HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

School of Information and communications technology

Final report

Eco Bike Rental Software

ITSS Software Development

Group 02

Phạm Lê Danh Chính – 20194492

Hoàng Xuân Bách – 2019

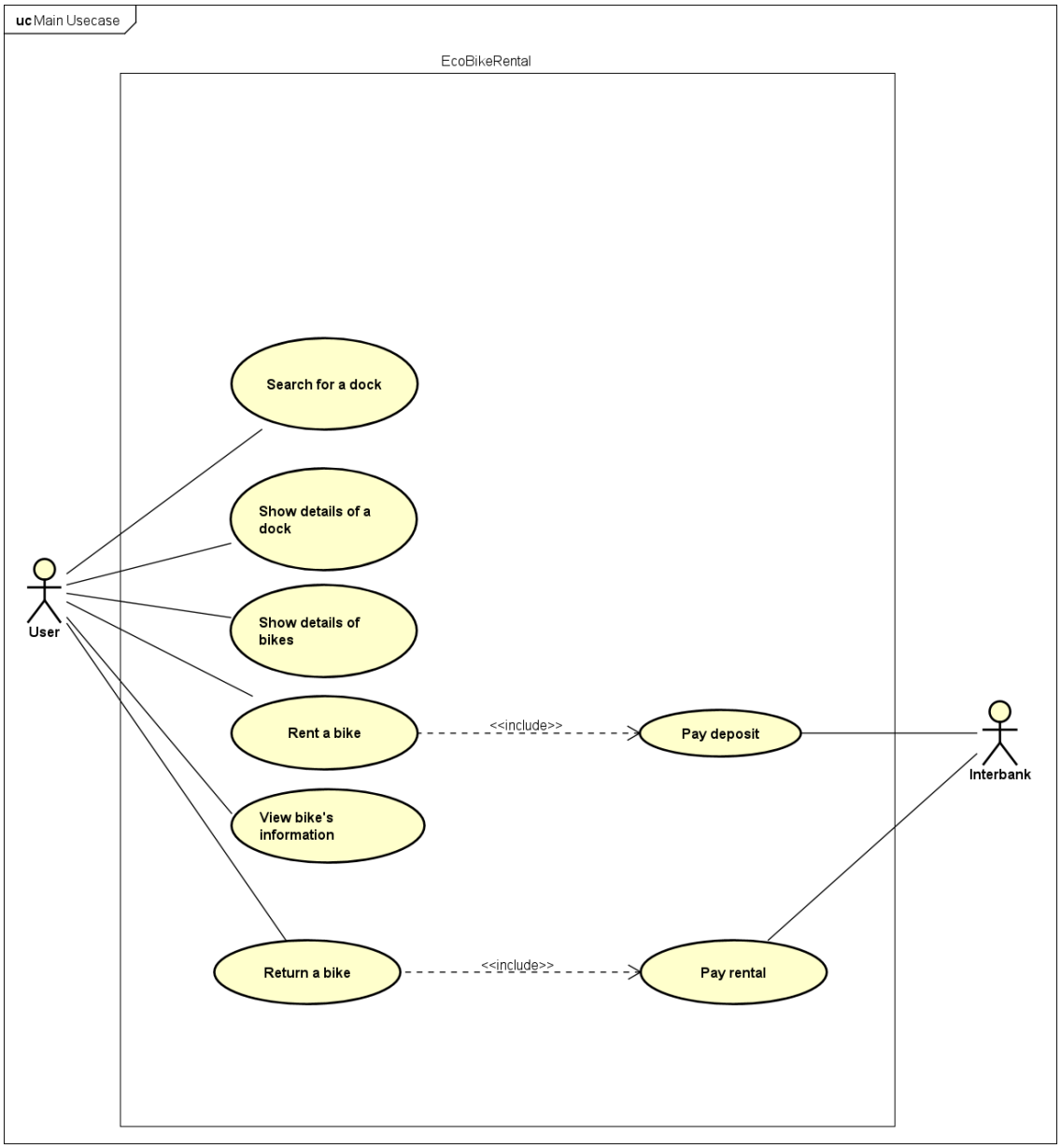
Trịnh Quốc Công - 2019

Hanoi, Feb 2023

Contents

1 Requirement analysis

1.1 Use case diagram



1.2 Use case specifications

1.2.1 Use case "Show details of a dock"

Use Case "Show details of a dock"

1. Use case code

UC001

2. Brief Description

This use case describes the interactions between user and EBR software when user wishes to view the detailed information of chosen dock.

3. Actors

3.1.User

4. Preconditions

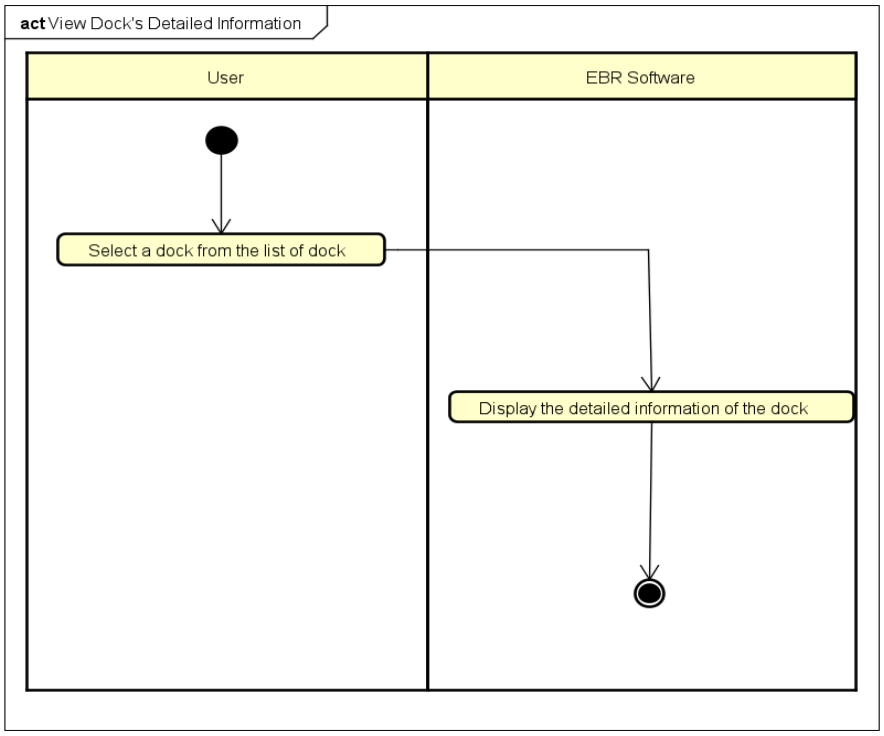
5. Basic Flow of Events

Step 1. The user chooses a dock from the list of docks.

Step 2. The software returns the information of the dock.

6. Alternative flows

7. Activity diagrams



8. Input data

9. Output data

Table 2 - Output data of view dock's detailed information

No	Data fields	Description	Display format	Example

1.	Name	Name of the chosen dock		Dock01
2.	Address	The address of this dock		
3.	Dock Area	The area of this dock	• Positive number • Left aligned	60
4.	Number of Available Bikes	Number of available bikes in this dock	• Positive integer • Left aligned	20
5.	Bike	Available bikes in the selected dock		Standard Bike 01
6.	Number of empty slots	The number of empty docking point	• Positive integer • Left aligned	10

10. Postconditions

1.2.2 Use case "Show details of a bike"

Use Case "View Bike's Detailed Information"

1. Use case code

UC002

2. Brief Description

This use case describes the interactions between user and EBR software when user wishes to view the detailed information of chosen bike.

3. Actors

3.1.User

4. Preconditions

5. Basic Flow of Events

Step 1. The user selects a bike from the list of bikes in dock view

Step 2. The software checks the information of the selected bike

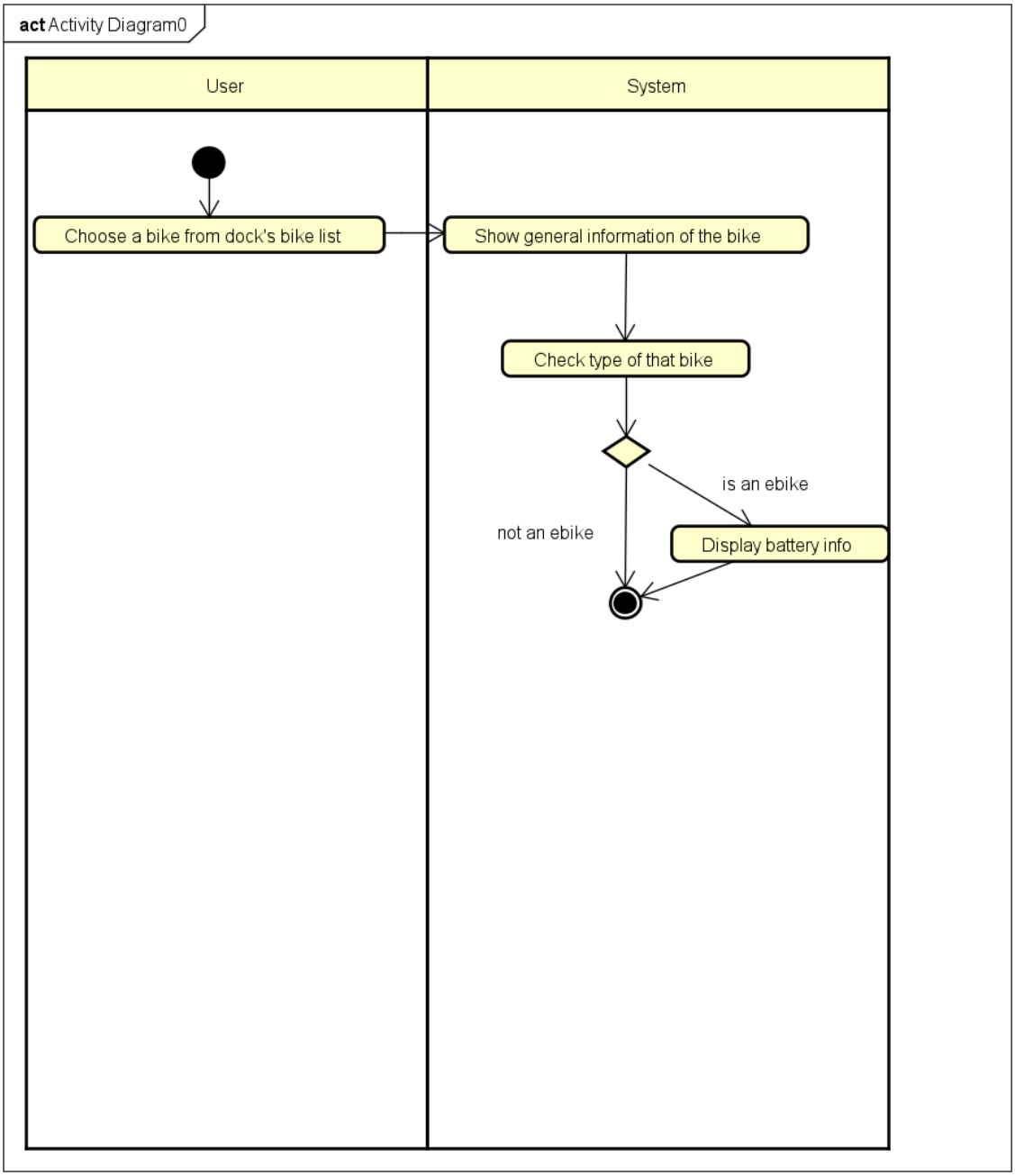
Step 3. The software returns the information of the bike

6. Alternative flows

Table 3 - Alternative flow of events for UC "Name of the Use Case"

No	Location	Condition	Action	Resume location
1	At Step 2	The selected bike is an e-bike	The software returns the percentage of remaining battery life	Resume at Step 3

7. Activity diagrams



8. **Input data**
9. **Output data**

Table 4 - Output data of view bike's detailed information

No	Data fields	Description	Display format	Example
1.	Name	Name of the selected bike		E-bike 01

2.	License plate	License plate of the bike		29G104567
2.	Type	Type of this bicycle		E-bike
3.	Saddle	Number of saddles of this bike	• Positive integer • Left aligned	1
4.	Pedals	Number of pair of pedals	• Positive integer • Left aligned	1
6.	Rear seat	Number of rear seats	• Positive integer • Left aligned	1
7.	Battery	The electric motor's battery percentage	• Positive number with percentage symbol • Left aligned	60%
8.	Time left	How much time is left	In minute	180 minutes

10. Postconditions

1.2.3 Use case "Rent a bike"

Use Case "Rent A Bike"

1. Use case code

UC003

2. Brief Description

This use case describes the interactions between user and EBR software when user wishes to rent a bike

3. **Actors**
   3.1. User
4. **Preconditions**
5. **Basic Flow of Events**

Step 1: User enters the corresponding barcode of the bike that he or she wants to rent from view dock screen

