

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

Viện Công nghệ thông tin và Truyền thông

Tài liệu thiết kế phần mềm

(Software Design Document – SDD)

PHẦN MỀM THUÊ XE ĐẠP THEO GIỜ

Môn: Thiết kế và xây dựng phần mềm

Nhóm 6

Phạm Trung Kiên	20170088
Phạm Minh Khiêm	20170084
Lê Vũ Lợi	20173240

Hà Nội, ngày 8 tháng 10 năm 2020

Table of Contents

1	Giới thiệu.....	4
1.1	Mục tiêu.....	4
1.2	Phạm vi.....	4
1.3	Từ điển thuật ngữ.....	4
1.4	Tài liệu tham khảo.....	4
2	Kiến trúc hệ thống và thiết kế kiến trúc.....	5
2.1	Mẫu thiết kế kiến trúc.....	5
2.2	Biểu đồ trình tự.....	5
2.3	Biểu đồ lớp phân tích.....	7
2.4	Biểu đồ lớp phân tích gộp.....	8
3	Thiết kế chi tiết.....	9
3.1	Thiết kế giao diện.....	9
3.1.1	Thiết kế giao diện người dùng.....	9
3.1.2	Thiết kế giao diện hệ thống.....	20
3.2	Mô hình hóa dữ liệu.....	24
3.2.1	Mô hình hóa dữ liệu mức khái niệm.....	24
3.2.2	Thiết kế cơ sở dữ liệu.....	24
3.3	Thiết kế chi tiết lớp.....	30
3.3.1	Biểu đồ lớp tổng quan.....	30
3.3.2	Biểu đồ lớp cho Interbank subsystem.....	31
3.3.3	Thiết kế chi tiết lớp.....	31
4	Các vấn đề thiết kế.....	50
4.1	Coupling and cohesion.....	50
4.1.1	High cohesion.....	50
4.1.2	Loose coupling.....	50
4.2	Các nguyên tắc thiết kế.....	51

4.2.1 Single Responsibility Principle.....	51
4.2.2 Open/Closed Principle.....	51
4.2.3 Liskov substitution principle.....	52
4.2.4 Interface segregation principle.....	52
4.2.5 Dependency Inversion principle.....	53
4.3 Design Pattern.....	53

1 Giới thiệu

1.1 Mục tiêu

- Mục tiêu của tài liệu này là đưa ra thiết kế phần mềm cho ứng dụng EcoBikeRental cho thuê xe đạp theo giờ với nhiều bãi để xe để thuê/trả xe tự động trong khu đô thị Ecopark.
- Tài liệu này phục vụ cho designers, developers và testers của hệ thống có thể dễ dàng đi tới các bước tiếp theo của quy trình phát triển ứng dụng này.

1.2 Phạm vi

Thuê xe đạp dạo quanh là một trong những dịch vụ được thu hút nhất tại khu đô thị sinh thái Ecopark. Hiện nay có 2 điểm cho thuê và đỗ xe đạp tại khu đô thị. Để dịch vụ này tiếp tục phát triển mở rộng ra, cần giải quyết khâu hạ tầng, bao gồm hệ thống làn đường cho xe đạp, điểm dừng, trông giữ, bảo quản, và đặc biệt là hệ thống thông tin thuê xe và trả xe tự động có thể hoạt động 24/7

1.3 Từ điển thuật ngữ

STT	Thuật ngữ, từ viết tắt	Giải thích
1	API	viết tắt của Application Programming Interface, là phần mềm trung gian cho phép kết nối 2 ứng dụng với nhau

1.4 Tài liệu tham khảo

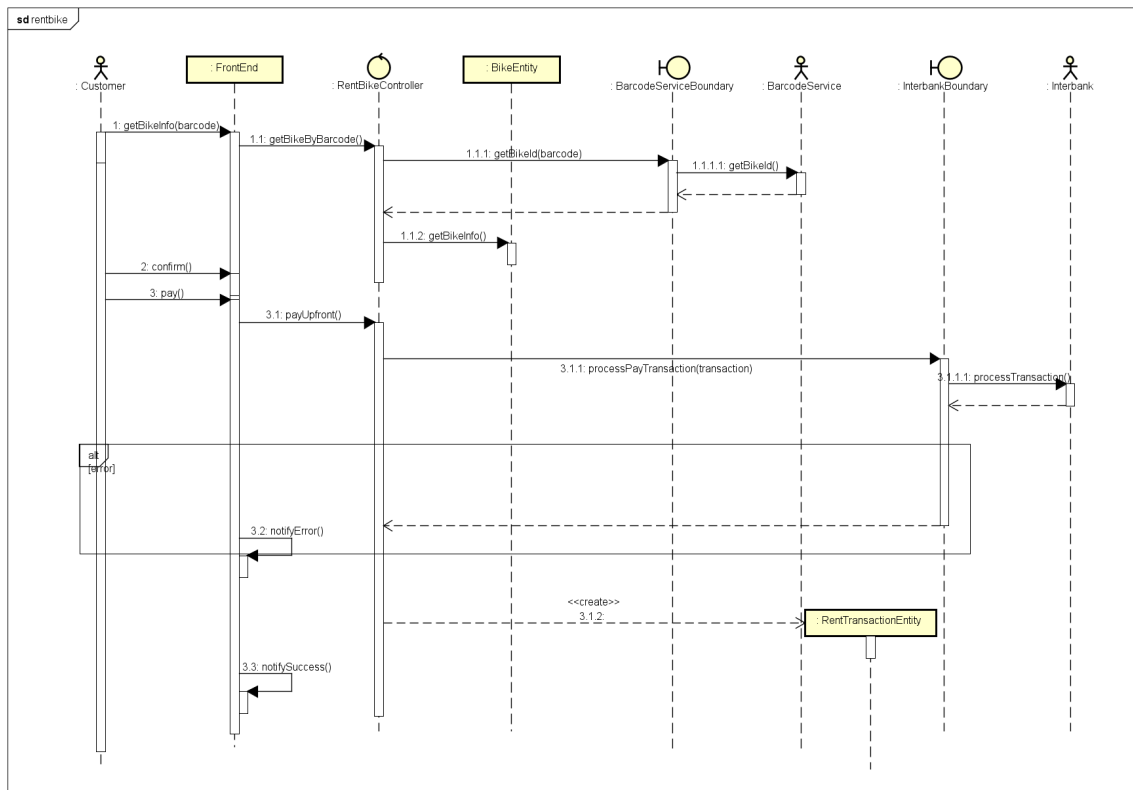
STT	Tên tài liệu
1	D. Budgen. Software Design, 2nd Edition. Addison-Wesley. 2004

