HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

School of Information and communications technology

Software Design Document

EcoBike Application

Subject: ITSS Software Management

Group 01

|  |  |  |
| --- | --- | --- |
| No. | Student Name | Student ID |
| 1 | Nguyen Quoc Anh | 20194726 |
| 2 | Pham Thanh Bien | 20194731 |
| 3 | Dinh Huu Dai | 20194735 |

*Hanoi, 02/2023*

# Table of Contents

[Table of Contents 1](#_Toc127222633)

[List of Figures 4](#_Toc127222634)

[List of Tables 6](#_Toc127222635)

[1. Introduction 8](#_Toc127222636)

[1.1. Objective 8](#_Toc127222637)

[1.2. Scope 8](#_Toc127222638)

[1.3. Glossary 8](#_Toc127222639)

[1.4. References 9](#_Toc127222640)

[2. Overall Description 11](#_Toc127222641)

[2.1. General Overview 11](#_Toc127222642)

[2.2. Assumptions/Constraints/Risks 12](#_Toc127222643)

[2.2.1. Assumptions 12](#_Toc127222644)

[2.2.2. Constraints 12](#_Toc127222645)

[2.2.3. Risks 12](#_Toc127222646)

[3. System Architecture and Architecture Design 13](#_Toc127222647)

[3.1. Architectural Patterns 13](#_Toc127222648)

[3.2. Interaction Diagrams 13](#_Toc127222649)

[3.2.1. Communication Diagrams 13](#_Toc127222650)

[3.2.2. Sequence Diagrams 16](#_Toc127222651)

[3.3. Analysis Class Diagrams 19](#_Toc127222652)

[3.4. Unified Analysis Class Diagram 22](#_Toc127222653)

[3.5. Security Software Architecture 22](#_Toc127222654)

[4. Detailed Design 23](#_Toc127222655)

[4.1. User Interface Design 23](#_Toc127222656)

[4.1.1. Screen Configuration Standardization 23](#_Toc127222657)

[4.1.2. Screen Transition Diagrams 23](#_Toc127222658)

[4.1.3. Screen Specifications 23](#_Toc127222659)

[4.1.3.1. Splash Screen 23](#_Toc127222660)

[4.1.3.2. Main Screen 24](#_Toc127222661)

[4.1.3.3. Dock Screen 24](#_Toc127222662)

[4.1.3.4. Bike Screen 25](#_Toc127222663)

[4.1.3.5. Payment Method Screen 26](#_Toc127222664)

[4.1.3.6. Deposit screen 26](#_Toc127222665)

[4.1.3.7. Payment screen 27](#_Toc127222666)

[4.2. Data Modeling 28](#_Toc127222667)

[4.2.1. Conceptual Data Modeling 28](#_Toc127222668)

[4.2.2. Database Design 28](#_Toc127222669)

[4.2.2.1. Database Management System 28](#_Toc127222670)

[4.2.2.2. Database Diagram 29](#_Toc127222671)

[4.2.2.3. Database Detail Design 29](#_Toc127222672)

[4.3. Non-Database Management System Files 36](#_Toc127222673)

[4.4. Class Design 36](#_Toc127222674)

[4.4.1. General Class Diagram 36](#_Toc127222675)

[4.4.2. Class Diagrams 37](#_Toc127222676)

[4.4.2.1. Class Diagram for Package BikeInformation 37](#_Toc127222677)

[4.4.2.2. Class Diagram for Subsystem RentBike 38](#_Toc127222678)

[4.4.2.3. Class Diagram for Subsystem InterBank 38](#_Toc127222679)

[4.4.3. Class Design 39](#_Toc127222680)

[4.4.3.1. Class RentBikeController 39](#_Toc127222681)

[4.4.3.2. Class ReturnBikeController 40](#_Toc127222682)

[4.4.3.3. Class PaymentController 42](#_Toc127222683)

[4.4.3.4. Class BikeInformationScreenHandler 44](#_Toc127222684)

[5. Design Considerations 46](#_Toc127222685)

[5.1. Goals and Guidelines 46](#_Toc127222686)

[5.2. Architectural Strategies 46](#_Toc127222687)

[5.3. Coupling and Cohesion 46](#_Toc127222688)

[5.3.1. Coupling 46](#_Toc127222689)

[5.3.1.1. Content coupling 46](#_Toc127222690)

[5.3.1.2. Common coupling 47](#_Toc127222691)

[5.3.1.3. Control coupling 47](#_Toc127222692)

[5.3.1.4. Stamp coupling 47](#_Toc127222693)

[5.3.1.5. Data coupling 48](#_Toc127222694)

[5.3.2. Cohesion 48](#_Toc127222695)

[5.3.2.1. Coincidental cohesion 48](#_Toc127222696)

[5.3.2.2. Logical cohesion 48](#_Toc127222697)

[5.3.2.3. Temporal cohesion 48](#_Toc127222698)

[5.3.2.4. Procedure cohesion 49](#_Toc127222699)

[5.3.2.5. Communicational cohesion 49](#_Toc127222700)

[5.3.2.6. Sequential cohesion 49](#_Toc127222701)

[5.3.2.7. Information cohesion 49](#_Toc127222702)

[5.3.2.8. Functional cohesion 49](#_Toc127222703)

[5.4. Design Principles 50](#_Toc127222704)

[5.5. Design Patterns 50](#_Toc127222705)

