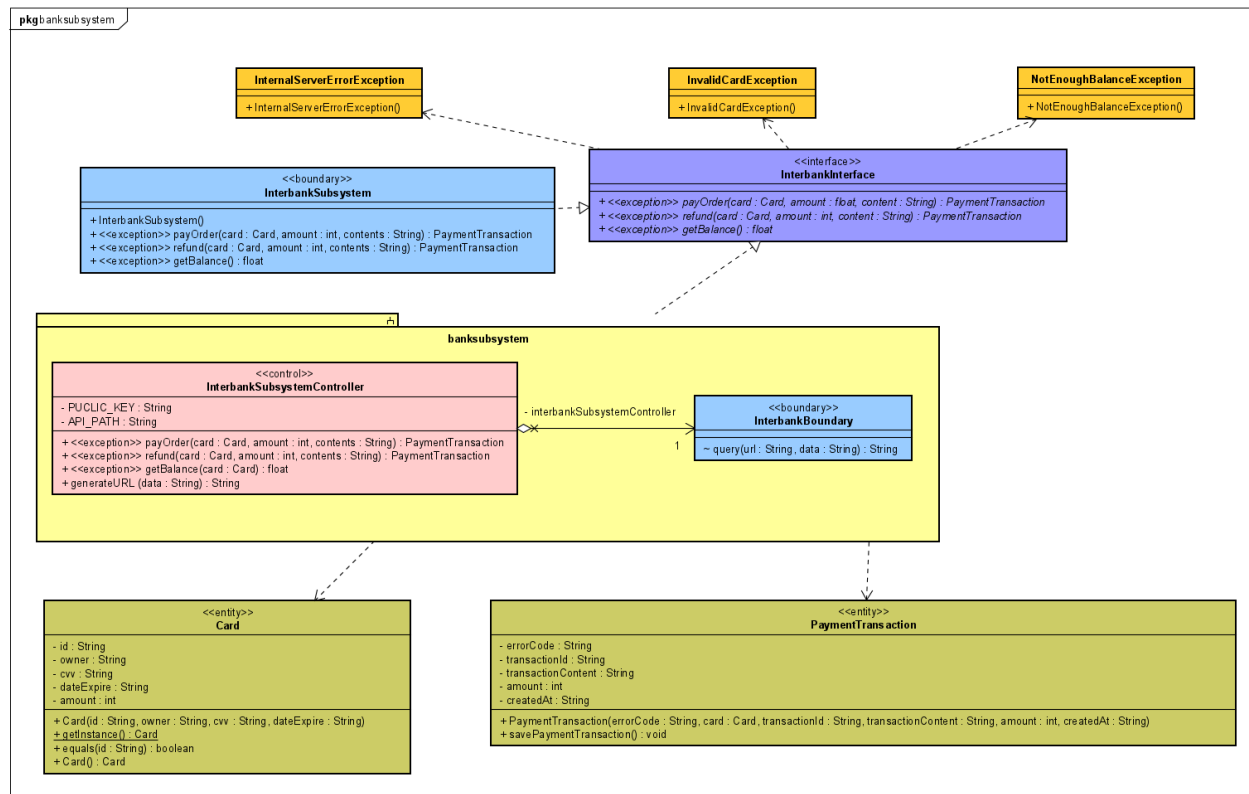


4.4.2.3 Class Diagram for model package

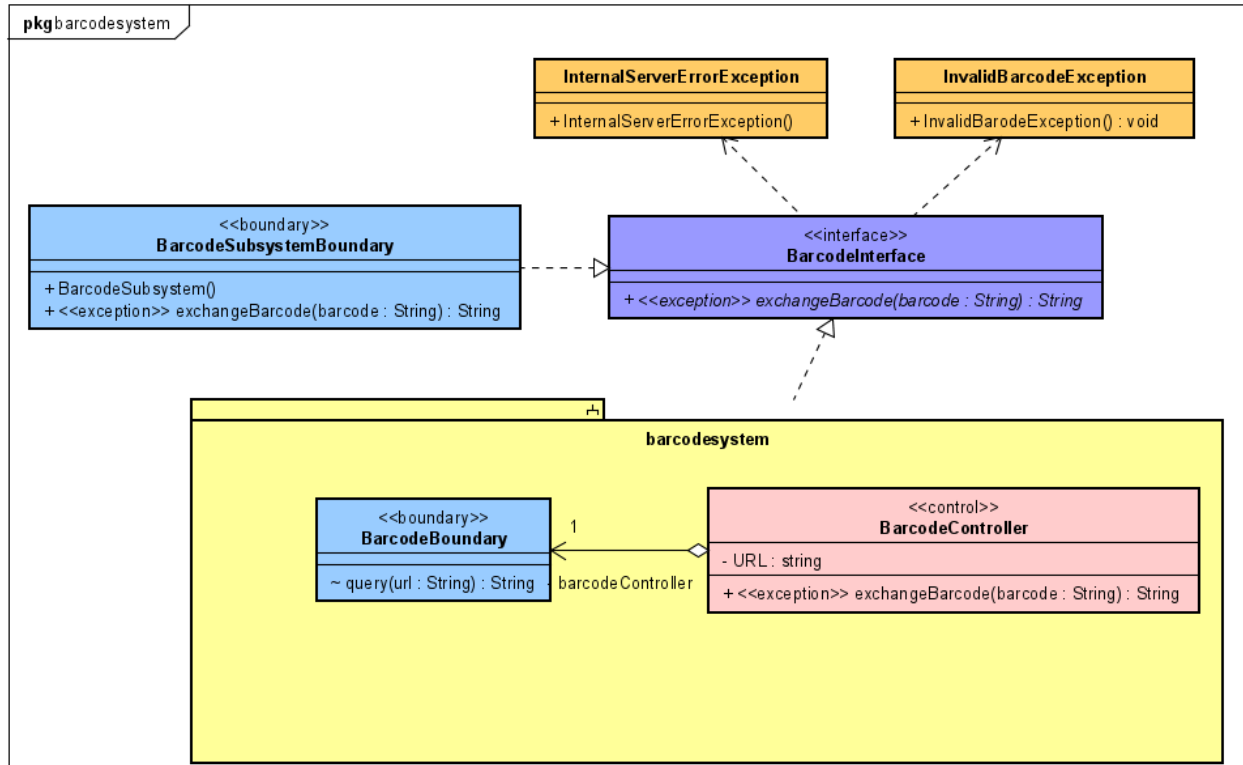




#### 4.4.2.4 Class Diagram for banksubsystem package

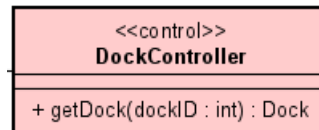


#### 4.4.2.5 Class Diagram for barcodesubsystem package



#### 4.4.3 Class Design

##### 4.4.3.1 Class “DockController”



*Attribute:*

Không có

*Operation:*

STT	Tên	Kiểu dữ liệu trả về	Mô tả (mục đích)
1	getDocks	List<Dock>	Lấy danh sách các bãi xe có trong hệ thống

*Parameter:* không

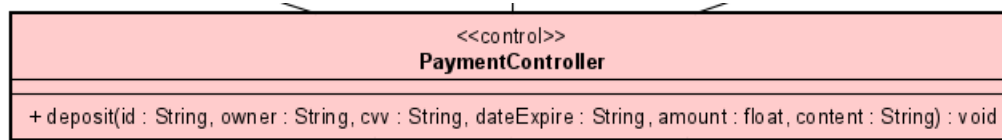


*Exception:* không

*Method:* không

*State:* không

#### 4.4.3.2 Class “PaymentController”



*Attribute:*

Không có

*Operation:*

STT	Tên	Kiểu dữ liệu trả về	Mô tả (mục đích)
1	Deposit	PaymentTransaction	Đặt cọc

Parameter:

*Transaction:* Transaction, giao dịch cần thanh toán hoặc hoàn trả tiền cọc

*Exception:* PaymentException: nếu mã lỗi trả về đã biết

*Method:*

không

*State:*

không

#### 4.4.3.3 Class “BikeController”



*Attribute:* Không có

*Operation:*

STT	Tên	Kiểu dữ liệu trả về	Mô tả (mục đích)
-----	-----	---------------------	------------------