2 Kiến trúc hệ thống và thiết kế kiến trúc

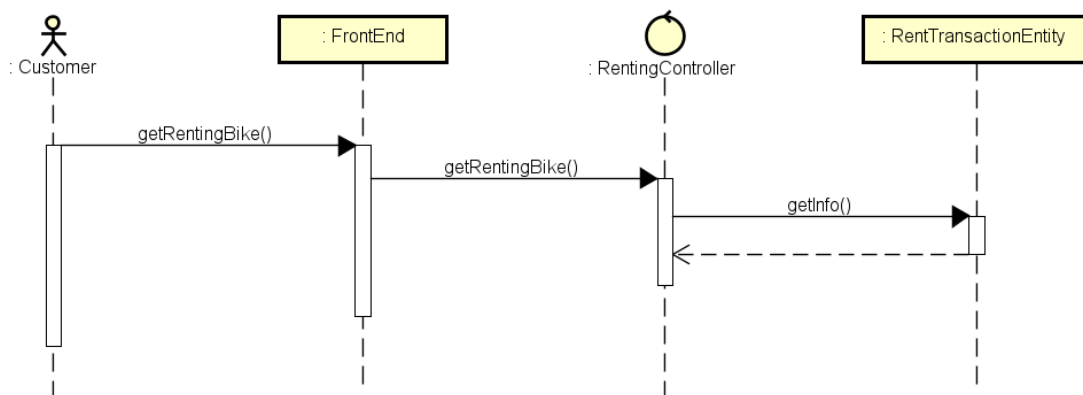
2.1 Mẫu thiết kế kiến trúc

Nhóm chọn thiết kế theo kiến trúc Frontend – Backend

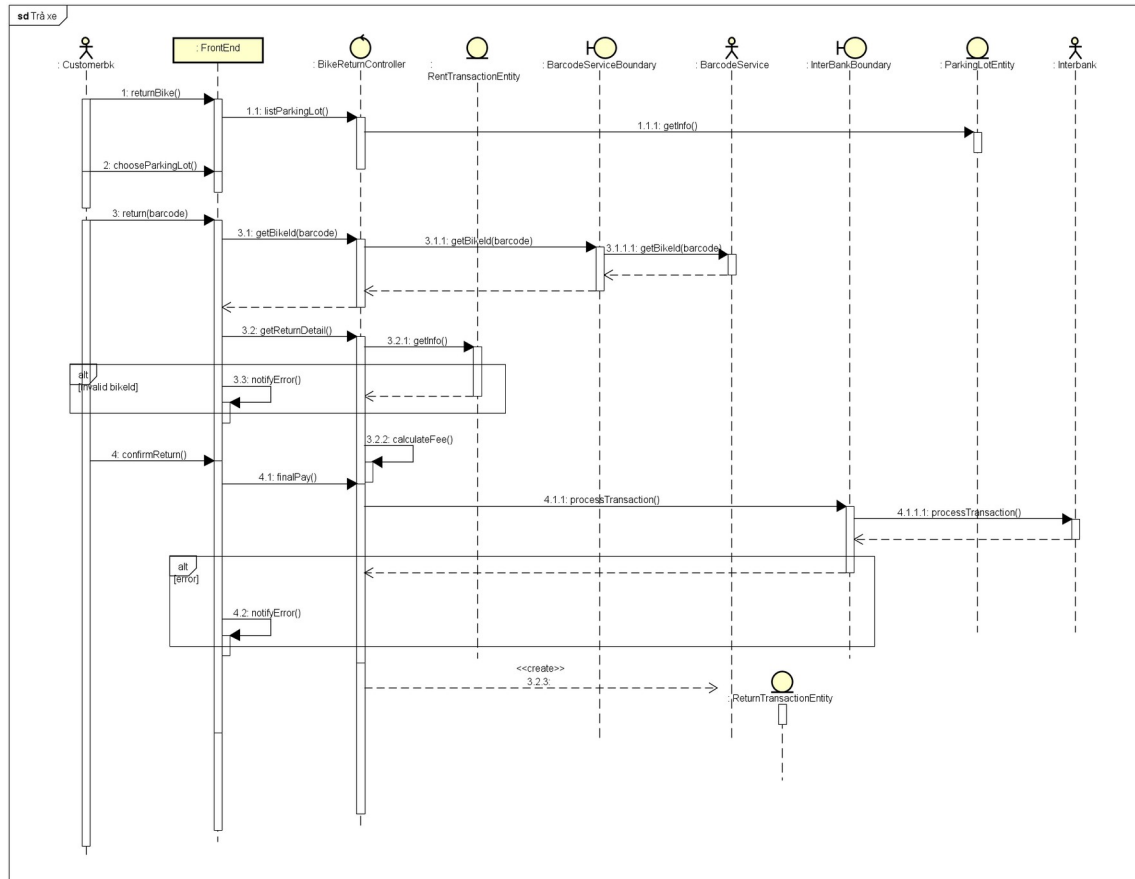
2.2 Biểu đồ trình tự



Hình : Biểu đồ trình tự cho thuê xe

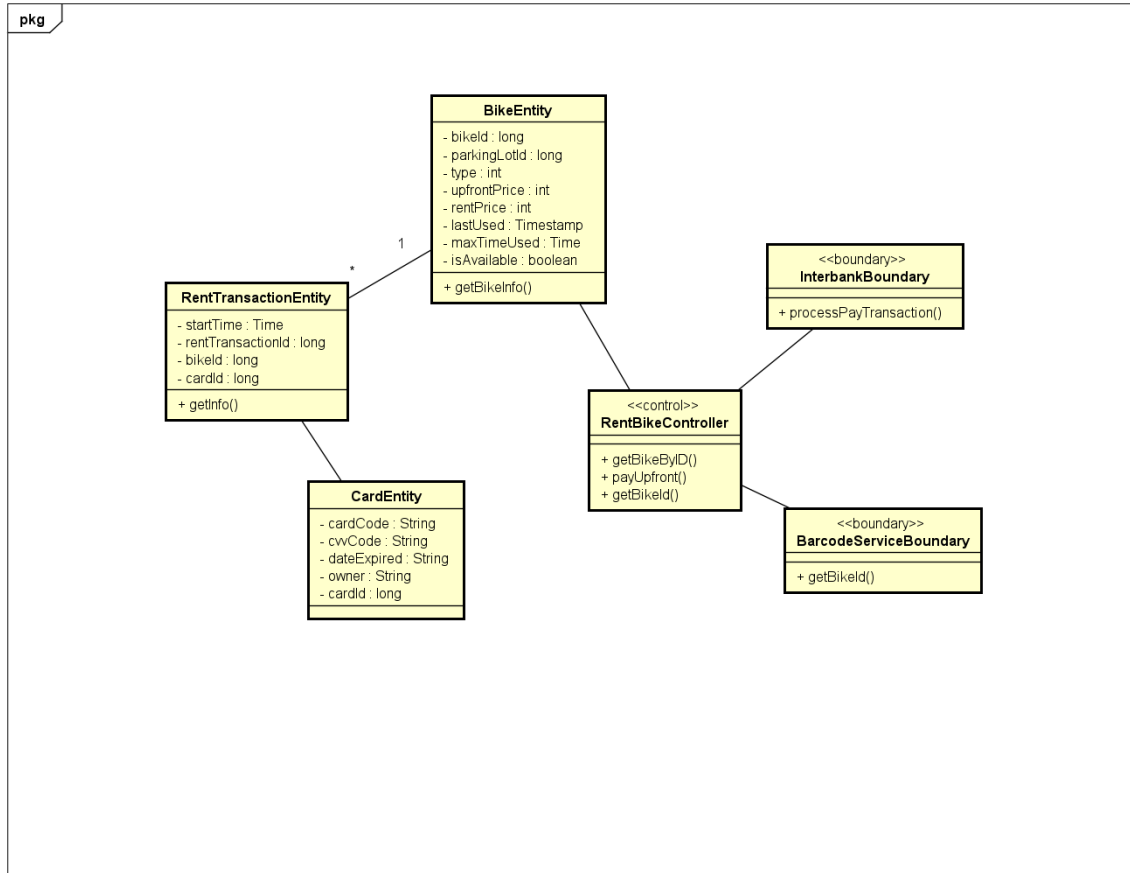


Hình : Biểu đồ trình tự cho xem xe đang thuê

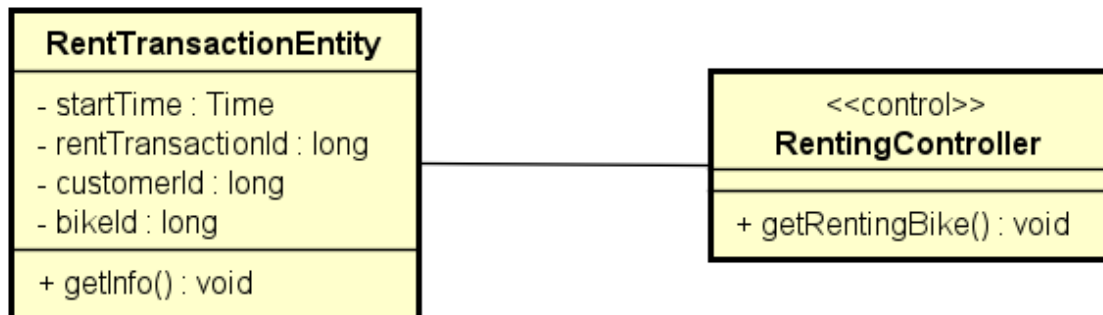


Hình : Biểu đồ trình tự cho trả xe

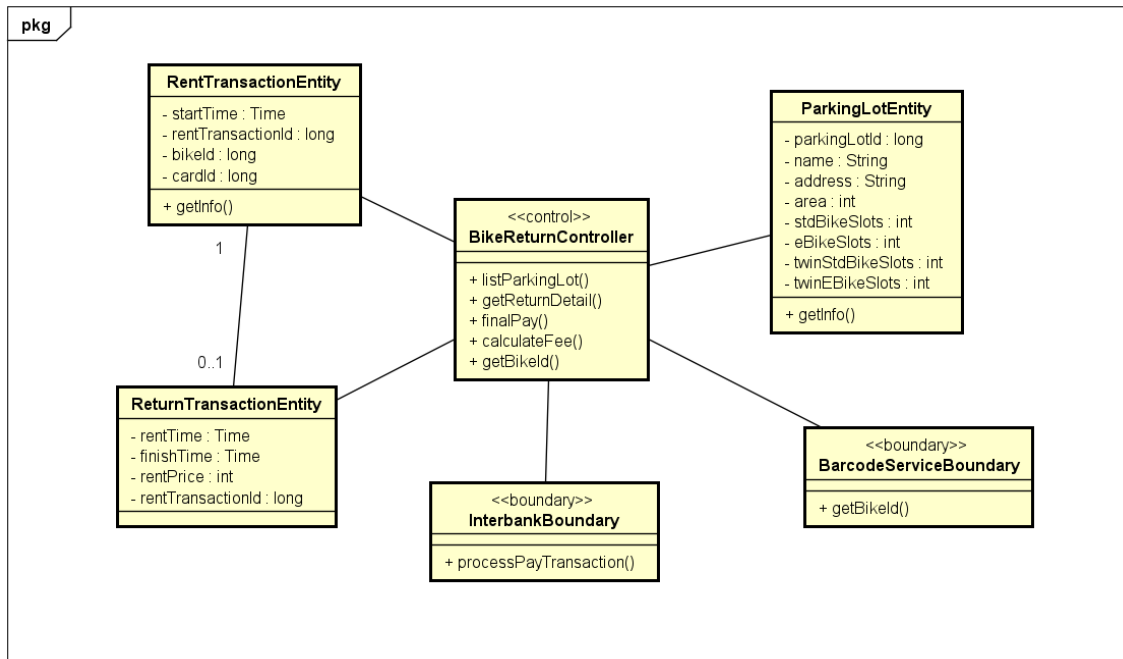
2.3 Biểu đồ lớp phân tích



Hình : Biểu đồ cho thuê xe

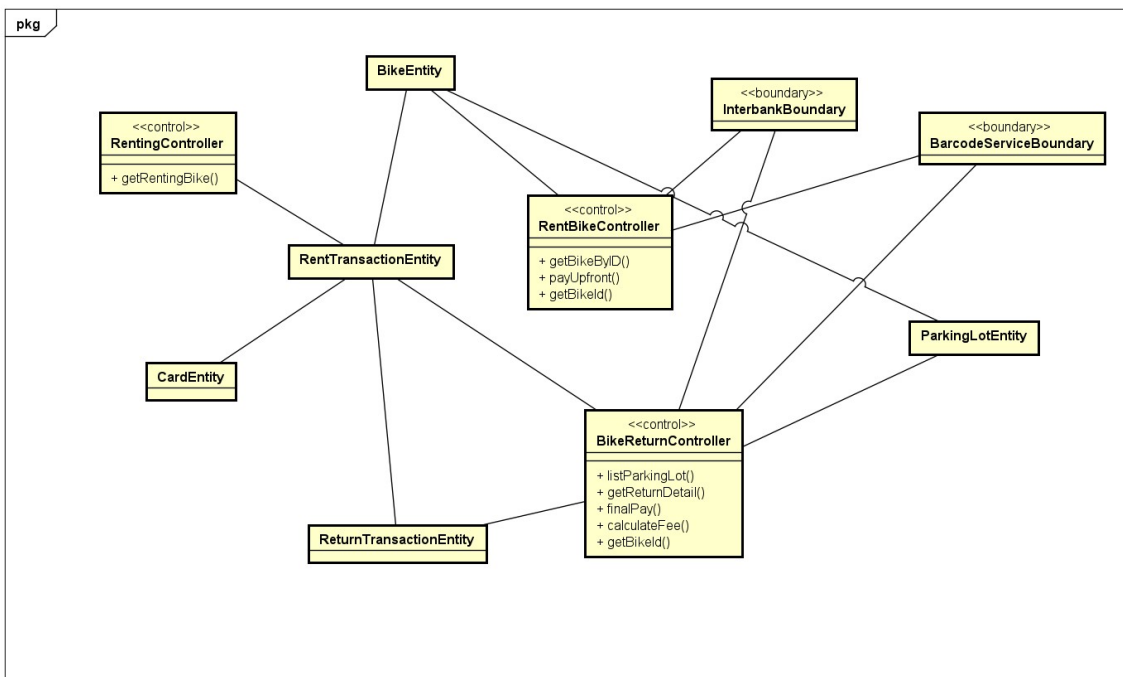


Hình : Biểu đồ cho xem xe đang thuê



Hình : Biểu đồ cho trả xe

2.4 Biểu đồ lớp phân tích gộp

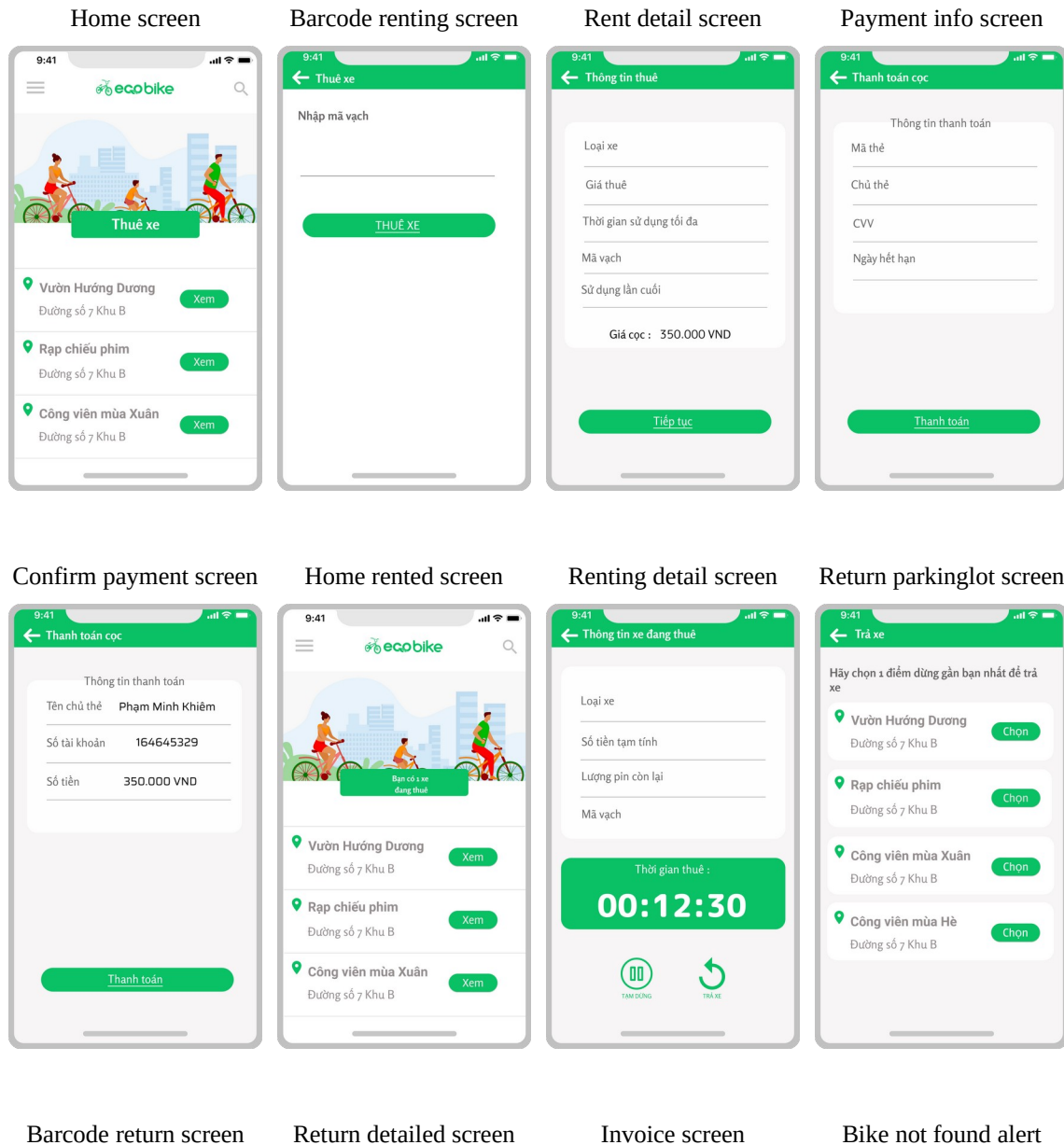


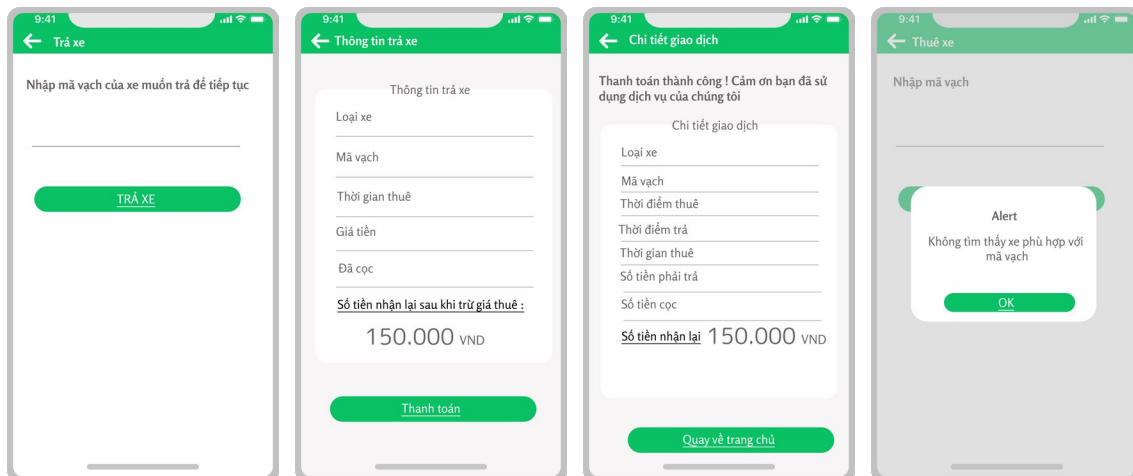
3 Thiết kế chi tiết

3.1 Thiết kế giao diện

3.1.1 Thiết kế giao diện người dùng

3.1.1.1 Danh sách các màn hình



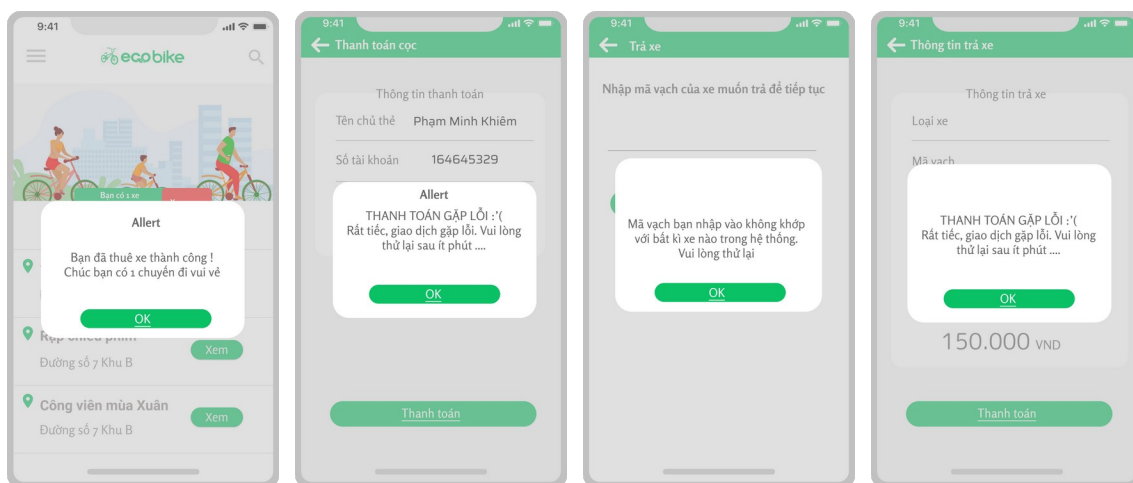


Success renting alert

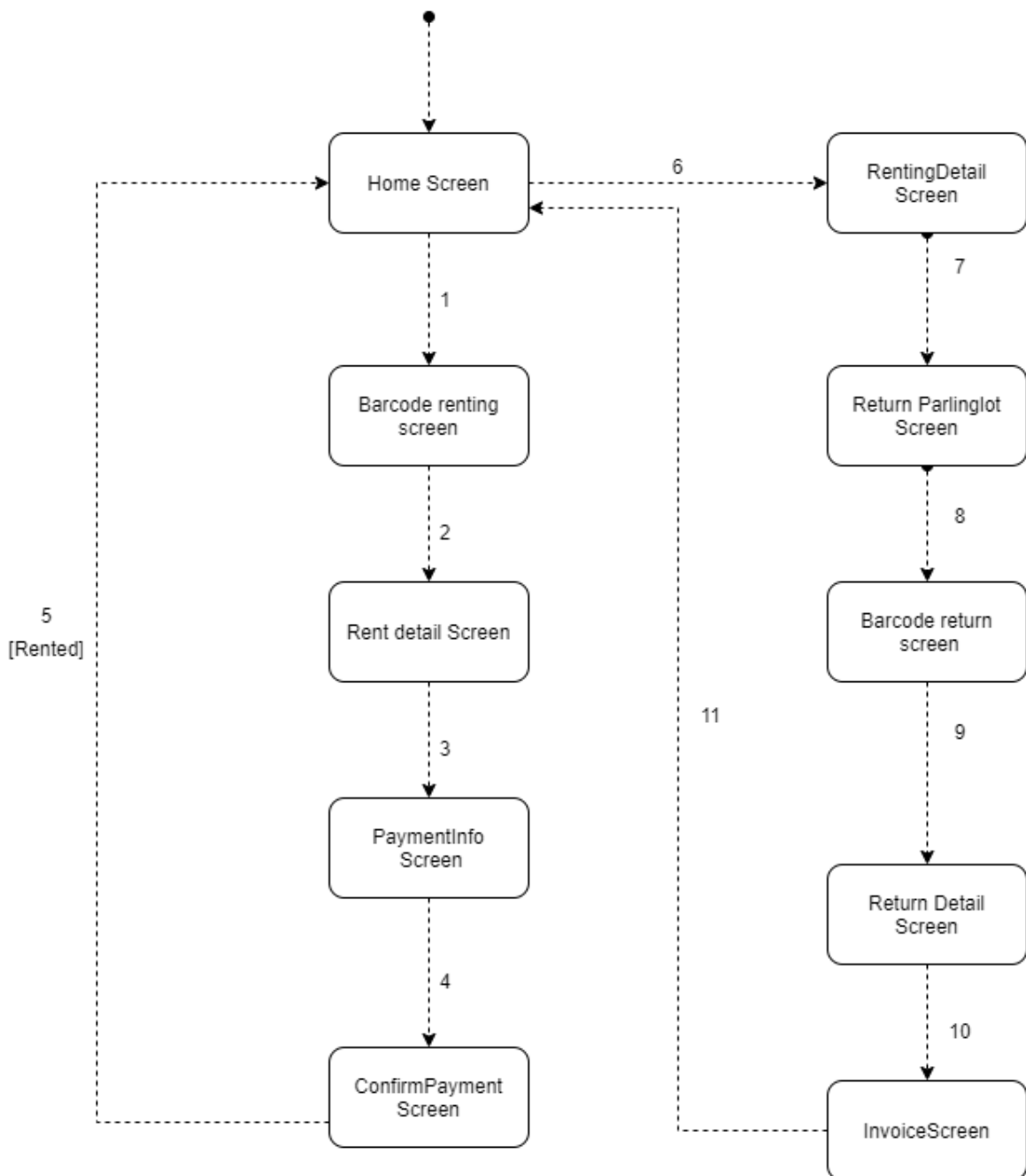
Rent payment failed alert

Barcode return error

Payment error alert





3.1.1.2 Sơ đồ dịch chuyển màn hình



3.1.1.3 Đặc tả màn hình


EcoBike		Date of creation	Approved by	Reviewed by	Person in charge
Screen specification	Home screen				Phạm Minh Khiêm

	<i>Control</i>	<i>Operation</i>	<i>Function</i>
	<i>Khu vực hiển thị các bãi đỗ xe</i>	<i>Initial</i>	<i>Hiển thị tên và địa chỉ của các bãi đỗ xe gần nhất</i>
	<i>Nút Xem</i>	<i>Click</i>	<i>Xem thông tin chi tiết bãi đỗ xe</i>
	<i>Nút Thuê xe</i>	<i>Click</i>	<i>Hiển thị màn hình nhập barcode</i>

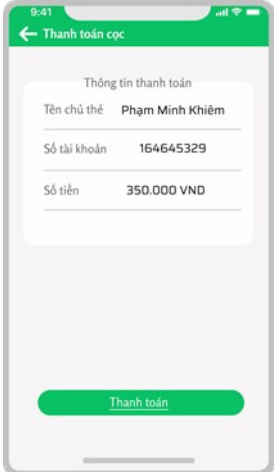
<i>EcoBike</i>		<i>Date of creation</i>	<i>Approved by</i>	<i>Reviewed by</i>	<i>Person in charge</i>
<i>Screen specification</i>	<i>Barcode renting screen</i>				<i>Phạm Minh Khiêm</i>
	<i>Control</i>	<i>Operation</i>	<i>Function</i>		
	<i>Nút Thuê xe</i>	<i>Click</i>	<i>Thuê xe có barcode tương ứng</i>		
	<i>Khu vực nhập mã vạch</i>	<i>Type</i>	<i>Nhập mã vạch</i>		


<i>EcoBike</i>		<i>Date of creation</i>	<i>Approved by</i>	<i>Reviewed by</i>	<i>Person in charge</i>
<i>Screen specification</i>	<i>Rent detail screen</i>				<i>Phạm Minh Khiêm</i>
		<i>Control</i>	<i>Operation</i>	<i>Function</i>	

	<i>Khu vực hiển thị thông tin xe</i>	<i>Initial</i>	<i>Hiển thị thông tin chi tiết của xe tương ứng</i>
	<i>Khu vực hiển thị giá cọc</i>	<i>Initial</i>	<i>Hiển thị giá cọc</i>
	<i>Nút Tiếp tục</i>	<i>Click</i>	<i>Hiển thị màn hình thanh toán</i>

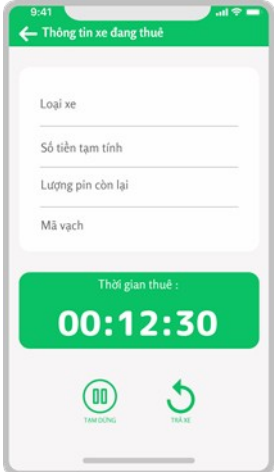
<i>EcoBike</i>		<i>Date of creation</i>	<i>Approved by</i>	<i>Reviewed by</i>	<i>Person in charge</i>
<i>Screen specification</i>	<i>Payment info screen</i>				<i>Phạm Minh Khiêm</i>
	<i>Control</i>	<i>Operation</i>	<i>Function</i>		
	<i>Khu vực nhập thông tin thanh toán</i>	<i>Initial</i>	<i>Nhập thông tin thanh toán</i>		
	<i>Nút Thanh toán</i>	<i>Click</i>	<i>Thực hiện đặt cọc</i>		