# List of Figures

*Figure 1.1: General use case diagram 11*

*Figure 3.1: Communication Diagram for Rent Bike Use Case 13*

*Figure 3.2: Communication Diagram for Deposit Use Case 13*

*Figure 3.3: Communication Diagram for Update Payment Method Use Case 14*

*Figure 3.4: Communication Diagram for Return Bike Use Case 14*

*Figure 3.5: Communication Diagram for Return Deposit Use Case 15*

*Figure 3.6: Communication Diagram for Pay For Rental Use Case 15*

*Figure 3.7: Sequence Diagram for Rent Bike Use Case 16*

*Figure 3.8: Sequence Diagram for Deposit Use Case 16*

*Figure 3.9: Sequence Diagram for Update Payment Method Use Case 17*

*Figure 3.10: Sequence Diagram for Return Bike Use Case 17*

*Figure 3.11: Sequence Diagram for Return Deposit Use Case 18*

*Figure 3.12: Sequence Diagram for Pay For Rental Use Case 18*

*Figure 3.13: Class Diagram for View Bike Use Case 19*

*Figure 3.14: Class Diagram for View Bike Use Case 19*

*Figure 3.15: Class Diagram for Deposit Use Case 20*

*Figure 3.16: Class Diagram for Return Bike Use Case 20*

*Figure 3.17: Class Diagram for Return Deposit Use Case 21*

*Figure 3.18: Class Diagram for Pay Rental Use Case 21*

*Figure 3.19: Unified Class Diagram for EcoBike Application 21*

*Figure 4.1: Screen Transition Diagram for EcoBike Application 23*

*Figure 4.2. ER Diagram for EcoBike Application 28*

*Figure 4.3. Database Diagram for EcoBike Application 29*

*Figure 4.4. General Class Diagram for EcoBike Application 36*

*Figure 4.5. Class Diagram for Package BikeInformation 37*

*Figure 4.6. Class Diagram for Subsystem RentBike 38*

*Figure 4.7. Class Diagram for Subsystem InterBank 38*

*Figure 4.8. RentBikeController Class Diagram 39*

*Figure 4.9. ReturnBikeController Class Diagram 40*

*Figure 4.10. PaymentControllerClass Diagram 42*

# List of Tables

*Table 1.1: Terms used in the document 10*

*Table 4.1. Splash Screen Specification 24*

*Table 4.2. Main Screen Specification 24*

*Table 4.3. View Dock Screen Specification 25*

*Table 4.4. View Bike Screen Specification 25*

*Table 4.5. Payment Method Screen Specification 26*

*Table 4.6. Deposit Screen Specification 26*

*Table 4.7. Payment Screen Specification 27*

*Table 4.8. Customer table design 29*

*Table 4.9. Administrator table design 30*

*Table 4.10. Dock table design* 30

*Table 4.11. Bike table design* 31

*Table 4.12. Bike In Dock table design* 31

*Table 4.13. Bike Status table design* 32

*Table 4.14. Invoice table design 32*

*Table 4.15. Transaction table design 33*

*Table 4.16. Rent Bike table design 33*

*Table 4.17. Credit Card table design 34*

*Table 4.18. RentBikeController attributes 39*

*Table 4.19. RentBikeController operations 40*

*Table 4.20. ReturnBikeController attributes 41*

*Table 4.21. ReturnBikeController operations 41*

*Table 4.22. PaymentController attributes 42*

*Table 4.23. PaymentController operations 43*

*Table 4.24.* BikeInformationScreenHandler *operations 45*

*Table 4.25.* BikeInformationScreenHandler *operations 45*

# 1. Introduction

## 1.1. Objective

This Software Design Document provides the design of EcoBike Application. It will explain the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate and how the system will react to external stimulation.

The expected audience is the user of the EcoBike Application, including Mrs. Trang and the ITSS Software development Course’s students, the developer of the project, and the people who will maintain the EcoBike Application.

## 1.2. Scope

This document contains a complete description of the design of EcoBike Application.

The EcoBike Application is a mobile application that allows the people of Hanoi to use the EcoBike service, including finding out nearby docks and renting bikes for personal usage and online payment for renting bikes.

The objective of the EcoBike Application is to serve a maximum of 10.000 users concurrently, with a friendly user interface with the aim of helping the user to find the most suitable service to rent or return the bike.

## 1.3. Glossary

|  |  |
| --- | --- |
| **Term** | **Definition** |
| Administrator | The person who uses EcoBike application system for the purposes of monitoring list of bicycles in the system |
| Admin | as “administrator” |
| Bicycle | The transportation mean to be rent in this application system |
| Bike | as “bicycle” |
| Card number | The ID number of the credit card, printed on the credit card |
| Cardholder name | The name of the owner of the credit card, printed on the credit card |
| Credit card | A card connected to the interbank, used for performing transaction |
| Customer | The person who uses EcoBike application system for the purposes of renting and returning bike |
| Database | Collection of all information monitored by this system |
| Deposit | An amount of money customer has to pay at first in order to rent a bike |
| Dock | A place where bikes are placed |
| Interbank | The organization in charges of performing payment and return deposit transactions in the system |
| Payment | An amount of money customer has to pay to rent a bike, including returned deposit and rental fee |
| Rent a bike | The action of using a bike in a period of time, with paying deposit and rental fee |
| Rental fee | An amount of money customer has to pay, outside of the deposit, which depends on the rental time |
| Rental time | The time period when the bike is being rented |
| Return a bike | The action of stopping using a bike after having rented |
| Software Requirement Specification | A document that completely describes all of the functions of a proposed system and the constraints under which it must operate. For example, this document. |
| Station | as “dock” |
| Transaction | The action of paying for bike deposit, bike rental or returning deposit |
| User | Customer or Administrator |

*Table 1.1: Terms used in the document*

## 1.4. References

|  |  |
| --- | --- |
| [1] | Centers for Medicare & Medicaid Services, "System Design Document Template," [Online]. Available: https://www.cms.gov/Research-Statistics-Data-and-Systems/CMS-Information-Technology/XLC/Downloads/SystemDesignDocument.docx. |