1	getBikes	List	Lấy danh sách các xe có trong bãi
---	----------	------	-----------------------------------

*Parameter:*

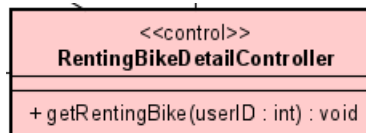
- type : int, id của kiểu xe muốn lấy ra danh sách
- barcode: String, id của xe

*Exception:* không

*Method:* không

*State:* không

#### 4.4.3.4 Class “RentingBikeController”



*Attribute:*

Không có

*Operation:*

STT	Tên	Kiểu dữ liệu trả về	Mô tả (mục đích)
1	getRentingBike	Bike	Lấy mã xe dựa trên userID

*Parameter:*

transaction: Transaction, giao dịch muốn thanh toán thuê xe name: String, tên người thuê xe

card: String, thông tin thanh toán

bike: Bike, xe muốn thuê

barcode: String, barcode xe đang thuê

*Exception:*

- Payment Exception: nếu mã lỗi trả về đã biết

*Method:*

không

*State:*

không

## **5 Design Considerations**

### **5.1 Goals and Guidelines**

#### **1. Mục tiêu:**

Phần mềm mô phỏng quá trình thuê và trả xe trong khu đô thị, 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.

#### **2.Cách dùng:**

Thuê xe trong thời gian ngắn vì số tiền ảo không đáp ứng được. Chỉ thuê 1 xe trong 1 thời điểm

### **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.

### **5.3 Coupling and Cohesion**

Trong hệ thống này, tính kết dính - cohesion khá cao. Các module trong hệ thống được tách theo các vai trò mà nó quản lý, không có lớp nào mang nhiều hơn hai trách nhiệm nghiệp vụ. Cụ thể ví dụ như lớp ReturnBikeController chỉ có một phương thức là refund() và một phương thức phụ trợ việc lưu giao dịch vào cơ sở dữ liệu là makeTransactionDao(). Bên cạnh đó, hệ thống chưa đạt được loose coupling. Các thành phần còn phụ thuộc vào nhau khá nhiều, ngoài ra một số module còn quản lý chung một số dữ liệu (Data Global) như PaymentTransaction. Cụ thể, RentingBikeController và ReturnBikeController đều có quyền chỉnh sửa trạng thái của lớp Singleton PaymentTransaction.