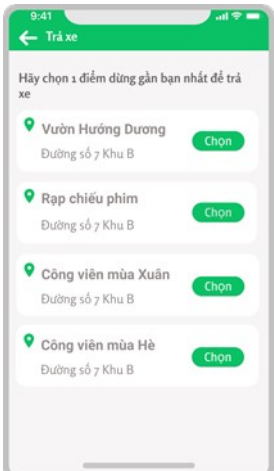
<i>EcoBike</i>		<i>Date of creation</i>	<i>Approved by</i>	<i>Reviewed by</i>	<i>Person in charge</i>
<i>Screen specification</i>	<i>Confirm payment screen</i>				<i>Phạm Minh Khiêm</i>
		<i>Control</i>	<i>Operation</i>	<i>Function</i>	

	<i>Khu vực hiển thị thông tin thanh toán</i>	<i>Initial</i>	<i>Hiển thị tên và địa chỉ của các bãi đỗ xe gần nhất</i>
	<i>Nút Thanh toán</i>	<i>Click</i>	<i>Xác nhận giao dịch</i>

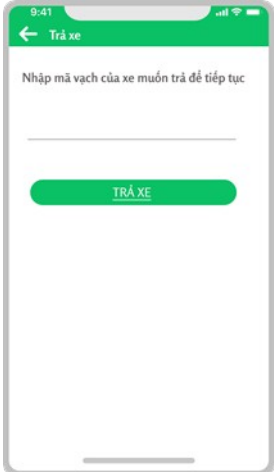
<i>EcoBike</i>		<i>Date of creation</i>	<i>Approved by</i>	<i>Reviewed by</i>	<i>Person in charge</i>
<i>Screen specification</i>	<i>Home rented screen</i>				<i>Phạm Minh Khiêm</i>
	<i>Control</i>	<i>Operation</i>	<i>Function</i>		
	<i>Nút Bạn đang có 1 xe đang thuê</i>	<i>Click</i>	<i>Xem thông tin xe đang thuê</i>		
	<i>Nút Xem</i>	<i>Click</i>	<i>Xem thông tin chi tiết bãi đỗ xe</i>		


<i>EcoBike</i>		<i>Date of creation</i>	<i>Approved by</i>	<i>Reviewed by</i>	<i>Person in charge</i>
<i>Screen specification</i>	<i>Renting detail screen</i>				<i>Phạm Minh Khiêm</i>
		<i>Control</i>	<i>Operation</i>	<i>Function</i>	

	<i>Khu vực hiển thị thông tin xe đang thuê</i>	<i>Initial</i>	<i>Hiển thị thông tin chi tiết của xe đang thuê</i>
	<i>Nút Tạm dừng</i>	<i>Click</i>	<i>Tạm dừng thuê xe</i>
	<i>Nút Trả xe</i>	<i>Click</i>	<i>Hiển thị màn hình trả xe</i>


<i>EcoBike</i>		<i>Date of creation</i>	<i>Approved by</i>	<i>Reviewed by</i>	<i>Person in charge</i>
<i>Screen specification</i>	<i>Return parkinglot screen</i>				<i>Phạm Minh Khiêm</i>
	<i>Control</i>	<i>Operation</i>	<i>Function</i>		
	<i>Khu vực hiển thị các bãi đỗ xe</i>	<i>Initial</i>	<i>Hiển thị tên và địa chỉ của các bãi đỗ xe gần nhất</i>		
	<i>Nút Chọn</i>	<i>Click</i>	<i>Chọn bãi đỗ xe để trả</i>		

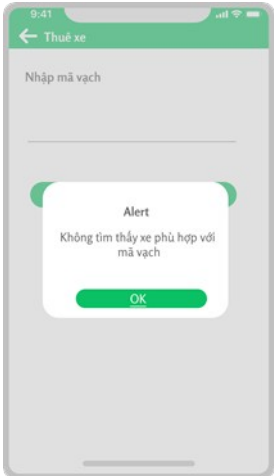
<i>EcoBike</i>		<i>Date of creation</i>	<i>Approved by</i>	<i>Reviewed by</i>	<i>Person in charge</i>
<i>Screen specification</i>	<i>Barcode return screen</i>				<i>Phạm Minh Khiêm</i>
		<i>Control</i>	<i>Operation</i>	<i>Function</i>	

	<i>Khu vực nhập mã vạch</i>	<i>Initial</i>	<i>Nhập mã vạch xe muốn trả</i>
	<i>Nút Trả xe</i>	<i>Click</i>	<i>Hiển thị màn hình trả xe</i>

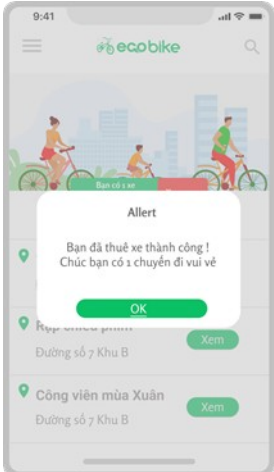
<i>EcoBike</i>		<i>Date of creation</i>	<i>Approved by</i>	<i>Reviewed by</i>	<i>Person in charge</i>
<i>Screen specification</i>	<i>Home screen</i>				<i>Phạm Minh Khiêm</i>
	<i>Control</i>	<i>Operation</i>	<i>Function</i>		
	<i>Khu vực hiển thị thông tin trả xe</i>	<i>Initial</i>	<i>Hiển thị thông tin chi tiết trả xe</i>		
	<i>Nút Thanh toán</i>	<i>Click</i>	<i>Xác nhận trả xe</i>		

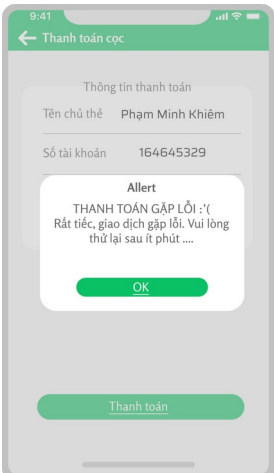
<i>EcoBike</i>		<i>Date of creation</i>	<i>Approved by</i>	<i>Reviewed by</i>	<i>Person in charge</i>
<i>Screen specification</i>	<i>Invoice screen</i>				<i>Phạm Minh Khiêm</i>
		<i>Control</i>	<i>Operation</i>	<i>Function</i>	