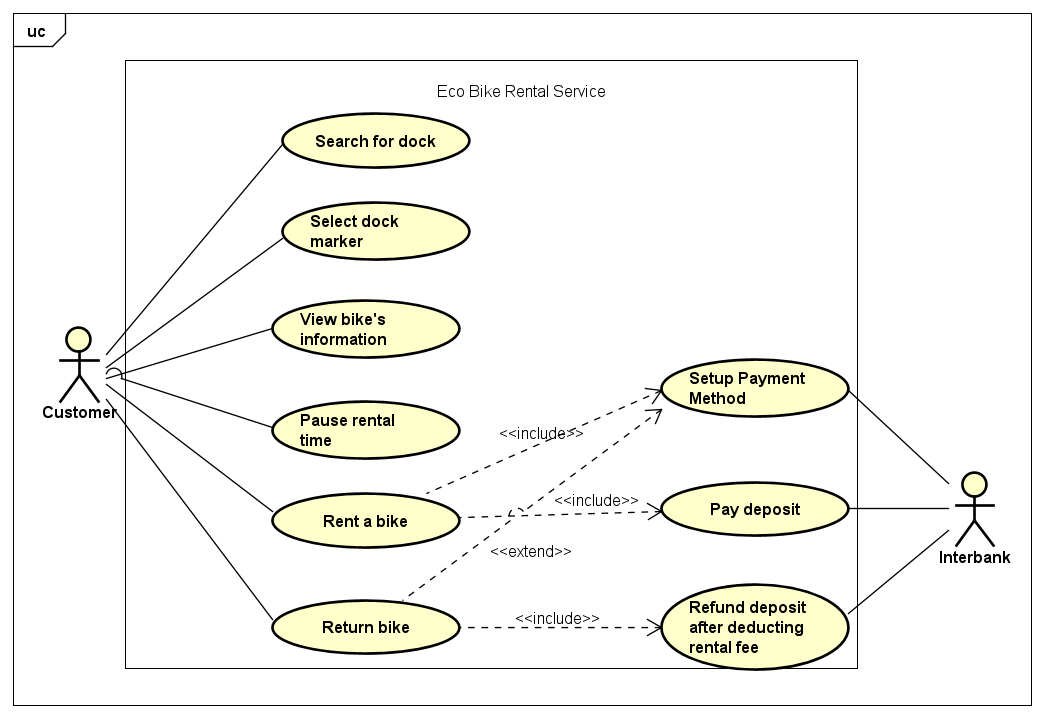
# 2. Overall Description

## 2.1. General Overview

EcoBike Application is a desktop application through which users can view docks and rent or return bikes. We design a clean and clear interface for users. Users can interact with the apps by clicking on the interface, and the request is processed by the controller.

Additionally, we have our own database to store information and data that is related to our system, as well as a subsystem to proceed payment transactions.

The below figure is the general use-case diagram for our design:



*Figure 1.1: General use case diagram*

## 2.2. Assumptions/Constraints/Risks

### 2.2.1. Assumptions

In order to use the application, users must have an internet connection as well as a personal computer to run the app. We would also require the latest version of JRE in order to ensure the application’s stability.

### 2.2.2. Constraints

· *Hardware or software environment*

· *End-user environment*

· *Availability or volatility of resources*

· *Standards compliance*

· *Interoperability requirements*

· *Interface/protocol requirements*

· *Licensing requirements*

· *Data repository and distribution requirements*

· *Security requirements (or other such regulations)*

· *Memory or other capacity limitations*

· *Performance requirements*

· *Network communications*

· *Verification and validation requirements (testing)*

· *Other means of addressing quality goals*

· *Other requirements described in the Requirements Document*

### 2.2.3. Risks

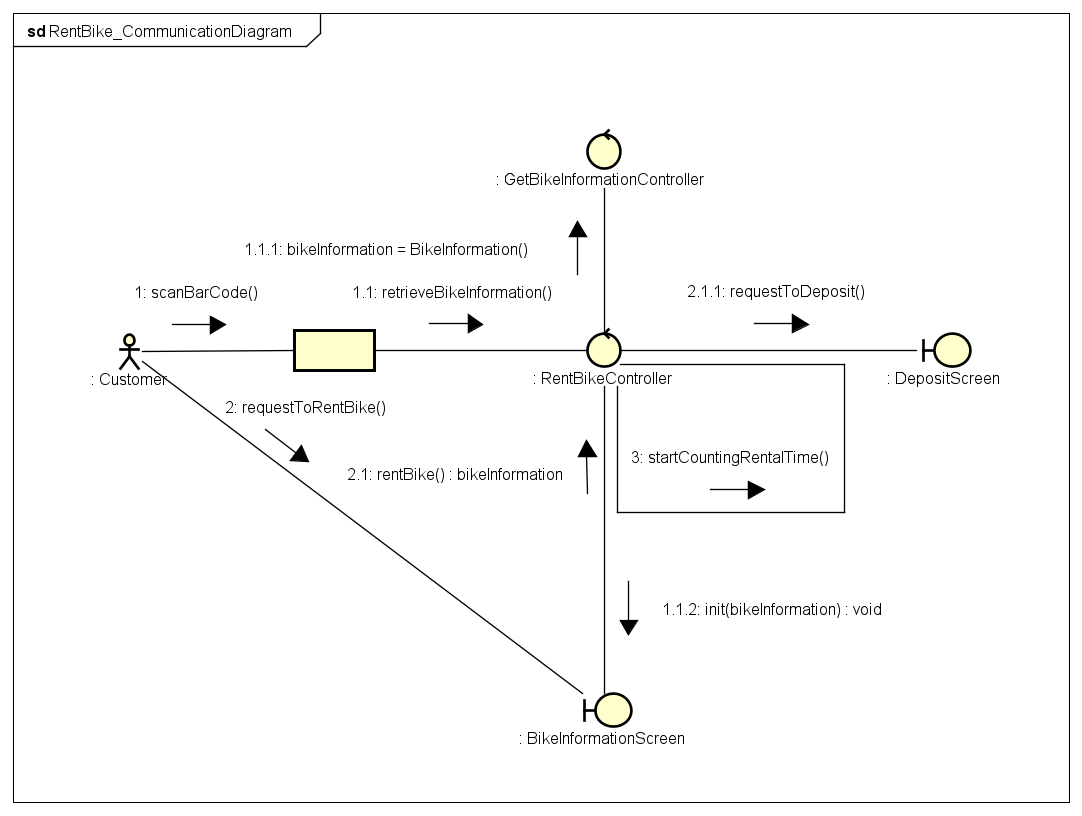
# 3. System Architecture and Architecture Design