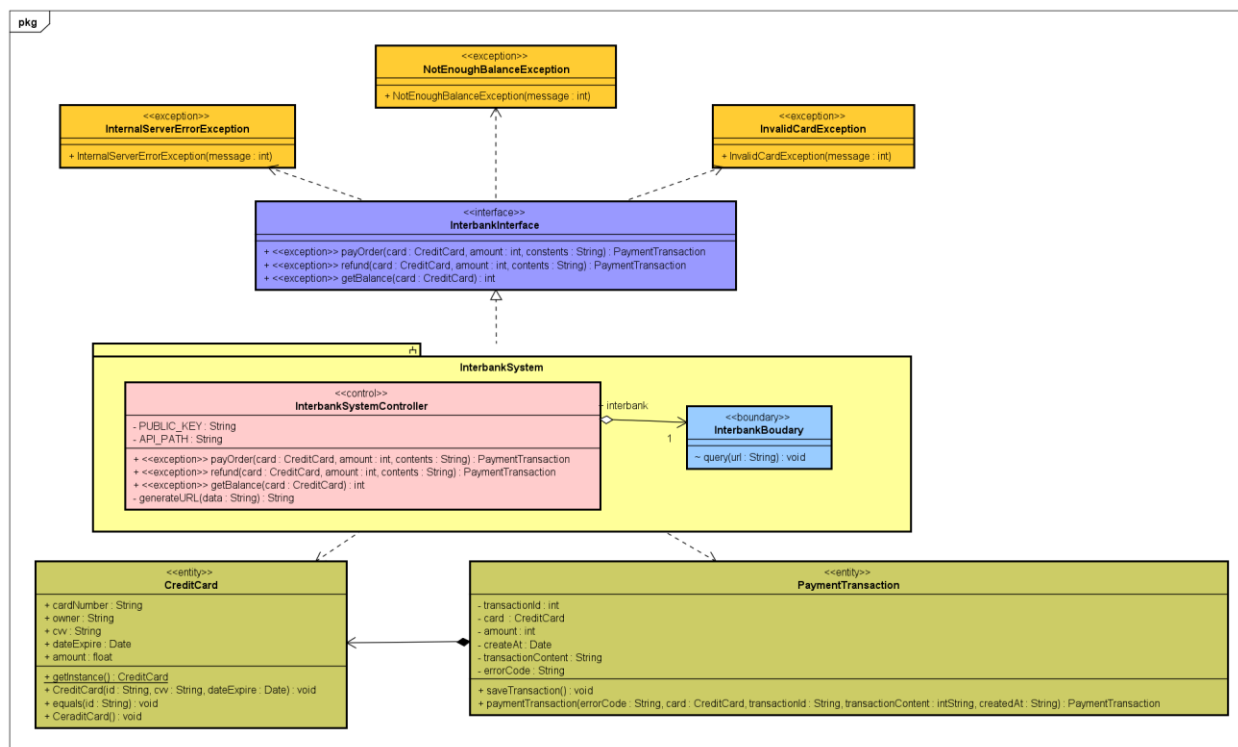
## 5.4 Design Principles

### 5.4.1 Single Responsibility Principle

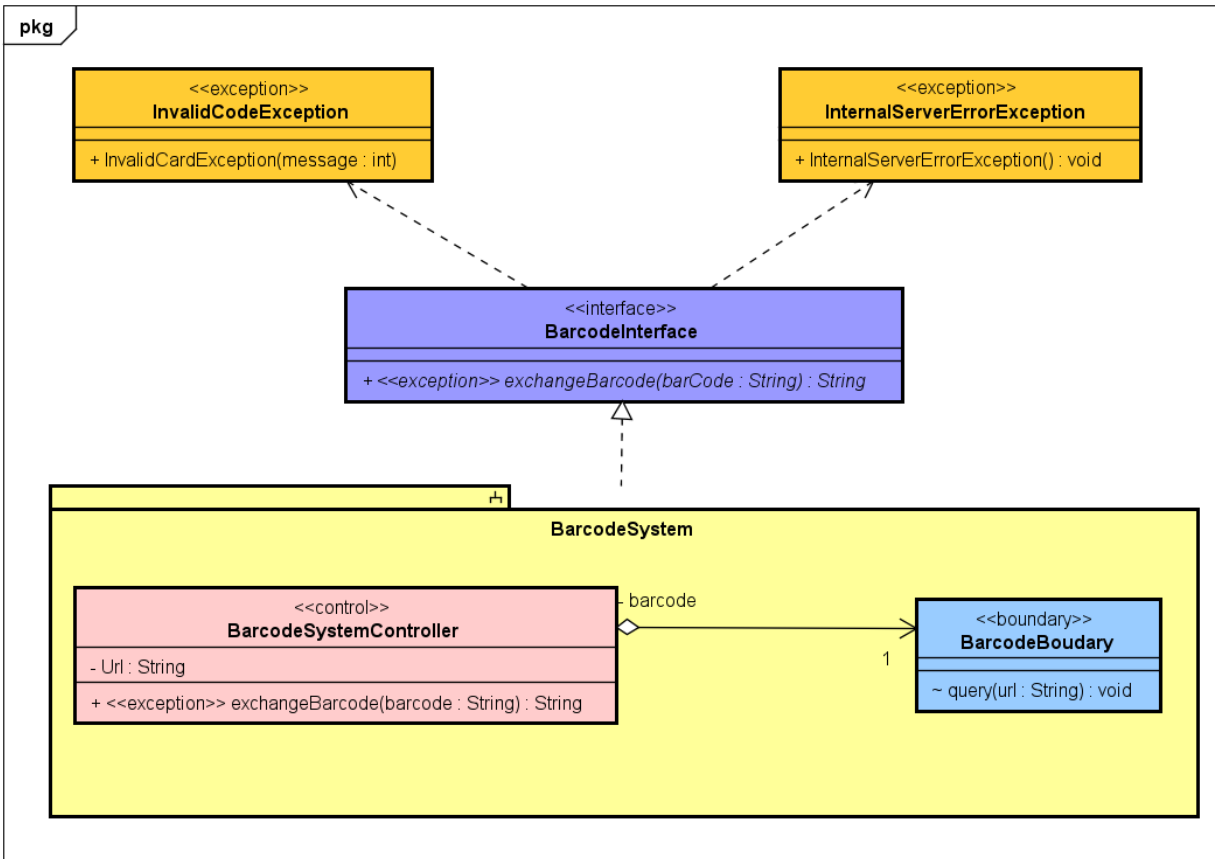
Trách nhiệm của hệ thống được phân bổ tới các package, các subsystem và trong mỗi package, subsystem, trách nhiệm được chia nhỏ cho từng Class, mỗi Class đảm nhận một trách nhiệm duy nhất.