	<i>Khu vực hiển thị thông tin giao dịch trả xe</i>	<i>Initial</i>	<i>Hiển thị thông tin giao dịch trả xe</i>
	<i>Nút Quay về trang chủ</i>	<i>Click</i>	<i>Về trang chủ</i>

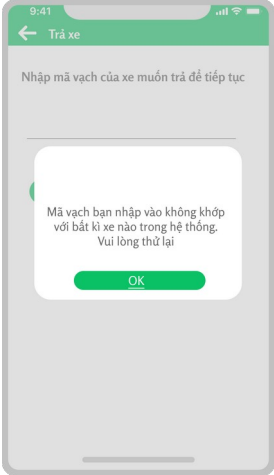
<i>EcoBike</i>		<i>Date of creation</i>	<i>Approved by</i>	<i>Reviewed by</i>	<i>Person in charge</i>
<i>Screen specification</i>	<i>Bike not found alert</i>				<i>Phạm Minh Khiêm</i>
	<i>Control</i>	<i>Operation</i>	<i>Function</i>		
	<i>Thông báo không thấy xe</i>	<i>Initial</i>	<i>Thông báo không thấy xe phù hợp mã vạch</i>		
	<i>Nút OK</i>	<i>Click</i>	<i>Đóng thông báo</i>		

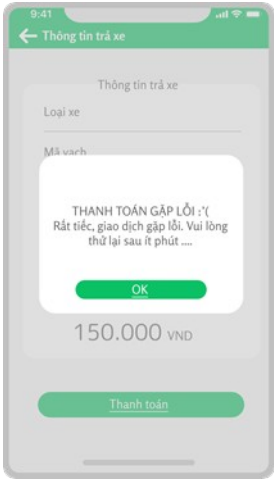
<i>EcoBike</i>		<i>Date of creation</i>	<i>Approved by</i>	<i>Reviewed by</i>	<i>Person in charge</i>
<i>Screen specification</i>	<i>Success renting alert</i>				<i>Phạm Minh Khiêm</i>
		<i>Control</i>	<i>Operation</i>	<i>Function</i>	

	Thông báo thuê xe thành công	Initial	Thông báo đã thuê xe thành công
	Nút OK	Click	Đóng thông báo

EcoBike		Date of creation	Approved by	Reviewed by	Person in charge
Screen specification	Rent payment failed alert				Phạm Minh Khiêm
		Control	Operation	Function	
		Thông báo thanh toán lỗi	Initial	Thông báo thanh toán gặp lỗi khi thuê xe	
		Nút OK	Click	Đóng thông báo	

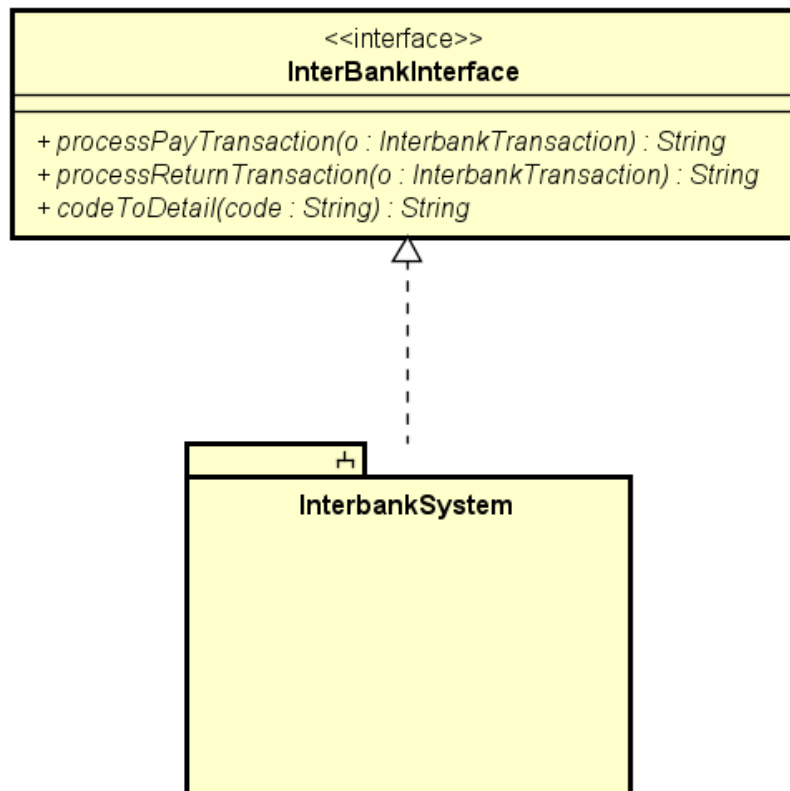
EcoBike		Date of creation	Approved by	Reviewed by	Person in charge
Screen specification	Barcode return error				Phạm Minh Khiêm

	<i>Control</i>	<i>Operation</i>	<i>Function</i>
	<i>Thông báo mã vạch không khớp</i>	<i>Initial</i>	<i>Thông báo mã vạch nhập vào không khớp với xe nào</i>
	<i>Nút OK</i>	<i>Click</i>	<i>Đóng thông báo</i>

<i>EcoBike</i>		<i>Date of creation</i>	<i>Approved by</i>	<i>Reviewed by</i>	<i>Person in charge</i>
<i>Screen specification</i>	<i>Payment error alert</i>				<i>Phạm Minh Khiêm</i>
	<i>Control</i>	<i>Operation</i>	<i>Function</i>		
	<i>Thông báo thanh toán lỗi</i>	<i>Initial</i>	<i>Thông báo thanh toán gặp lỗi khi trả xe</i>		
	<i>Nút OK</i>	<i>Click</i>	<i>Đóng thông báo</i>		

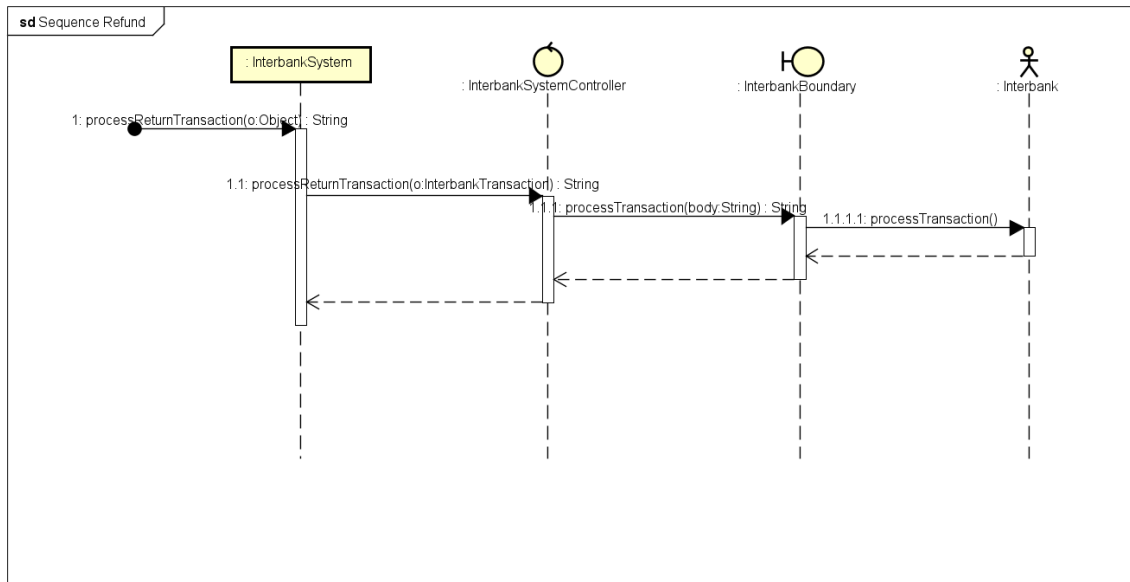
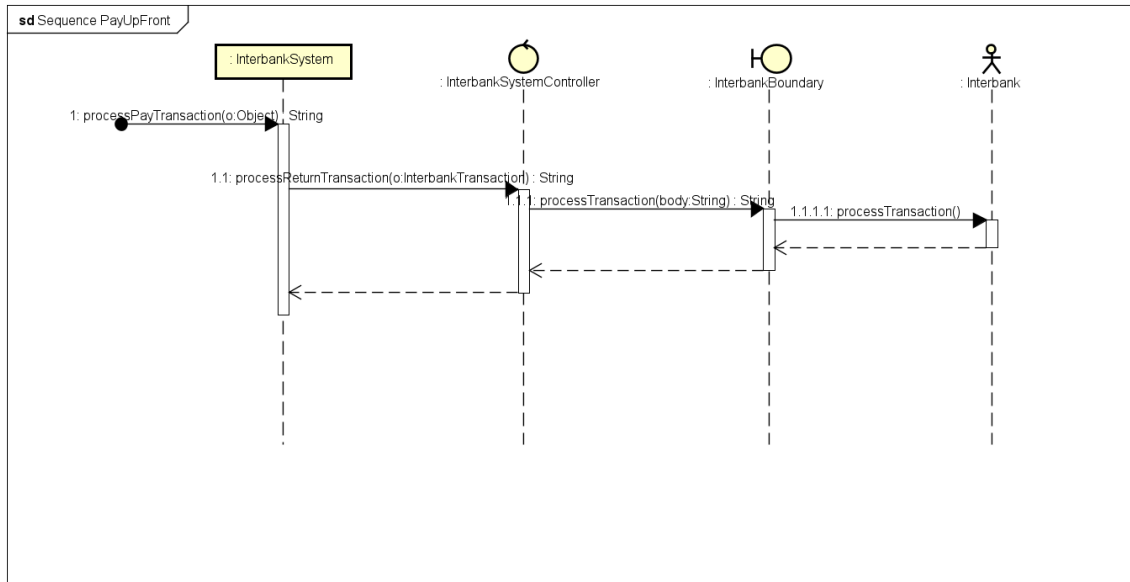
3.1.2 Thiết kế giao diện hệ thống

3.1.2.1 Thiết kế interface cho Interbank subsystem:

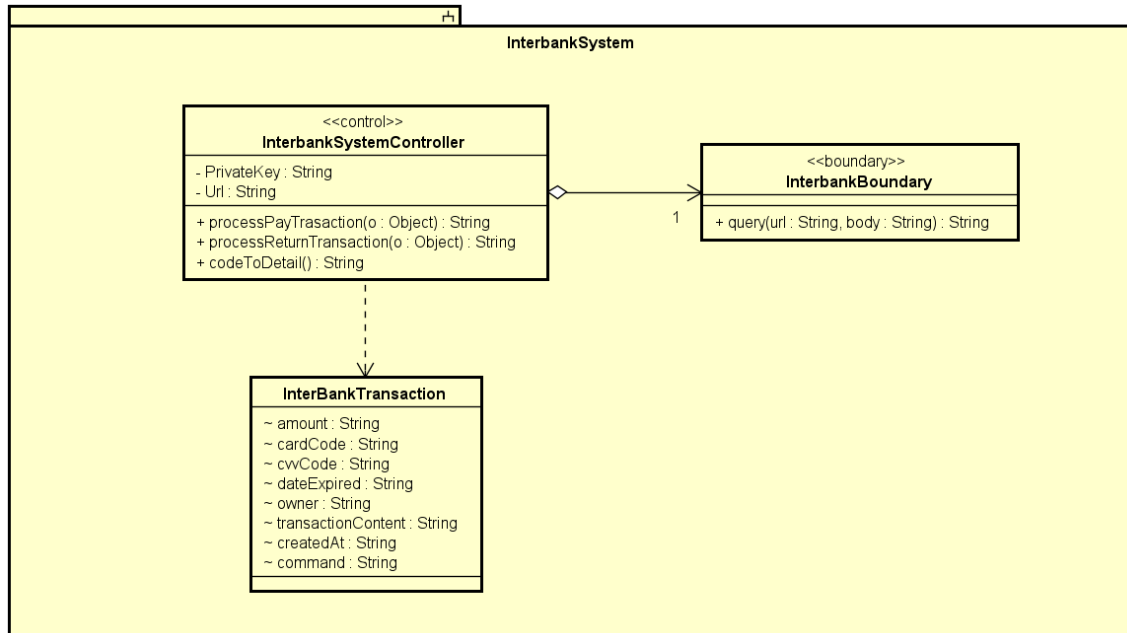


3.1.2.2 Thiết kế *Interbank subsystem*:

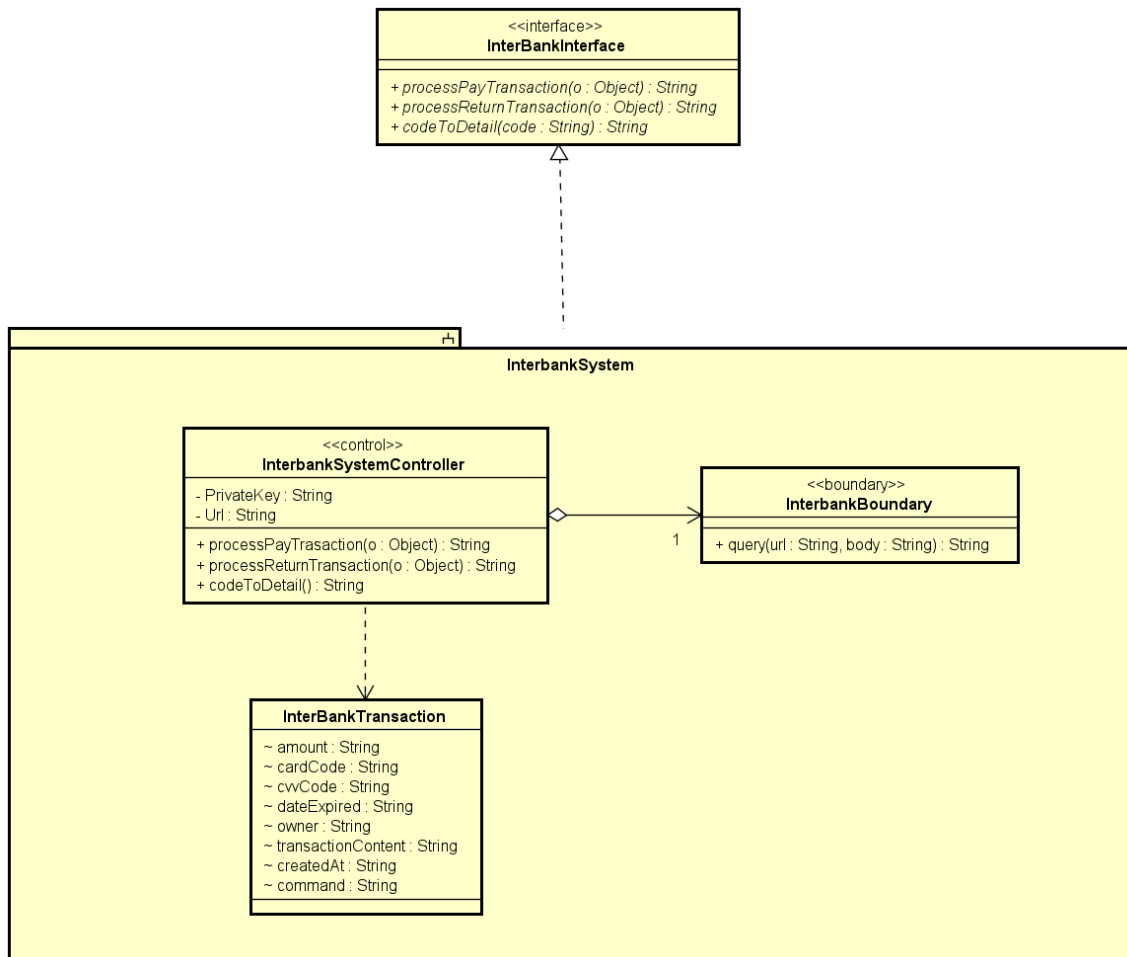
3.1.2.2.1 Phân bố hành vi của subsystem cho các thành phần bên trong subsystem