## 3.1. Architectural Patterns

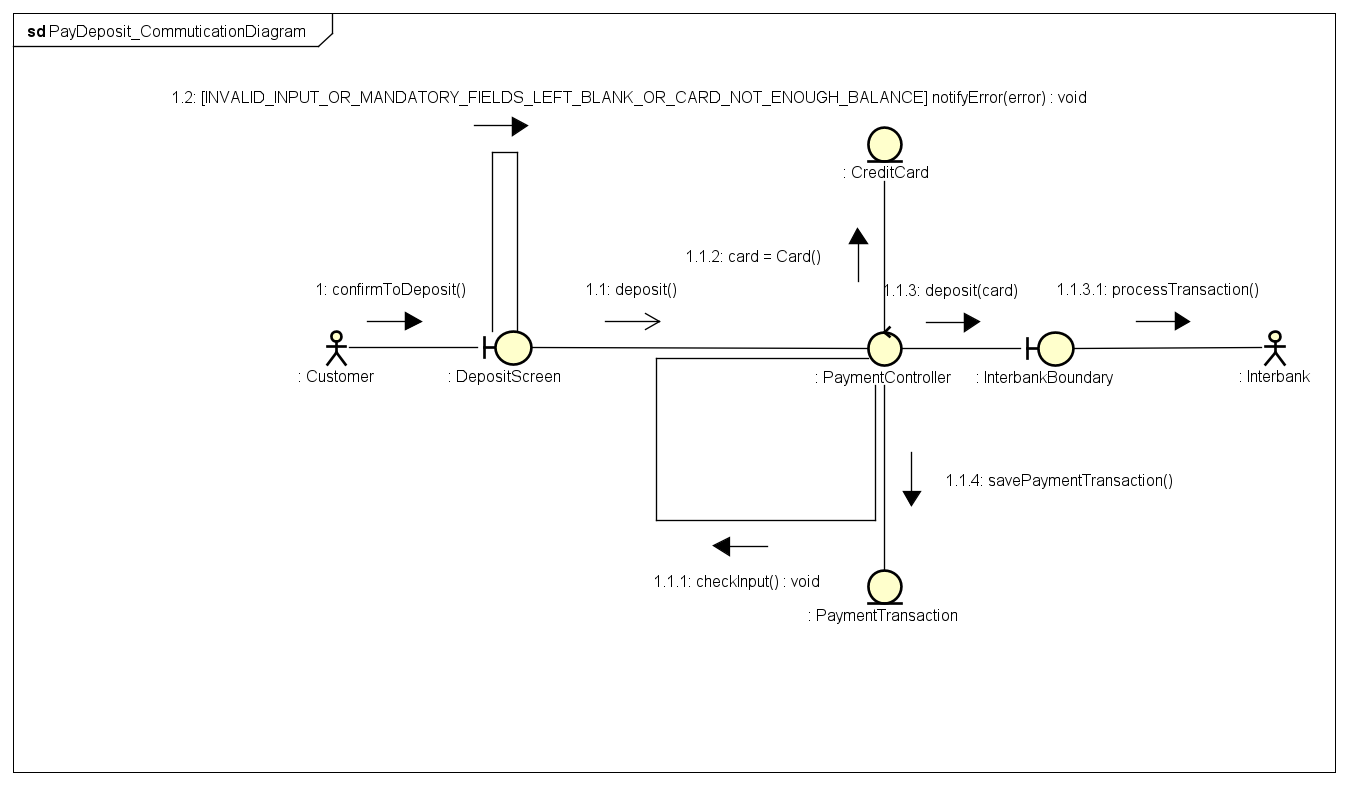
*<Specify and briefly describe the chosen architectural patterns and the reasons why they were chosen>*

## 3.2. Interaction Diagrams

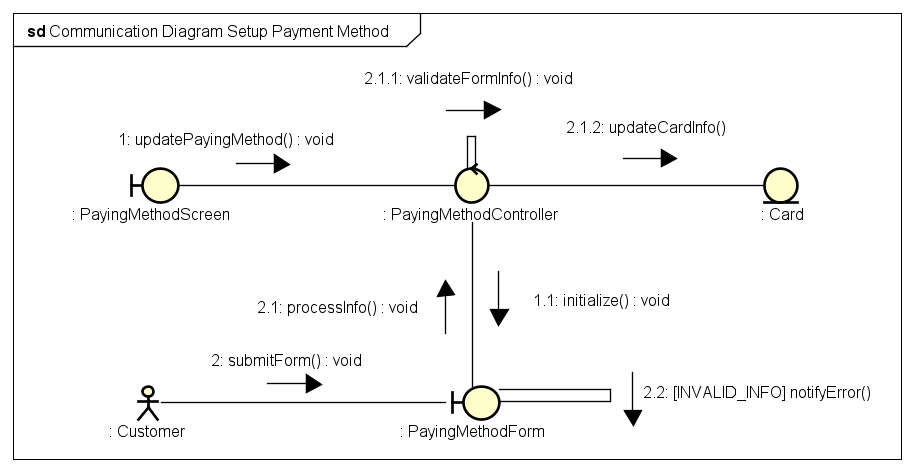
### 3.2.1. Communication Diagrams



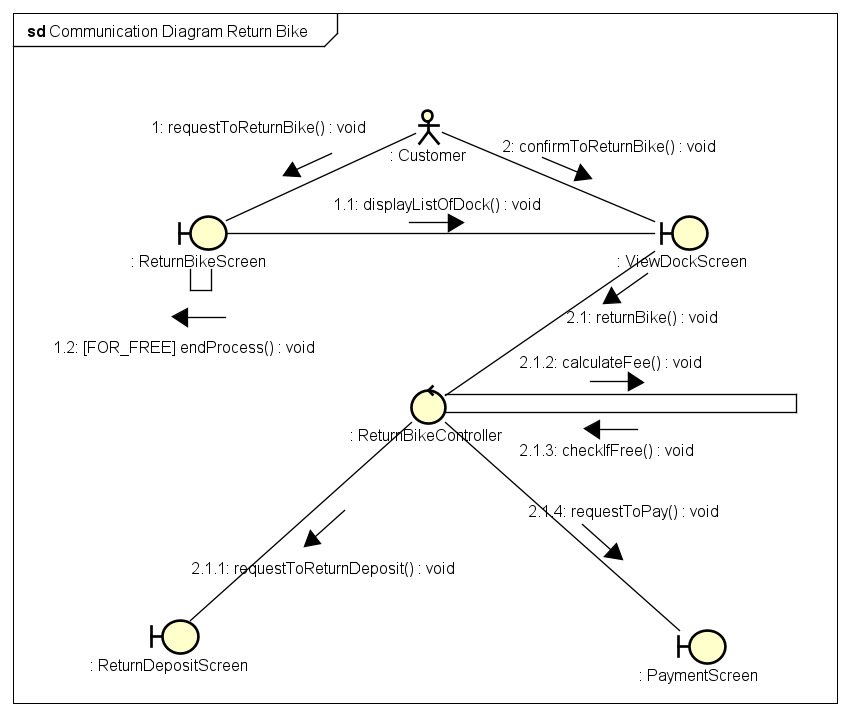
*Figure 3.1: Communication Diagram for Rent Bike Use Case*