### 5.4.2 Open/Closed Principle

Interbank subsystem implement các phương thức được định nghĩa trong Interbank interface. Các lớp của hệ thống chỉ phụ thuộc vào Interbank Interface chứ không phụ thuộc trực tiếp vào Interbank subsystem. Do đó, có thể dễ dàng thay thế subsystem sẵn có bằng một subsystem khác hoặc thêm một số phương thức khác cho InterbankInterface và implement các phương thức này trong subsystem. Các thay đổi bên phía subsystem hoàn toàn trong suốt với các bên liên quan sử dụng giao diện của Interbank interface.

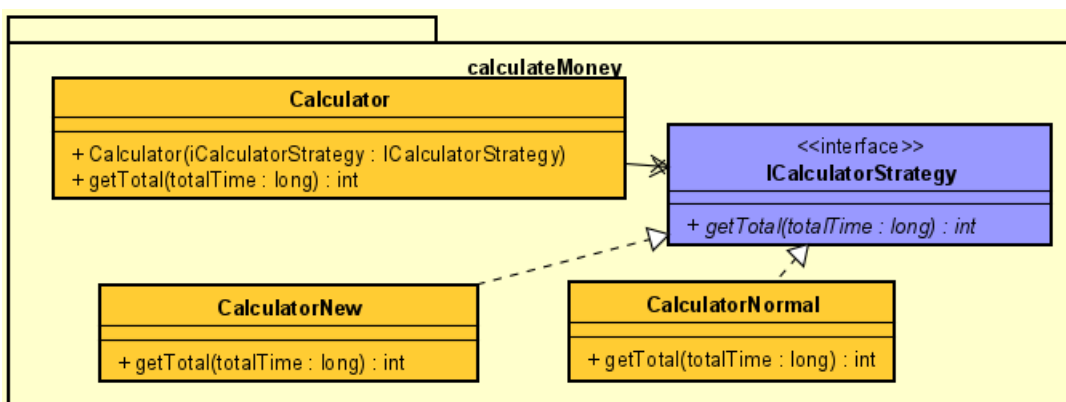


Tương tự với Barcode Subsystem.