3.1.2.2.2 Mô tả các thành phần của subsystem

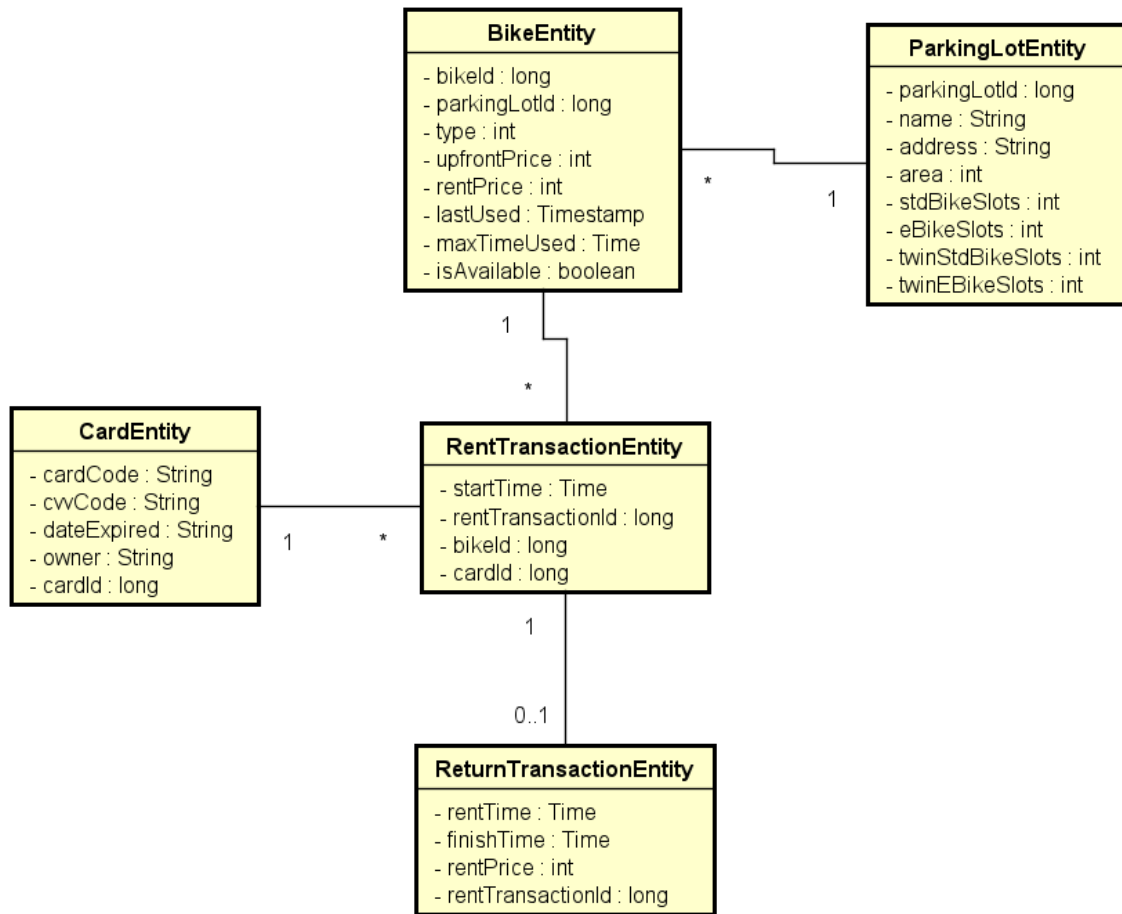


3.1.2.2.3 Tổng hợp thiết kế của subsystem



3.2 Mô hình hóa dữ liệu

3.2.1 Mô hình hóa dữ liệu mức khái niệm

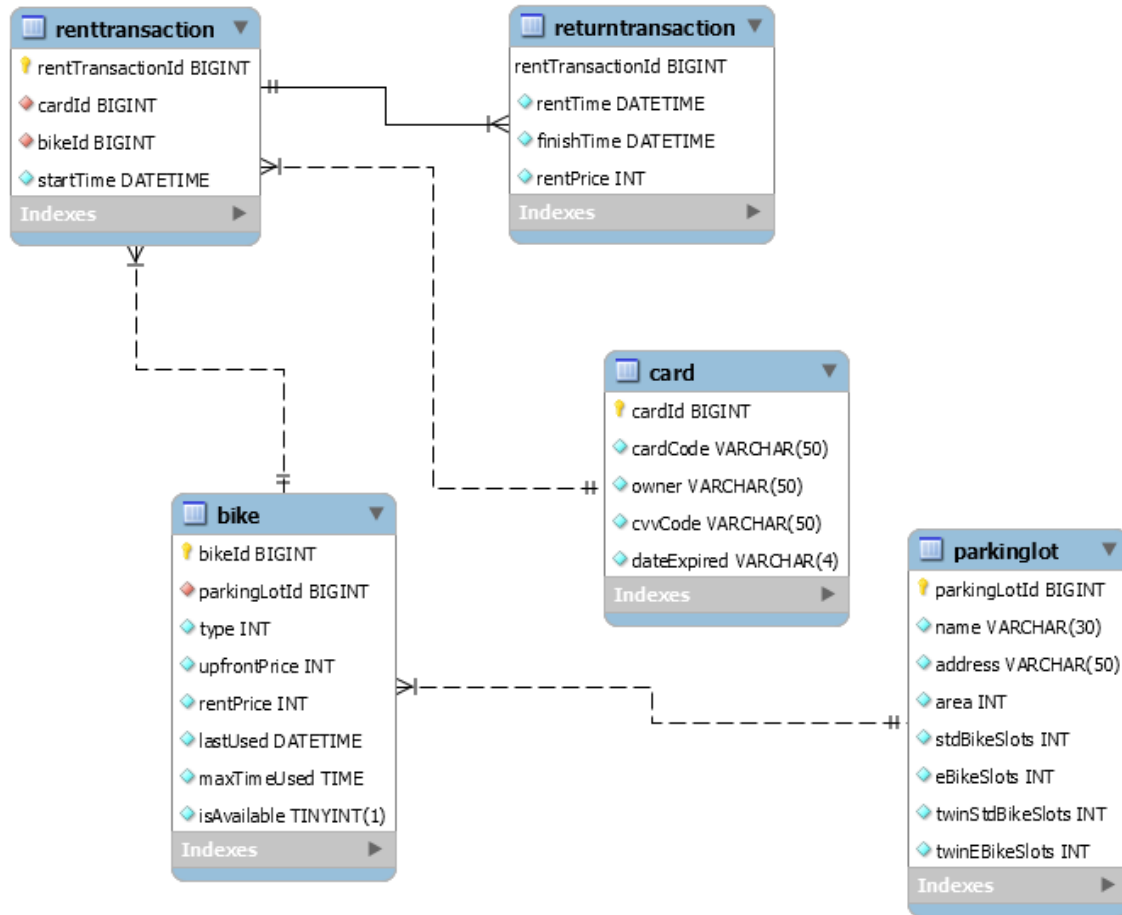


3.2.2 Thiết kế cơ sở dữ liệu

3.2.2.1 Hệ quản trị cơ sở dữ liệu

Hệ quản trị dữ liệu *mySQL*

3.2.2.2 Mô hình hóa dữ liệu mức logic



3.2.2.3 Mô hình hóa dữ liệu mức vật lý

3.2.2.3.1 Schemas

Field	Type	Null	Key	Default	Extra
parkingLotId	bigint	NO	PRI	NULL	auto_increment
name	varchar(30)	NO		NULL	
address	varchar(50)	NO		NULL	
area	int	NO		NULL	
stdBikeSlots	int	NO		NULL	
eBikeSlots	int	NO		NULL	
twinStdBikeSlots	int	NO		NULL	
twinEBikeSlots	int	NO		NULL	

Bang:

Field	Type	Null	Key	Default	Extra
cardId	bigint	NO	PRI	NULL	auto_increment
cardCode	varchar(50)	NO		NULL	
owner	varchar(50)	NO		NULL	

cvvCode	varchar(50)	NO		NULL	
dateExpired	varchar(4)	NO		NULL	

Bang:

Field	Type	Null	Key	Default	Extra
bikeId	bigint	NO	PRI	NULL	auto_increment
parkingLotId	bigint	NO	MUL	NULL	
type	int	NO		NULL	
upfrontPrice	int	NO		NULL	
rentPrice	int	NO		NULL	
lastUsed	datetime	NO		NULL	
maxTimeUsed	time	NO		NULL	
isAvailable	tinyint(1)	NO		NULL	

Bang:

Field	Type	Null	Key	Default	Extra
rentTransactionId	bigint	NO	PRI	NULL	auto_increment
cardId	bigint	NO	MUL	NULL	
bikeId	bigint	NO	MUL	NULL	
startTime	datetime	NO		NULL	

Bang:

Field	Type	Null	Key	Default	Extra
rentTransactionId	bigint	NO	PRI	NULL	
rentTime	datetime	NO		NULL	
finishTime	datetime	NO		NULL	
rentPrice	int	NO		NULL	

3.2.2.3.2 SQL Scripts

Create schemas

```
create table parkinglot (
    parkingLotId bigint primary key auto_increment,
    name varchar(30) not null,
    address varchar(50) not null,
    area int not null,
    stdBikeSlots int not null,
    eBikeSlots int not null,
    twinStdBikeSlots int not null,
    twinEBikeSlots int not null
);
```

```

create table card (
    cardId bigint primary key auto_increment,
    cardCode varchar(50) not null,
    owner varchar(50) not null,
    cvvCode varchar(50) not null,
    dateExpired varchar(4) not null
);

create table bike (
    bikeId bigint primary key auto_increment,
    parkingLotId bigint not null,
    type int not null,
    upfrontPrice int not null,
    rentPrice int not null,
    lastUsed datetime not null,
    maxTimeUsed time not null,
    isAvailable bool not null,
    foreign key (parkingLotId) references parkinglot(parkingLotId)
);

create table renttransaction (
    rentTransactionId bigint primary key auto_increment,
    cardId bigint not null,
    bikeId bigint not null,
    startTime datetime not null,
    foreign key (cardId) references card(cardId),
    foreign key (bikeId) references bike(bikeId)
);

```

```

create table returntransaction (
    rentTransactionId bigint primary key,
    rentTime datetime not null,
    finishTime datetime not null,
    rentPrice int not null,
    foreign key (rentTransactionId) references renttransaction(rentTransactionId)
);

```

Insert data

```

insert into parkinglot(name, address, area, stdBikeSlots, eBikeSlots, twinStdBikeSlots,
twinEBikeSlots)

```

```

values ('Vườn hoa Hướng Dương', 'Ha Noi', 1000, 30, 30, 30, 30);

```

```

insert into parkinglot(name, address, area, stdBikeSlots, eBikeSlots, twinStdBikeSlots,
twinEBikeSlots)

```

```

values ('Hồ con cá', 'Ha Noi', 1200, 40, 30, 30, 30);

```

```

insert into parkinglot(name, address, area, stdBikeSlots, eBikeSlots, twinStdBikeSlots,
twinEBikeSlots)

```

```

values ('Công viên Thống Nhất', 'Ha Noi', 1200, 30, 30, 40, 30);

```

```

insert into parkinglot(name, address, area, stdBikeSlots, eBikeSlots, twinStdBikeSlots,
twinEBikeSlots)

```

```

values ('Viện bảo tàng', 'Ha Noi', 1200, 30, 40, 30, 30);

```

```

insert into parkinglot(name, address, area, stdBikeSlots, eBikeSlots, twinStdBikeSlots,
twinEBikeSlots)

```

```

values ('Parking Lot 5', 'Ha Noi', 1200, 30, 30, 30, 40);

```

```

-- -----

```

```

insert into bike(parkingLotId, type, upfrontPrice, rentPrice, lastUsed, maxTimeUsed,isAvailable)

```

```

values (1, 1, 100, 20, '2018-01-01', '6:00', true);

```

```

insert into bike(parkingLotId, type, upfrontPrice, rentPrice, lastUsed, maxTimeUsed,isAvailable)

```

```

values (2, 1, 100, 20, '2018-01-01', '6:00', true);