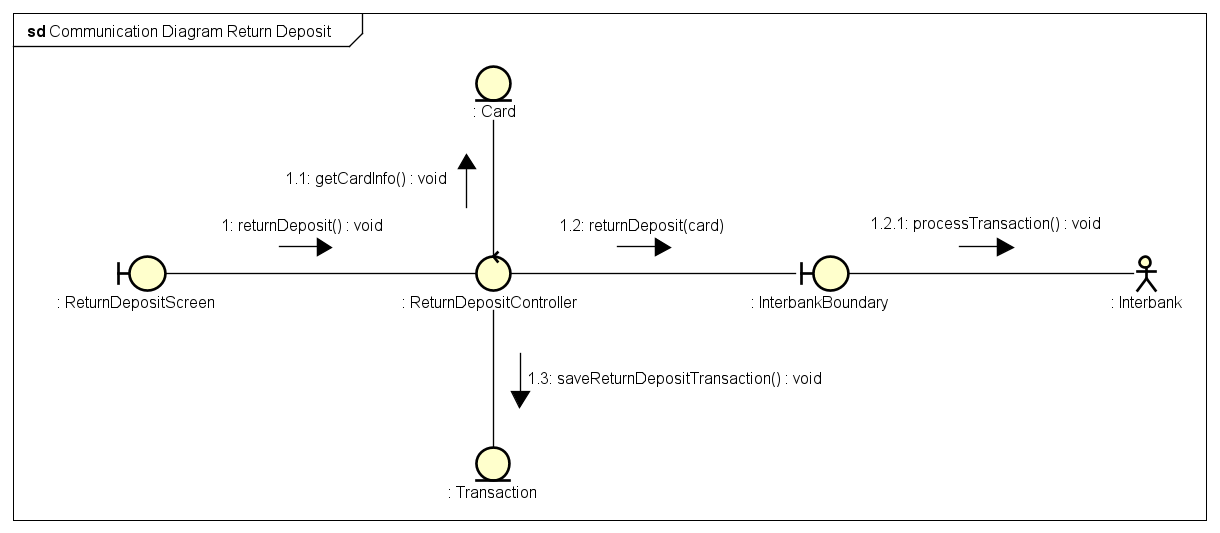
*Figure 3.2: Communication Diagram for Deposit Use Case*



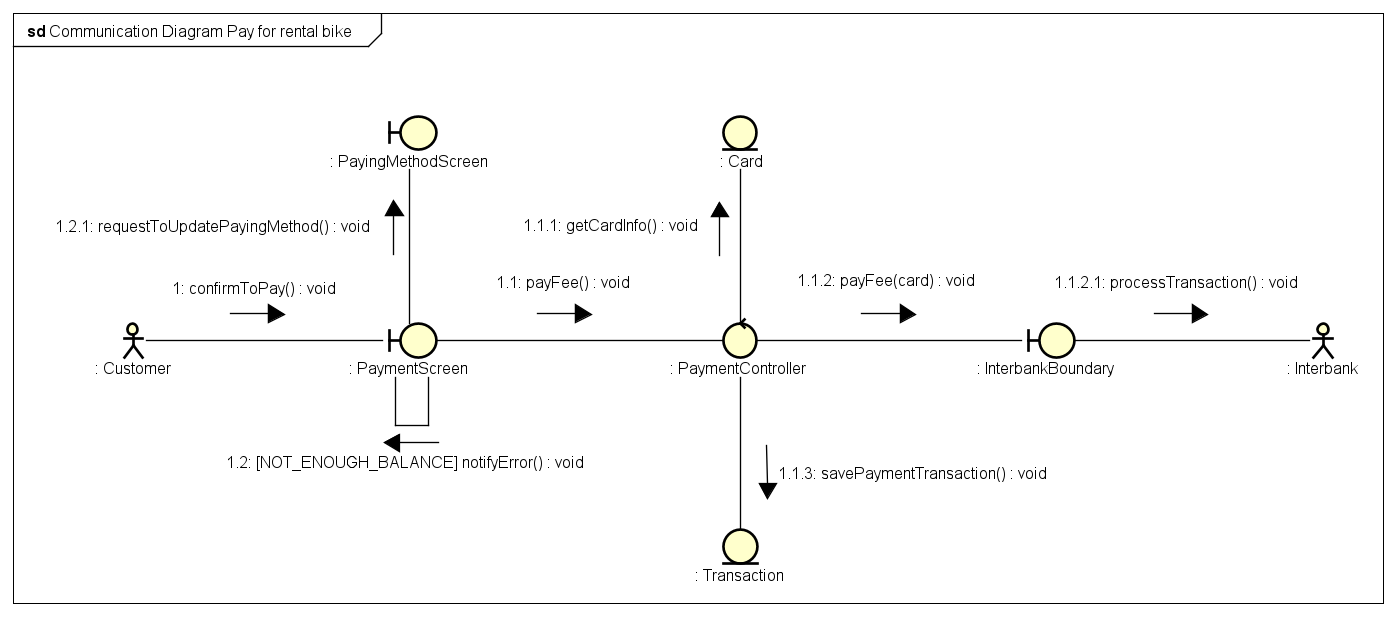
*Figure 3.3: Communication Diagram for Setup Payment Method Use Case*