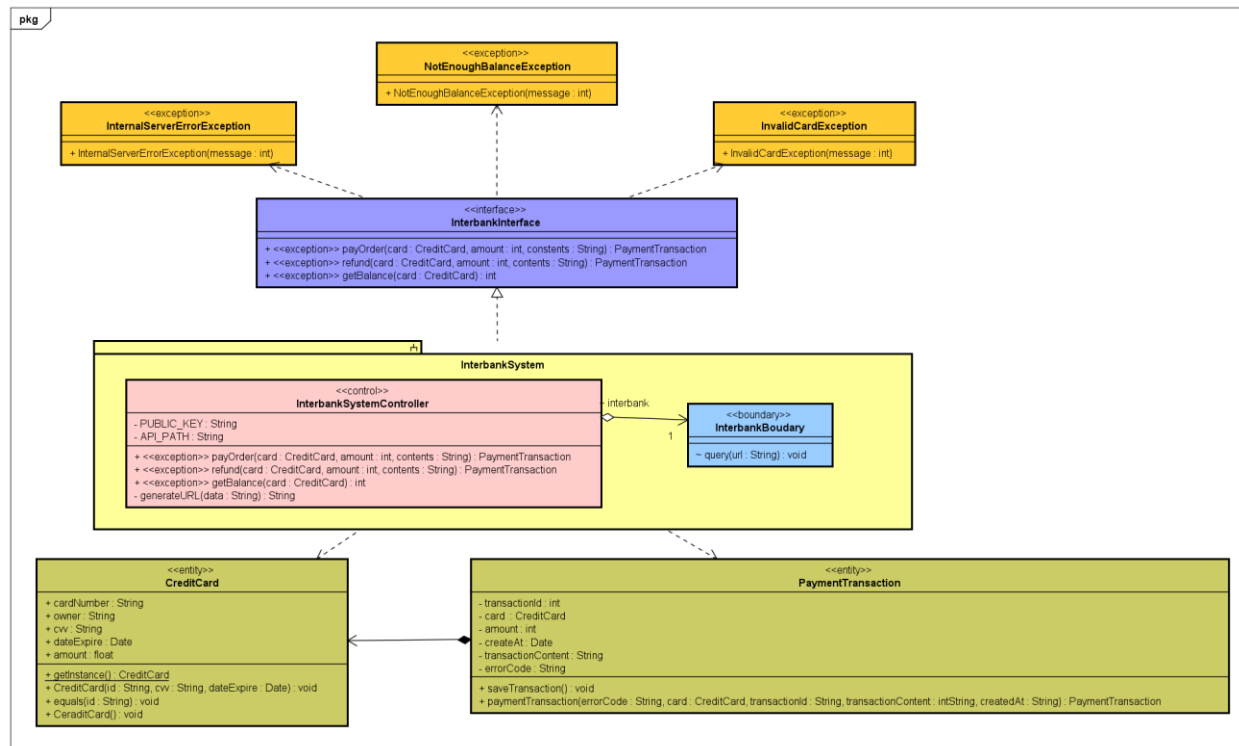
### 5.4.3 Liskov substitution principle

Nguyên tắc này nghe có vẻ phức tạp, nhưng thực chất có thể diễn ý đơn giản lại như sau: Nếu một class có sử dụng một implemtation của một interface, thì nó phải được thay thế dễ dàng bởi các implementation của interface đó mà không cần sửa gì thêm. 34 Ở đây ta có thể dễ dàng thay thế `IcalculatorStrategy` bởi 1 trong 2 `CalculatorNew` hoặc `CalculatorNormal`



#### 5.4.4 Interface segregation principle

Thiết kế hiện tại về cơ bản chưa đáp ứng được nguyên tắc này. Ví dụ trong InterbankInterface vẫn có 2 method nhỏ chứ chưa tách biệt.



#### 5.4.5 Dependency Inversion principle

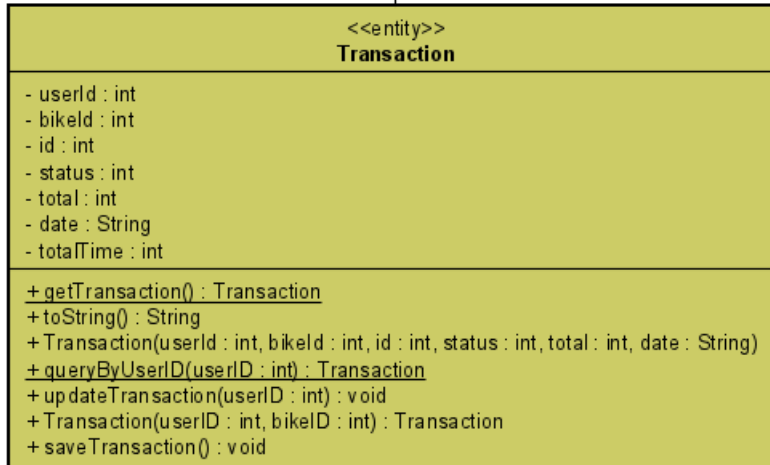
Hiện tại, PaymentController đang phụ thuộc chặt chẽ vào lớp Card, sau này giả sử không sử dụng Card để thanh toán mà sử dụng một loại phương thức thanh toán khác, ví dụ như domestic debit card... như vậy thiết kế hiện tại đã vi phạm nguyên lý Dependency Inversion.