```

```

insert into bike(parkingLotId, type, upfrontPrice, rentPrice, lastUsed, maxTimeUsed,isAvailable)

```

```

values (2, 2, 200, 25, '2018-01-01', '6:00', true);

```

```

insert into bike(parkingLotId, type, upfrontPrice, rentPrice, lastUsed, maxTimeUsed,isAvailable)

```

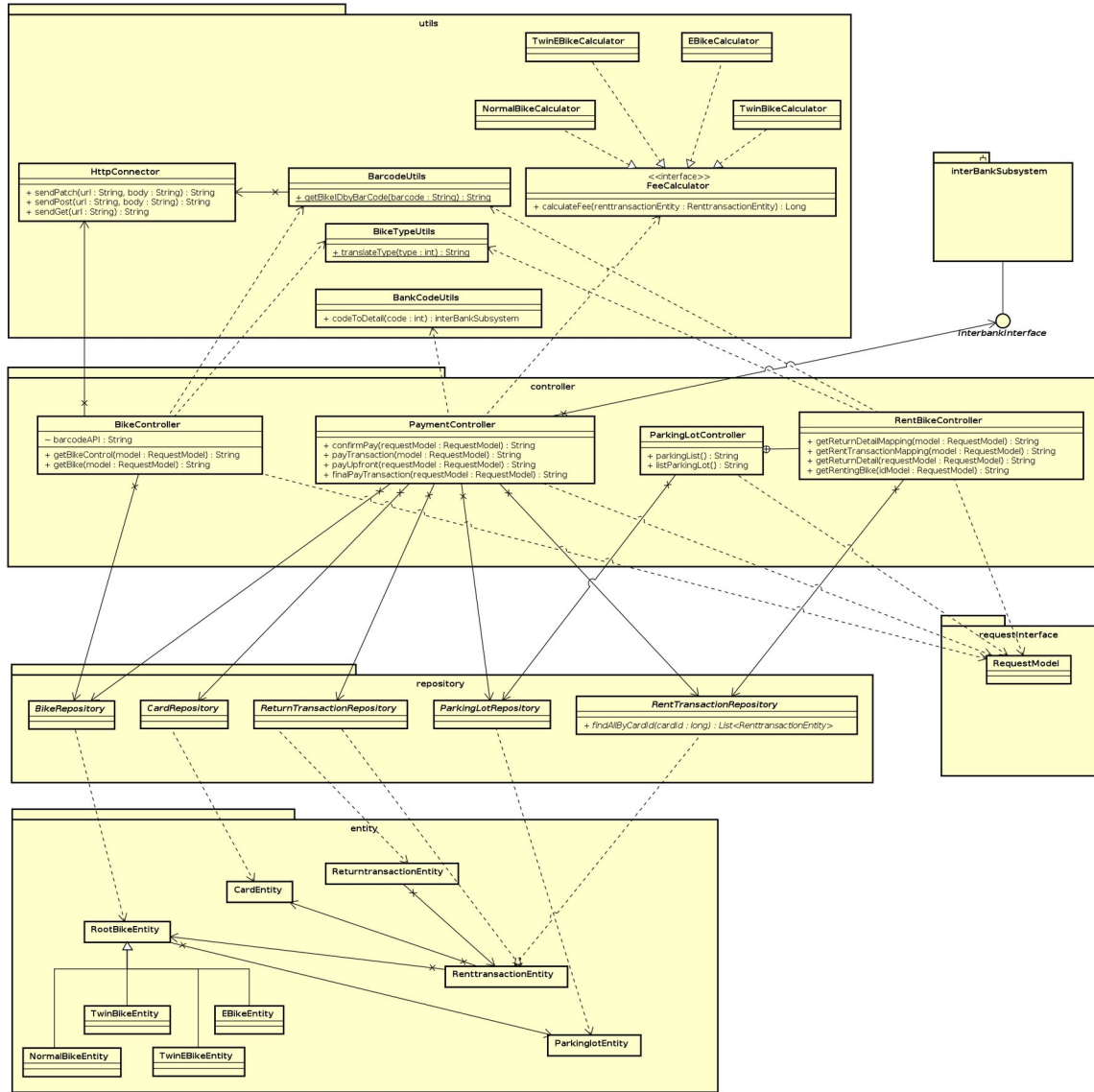
```

values (2, 2, 200, 25, '2018-01-01', '6:00', true);
insert into bike(parkingLotId, type, upfrontPrice, rentPrice, lastUsed, maxTimeUsed,isAvailable)
values (1, 3, 300, 30, '2018-01-01', '6:00', true);
insert into bike(parkingLotId, type, upfrontPrice, rentPrice, lastUsed, maxTimeUsed,isAvailable)
values (3, 3, 300, 30, '2018-01-01', '6:00', true);
insert into bike(parkingLotId, type, upfrontPrice, rentPrice, lastUsed, maxTimeUsed,isAvailable)
values (3, 4, 400, 35, '2018-01-01', '6:00',true);
insert into bike(parkingLotId, type, upfrontPrice, rentPrice, lastUsed, maxTimeUsed,isAvailable)
values (2, 1, 100, 20, '2018-01-01', '6:00',true);
insert into bike(parkingLotId, type, upfrontPrice, rentPrice, lastUsed, maxTimeUsed,isAvailable)
values (1, 4, 400, 35, '2018-01-01', '6:00',true);
insert into bike(parkingLotId, type, upfrontPrice, rentPrice, lastUsed, maxTimeUsed,isAvailable)
values (3, 3, 300, 30, '2018-01-01', '6:00',true);
insert into bike(parkingLotId, type, upfrontPrice, rentPrice, lastUsed, maxTimeUsed,isAvailable)
values (3, 3, 150000, 30000, '2018-01-01', '6:00',true);

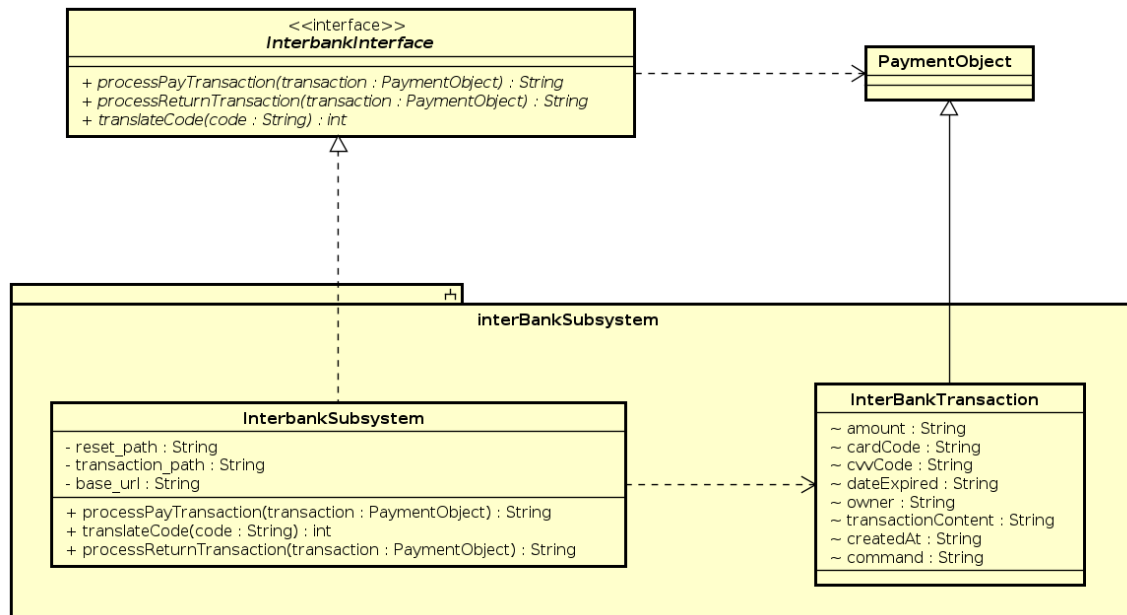
```

3.3 Thiết kế chi tiết lớp

3.3.1 Biểu đồ lớp tổng quan



3.3.2 Biểu đồ lớp cho Interbank subsystem



3.3.3 Thiết kế chi tiết lớp

3.3.3.1 Gói controller

3.3.3.1.1 Lớp BikeController

Mục đích sử dụng

Điều khiển các tác vụ liên quan đến đối tượng xe.

Định nghĩa REST API tương ứng để front-end có thể tương tác đến.

Attribute

#	Tên	Kiểu dữ liệu	Giá trị mặc định	Mô tả
1	bikeRepository	BikeRepository		Đối tượng sử dụng để truy vấn bảng Bike trong cơ sở dữ liệu
2	barcodeAPI	String	https://barcodeservicebykv2.herokuapp.com/barcode	Đường dẫn đến API chuyển mã vạch
3	httpConnector	HttpConnector		gửi request đến API barcode yêu cầu chuyển mã vạch thành mã xe

Operation

1	Tên	getBikeControl	Kiểu dữ liệu trả về	String
	Mô tả	Tạo API để front-end truy vấn thông tin xe: <ul style="list-style-type: none">Method: POSTPath: /getBikeByBarcode		
	Danh sách tham số	Tên	Kiểu dữ liệu	Mô tả
		model	RequestModel	Đóng gói các thông tin cần thiết để yêu cầu thông tin về một xe muốn thuê, bao gồm: <ul style="list-style-type: none">barcode

2	Tên	getBikeByBarcode	Kiểu dữ liệu trả về	String
	Mô tả	Lấy thông tin xe tương ứng với barcode truyền vào		
	Danh sách tham số	Tên	Kiểu dữ liệu	Mô tả
		model	RequestModel	Tương tự getBikeControl

Method

Không

State

Không

3.3.3.1.2 Lớp PaymentController

Mục đích sử dụng

Lớp điều khiển các tác vụ liên quan đến thanh toán hóa đơn thuê và trả xe.

Định nghĩa REST API tương ứng để front-end có thể tương tác đến.

Attribute

#	Tên	Kiểu dữ liệu	Giá trị mặc định	Mô tả
1	parkinglotRepository	ParkingLotRepository		Đối tượng sử dụng để truy vấn bảng ParkingLot trong cơ sở dữ liệu
2	bikeRepository	BikeRepository		Đối tượng sử dụng để truy vấn bảng Bike trong cơ sở dữ liệu
3	rentTransactionRepository	RentTransactionRepository		Đối tượng sử dụng để truy vấn bảng RentTransaction trong cơ sở dữ liệu

4	returnTransactionRepository	ReturnTransactionRepository		Đối tượng sử dụng để truy vấn bảng ReturnTransaction trong cơ sở dữ liệu
5	cardRepository	CardRepository		Đối tượng sử dụng để truy vấn bảng Card trong cơ sở dữ liệu
6	interbankSubsystem	InterbankInterface		Đối tượng sử dụng để giao tiếp với Interbank API

Operation

1	Tên	payUpFrontControl	Kiểu dữ liệu trả về	String
	Mô tả	Định nghĩa API để yêu cầu front-end thanh toán cọc: <ul style="list-style-type: none"> Method: POST Path: /payUpFront 		
	Danh sách tham số	Tên	Kiểu dữ liệu	Mô tả
		model	RequestModel	Đóng gói các trường dữ liệu liên quan đến thông tin thanh toán, bao gồm: <ul style="list-style-type: none"> barcode cardCode cardOwner cvv expireDate
2	Tên	finalPayControl	Kiểu dữ liệu trả về	String
	Mô tả	Định nghĩa API yêu cầu thanh toán trả xe		
	Danh sách tham số	Tên	Kiểu dữ liệu	Mô tả
		model	RequestModel	Đóng gói các trường dữ liệu liên quan đến việc xác nhận người thanh toán (người thuê xe phải đúng là người trả xe), bao gồm: <ul style="list-style-type: none"> barcode cardID

				<ul style="list-style-type: none"> parkinglotID
3	Tên	payUpFront	Kiểu dữ liệu trả về	String
	Mô tả	Phương thức dùng để thanh toán cọc		
	Danh sách tham số	Tên	Kiểu dữ liệu	Mô tả
		model	RequestModel	Tương tự payUpFrontControl
4	Tên	finalPay	Kiểu dữ liệu trả về	String
	Mô tả	Phương thức thanh toán khi người dùng trả xe		
	Danh sách tham số	Tên	Kiểu dữ liệu	Mô tả
		model	RequestModel	Tương tự finalPayControl

Method

Không

State

Không

3.3.3.1.3 Lớp ParkingLotController

Mục đích sử dụng

Lớp điều khiển các tác vụ liên quan đến truy vấn các bãi xe.

Định nghĩa REST API tương ứng để front-end có thể tương tác đến.

Attribute

#	Tên	Kiểu dữ liệu	Giá trị mặc định	Mô tả
1	parkinglotRepository	ParkingLotRepository		Đối tượng sử dụng để truy vấn bảng ParkingLot trong cơ sở dữ liệu

Operation

1	Tên	parkingList	Kiểu dữ liệu trả về	String
	Mô tả	Định nghĩa API để front-end truy vấn thông tin xe:		

		<ul style="list-style-type: none"> • Method: GET • Path: /listParkingLot 		
	Danh sách tham số	<i>Tên</i>	<i>Kiểu dữ liệu</i>	<i>Mô tả</i>
2	Tên	listParkingLot	Kiểu dữ liệu trả về	String
	Mô tả	Lấy thông tin danh sách các bãi xe trong hệ thống		
	Danh sách tham số	<i>Tên</i>	<i>Kiểu dữ liệu</i>	<i>Mô tả</i>

Method

Không

State

Không

3.3.3.1.4 Lớp RentBikeController

Mục đích sử dụng

Lớp điều khiển các tác vụ liên quan đến thông tin thuê và trả xe.

Định nghĩa REST API tương ứng để front-end có thể tương tác đến.