Step 2: The software checks the availability of the bike in the dock

Step 3: The software calculate deposit

Step 4: The software displays the invoice

Step 5: User confirms to rent bike

Step 6: EcoBike call UC PayRental

Step 7: The software saves the rental

Step 8: The AIMS software removes the bike from the dock

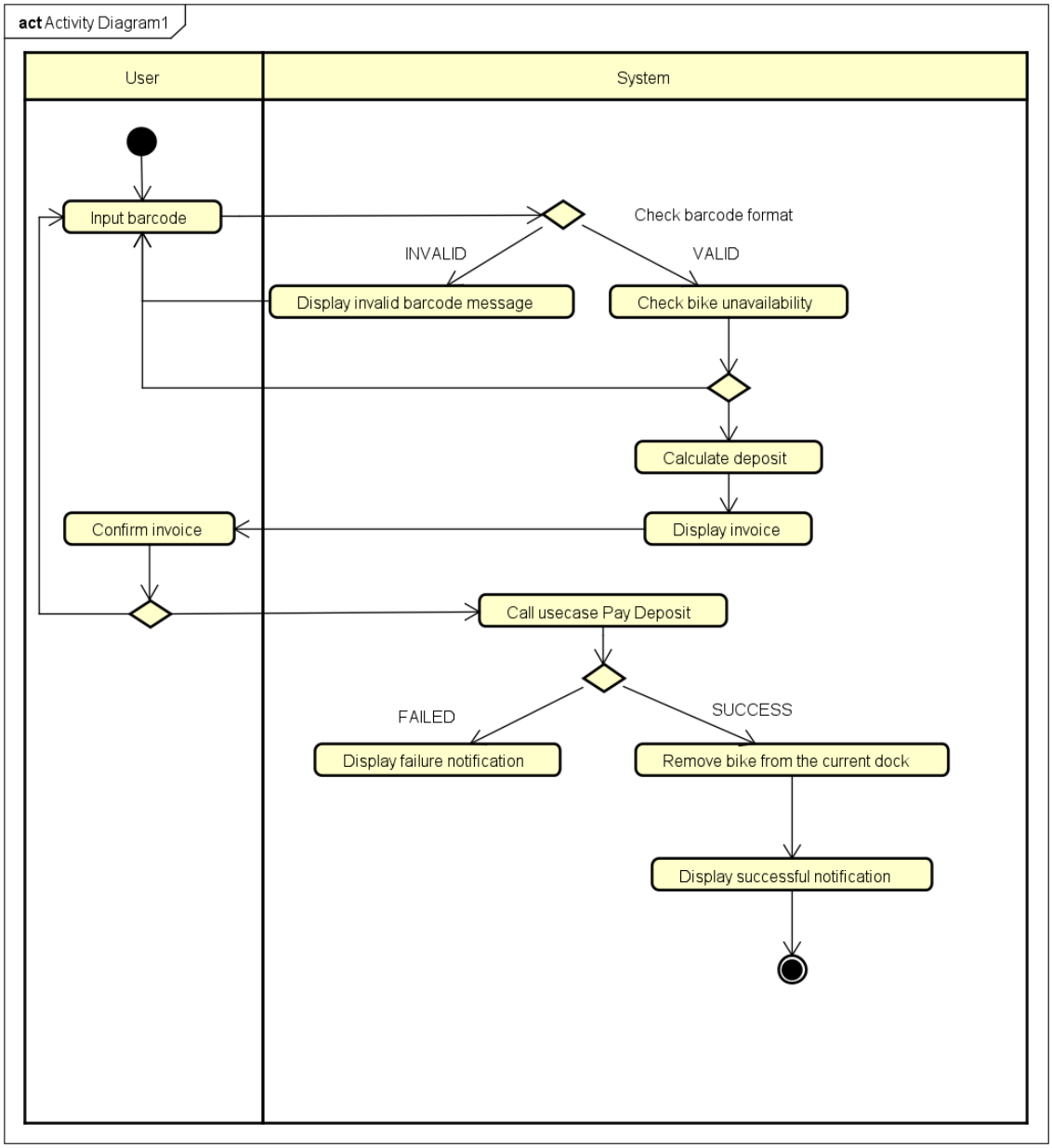
Step 9: The AIMS software displays successful notification.

6. **Alternative flows**

Table 5 - Alternative flow of events for UC "Rent A Bike"

No	Location	Condition	Action	Resume location
1	Step 2	The barcode is in incorrect format	The software notifies that the input barcode is incorrect	Step 1
2	Step 3	The bike is not available in the dock	The software notifies that the bike is not available	Step 1

7. Activity diagrams



8. Input data

Table 6 - Input data of bike rental

No	Data fields	Description	Mandatory	Valid condition	Example

1	Barcode	Bike's barcode	Yes	UUID	123e4567-e89b-12d3-a456-426614174000

9. Output data

10. Postconditions

1.2.4 Use case "Return bike"

Use Case "Return a bike"

1. Use case code

UC004

2. Brief Description

This use case describes the interactions between User and EBR software when User wishes to return a bike.

3. Actors

3.1. User

4. Preconditions

User already rented a bike

5. Basic Flow of Events

Step 1. User requests to return a bike

Step 2. The software requests user to choose a dock to return bike

Step 3. User pick a dock in the list, enter barcode of a lock

Step 4. The software check whether the lock is in "RELEASED" state

Step 5. The software attaches the bike to the lock

Step 6. The software display the rental invoice

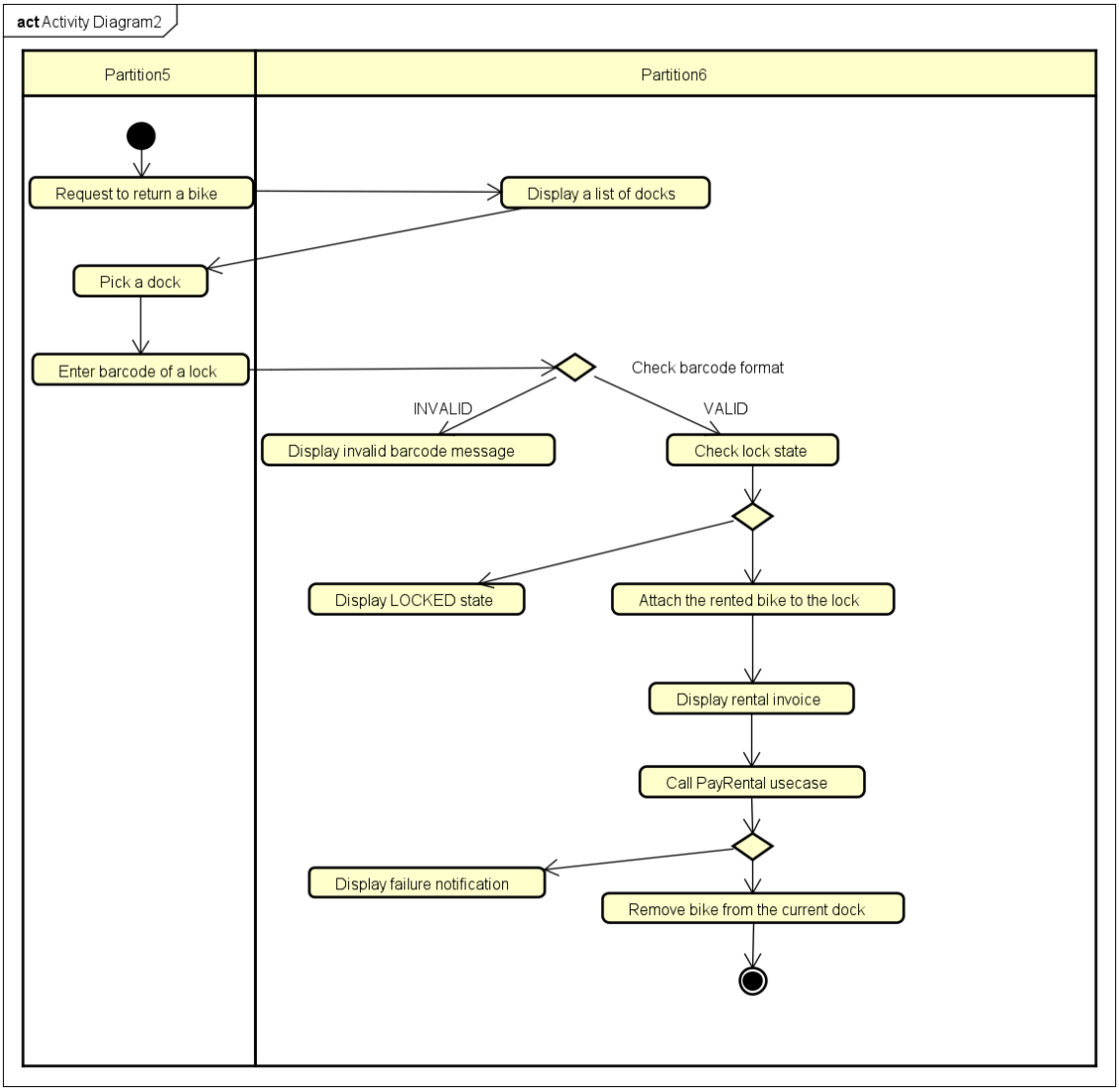
Step 7. The software calls Pay Rental use case

6. Alternative flows

Table 10 - Alternative flow of events for UC "Name of the Use Case"

No	Location	Condition	Action	Resume location
1	At step 4	The lock is not released	Notify lock state	Step 1
2	At step 7	Transaction fail	Notify transaction fail	End of the use case

7. Activity diagrams



8. Input data

Table 11 - Input data of "Return a Bike"

No	Data fields	Description	Mandatory	Valid condition	Example
1	Barcode	Bike's barcode	Yes	String	1231abc212

9. Output data

Table 12 - Output data of "Return a Bike"

No	Data fields	Description	Display format	Example
1	End time	Ending time of the User's renting session	hh : mm DD/MM/YY	12:20 10/10/2020
2	Usage time		. A number of minutes usage	120
3	Renting fee		. Comma for thousands separator . Positive integer . Left aligned	123,000
4	Refund deposit	deposit after deducting rental fee	. Comma for thousands separator . Positive integer . Left aligned	100,000

10. Postconditions

1.2.5 Use case "Pay rental"

Use Case "Pay rental"

1. Use case code

UC005

2. Brief Description

This use case describes the interactions between EBR software and Interbank when EBR software wishes to refund deposit to User.

3. Actors

3.1. User

3.2. Interbank

3.3. System

4. Preconditions

5. Basic Flow of Events

Step 1. The software display payment form

Step 2. User enter credit card information

Step 3. The software validate and verify the card information

Step 4. Interbank proceeds transaction

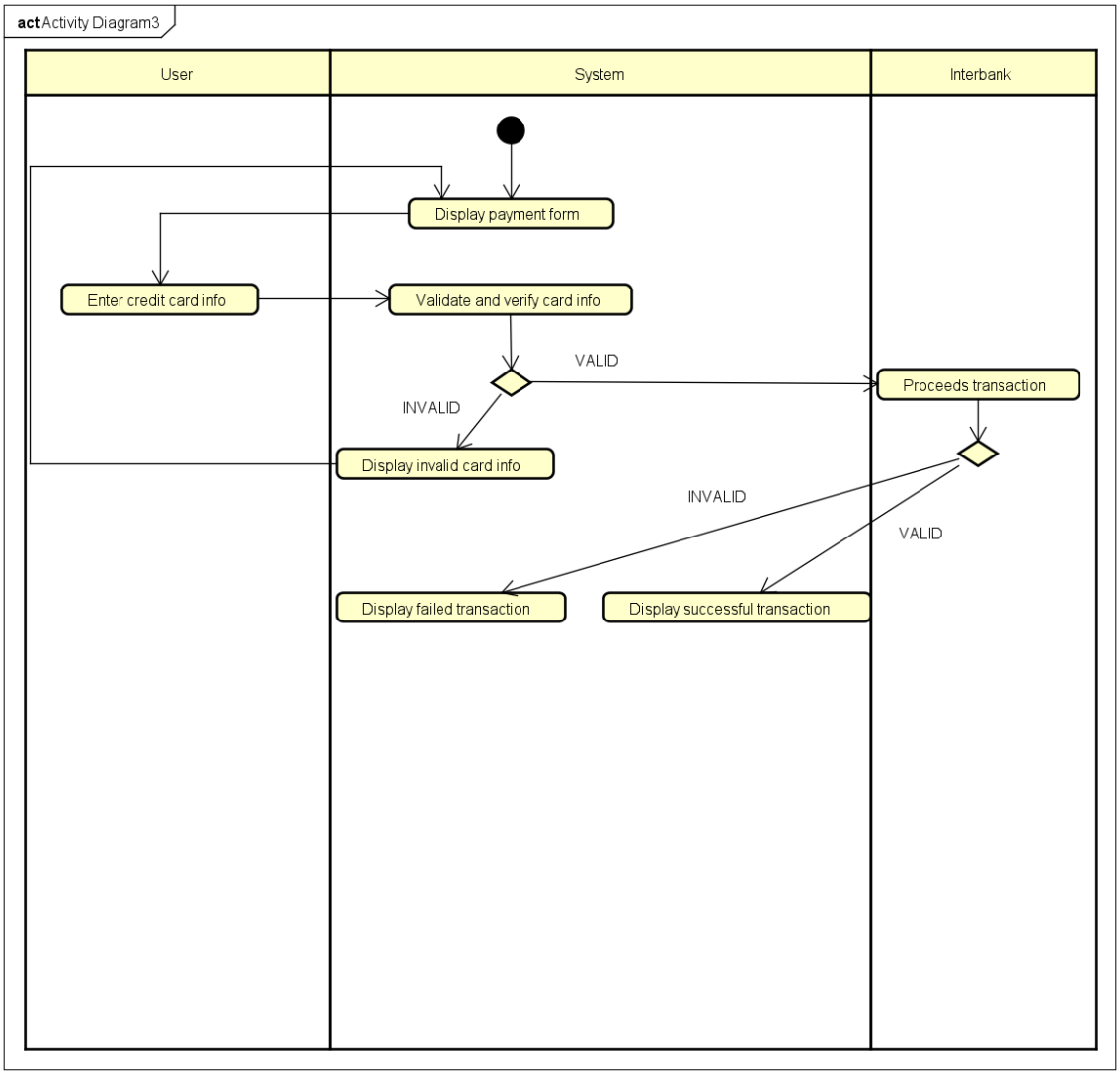
Step 5. Sends the transaction result

6. Alternative flows

Table 13 - Alternative flow of events for UC "Refund Deposit"