### 5.5 Design Patterns

Thiết kế áp dụng 3 design pattern là Singleton pattern, DAO pattern và Strategy pattern

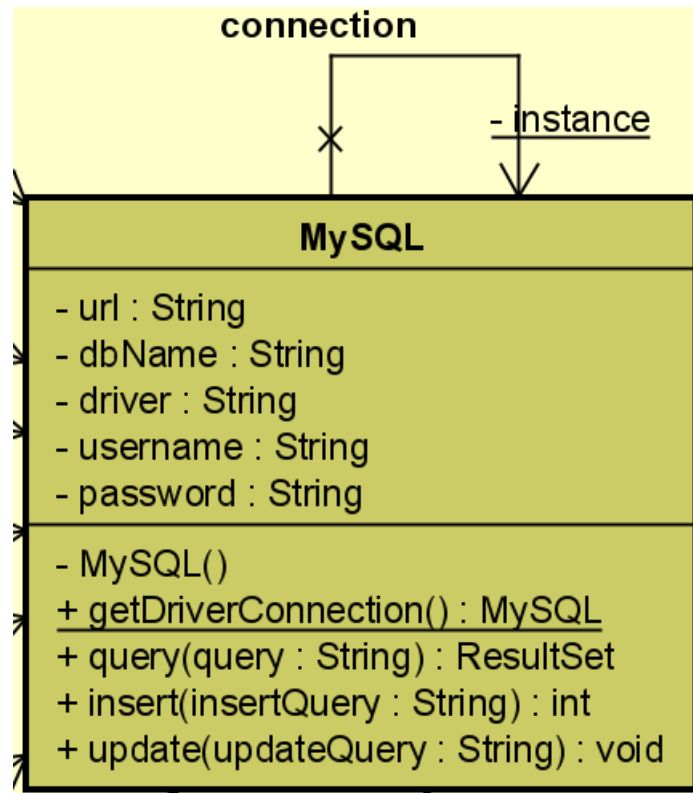
#### 5.5.1 Singleton

Single Pattern là một design pattern mà đảm bảo rằng một class chỉ có duy nhất một instance và cung cấp một cách toàn cầu để truy cập tới instance đó. Thiết kế áp dụng Singleton pattern cho lớp Transaction, Card và MySQLDriver.



Với lớp Transaction, áp dụng thiết kế như vậy nhằm mục đích với mỗi khách hàng, cùng một thời điểm chỉ được đặt một xe. Điều này áp dụng tốt trong thực tế vì mỗi khách chỉ có thể dùng 1 xe tại một thời điểm, tránh gây mất mát tài sản vì trong tương lai, hệ thống sẽ có thể có thêm chức năng theo dõi vị trí của khách hàng khi thuê xe qua app điện thoại.

Lớp Card, áp dụng Singleton pattern để mỗi khách hàng chỉ có thể dùng 1 thẻ để thanh 36 toán. Thiết kế này nếu áp dụng vào thực tế thì sẽ không hợp lí vì khách có thể có nhiều thẻ thanh toán khác nhau. Tuy nhiên, với phần mềm demo ở hiện tại, thiết kế này được coi là khả thi.