*Figure 3.4: Communication Diagram for Return Bike Use Case*

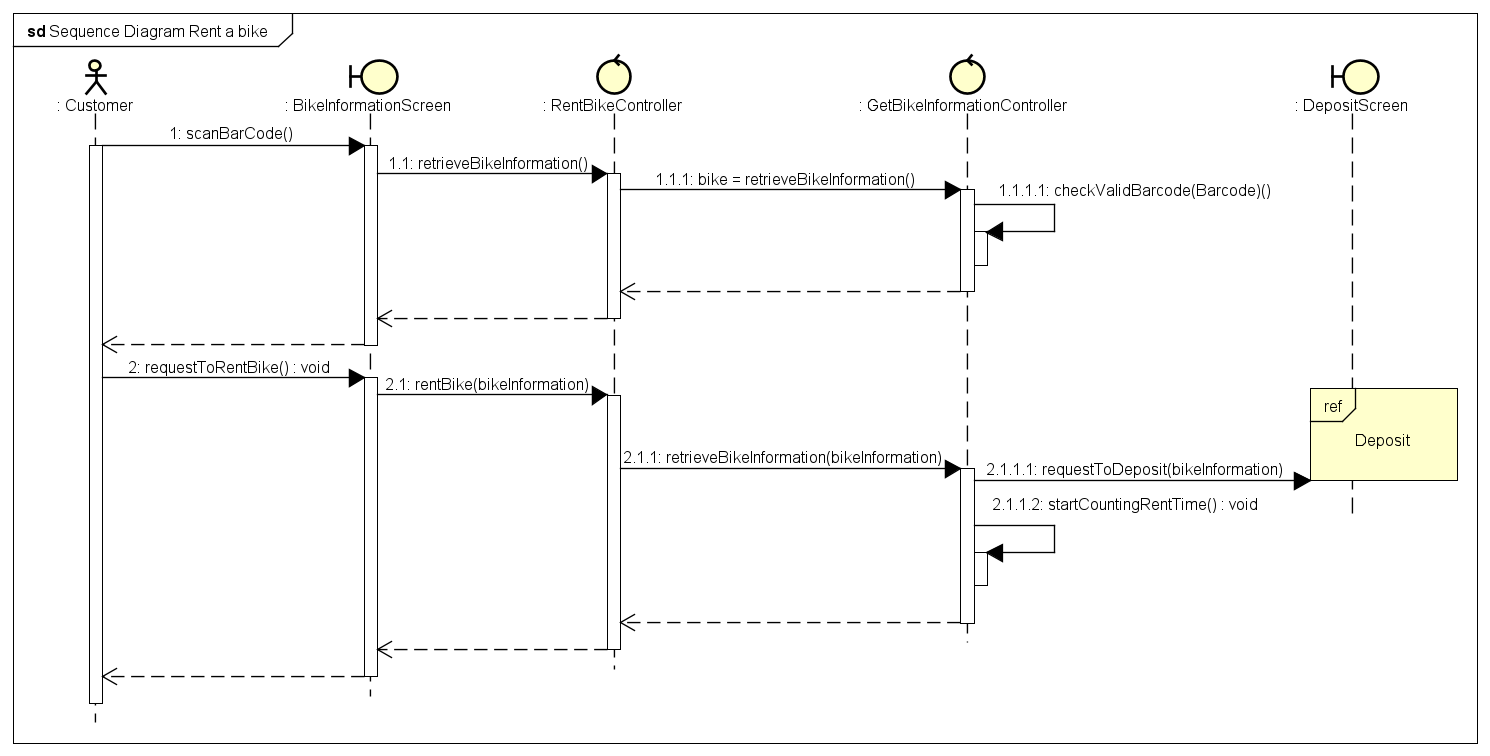


*Figure 3.5: Communication Diagram for Return Deposit Use Case*

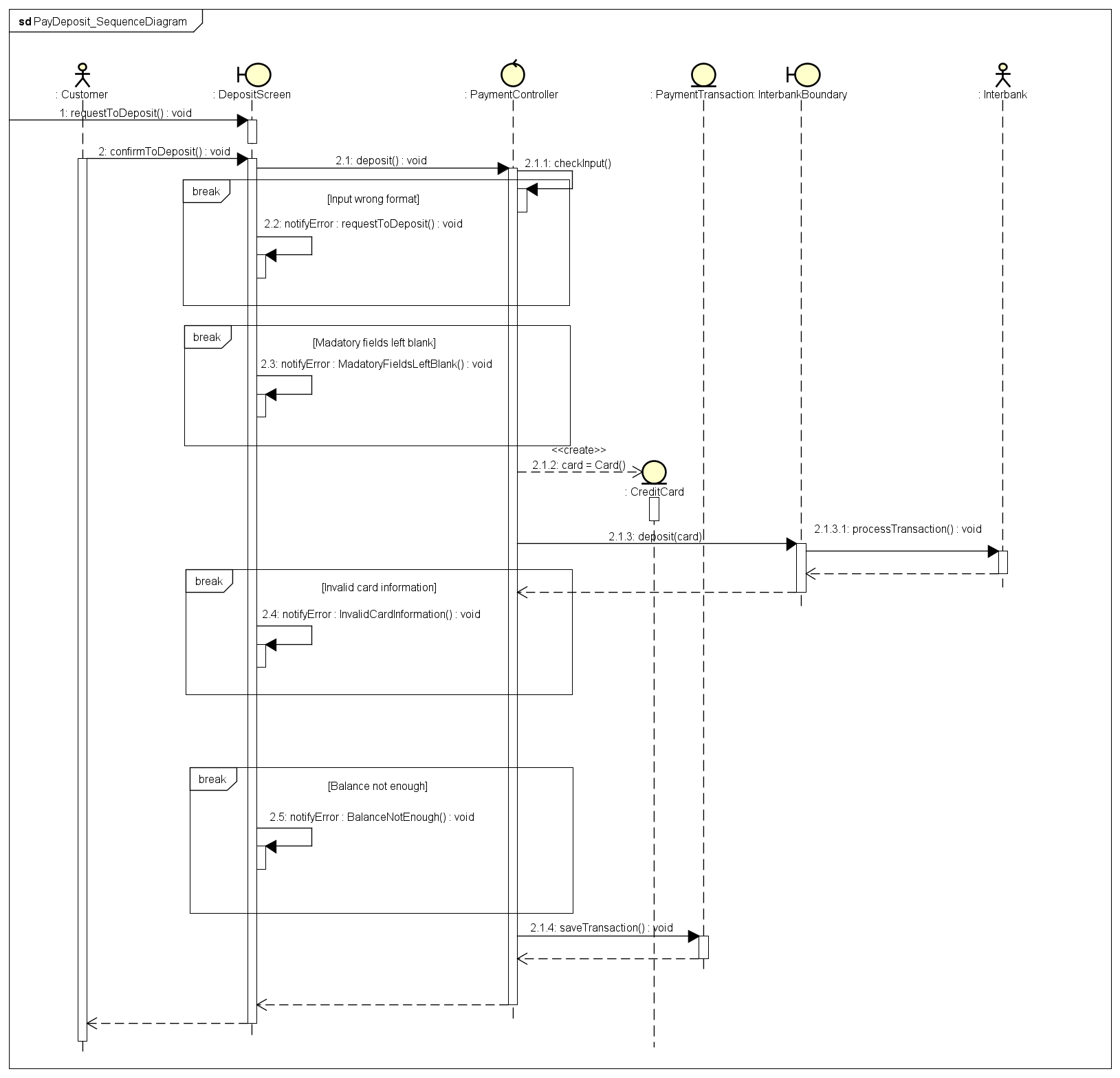