No	Location	Condition	Action	Resume location
1	At step 3	Invalid card format	Notify invalid format	step 1
2	At Step 3	Invalid card info	Notify invalid card info	step 1
3	At Step 4	Not enough balance	Notify not enough balance	step 1

7. Activity diagrams



8. Input data

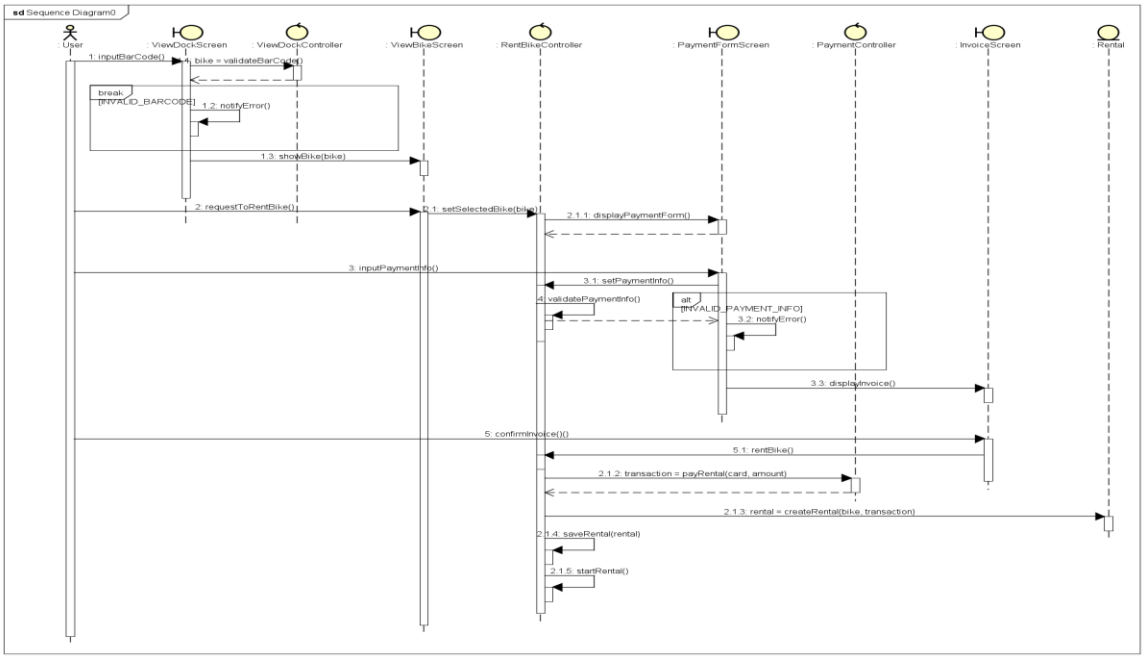
No	Data fields	Description	Mandatory	Valid condition	Example
1	Card owner		yes		PHAM LE DANH CHINH
2	Card code		yes	16-digit	1234 5678 9012 3456
3	Expiry date		yes	mm/yyyy	12/25
3	Cvv code		yes		231413245