Attribute

#	Tên	Kiểu dữ liệu	Giá trị mặc định	Mô tả
1	rentTransactionRepository	RentTransactionRepository		Đối tượng sử dụng để truy vấn bảng RentTransaction trong cơ sở dữ liệu

Operation

1	Tên	getReturnDetailControl	Kiểu dữ liệu trả về	String
	Mô tả	Định nghĩa API truy vấn thông tin trả xe:		

		<ul style="list-style-type: none">Method: POSTPath: /returnDetail		
	Danh sách tham số	<i>Tên</i>	<i>Kiểu dữ liệu</i>	<i>Mô tả</i>
		model	RequestModel	Đóng gói các trường dữ liệu liên quan đến trả xe, bao gồm: <ul style="list-style-type: none">barcodecardID
2	Tên	getRentTransactionControl	Kiểu dữ liệu trả về	String
	Mô tả	Định nghĩa API truy vấn thông tin xe đang thuê: <ul style="list-style-type: none">Method: POSTPath: /getRentTransaction		
	Danh sách tham số	<i>Tên</i>	<i>Kiểu dữ liệu</i>	<i>Mô tả</i>
		model	RequestModel	Đóng gói các trường dữ liệu liên quan đến yêu cầu thông tin giao dịch thuê xe: <ul style="list-style-type: none">cardID
3	Tên	getReturnDetail	Kiểu dữ liệu trả về	String
	Mô tả	Lấy thông tin dùng để trả xe		
	Danh sách tham số	<i>Tên</i>	<i>Kiểu dữ liệu</i>	<i>Mô tả</i>
		model	RequestModel	Tương tự getReturnDetailControl
4	Tên	getRentTransaction	Kiểu dữ liệu trả về	String
	Mô tả	Lấy thông tin xe đang được thuê		
	Danh	<i>Tên</i>	<i>Kiểu dữ liệu</i>	<i>Mô tả</i>

	sách tham số	model	RequestModel	Tương tự getRentTransactionControl
--	-------------------------	-------	--------------	---------------------------------------

Method

Không

State

Không

3.3.3.2 Gói repository

Sử dụng Java Persistence API để thực hiện các thao tác CRUD đến database bằng cách tạo ra các repository interface:

- BikeRepository
- CardRepository
- ParkingLotRepository
- RentTransactionRepository
- ReturnTransactionRepository

3.3.3.3 Gói entity

3.3.3.3.1 Lớp BikeEntity

Mục đích sử dụng

Định nghĩa một đối tượng Bike, mapping với các trường thông tin tương ứng trong CSDL.

Attribute

#	Tên	Kiểu dữ liệu	Giá trị mặc định	Mô tả
1	bikeld	long		Mã định danh xe
2	parkingLotId	long		Mã bãi xe
3	type	int		Loại xe
4	upfrontPrice	int		Giá cọc của xe
5	rentPrice	int		Giá thuê
6	lastUsed	Timestamp		Lần cuối sử dụng
7	maxTimeUsed	Time		Thời gian sử dụng tối đa (chỉ có với xe điện)

8	isAvailable	byte		Có đang được thuê hay không?
8	parkinglotByParkingLotId	ParkingLotEntity		Bãi xe tương ứng với xe
9	renttransactionsByBikeId	Collection<RentTransactionEntity>		Danh sách các giao dịch thuê xe tương ứng với xe

Operation

1	Tên	getParkingLotByParkingLotId	Kiểu dữ liệu trả về	ParkingLot
	Mô tả	Lấy về bãi xe tương ứng với xe		
	Danh sách tham số	Tên	Kiểu dữ liệu	Mô tả
2	Tên	getRentTransactionsByBikeId	Kiểu dữ liệu trả về	Collection<RentTransactionEntity>
	Mô tả	Lấy về danh sách các giao dịch thuê xe tương ứng với xe		
	Danh sách tham số	Tên	Kiểu dữ liệu	Mô tả

Method

Không

State

Không

3.3.3.3.2 Lớp CardEntity

Mục đích sử dụng

Định nghĩa một đối tượng Card, mapping với các trường thông tin tương ứng trong CSDL.

Attribute

#	Tên	Kiểu dữ liệu	Giá trị	Mô tả
---	-----	--------------	---------	-------

			<i>mặc định</i>	
1	cardId	long		ID định danh thẻ mỗi lần thanh toán
2	cardCode	String		Mã định danh thẻ
3	owner	String		Tên chủ thẻ
4	cvvCode	String		Card Verification Value
5	dateExpired	String		Ngày hết hạn
6	renttransactionsByCardId	Collection<RentTransactionEntity>		Danh sách các giao dịch thuê xe sử dụng thẻ cardID

Operation

1	Tên	getRentTransactionByCardId	Kiểu dữ liệu trả về	Collection<RentTransactionEntity>
	Mô tả	Lấy về danh sách giao dịch thuê xe sử dụng thẻ có cardID tương ứng		
	Danh sách tham số	Tên	Kiểu dữ liệu	Mô tả

Method

Không

State

Không

3.3.3.3.3 Lớp ReturnTransactionEntity

Mục đích sử dụng

Định nghĩa một đối tượng ReturnTransaction, mapping với các trường thông tin tương ứng trong CSDL.

Attribute

#	Tên	Kiểu dữ liệu	Giá trị	Mô tả
---	-----	--------------	---------	-------

			<i>mặc định</i>	
1	rentTransactionId	long		Mã giao dịch thuê xe tương ứng
2	rentTime	Timestamp		Thời điểm bắt đầu thuê xe
3	finishTime	Timestamp		Thời điểm trả xe
4	rentPrice	int		Tổng tiền thuê xe
5	renttransactionsByRentTransactionId	RenttransactionEntity		Đối tượng RentTransactionEntity của giao dịch thuê xe tương ứng.

Operation

1	Tên	getRenttransactionsByRentTransactionId	Kiểu dữ liệu trả về	RentTransactionEntity
	Mô tả	Lấy về đối tượng RentTransactionEntity của giao dịch thuê xe tương ứng		
	Danh sách tham số	<i>Tên</i>	<i>Kiểu dữ liệu</i>	<i>Mô tả</i>

Method

Không

State

Không

3.3.3.3.4 Lớp ParkinglotEntity

Mục đích sử dụng

Định nghĩa một đối tượng ParkingLot, mapping với các trường thông tin tương ứng trong CSDL.

Attribute

#	Tên	Kiểu dữ liệu	Giá trị mặc định	Mô tả
---	-----	--------------	------------------	-------

1	parkingLotId	long		Mã bãi xe
2	name	String		Tên bãi xe
3	address	String		Địa chỉ bãi xe
4	area	int		Diện tích bãi xe
5	stdBikeSlots	int		Số lượng chỗ trống xe đạp đơn
6	eBikeSlots	int		Số lượng chỗ trống xe đạp điện đơn
7	twinStdBikeSlots	int		Số lượng chỗ trống xe đạp đôi
8	twinEBikeSlots	int		Số lượng chỗ trống của xe đạp điện đôi
9	bikesByParkingLotId	Collection<BikeEntity>		Danh sách các xe có trong bãi

Operation

1	Tên	getBikeByParkinglotId	Kiểu dữ liệu trả về	Collection<BikeEntity>
	Mô tả	Lấy về danh sách các xe có trong bãi		
	Danh sách tham số	Tên	Kiểu dữ liệu	Mô tả

Method

Không

State

Không

3.3.3.3.5 Lớp RentTransactionEntity

Mục đích sử dụng

Định nghĩa một đối tượng RentTransaction, mapping với các trường thông tin tương ứng trong CSDL.

Attribute

#	Tên	Kiểu dữ liệu	Giá trị mặc định	Mô tả
---	-----	--------------	------------------	-------

1	rentTransactionId	long		Mã giao dịch thuê xe
2	cardId	long		ID định danh thẻ mỗi lần thanh toán
3	bikeId	long		Mã định danh xe
4	startTime	Timestamp		Thời điểm bắt đầu thuê xe
5	cardByCardId	CardEntity		Thông tin thẻ tương ứng với cardId
6	bikeByBikeId	BikeEntity		Thông tin xe đang thuê tương ứng

Operation

1	Tên	getCardByCardId	Kiểu dữ liệu trả về	CardEntity
	Mô tả	Lấy về thông tin thẻ tương ứng với cardId		
	Danh sách tham số	Tên	Kiểu dữ liệu	Mô tả
2	Tên	getBikeByBikeId	Kiểu dữ liệu trả về	BikeEntity
	Mô tả	Lấy về thông tin xe tương ứng với xe được thuê		
	Danh sách tham số	Tên	Kiểu dữ liệu	Mô tả

Method

Không

State

Không

3.3.3.4 Gói utils

3.3.3.4.1 Lớp HttpConnector

Mục đích sử dụng

Thực hiện HTTP connect, các method POST, GET, PATCH, ...

Attribute

Không

Operation

1	Tên	sendPatch	Kiểu dữ liệu trả về	String
	Mô tả	Gửi một HTTP request dạng PATCH và lấy kết quả trả về		
	Danh sách tham số	Tên	Kiểu dữ liệu	Mô tả
		url	String	Đường dẫn tới máy chủ cần request
		body	String	Nội dung gói tin gửi trong request
2	Tên	sendPost	Kiểu dữ liệu trả về	String
	Mô tả	Gửi một HTTP request dạng POST và lấy kết quả trả về		
	Danh sách tham số	Tên	Kiểu dữ liệu	Mô tả
		url	String	Đường dẫn tới máy chủ cần request
		body	String	Nội dung gói tin gửi trong request

Method

Không

State

Không

3.3.3.4.2 Lớp BikeTypeUtils

Mục đích sử dụng

Mapping từ mã loại xe (dạng int) sang tên loại xe (dạng String) tương ứng.

Attribute

Không

Operation

1	Tên	translateType	Kiểu dữ liệu trả về	String
	Mô tả	Mapping từ mã loại xe (dạng int) sang tên loại xe (dạng String) tương ứng.		
	Danh sách tham số	Tên	Kiểu dữ liệu	Mô tả
		type	int	Mã loại xe: <ul style="list-style-type: none"> 1: Xe đạp đơn

				<ul style="list-style-type: none"> • 2: Xe đạp đôi • 3: Xe đạp điện • 4: Xe đạp điện đôi
--	--	--	--	---

Method

Không

State

Không

3.3.3.4.3 Lớp BarcodeUtils

Mục đích sử dụng

Gọi Barcode API, chuyển mã vạch thành mã xe tương ứng.

Attribute

#	Tên	Kiểu dữ liệu	Giá trị mặc định	Mô tả
1	httpConnector	HttpConnector		Đối tượng sử dụng để thực hiện gửi HTTP request đến Barcode API

Operation

1	Tên	getBikeByBarcode	Kiểu dữ liệu trả về	String
	Mô tả	Lấy về thông tin xe có mã vạch tương ứng		
	Danh sách tham số	Tên	Kiểu dữ liệu	Mô tả
		barcode	String	Mã vạch của xe

Method

Không

State

Không

3.3.3.4.4 Lớp FeeCalculator

Mục đích sử dụng

Tính toán chi phí thuê xe

Attribute

Không

Operation

1	Tên	calculateFee	Kiểu dữ liệu trả về	Long
	Mô tả	Tính toán chi phí thuê xe		
	Danh sách tham số	Tên	Kiểu dữ liệu	Mô tả
		renttransactionEntity	RentTransactionEntity	Giao dịch thuê xe tương ứng

Method

Không

State

Không

3.3.3.5 Gói requestInterface

3.3.3.5.1 Lớp RequestModel

Mục đích sử dụng

Chứa thông tin nhận được từ front-end

Attribute

#	Tên	Kiểu dữ liệu	Giá trị mặc định	Mô tả
1	cardID	String	NULL	ID định danh thẻ mỗi lần thanh toán
2	barcode	String	NULL	Mã vạch của xe
3	parkinglotID	String	NULL	Mã bãi xe
4	cardCode	String	NULL	Mã thẻ thanh toán
5	cardOwner	String	NULL	Tên chủ thẻ
6	cvv	String	NULL	Card Verification Value
7	expireDate	String	NULL	Ngày hết hạn

Operation

Không

Method

Không

State

Không

3.3.3.6 Interbank subsystem

3.3.3.6.1 InterbankInterface

Mục đích sử dụng

Định nghĩa các phương thức để kết nối thanh toán

Attribute

Không

Operation

1	Tên	processPayTransaction	Kiểu dữ liệu trả về	String
	Mô tả	Xử lý giao dịch loại thanh toán		
	Danh sách tham số	Tên	Kiểu dữ liệu	Mô tả
		transaction	Object	Thông tin thanh toán
2	Tên	processReturnTransaction	Kiểu dữ liệu trả về	String
	Mô tả	Xử lý giao dịch loại hoàn tiền		
	Danh sách tham số	Tên	Kiểu dữ liệu	Mô tả
		transaction	Object	Thông tin thanh toán
3	Tên	codeToDetail	Kiểu dữ liệu trả về	String
	Mô tả	Hàm giải mã lỗi tương ứng		
	Danh sách tham số	Tên	Kiểu dữ liệu	Mô tả
		code	String	Mã lỗi trả về

Method

Không

State

Không

3.3.3.6.2 InterbankSystemController

Mục đích sử dụng

Điều khiển các phương thức thanh toán với Interbank

Attribute

#	Tên	Kiểu dữ liệu	Giá trị mặc định	Mô tả
1	transactionPath	String	api/card/ processTransaction	Đường dẫn tới API xử lý giao dịch
2	baseUrl	String	https://ecopark-system- api.herokuapp.com/	Tên miền của API
3	secretKey	String		Mã bí mật của app với API
4	interbankBoundary	InterbankBoundary		Đối tượng sử dụng để giao tiếp với API

Operation

1	Tên	processTransaction	Kiểu dữ liệu trả về	String
	Mô tả	Phương thức thực hiện giao dịch		
	Danh sách tham số	Tên	Kiểu dữ liệu	Mô tả
		transaction	InterbankTransaction	Thông tin giao dịch thanh toán
2	Tên	processReturnTransaction	Kiểu dữ liệu trả về	String
	Mô tả	Phương thức thực hiện giao dịch hoàn tiền		
	Danh sách tham số	Tên	Kiểu dữ liệu	Mô tả
		transaction	Object	Thông tin giao dịch hoàn tiền
3	Tên	processPayTransaction	Kiểu dữ liệu trả về	String
	Mô tả	Phương thức thực hiện giao dịch thanh toán		
	Danh sách tham số	Tên	Kiểu dữ liệu	Mô tả
		transaction	Object	Thông tin giao dịch hoàn tiền

4	Tên	codeToDetail	Kiểu dữ liệu trả về	String
	Mô tả	Hàm chuyển mã lỗi thành message tương ứng		
	Danh sách tham số	Tên	Kiểu dữ liệu	Mô tả
		code	String	Mã lỗi

Method

Không

State

Không

3.3.3.6.3 Lớp InterbankBoundary

Mục đích sử dụng

Giao tiếp với API Interbank, trả về mã lỗi.