*Figure 3.6: Communication Diagram for Pay For Rental Use Case*

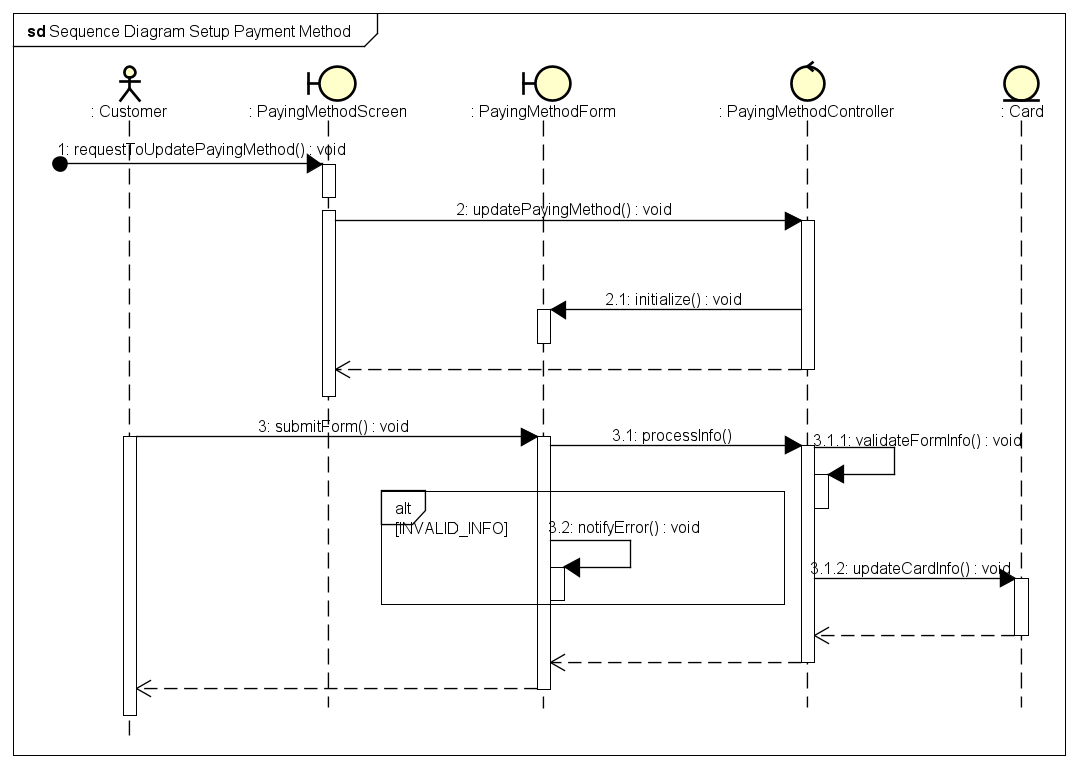
### 3.2.2. Sequence Diagrams



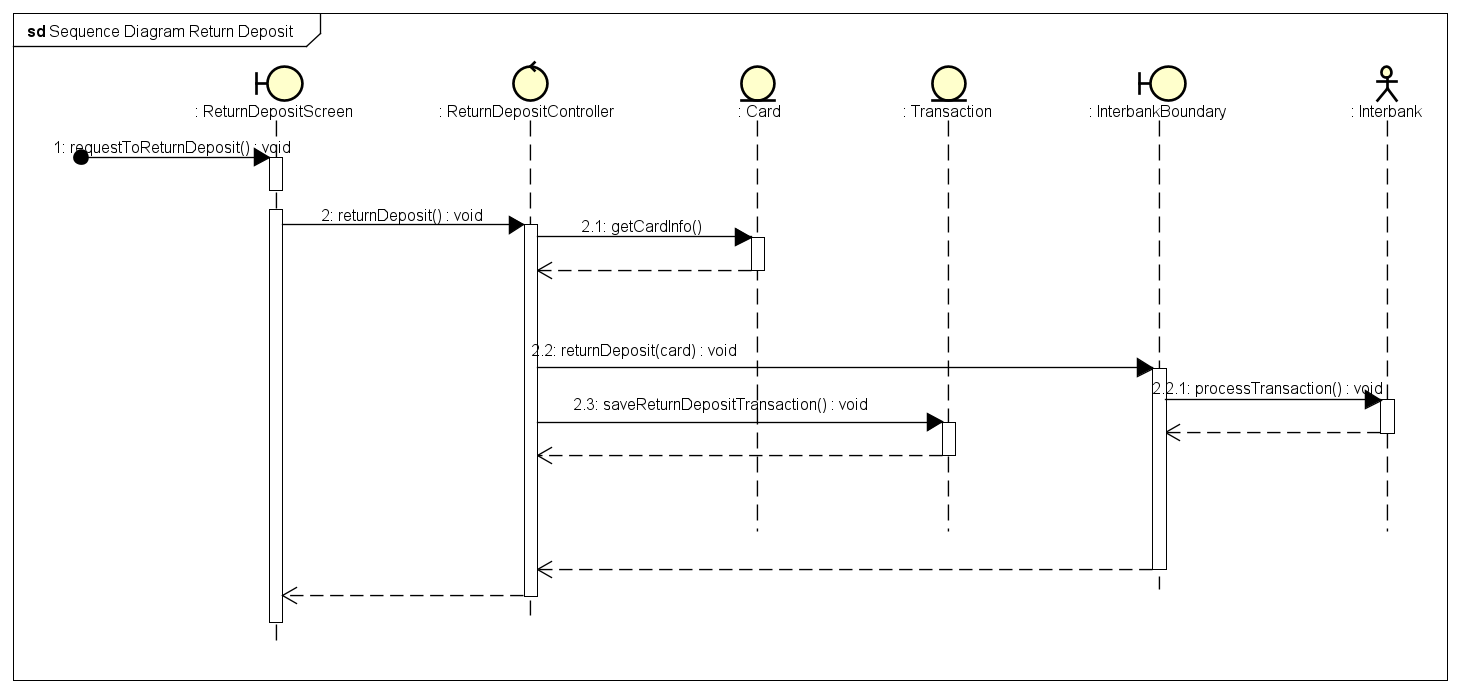
*Figure 3.7: Sequence Diagram for Rent A Bike Use Case*