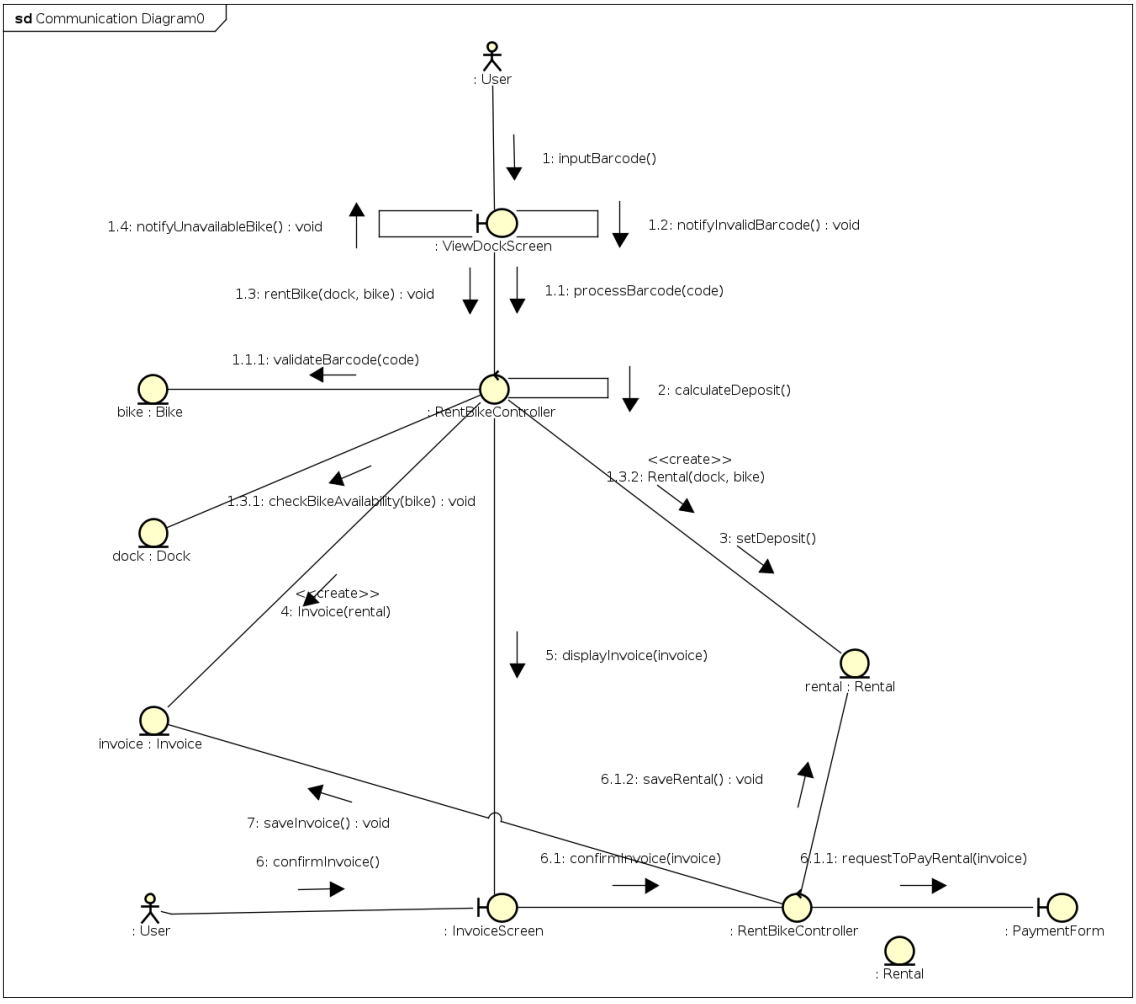
9. **Output data**

10. Postconditions

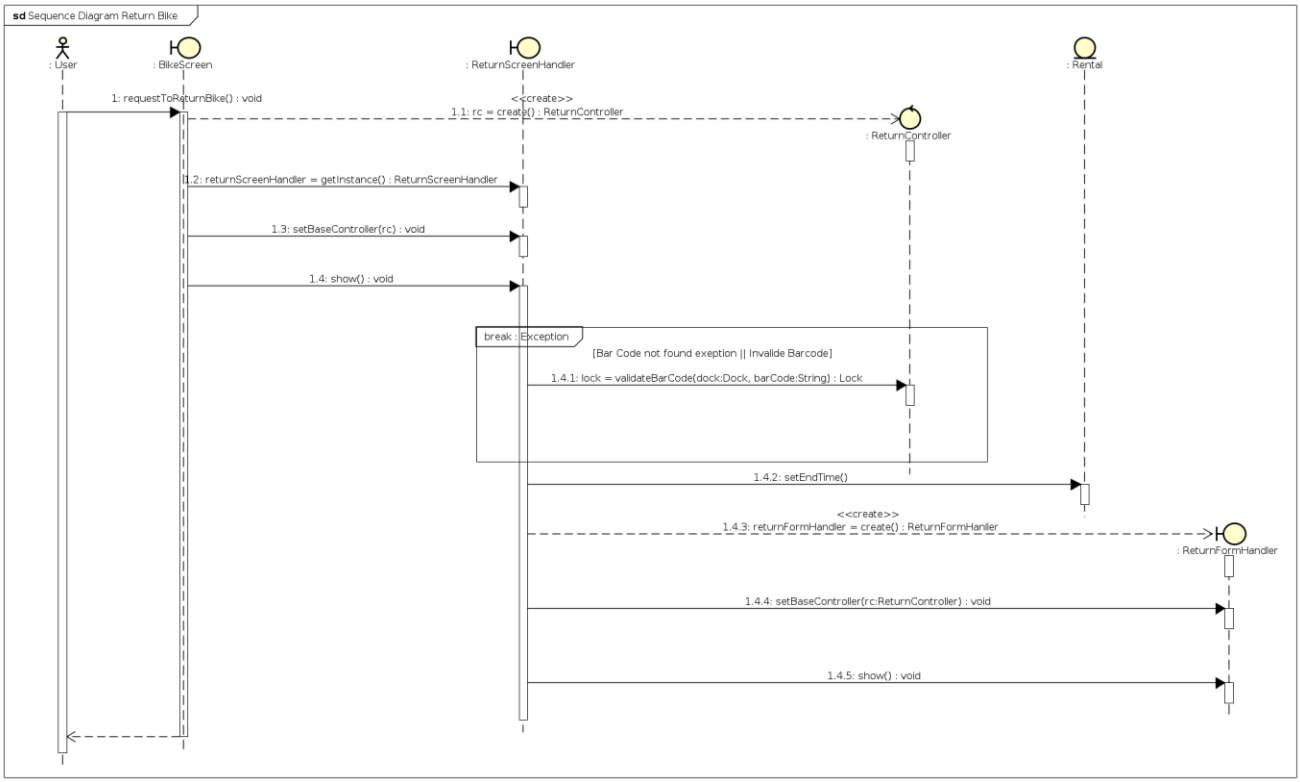
2 Interaction diagram

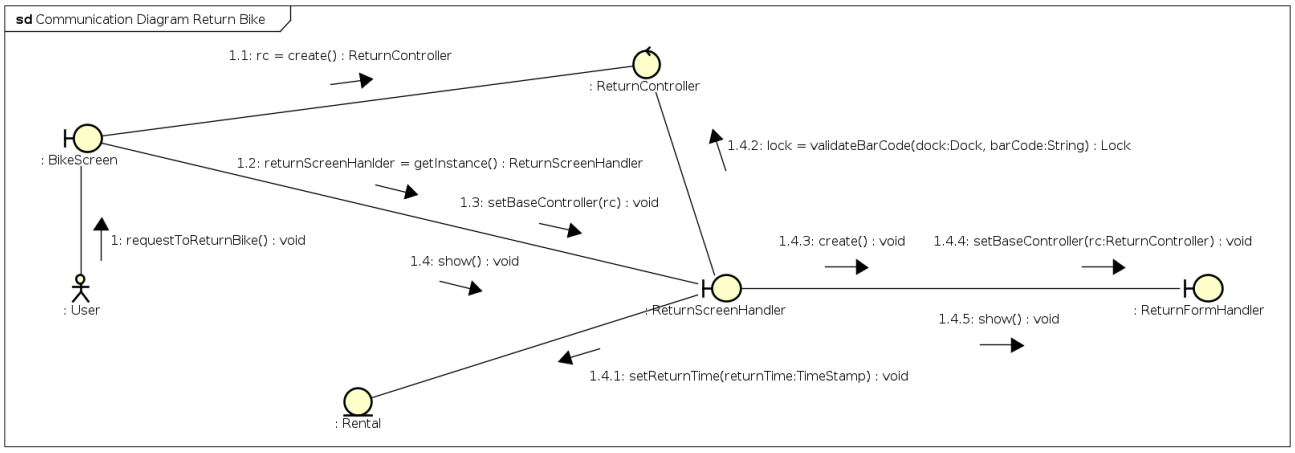
2.1 Rent bike



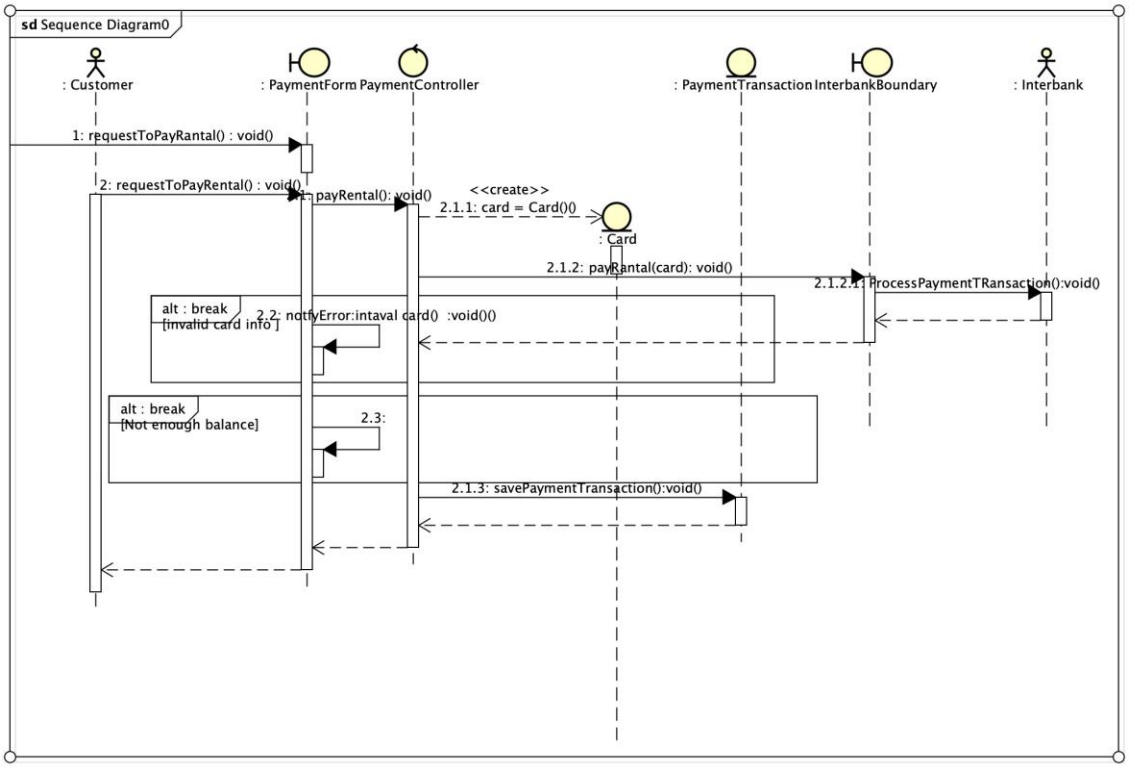

sd Communication Diagram0
: User
1: inputBarcode()
1.4: notifyUnavailableBike() : void
: ViewDockScreen
1.2: notifyInvalidBarcode() : void
1.3: rentBike(dock, bike) : void
1.1: processBarcode(code)
1.1.1: validateBarcode(code)
bike : Bike
: RentBikeController
2: calculateDeposit()
<<create>>
1.3.2: Rental(dock, bike)
1.3.1: checkBikeAvailability(bike) : void
dock : Dock
3: setDeposit()
<<create>>
4: Invoice(rental)
5: displayInvoice(invoice)
rental : Rental
invoice : Invoice
6.1.2: saveRental() : void
7: saveInvoice() : void
6: confirmInvoice()
6.1: confirmInvoice(invoice)
6.1.1: requestToPayRental(invoice)
: User
: InvoiceScreen
: RentBikeController
: PaymentForm
: Rental

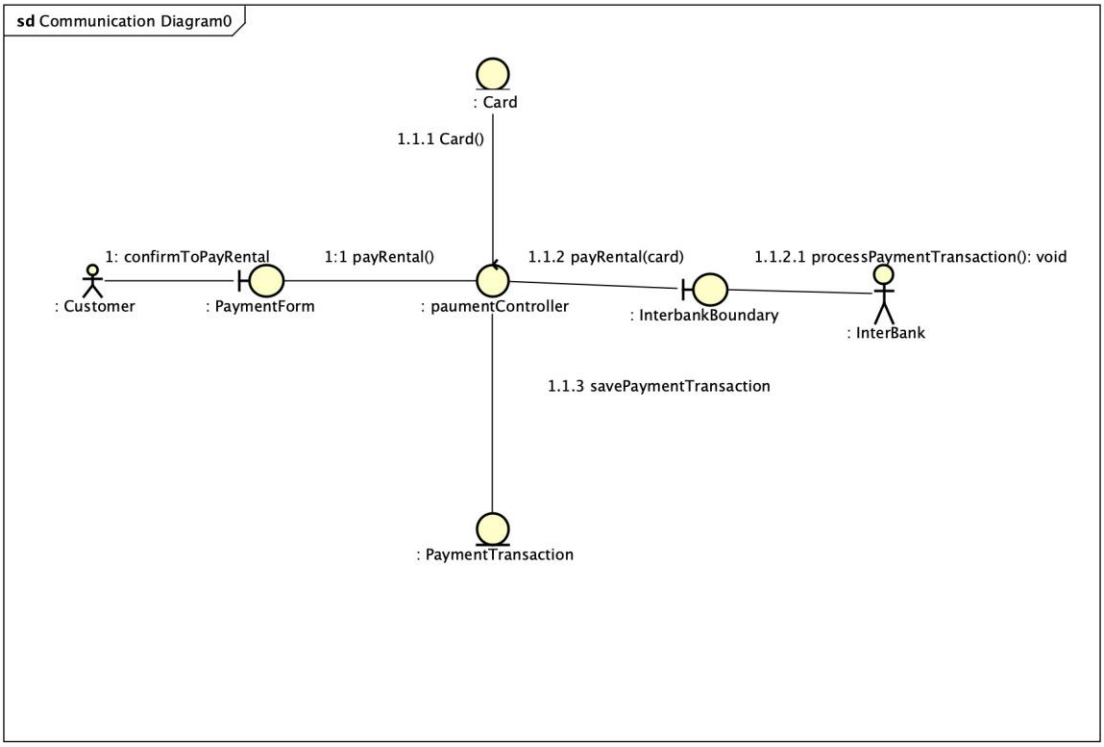
2.2 Return bike





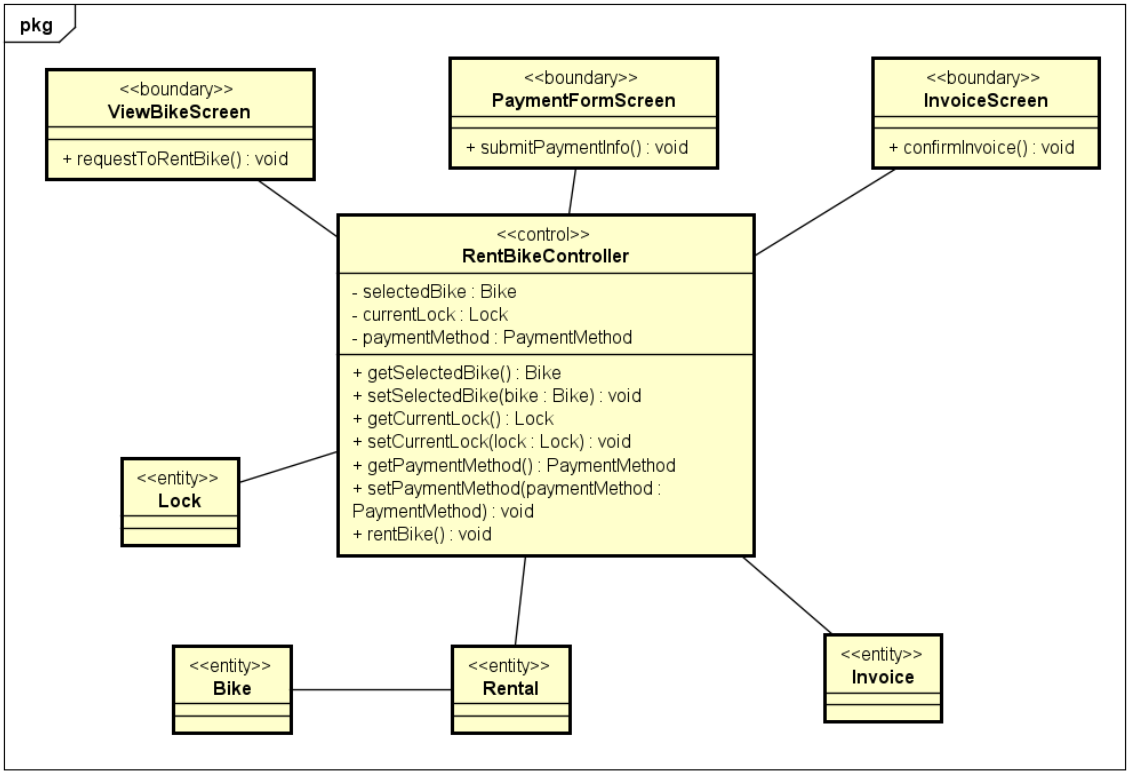
2.3 Pay rental



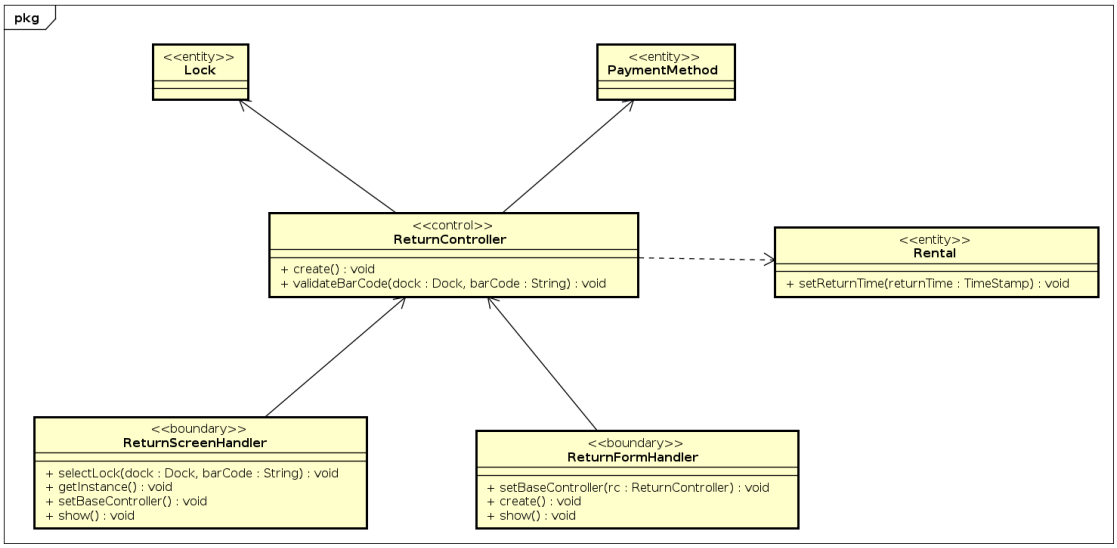

sd Communication Diagram0
: Card
1.1.1 Card()
1: confirmToPayRental
1:1 payRental()
1.1.2 payRental(card)
1.1.2.1 processPaymentTransaction(): void
: Customer
: PaymentForm
: paumentController
: InterbankBoundary
: InterBank
1.1.3 savePaymentTransaction
: PaymentTransaction