Attribute

Không

Operation

1	Tên	processTransaction	Kiểu dữ liệu trả về	String
	Mô tả	Phương thức truy vấn API		
	Danh sách tham số	Tên	Kiểu dữ liệu	Mô tả
		url	String	Địa chỉ của API thanh toán
		body	String	Nội dung của giao dịch thanh toán

Method

Không

State

Không

3.3.3.6.4 Lớp InterbankTransaction

Mục đích sử dụng

Chức các thông tin dùng để thực hiện thanh toán với Interbank

Attribute

#	Tên	Kiểu dữ liệu	Giá trị mặc định	Mô tả
1	amount	String	NULL	Số tiền giao dịch
2	cardCode	String	NULL	Mã định danh thẻ
3	cvvCode	String	NULL	Card Verification Value
4	dateExpired	String	NULL	Ngày hết hạn
5	owner	String	NULL	Tên chủ thẻ
6	transactionContent	String	NULL	Nội dung giao dịch
7	createdAt	String	NULL	Thời điểm tạo
8	command	String	NULL	Loại giao dịch: thanh toán hoặc hoàn tiền

Operation

Không

Method

Không

State

Không

4 Các vấn đề thiết kế

4.1 Giải quyết yêu cầu phát sinh

Nếu có một loại xe mới trong hệ thống, ta chỉ cần tạo ra một lớp con kế thừa từ lớp Bike, đồng thời tạo một lớp mới để tính tiền thuê xe cho loại xe này, implement FeeCalculator interface.

4.2 Coupling and cohesion

4.2.1 Coupling

4.2.1.1 Content coupling

Thiết kế hiện tại không có loại coupling này

4.2.1.2 Common coupling

Thiết kế hiện tại không có loại coupling này.

4.2.1.3 Control coupling

Thiết kế hiện tại sử dụng Strategy pattern trong việc tính tiền thuê xe, điều này giúp tránh được control coupling vì lớp caller là PaymentController không cần truyền tham số điều khiển cho callee là FeeCalculator.

4.2.1.4 Stamp coupling

Thiết kế hiện tại đang có loại coupling này, cụ thể, các lớp controller khi giao tiếp với các hệ thống ngoài sẽ sử dụng kiểu dữ liệu RequestModel. Mỗi controller chỉ sử dụng một phần các thuộc tính của lớp RequestModel. Có thể chuyển dạng coupling này sang Data coupling bằng việc truyền danh sách tham số có kiểu primitive cho các controller. Tuy nhiên, nhược điểm của việc này là danh sách tham số sẽ quá dài, phức tạp, khó quản lý. Do đó, nhóm vẫn giữ nguyên cách thiết kế sử dụng RequestModel.

4.2.1.5 Data coupling

Đa số các thành phần của hệ thống đều ở mức coupling này.

4.2.2 Cohesion

4.2.2.1 Coincidental cohesion

Trong thiết kế hiện tại, gói utils chứa khá nhiều lớp. Các lớp này không có mối quan hệ gì nhiều với nhau ngoại trừ việc chúng đều là các lớp tiện ích cung cấp dịch vụ các thành phần khác trong hệ thống.

4.2.2.2 Logical cohesion

Các lớp trong gói controller không có liên quan về mặt chức năng nhưng được đặt trong cùng một package. Tuy nhiên việc này là chấp nhận được ở mức package.

Trong lớp PaymentController, 2 phương thức *payUpFront* (thanh toán cọc) và *finalPay* (thanh toán thuê xe) không có liên quan đến nhau về mặt chức năng, chúng chỉ được đặt cùng trong một class vì class này phụ trách tất cả nhiệm vụ liên quan đến thanh toán.

4.2.2.3 Temporal cohesion

Hệ thống hiện tại không có loại cohesion này.

4.2.2.4 Procedural cohesion

Hệ thống hiện tại không có loại cohesion này.

4.2.2.5 Communicational cohesion

Hệ thống hiện tại có loại cohesion này, cụ thể: 2 phương thức *processPayTransaction* và *processReturnTransaction* cùng được đặt trong lớp *InterbankSystemController*, cùng nhận vào một giao dịch và trả về kết quả giao dịch.

4.2.2.6 Sequential cohesion

Hệ thống hiện tại không có loại cohesion này.

4.2.2.7 Informational cohesion

Hệ thống hiện tại có loại cohesion này. Cụ thể: các phương thức khác nhau trong lớp *PaymentController* cùng sử dụng thuộc tính *bikeRepository* để truy vấn các thông tin về xe trong cơ sở dữ liệu.

4.2.2.8 Functional cohesion

Hệ thống hiện tại có loại cohesion này. Cụ thể, trong lớp *InterbankSystemController*, các phương thức *translateCode* và *processTransaction* đều phục vụ mục đích thực hiện giao dịch với phía ngân hàng.

4.3 Các nguyên tắc thiết kế

4.3.1 Single Responsibility Principle

Nguyên tắc này nói rằng mỗi một lớp chỉ nên đảm nhận một trách nhiệm duy nhất. Nếu nó đảm nhận nhiều hơn một trách nhiệm, lớp đó nên được thành ra thành một vài lớp khác nhau.

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 lớp, mỗi lớp đảm nhận một

trách nhiệm duy nhất. Bảng dưới đây mô tả trách nhiệm của một số lớp đại diện cho các package sử dụng trong hệ thống.

<i>Class</i>	<i>Package</i>	<i>Responsibility</i>
PaymentController	controller	Lớp điều khiển các tác vụ liên quan đến thanh toán hóa đơn thuê và trả xe. Lớp này sẽ không phụ trách các nghiệp vụ không liên quan đến thanh toán.
BikeRepository	repository	Lớp phụ trách việc thực hiện các truy vấn đến bảng Bike trong cơ sở dữ liệu. Mỗi bảng trong cơ sở dữ liệu được truy vấn bởi một lớp Repository khác nhau
BikeEntity	entity	Lớp này mapping với bảng entity trong cơ sở dữ liệu. Mỗi bảng trong cơ sở dữ liệu được mapping tương ứng đến một lớp Entity.
HttpConnector	utils	Lớp này đảm nhận việc thực hiện các HTTP request đến một server. Tất cả các hàm của lớp đều phục vụ mục đích này, ví dụ: sendPatch, sendPost; đồng thời lớp cũng không đảm nhận chức năng nào khác.

4.3.2 Open/Closed Principle

Open/Closed principle được hiểu là open for extension và closed for modification. Các thay đổi của một lớp, hay một component phải là trong suốt với các lớp, các component phụ thuộc (close for modification). Đồng thời, việc mở rộng mã nguồn không nên làm ảnh hưởng đến các thành phần đã có (open for extension). Nguyên tắc này được thực hiện bằng việc sử dụng các design pattern (template method, strategy, ...). Để thay đổi hay mở rộng các chức năng cung cấp bởi một interface, ta chỉ cần viết một lớp mới implement lại interface này.

Các thành phần của hệ thống thể hiện nguyên tắc Open/Closed

- ◆ Khi tính giá thuê xe, lớp PaymentController chỉ phụ thuộc vào interface FeeCalculator, và các lớp NormalBikeCalculator, EbikeCalculator, TwinBikeCalculator, TwinEBikeCalculator sẽ implement interface FeeCalculator. Điều này đảm bảo lớp PaymentController không phụ thuộc vào một lớp tính tiền thuê xe cụ thể nào, và khi có các cách tính tiền mới, các đoạn mã nguồn đã viết trong PaymentController không cần phải thay đổi, mà chỉ cần thêm implementation cho interface FeeCalculator.

4.3.3 Liskov substitution principle

Nguyên tắc này yêu cầu một lớp con kế thừa từ một lớp cha sẽ có thể sử dụng để thay thế lớp cha trong bất kì tình huống nào. Để đảm bảo nguyên tắc này thì các phương thức ghi đè hoặc các phương thức implement lại một phương thức abstract phải đảm bảo các yêu cầu sau:

- Các tham số của phương thức ghi đè phải có miền giá trị rộng hơn tham số của phương thức bị ghi đè. Điều này đảm bảo một tham số truyền cho lớp cha cũng có thể truyền cho lớp con mà không gây nên lỗi.
- Giá trị trả về của phương thức ghi đè phải giống với hoặc là lớp con của giá trị trả về của phương thức bị ghi đè.

Ví dụ:

Trong thiết kế hiện tại, có các lớp NormalBikeEntity, EbikeEntity, TwinBikeEntity, TwinEBikeEntity là lớp con kế thừa lớp RootBikeEntity. Các lớp này đã đảm bảo tuân theo nguyên tắc Liskov substitution principle.

4.3.4 Interface segregation principle

Nguyên tắc chỉ ra rằng các interface không nên chứa quá nhiều phương thức, vì nếu chúng ta cần 2 implementations của 1 interface với sự khác nhau rất ít (có thể chỉ khác nhau một phương thức duy nhất), ta sẽ phải copy code của toàn bộ những phương thức giống nhau giữa 2 implementations này 2 lần. Điều tương tự xảy ra nếu chúng ta có nhiều implementations của cùng một interface nhưng các implementations này lại không có quá nhiều khác biệt.

Các thành phần của hệ thống thể hiện nguyên tắc Interface segregation:

- ◆ Trong thiết kế của hệ thống hiện tại, toàn bộ gói repository đều là các interface. Các interface này là các Java Persistence API. Chúng định nghĩa các phương thức liên quan đến object-relational-mapping. Một implementation của API này tương ứng với một loại cơ sở dữ liệu quan hệ. Do đó, các implementations khác nhau sẽ không có các phương thức bị lặp lại ở nhiều hơn một implementation. Điều này có nghĩa là các phương thức định nghĩa bởi API đã được phân tách một cách hợp lý và đảm bảo đúng nguyên tắc Interface segregation.
- ◆ Trong thiết kế của hệ thống, interface InterbankInterface có 2 phương thức được định nghĩa là: processTransaction, translateCode. Đây là 2 phương thức mà bất kì ngân hàng nào cũng cần có, tức là interface này đã được phân tách đủ nhỏ và thiết kế đảm bảo tuân theo nguyên tắc Interface segregation.

4.3.5 Dependency Inversion principle

Nguyên tắc này nói rằng các lớp ở tầng trên không nên sử dụng trực tiếp dịch vụ cung cấp bởi các lớp ở tầng dưới mà nên sử dụng thông qua một interface. Việc này giúp phân tách hoạt động giữa các tầng (layer) trong hệ thống, làm giảm coupling giữa các thành phần trong hệ thống. Nếu không tuân theo nguyên tắc này, hệ thống sẽ trở nên khó sửa chữa và mở rộng.

Các thành phần thể hiện nguyên tắc Dependency Inversion trong hệ thống:

- ◆ Trong thiết kế của hệ thống hiện tại, các lớp sử dụng dịch vụ của gói repository sẽ không cần phải thay đổi nếu hệ thống sử dụng SQLServer thay vì MySQL. Như vậy, các lớp phụ trách nghiệp vụ như các lớp trong gói controller không phụ thuộc vào các module cấp thấp như module kết nối cơ sở dữ liệu. Thay vào đó, các lớp của controller sử dụng các interface của gói repository và các module kết nối cơ sở dữ liệu sẽ implement các phương thức truy vấn cơ sở dữ liệu cụ thể. Điều này tuân theo nguyên tắc Dependency Inversion.
- ◆ Khi tính giá thuê xe, lớp PaymentController chỉ phụ thuộc vào interface FeeCalculator, và các lớp NormalBikeCalculator, EbikeCalculator, TwinBikeCalculator, TwinEBikeCalculator sẽ implement interface FeeCalculator. Điều này đảm bảo lớp PaymentController không phụ thuộc vào một lớp tính tiền thuê xe cụ thể nào, và khi có các cách tính tiền mới, các đoạn mã nguồn đã viết trong PaymentController không cần phải thay đổi, mà chỉ cần thêm implementation cho interface FeeCalculator.

4.4 Design Pattern

Hệ thống sử dụng các thư viện và framework và các thành phần này có sử dụng một số design pattern như sau:

- Simple Logging Facade for Java (SLF4J): Thư viện này sử dụng các design pattern như Singleton, Facade, Factory method, cụ thể:
 - ✓ Hàm static LoggerFactory.getLogger(Class<>) trả về một đối tượng log tương ứng với class truyền vào. Hàm này sử dụng singleton design pattern.
 - ✓ Toàn bộ các chức năng cung cấp bởi thư viện được đóng gói và cung cấp cho người dùng một API để có thể dễ dàng sử dụng mà không cần quan tâm đến logic phức tạp bên trong. Đây chính là facade design pattern.
 - ✓ Thư viện này sử dụng một LoggerFactory để tạo ra các logger. Người dùng chỉ cần truyền vào một tham số là tên lớp và mọi công việc khởi tạo phức tạp đằng sau sẽ được factory xử lý. Đây là chính là Factory method design pattern.

- Spring boot: Thư viện này sử dụng Singleton design pattern, cụ thể:
- ✓ Spring boot framework sử dụng annotation `@Autowired` cho các đối tượng repository và các đối tượng được gắn annotation này sẽ chỉ được tạo ra đúng một lần.

Các design pattern do nhóm sử dụng:

- Singleton design pattern: sử dụng cho các interface trong gói repository.
- Strategy design pattern: sử dụng các cách tính tiền khác nhau cho các loại Bike.