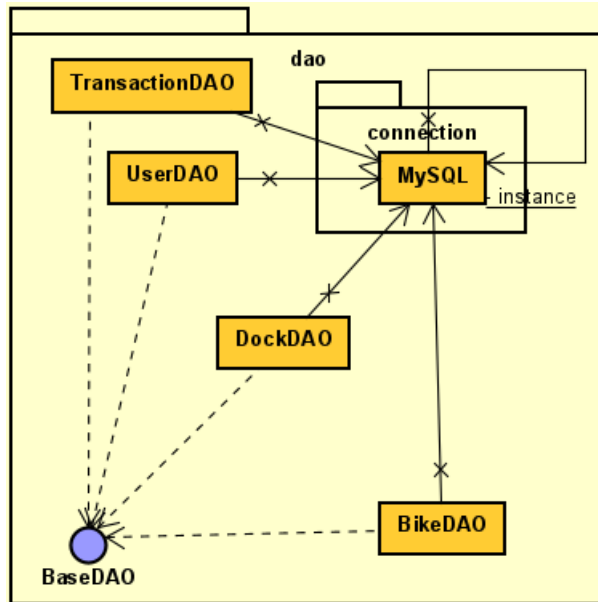


Với lớp MySQLDriver áp dụng pattern này cho MySQLDriver nhằm giúp hệ thống hoạt động tránh gặp lỗi hay xung đột nếu nhà phát triển chẳng may tạo nhiều thực thể MySQLDriver khác nhau tại nhiều vị trí trong phần mềm.

### 5.5.2 DAO - Data Access Object pattern

Data Access Object (DAO) Pattern là một trong những Pattern thuộc nhóm cấu trúc (Structural Pattern). Mẫu thiết kế DAO được sử dụng để phân tách logic lưu trữ dữ liệu trong một lớp riêng biệt. Theo cách này, các service được che dấu về cách các hoạt động cấp thấp để truy cập cơ sở dữ liệu được thực hiện. Nó còn được gọi là nguyên tắc Tách logic (Separation of Logic).



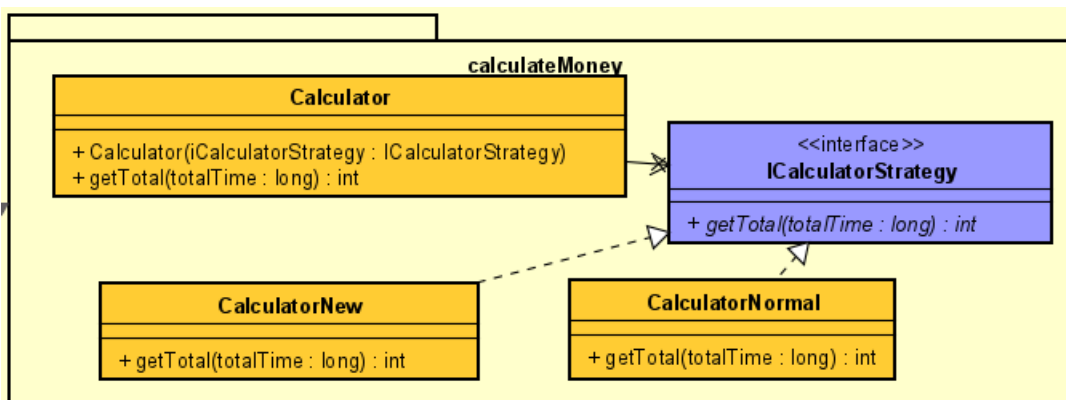


Lớp Interface BaseDAO là một interface định nghĩa các phương thức trừu tượng việc triển khai truy cập dữ liệu cơ bản cho BusinessObject để cho phép truy cập vào nguồn dữ liệu (DataSource).

DockDAO, BikeDAO, TransactionDAO, UserDao cài đặt các phương thức được định nghĩa trong DAO, lớp này sẽ thao tác trực tiếp với nguồn dữ liệu (DataSource).

### 5.5.3 Strategy Pattern

Strategy Pattern là một behavior design pattern. Strategy Pattern được phát biểu như sau: "Xác định một họ chức năng, gói gọn từng chức năng và làm cho chúng có thể thay thế cho nhau". Khi áp dụng Strategy Pattern thì các hành vi hoặc giải thuật của một class có thể thay đổi ở runtime.



Ở modul tính toán, ta có 2 cách tính tiền đó là tính tiền bình thường(CalculatorNormal) và tính tiền mới(CalculatorNew). Ta sẽ tạo 1 Interface là ICalculatorStrategy, interface này định nghĩa method getTotal(). Sau đó ta tạo 2 class CalculatorNormal và CalculatorNew implement ICalculatorStrategy. Và cuối cùng tạo class Calculator để sử dụng Strategy