3 Analysis Class Diagrams

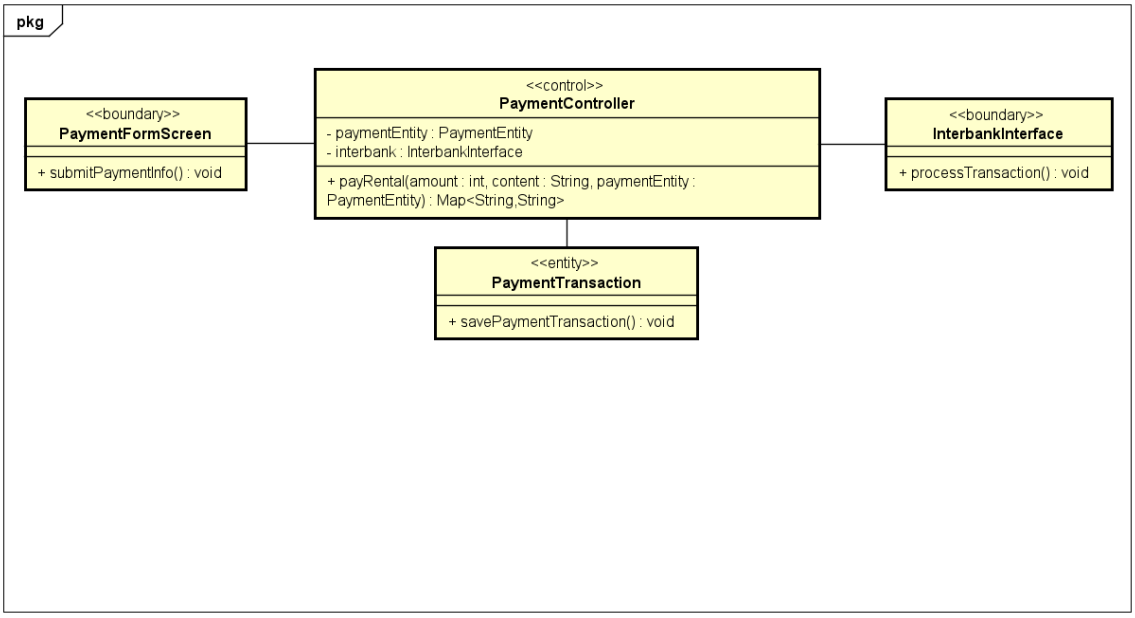
3.1 Use case Rent bike



3.2 Use case Return bike



3.3 Use case Pay Rental



4 Detailed Design

4.1 User interface design

4.1.1 Screen configuration standardized.

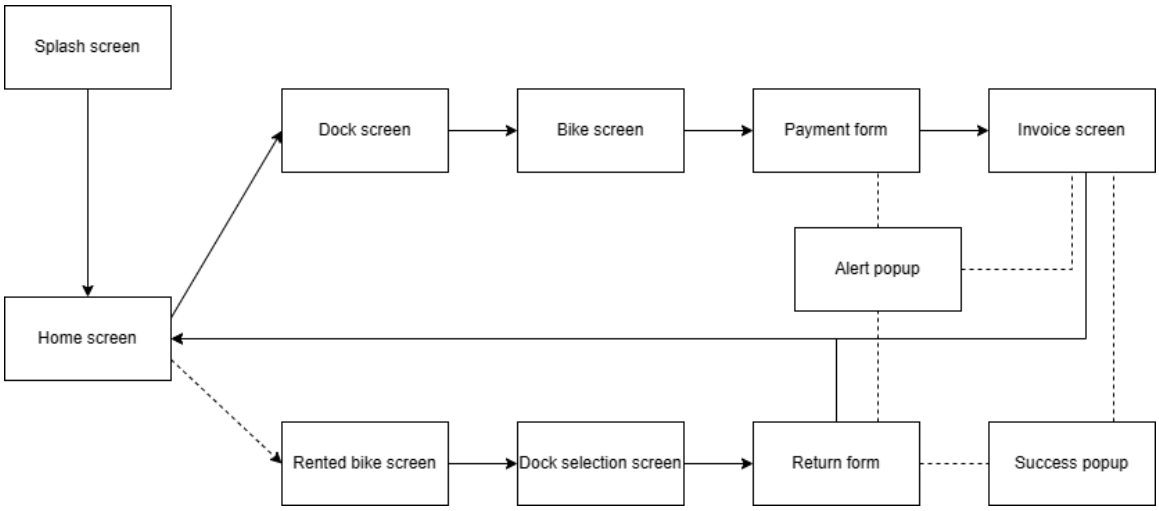
Display:

- Resolution: $1366 \times 768\ pixel$

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 center-top of the screen and on the top-left of the window bar.

4.1.2 Screen Transition Diagrams

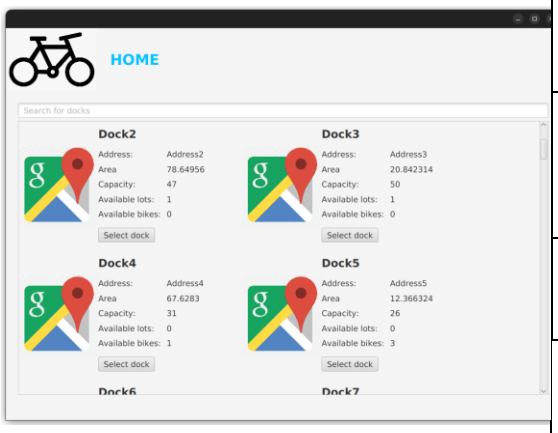


4.1.3 Screen Specifications

4.1.3.1 Home screen

Capstone Software		Date of creation	Approved by	Reviewed by	Person in charge
Screen specification	Home screen	03/01/2023			Phạm Lê

					Danh Chính



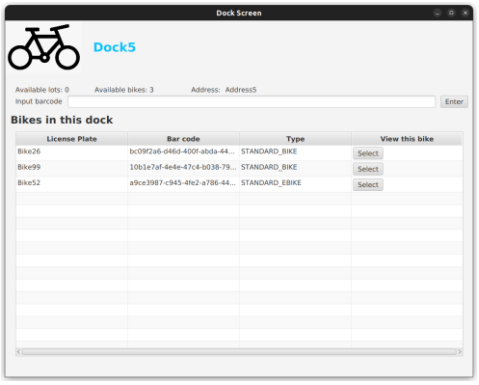
Control	Operation	Function
Area for displaying all docks	initial	Display all docks
Address	Initial	Address of the dock
Available bikes	Initial	Number of available bikes in the dock
Available lots	Initial	Number of available locks not occupied in the dock
Search bar	Text input	Filter dock by name and address
Select dock	Click	Display details of a dock

Defining the field attributes

Screen name	Home screen			
Item name	Number of digits(bytes)	Type	Field attribute	Remarks
Dock name	100	String	Black	Left-justified
Available bikes	20	Integer	Black	Left-justified
Available lots	20	Integer	Black	Left-justified
Address	100	String	Black	Left-justified
Capacity	20	Integer	Black	Left-justified
Area	20	Double	Black	Left-justified

4.1.3.2 View dock screen

Capstone Software		Date of creation	Approved by	Reviewed by	Person in charge
Screen specification	View dock screen	03/01/2023			Phạm Lê Danh Chính



Control	Operation	Function
Address	Initial	Address of the dock
Available bikes	Initial	Number of available bikes in the dock
Available lots	Initial	Number of available locks not occupied in the dock
Barcode input field	Text input	Input field for barcode
Barcode submit button	Click	Submit input barcode
Table of available bikes	Initial	Available bikes in the current dock
View bike button	Click	View bike details

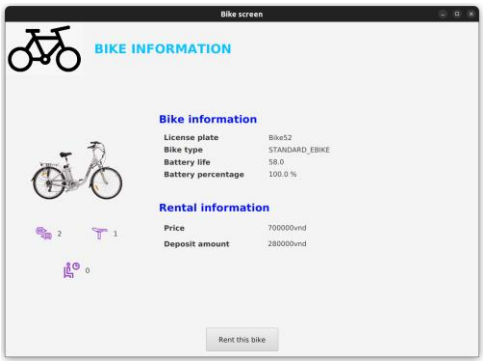
Defining the field attributes

Screen name	View dock screen			
Item name	Number of digits(bytes)	Type	Field attribute	Remarks
Dock name	100	String	Black	Left-justified
Available bikes	20	Integer	Black	Left-justified
Available lots	20	Integer	Black	Left-justified

Address	100	String	Black	Left-justified
Capacity	20	Integer	Black	Left-justified
Area	20	Double	Black	Left-justified
Barcode	16	String	Black	Left-justified

4.1.3.3 View bike screen

Capstone Software		Date of creation	Approved by	Reviewed by	Person in charge
Screen specification	View bike screen	03/01/2023			Phạm Lê Danh Chính



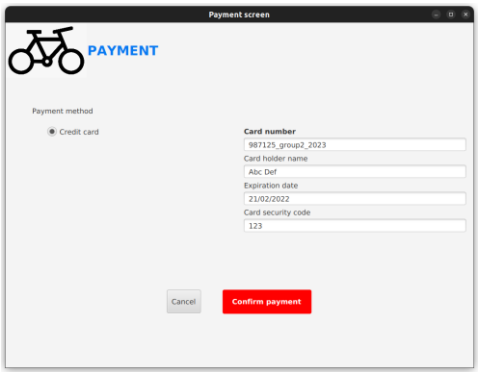
Control	Operation	Function
Bike image	Initial	
License plate	Initial	Identity license number of the bike
Bike type	Initial	Type of the bike
Battery life	Initial	For Standard EBike only
Battery percentage	Initial	For Standard EBike only
Price	Initial	Price of the bike
Deposit amount	Initial	Amount of deposit derived from price
Rent bike button	Click	Start rental process

Defining the field attributes

Screen name	View bike screen	

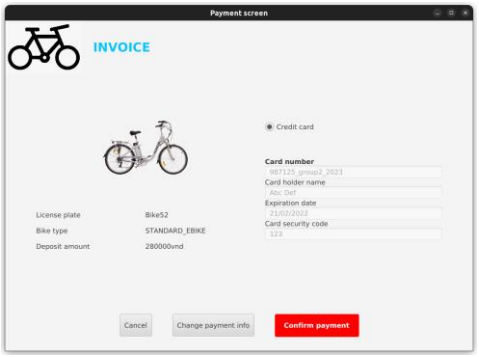
Item name	Number of digits(bytes)	Type	Field attribute	Remarks
License plate	20	String	Black	Left-justified
Bike type	20	String	Black	Left-justified
Battery life	4	Float	Black	Left-justified
Battery percentage	4	Float	Black	Left-justified
Price	4	Integer	Black	Left-justified
Deposit amount	4	Integer	Black	Left-justified

4.1.3.4 Payment form

<table>
<tr><td colspan="2">Capstone Software</td><td>Date of creation</td><td>Approved by</td><td>Reviewed by</td><td>Person in charge</td></tr>
<tr><td>Screen specification</td><td>Payment form</td><td>03/01/2023</td><td></td><td></td><td>Phạm Lê Danh Chính</td></tr>
<tr><td colspan="6" rowspan="8"> 
</td></tr>
<tr><td>Control</td><td>Operation</td><td colspan="2">Function</td></tr>
<tr><td>Payment method selection button</td><td>Click</td><td colspan="2">Choose one of available payment methods</td></tr>
<tr><td>Card number</td><td>Text input</td><td colspan="2">Card code</td></tr>
<tr><td>Card owner</td><td>Text input</td><td colspan="2">Card owner</td></tr>
<tr><td>Expiry date</td><td>Text input</td><td colspan="2">Card expiry date</td></tr>
<tr><td>Cvv code</td><td>Text input</td><td colspan="2">Card cvv code</td></tr>
<tr><td>Cancel button</td><td>Click</td><td colspan="2">Cancel rental and return to previous bike screen</td></tr>
<tr><td>Confirm button</td><td>Click</td><td colspan="2">Submit payment info</td></tr>
</table>

4.1.3.5 Deposit invoice screen

Capstone Software		Date of creation	Approved by	Reviewed by	Person in charge
Screen specification	Deposit invoice screen	03/01/2023			Phạm Lê Danh Chính



Control	Operation	Function
License plate	Initial	Identity license number of the bike
Bike type	Initial	Type of the bike
Deposit amount	Initial	Amount of deposit derived from price
Card number	Text input	Card code
Card owner	Text input	Card owner
Expiry date	Text input	Card expiry date
Cvv code	Text input	Card cvv code
Cancel button	Click	Cancel rental and return to previous bike screen
Change payment info	Click	Return to previous screen and change payment info
Confirm button	Click	Confirm and start rental

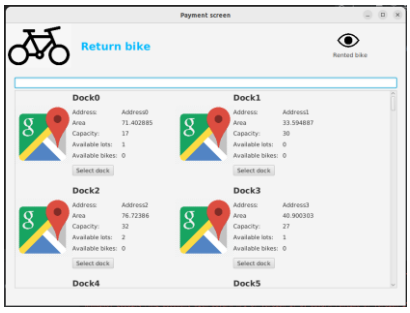
Defining the field attributes

Screen name	Deposit invoice screen	

Item name	Number of digits(bytes)	Type	Field attribute	Remarks
License plate	20	String	Black	Left-justified
Bike type	20	String	Black	Left-justified
Battery life	4	Float	Black	Left-justified
Battery percentage	4	Float	Black	Left-justified
Price	4	Integer	Black	Left-justified
Card number	20	String	Black	Left-justified
Card owner	20	String	Black	Left-justified
Expiry date	10	Date format (dd/mm/yyyy)	Black	Left-justified
Cvv code	3	Integer	Black	Left-justified

4.1.3.6 Selecting dock to return screen

Screen specification – select dock return

Capstone Software		Date of creation	Approved by	Reviewed by	Person in charge
Screen specification	select dock return	03/01/2023			Hoàng Xuân Bách
		Control	Operation		
		Khu vực hiển thị tên bãi xe	Initial		
		Available bikes	Initial	Hiển thị số lượng xe hiện có của bãi xe	
		Select dock	Click	Chọn bãi xe muốn gửi	

	Available lots	Initial	Khu vực hiển thị số lượng ô trống
	Address	Initial	Khu vực hiển thị địa chi bãi xe
	Area	Initial	Khu vực hiển thị diện tích bãi xe
	Capacity	Initial	Khu vực hiển thị sức chứa của bãi
	Avatar app	Click	Nút quay về trang mượn xe
	Rented bike	Click	Nút quay về trang home
	Nhập barcode của locks muốn gửi	Typing	

Defining the field attributes

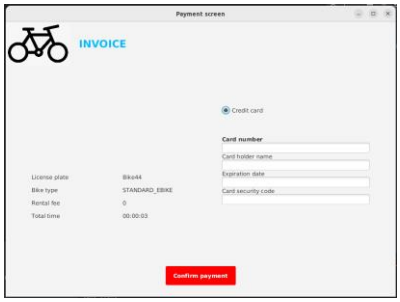
Screen name	Select dock to return			
Item name	Number of digits(bytes)	Type	Field attribute	Remarks
Dock name	100	String	Blue	Left-justified
Available bikes	20	Integer	Blue	Left-justified
Available lots	20	Integer	Blue	Left-justified

Address	100	String	Blue	Left-justified
Capacity	20	Integer	Blue	Left-justified
Area	20	Double	Blue	Left-justified

4.1.3.7 Return invoice screen

Screen specification - return invoice screen

Capstone Software		Date of creation	Approved by	Reviewed by	Person in charge
Screen specification	return invoice screen	03/01/2023			Hoàng Xuân Bách



Control	Operation	
License plate	Initial	Hiển thị biển số xe mượn
Bike type	Initial	Hiển thị loại xe
Rental fee	Initial	Hiển thị phí gửi xe
Retal time	Initial	Hiển thị thời gian mượn xe
Card number	Typing	Nhập số thẻ thanh toán
Card holder name	Typing	Nhập tên chủ thẻ
Expiration date	Typing	Nhập ngày thẻ hết hạn

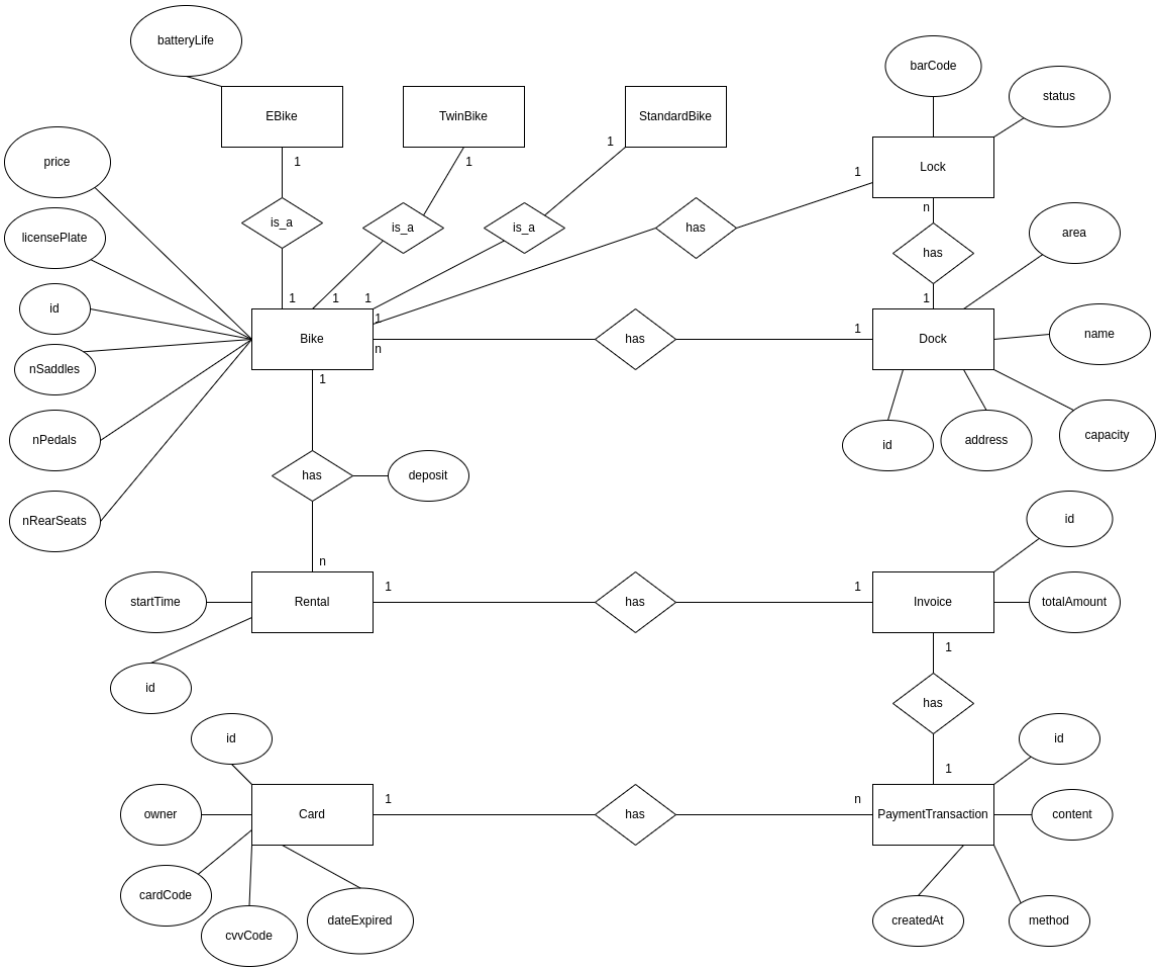
	Security code	Typing	Nhập mã bảo mật thẻ
	Credit card	Click	Chọn loại thẻ thanh toán

Defining the field attributes

Screen name	Return invoice screen			
Item name	Number of digits(bytes)	Type	Field attribute	Remarks
Bike type	20	String	Blue	Left-justified
Rental fee	20	String	Blue	Left-justified
Retal time	20	String	Blue	Left-justified
License plate	20	String	Blue	Left-justified

4.2 Data modeling

4.2.1 Conceptual data modeling

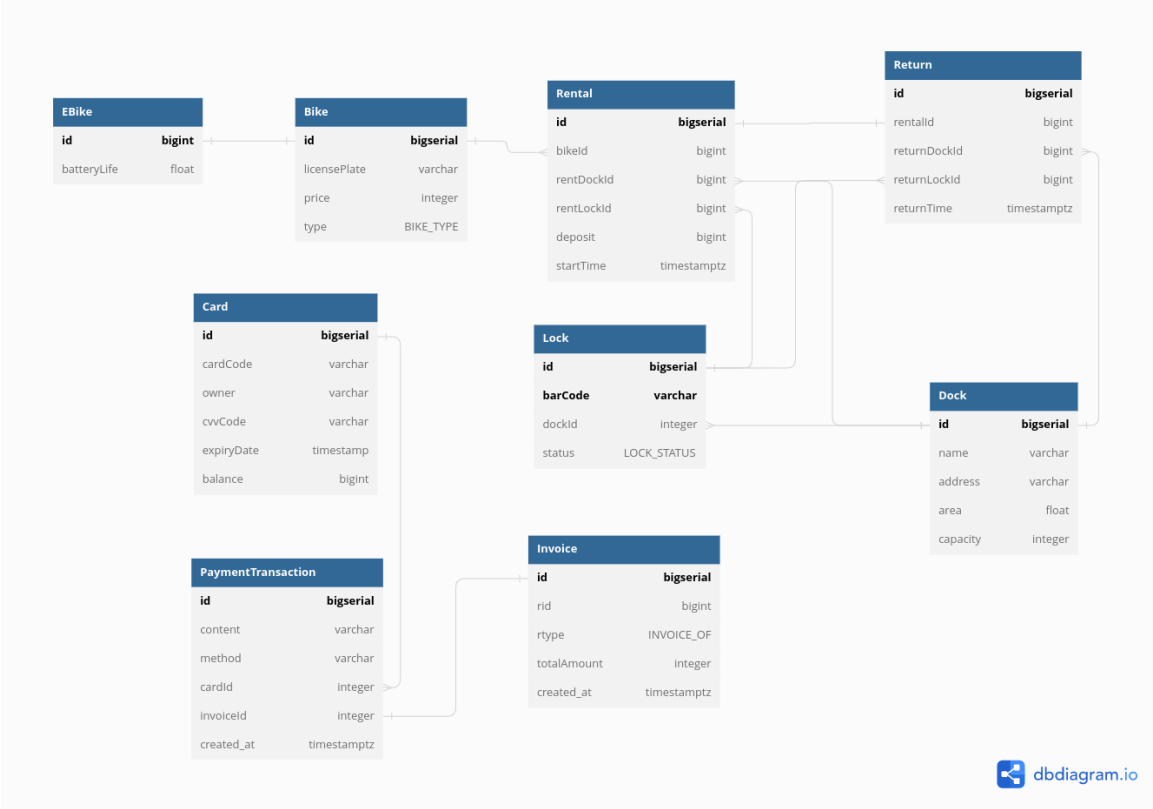


4.2.2 Database design

4.2.2.1 Database management system

Database Management System: PostgreSQL

4.2.2.2 Logical data model



4.2.2.3 Physical Data Model

```
CREATE TYPE "BIKE_TYPE" AS ENUM (
  'STANDARD_BIKE',
  'STANDARD_EBIKE',
  'TWIN_BIKE'
);

CREATE TYPE "LOCK_STATUS" AS ENUM (
  'LOCKED',
  'RELEASED'
);

CREATE TYPE "INVOICE_OF" AS ENUM (
  'RENTAL',
  'RETURN'
```

```
);

CREATE TABLE "Bike" (
  "id" bigserial PRIMARY KEY,
  "licensePlate" varchar NOT NULL,
  "price" integer NOT NULL,
  "type" "BIKE_TYPE" NOT NULL
);

CREATE TABLE "EBike" (
  "id" bigint PRIMARY KEY,
  "batteryLife" float NOT NULL
);

CREATE TABLE "Dock" (
  "id" bigserial PRIMARY KEY,
  "name" varchar NOT NULL,
  "address" varchar NOT NULL,
  "area" float NOT NULL,
  "capacity" integer NOT NULL
);

CREATE TABLE "Lock" (
  "id" bigserial PRIMARY KEY,
  "barCode" varchar,
  "bikeId" bigint,
  "dockId" bigint NOT NULL,
  "status" "LOCK_STATUS" NOT NULL DEFAULT 'RELEASED'
);
```

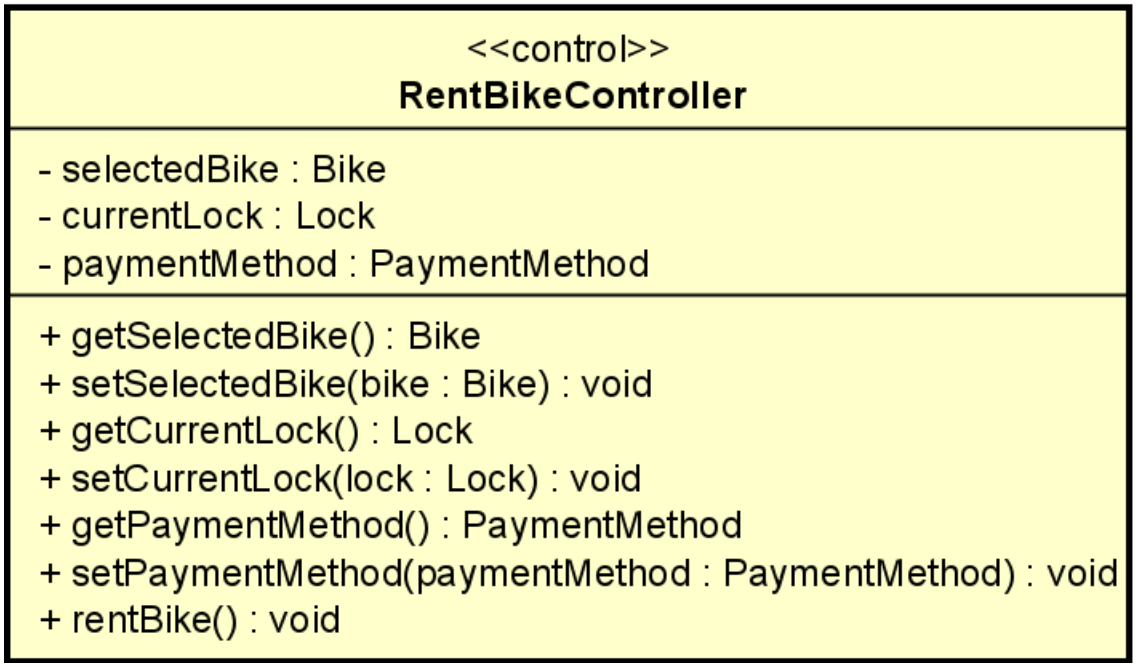
```
CREATE TABLE "Rental" (
  "id" bigserial PRIMARY KEY,
  "bikeId" bigint NOT NULL,
  "rentDockId" bigint NOT NULL,
  "rentLockId" bigint NOT NULL,
  "deposit" bigint NOT NULL,
  "startTime" timestamptz NOT NULL
);

CREATE TABLE "Return" (
  "id" bigserial PRIMARY KEY,
  "rentalId" bigint NOT NULL,
  "returnDockId" bigint NOT NULL,
  "returnLockId" bigint NOT NULL,
  "returnTime" timestamptz NOT NULL
);

CREATE TABLE "Invoice" (
  "id" bigserial PRIMARY KEY,
  "rid" bigint NOT NULL,
  "rtype" "INVOICE_OF" NOT NULL,
  "totalAmount" integer NOT NULL,
  "created_at" timestamptz NOT NULL DEFAULT (now())
);

CREATE TABLE "PaymentTransaction" (
  "id" bigserial PRIMARY KEY,
  "content" varchar NOT NULL DEFAULT ",
```

```
  "method" varchar NOT NULL DEFAULT 'credit card',
  "cardId" integer NOT NULL,
  "invoiceId" integer NOT NULL,
  "created_at" timestamptz NOT NULL DEFAULT (now())
);

CREATE TABLE "Card" (
  "id" bigserial PRIMARY KEY,
  "cardCode" varchar NOT NULL,
  "owner" varchar NOT NULL,
  "cvvCode" varchar NOT NULL,
  "expiryDate" timestamp NOT NULL,
  "balance" bigint NOT NULL
);

COMMENT ON COLUMN "Bike"."price" IS 'must ge greater than 0';

COMMENT ON COLUMN "EBike"."batteryLife" IS 'must be greater than 0';

COMMENT ON COLUMN "Dock"."area" IS 'must be greater than 0';

COMMENT ON COLUMN "Dock"."capacity" IS 'must be greater than 0';

COMMENT ON COLUMN "Rental"."deposit" IS 'must be greater than 0';

COMMENT ON COLUMN "Return"."returnTime" IS 'must be later than startTime';

COMMENT ON COLUMN "Invoice"."rid" IS 'id of corresponding rental or return';
```

```
COMMENT ON COLUMN "Invoice"."totalAmount" IS 'must be greater than 0';

COMMENT ON COLUMN "Card"."balance" IS 'must be greater than 0';

ALTER TABLE "EBike" ADD FOREIGN KEY ("id") REFERENCES "Bike" ("id");

ALTER TABLE "Lock" ADD FOREIGN KEY ("dockId") REFERENCES "Dock" ("id");

ALTER TABLE "Rental" ADD FOREIGN KEY ("bikeId") REFERENCES "Bike" ("id");

ALTER TABLE "Rental" ADD FOREIGN KEY ("rentDockId") REFERENCES "Dock"
("id");

ALTER TABLE "Rental" ADD FOREIGN KEY ("rentLockId") REFERENCES "Lock"
("id");

ALTER TABLE "Return" ADD FOREIGN KEY ("rentalId") REFERENCES "Rental"
("id");

ALTER TABLE "Return" ADD FOREIGN KEY ("returnDockId") REFERENCES
"Dock" ("id");

ALTER TABLE "Return" ADD FOREIGN KEY ("returnLockId") REFERENCES "Lock"
("id");

ALTER TABLE "PaymentTransaction" ADD FOREIGN KEY ("cardId") REFERENCES
"Card" ("id");

ALTER TABLE "PaymentTransaction" ADD FOREIGN KEY ("invoiceId")
REFERENCES "Invoice" ("id");
```

4.3 Class design

4.3.1 Class RentBikeController



Attributes

#	Name	Data type	Default value	Description
1	selectedBike	Bike	Null	Represent the currently rented bike
2	currentLock	Lock	Null	Represent the lock that selectedBike currently attached to
3	paymentMethod	PaymentMethod	null	Represent the currently selected payment method

Operations

#	Name	Return type	Description
1	rentBike	void	Process to rent the selected Bike: start a rental session, update database, make a call to interbank api

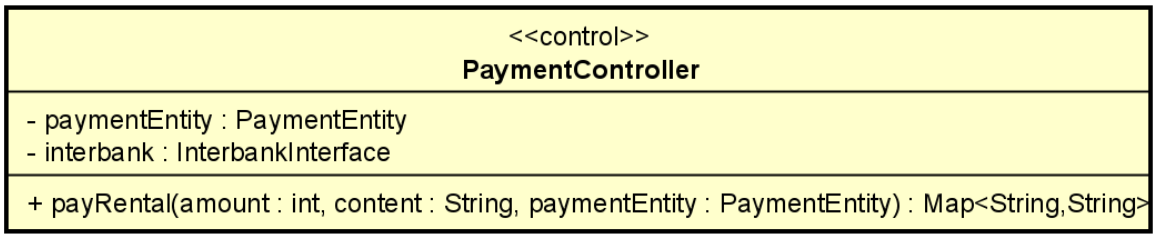
Methods

None

States

None

4.3.2 Class PaymentController



Attributes

#	Name	Data type	Default value	Description
1	paymentEntity	PaymentEntity	null	Represent the payment entity (card/e-wallet/…) used for payment
2	interbank	InterbankInterface	null	Represent the Interbank subsystem

Operations

#	Name	Return type	Description
1	payRental	Map<String, String>	Pay rental, then return the result with a message

Parameter:

- amount-the amount to pay
- contents-the transaction contents
- paymentEntity – the entity (credit card/e-wallet/...) used for payment

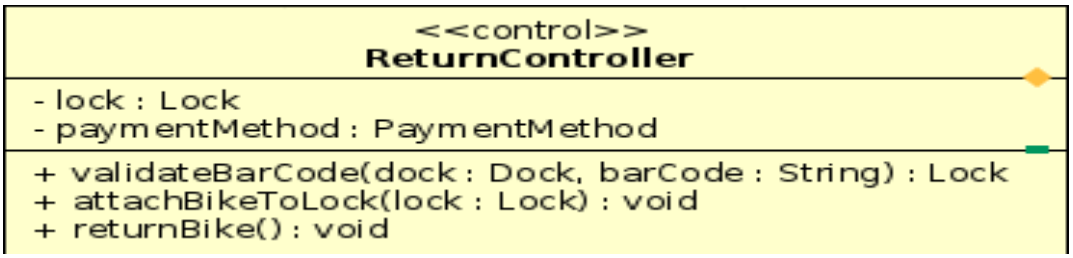
Methods

None

States

None

4.3.3 Class ReturnBikeController



Attributes

#	Name	Data type	Default value	Description
1	lock	Lock	null	lock that the user returns the car
2	paymentMethod	aymentMethod	null	How to calculate car rental

Operations

#	Name	Return type	Description

1	VarlidateBarCode	Lock	Check if the barcode is valid
2	attachBikeToLock	void	put the bike back to the selected dock in
3	returnBike		complete the car return operations

Parameter:

- dock-the dock return bike
- barCode -bar code of lock in dock

Exceptions:

- InvalidBarcodeException if barcode is invalid
- BarCodeNotFoundException: if the lock is not found in this dock
- LockNotFreeException : if lock is in use

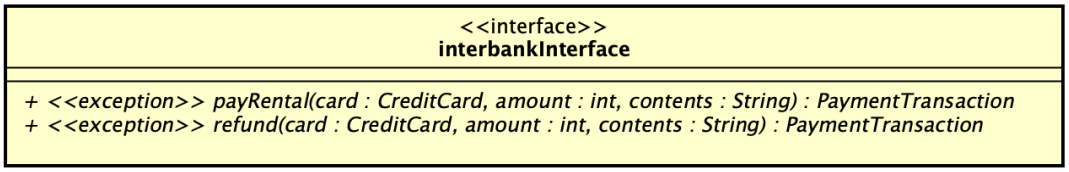
Methods

None

States

None

4.3.4 Class InterbankInterface



Attributes

None

Operations

#	Name	Return type	Description

1	payRental	PaymentTransaction	PayRental , and then return the payment transaction
2	refund	PaymentTransaction	Refund, and then return the payment transaction

Parameters:

- Card- the credit card used for payment/refund
- Amount-the amount to pay/refund
- Contents - the transaction contents

Exceptions:

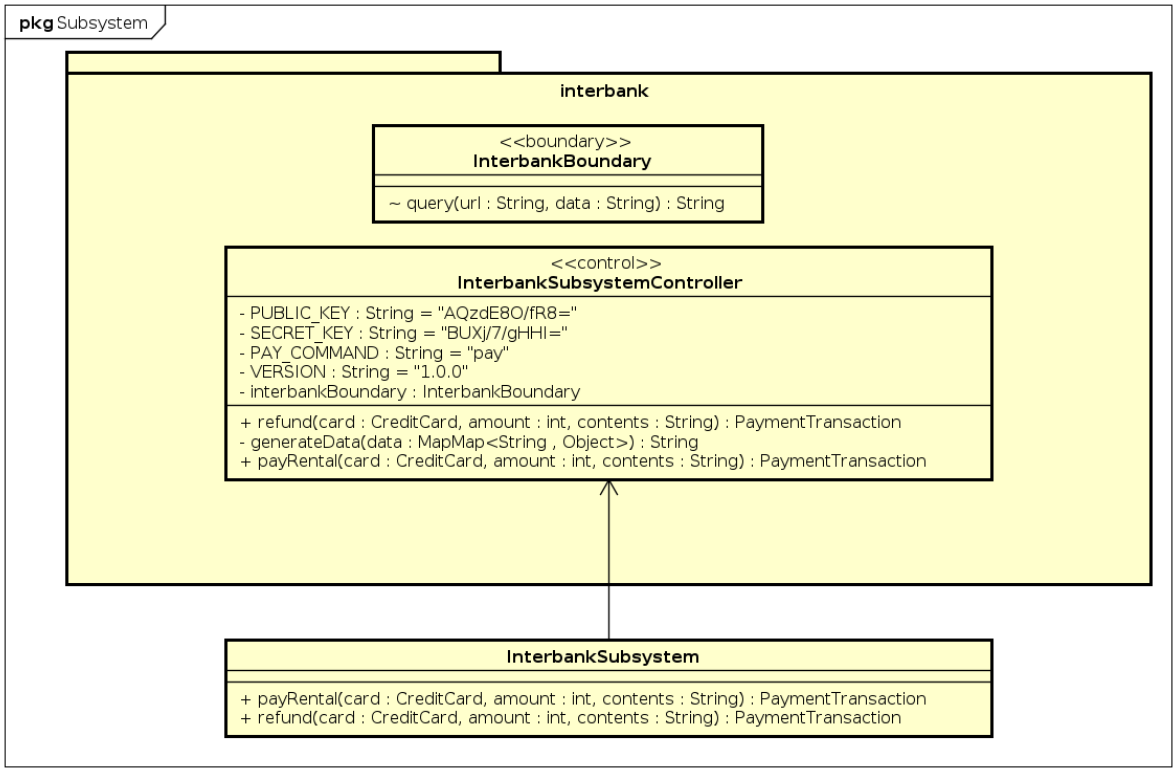
- PaymentException - if responded with a pre-defind error code
- UnrecognizedException: if responded with an unknown error code or something goes wrong

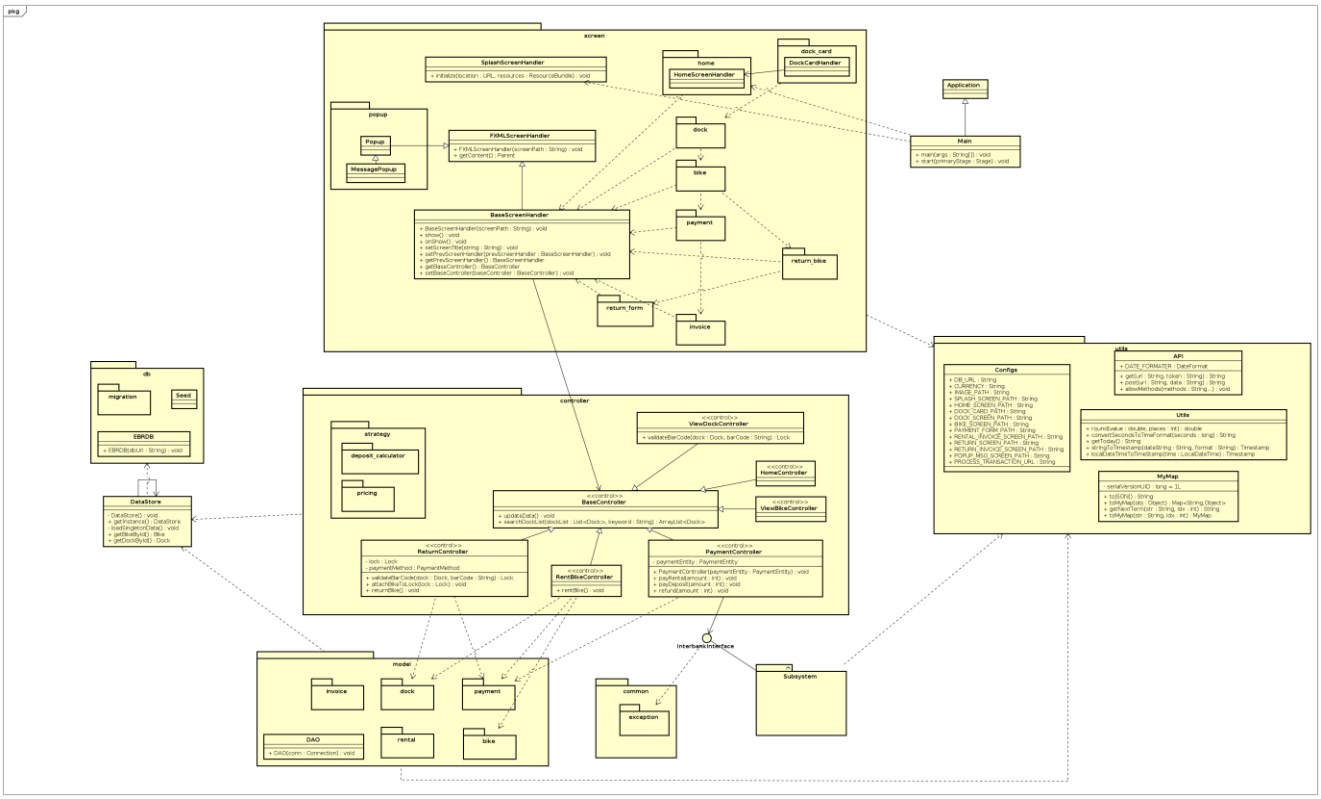
Methods

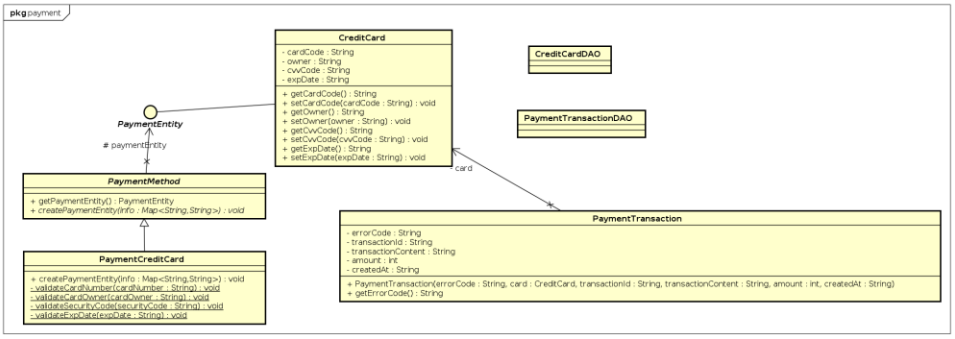
None

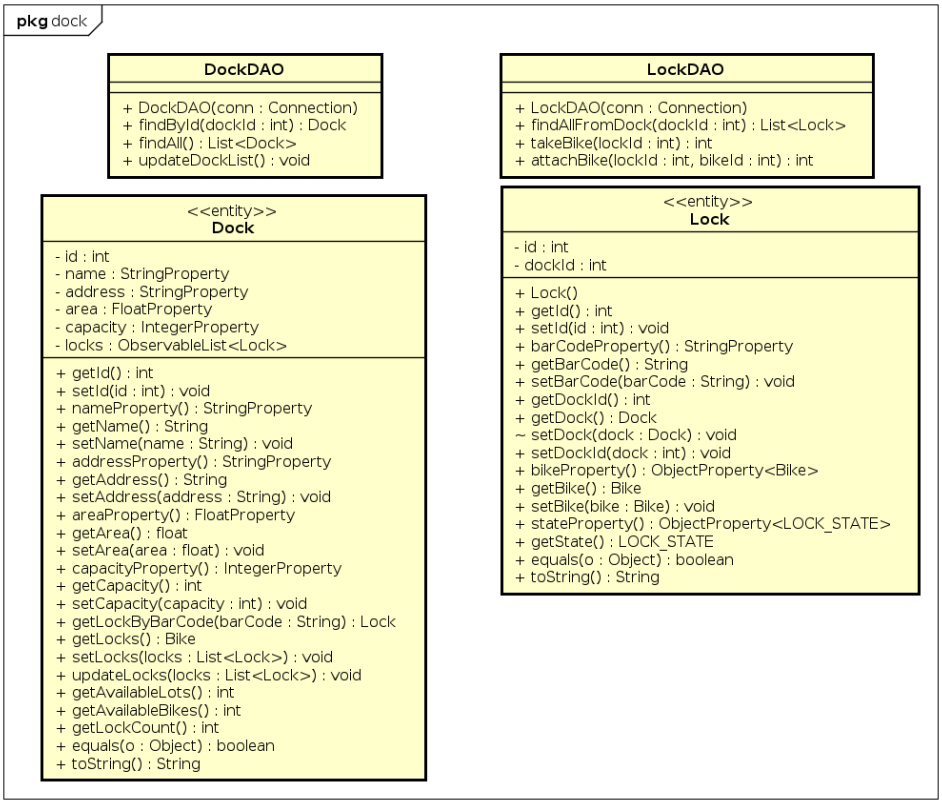
States

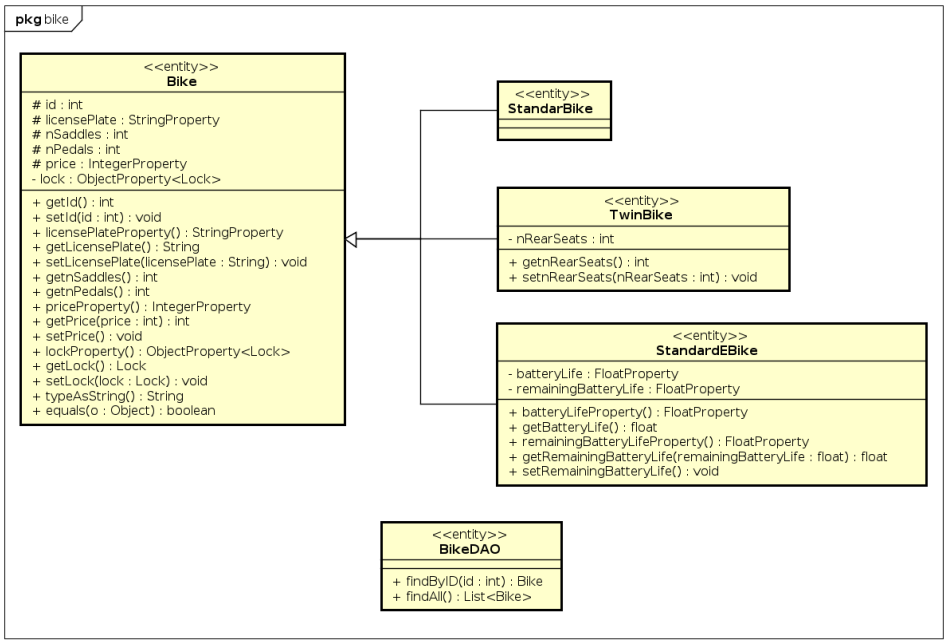
None




pkg
screen
SplashScreenHandler
+ initialize(location : URL, resources : ResourceBundle) : void
home
HomeScreenHandler
dock_card
DockCardHandler
Application
popup
Popup
MessagePopup
FXMLScreenHandler
+ FXMLScreenHandler(screenPath : String) : void
+ getContent() : Parent
dock
bike
payment
return_bike
return_form
invoice
Main
+ main(args : String[]) : void
+ start(primaryStage : Stage) : void
BaseScreenHandler
+ BaseScreenHandler(screenPath : String) : void
+ show() : void
+ setNode() : void
+ setScreenTitle(string : String) : void
+ setPrevScreenHandler(prevScreenHandler : BaseScreenHandler) : void
+ getPrevScreenHandler() : BaseScreenHandler
+ getBaseController() : BaseController
+ setBaseController(baseController : BaseController) : void
db
migration
Seed
ERROB
+ ERROB(dbUrl : String) : void
DataStore
- DataStore() : void
+ getInstance() : DataStore
- loadSingletonData() : void
+ getBikeByid() : Bike
+ getDockById() : Dock
controller
strategy
deposit_calculator
pricing
«controller»
ViewDockController
+ validateBarCode(dock : Dock, barCode : String) : Lock
«controller»
HomeController
«controller»
BaseController
+ updateData() : void
+ searchDockListBasisList : List<Dock>, keyword : String) : ArrayList<Dock>
«controller»
ViewBikeController
«controller»
ReturnController
- lock : Lock
- paymentMethod : PaymentMethod
+ validateBarCode(dock : Dock, barCode : String) : Lock
+ attachBikeToLock(lock : Lock) : void
+ returnBike() : void
«controller»
RentBikeController
+ rentBike() : void
«controller»
PaymentController
- paymentEntity : PaymentEntity
+ PaymentController(paymentEntity : PaymentEntity) : void
+ payRental(amount : int) : void
+ payDeposit(amount : int) : void
+ refund(amount : int) : void
utils
Configs
+ DB_URL : String
+ CURRENCY : String
+ IMAGE_PATH : String
+ SPLASH_SCREEN_PATH : String
+ DOCK_SCREEN_PATH : String
+ DOCK_CARD_PATH : String
+ DOCK_SCREEN_PATH : String
+ BIKE_SCREEN_PATH : String
+ PAYMENT_FORM_PATH : String
+ RENTAL_INVOICE_SCREEN_PATH : String
+ RETURN_SCREEN_PATH : String
+ RETURN_INVOICE_SCREEN_PATH : String
+ POPUP_MSG_SCREEN_PATH : String
+ PROCESS_TRANSACTION_URL : String
API
+ DATE_FORMATTER : DateFormat
+ getUrl : String, token : String) : String
+ postUrl : String, data : String) : String
+ allowMethods(@methods : String...) : void
Utils
+ roundValue : double, places : int) : double
+ convertSecondsToTimeFormat(seconds : long) : String
+ getToday() : String
+ stringToTimestamp(dateString : String, format : String) : Timestamp
+ localDateTimeToTimeStamp(time : LocalDateTime) : Timestamp
MyMap
- serialVersionUID : long = 1L
+ toJSON() : String
+ convert(object) : Map<String,Object>
+ getNextTerm(str : String, idx : int) : String
+ toMyMap(str : String, idx : int) : MyMap
model
invoice
dock
payment
DAO
+ DAO(conn : Connection) : void
rental
bike
InterbankInterface
common
exception
Subsystem


pkg payment
CreditCard
- cardCode : String
- owner : String
- cvvCode : String
- expDate : String
+ getCardCode() : String
+ setCardCode(cardCode : String) : void
+ getOwner() : String
+ setOwner(owner : String) : void
+ getCvvCode() : String
+ setCvvCode(cvvCode : String) : void
+ getExpDate() : String
+ setExpDate(expDate : String) : void
CreditCardDAO
PaymentTransactionDAO
PaymentEntity
paymentEntity
card
PaymentMethod
+ getPaymentEntity() : PaymentEntity
+ createPaymentEntity(info : Map<String,String>) : void
PaymentCreditCard
+ createPaymentEntity(info : Map<String,String>) : void
- validateCardNumber(cardNumber : String) : void
- validateCardOwner(cardOwner : String) : void
- validateSecurityCode(securityCode : String) : void
- validateExpDate(expDate : String) : void
PaymentTransaction
- errorCode : String
- transactionId : String
- transactionContent : String
- amount : int
- createdAt : String
+ PaymentTransaction(errorCode : String, card : CreditCard, transactionId : String, transactionContent : String, amount : int, createdAt : String)
+ getErrorCode() : String


pkg dock
DockDAO
+ DockDAO(conn : Connection)
+ findById(dockId : int) : Dock
+ findAll() : List<Dock>
+ updateDockList() : void
LockDAO
+ LockDAO(conn : Connection)
+ findAllFromDock(dockId : int) : List<Lock>
+ takeBike(lockId : int) : int
+ attachBike(lockId : int, bikeId : int) : int
<<entity>>
Dock
- id : int
- name : StringProperty
- address : StringProperty
- area : FloatProperty
- capacity : IntegerProperty
- locks : ObservableList<Lock>
+ getId() : int
+ setId(id : int) : void
+ nameProperty() : StringProperty
+ getName() : String
+ setName(name : String) : void
+ addressProperty() : StringProperty
+ getAddress() : String
+ setAddress(address : String) : void
+ areaProperty() : FloatProperty
+ getArea() : float
+ setArea(area : float) : void
+ capacityProperty() : IntegerProperty
+ getCapacity() : int
+ setCapacity(capacity : int) : void
+ getLockByBarCode(barCode : String) : Lock
+ getLocks() : Bike
+ setLocks(locks : List<Lock>) : void
+ updateLocks(locks : List<Lock>) : void
+ getAvailableLots() : int
+ getAvailableBikes() : int
+ getLockCount() : int
+ equals(o : Object) : boolean
+ toString() : String
<<entity>>
Lock
- id : int
- dockId : int
+ Lock()
+ getId() : int
+ setId(id : int) : void
+ barCodeProperty() : StringProperty
+ getBarCode() : String
+ setBarCode(barCode : String) : void
+ getDockId() : int
+ getDock() : Dock
~ setDock(dock : Dock) : void
+ setDockId(dock : int) : void
+ bikeProperty() : ObjectProperty<Bike>
+ getBike() : Bike
+ setBike(bike : Bike) : void
+ stateProperty() : ObjectProperty<LOCK_STATE>
+ getState() : LOCK_STATE
+ equals(o : Object) : boolean
+ toString() : String


pkg bike
<<entity>>
Bike
id : int
licensePlate : StringProperty
nSaddles : int
nPedals : int
price : IntegerProperty
- lock : ObjectProperty<Lock>
+ getId() : int
+ setId(id : int) : void
+ licensePlateProperty() : StringProperty
+ getLicensePlate() : String
+ setLicensePlate(licensePlate : String) : void
+ getnSaddles() : int
+ getnPedals() : int
+ priceProperty() : IntegerProperty
+ getPrice(price : int) : int
+ setPrice() : void
+ lockProperty() : ObjectProperty<Lock>
+ getLock() : Lock
+ setLock(lock : Lock) : void
+ typeAsString() : String
+ equals(o : Object) : boolean
<<entity>>
StandarBike
<<entity>>
TwinBike
- nRearSeats : int
+ getnRearSeats() : int
+ setnRearSeats(nRearSeats : int) : void
<<entity>>
StandardEBike
- batteryLife : FloatProperty
- remainingBatteryLife : FloatProperty
+ batteryLifeProperty() : FloatProperty
+ getBatteryLife() : float
+ remainingBatteryLifeProperty() : FloatProperty
+ getRemainingBatteryLife(remainingBatteryLife : float) : float
+ setRemainingBatteryLife() : void
<<entity>>
BikeDAO
+ findByID(id : int) : Bike
+ findAll() : List<Bike>