**

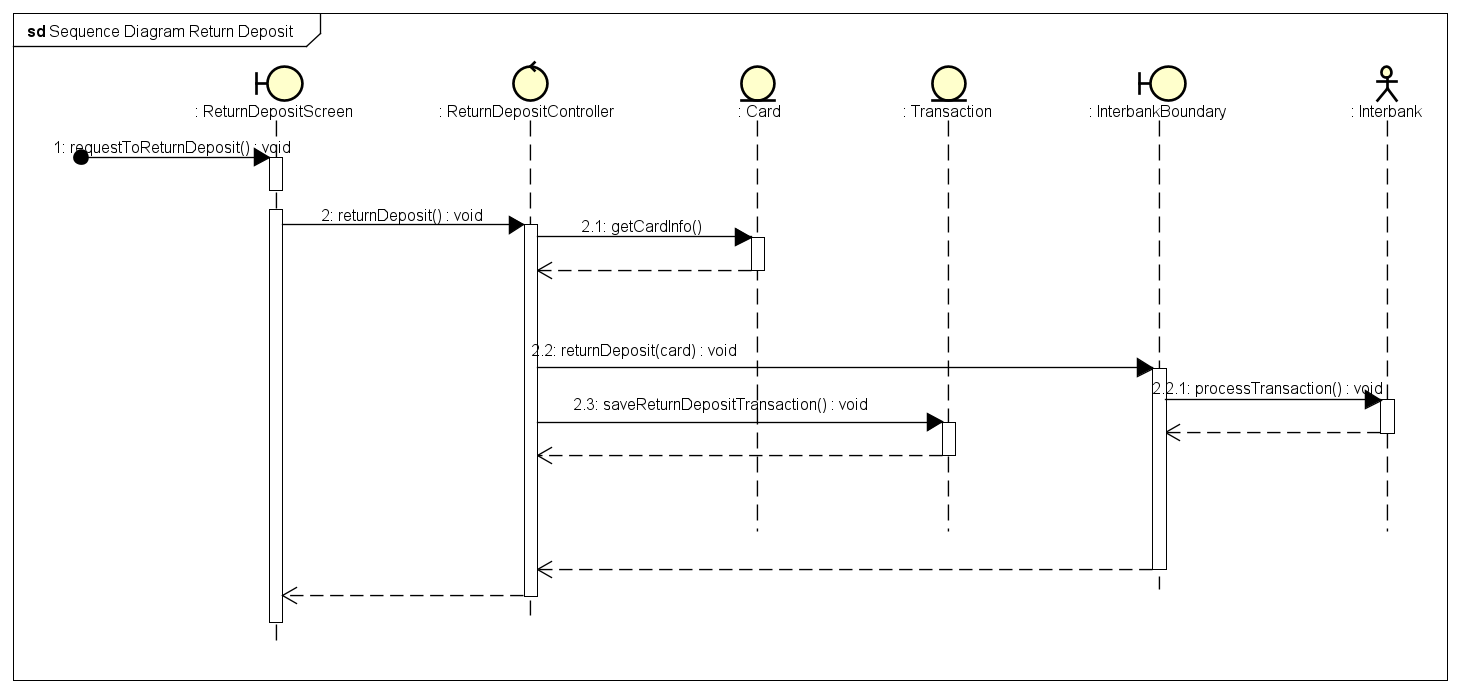
*Figure 3.8: Sequence Diagram for Deposit Use Case*

**

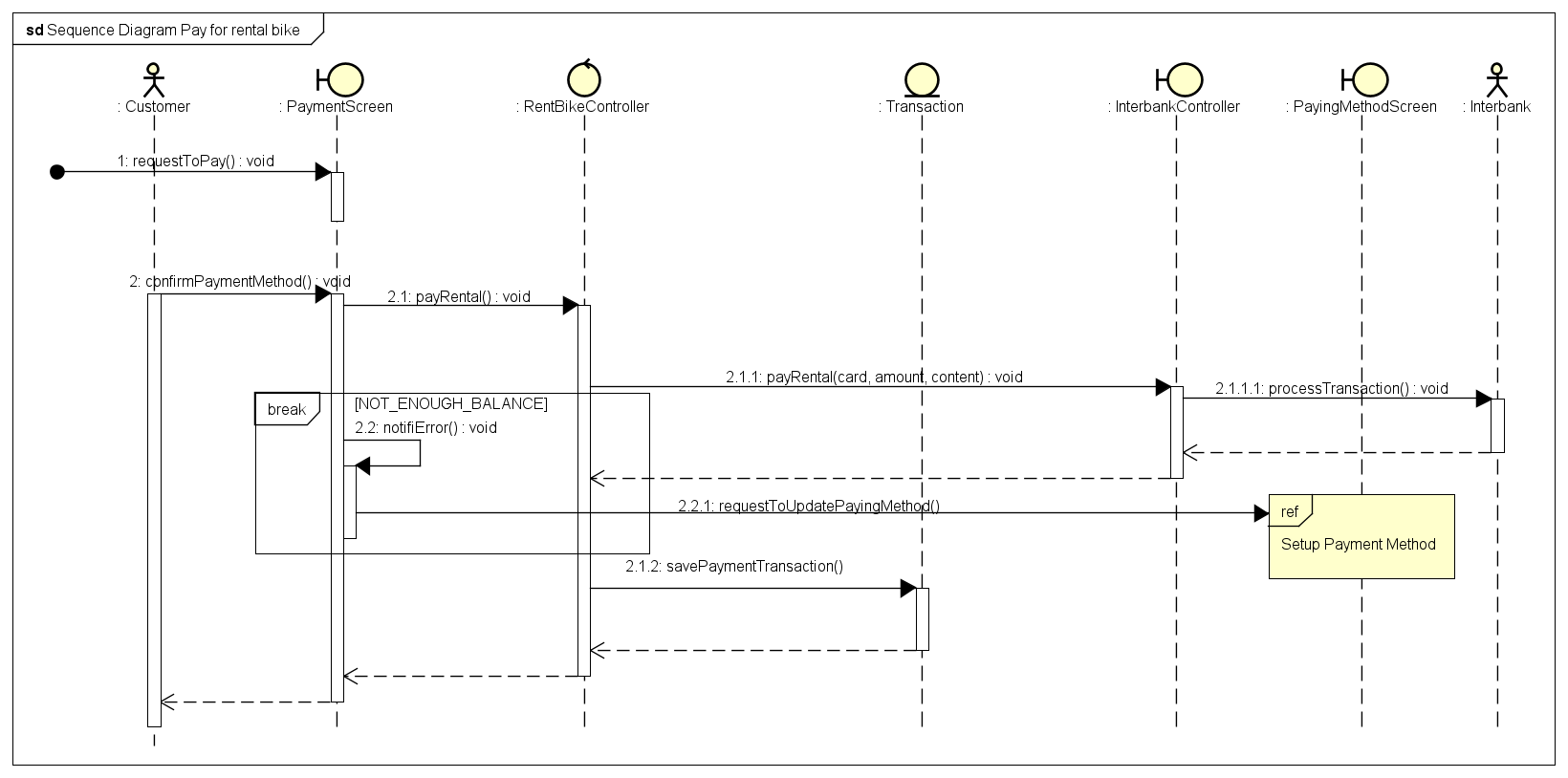
*Figure 3.9: Sequence Diagram for Setup Payment Method Use Case*

**

*Figure 3.10: Sequence Diagram for Return Bike Use Case*

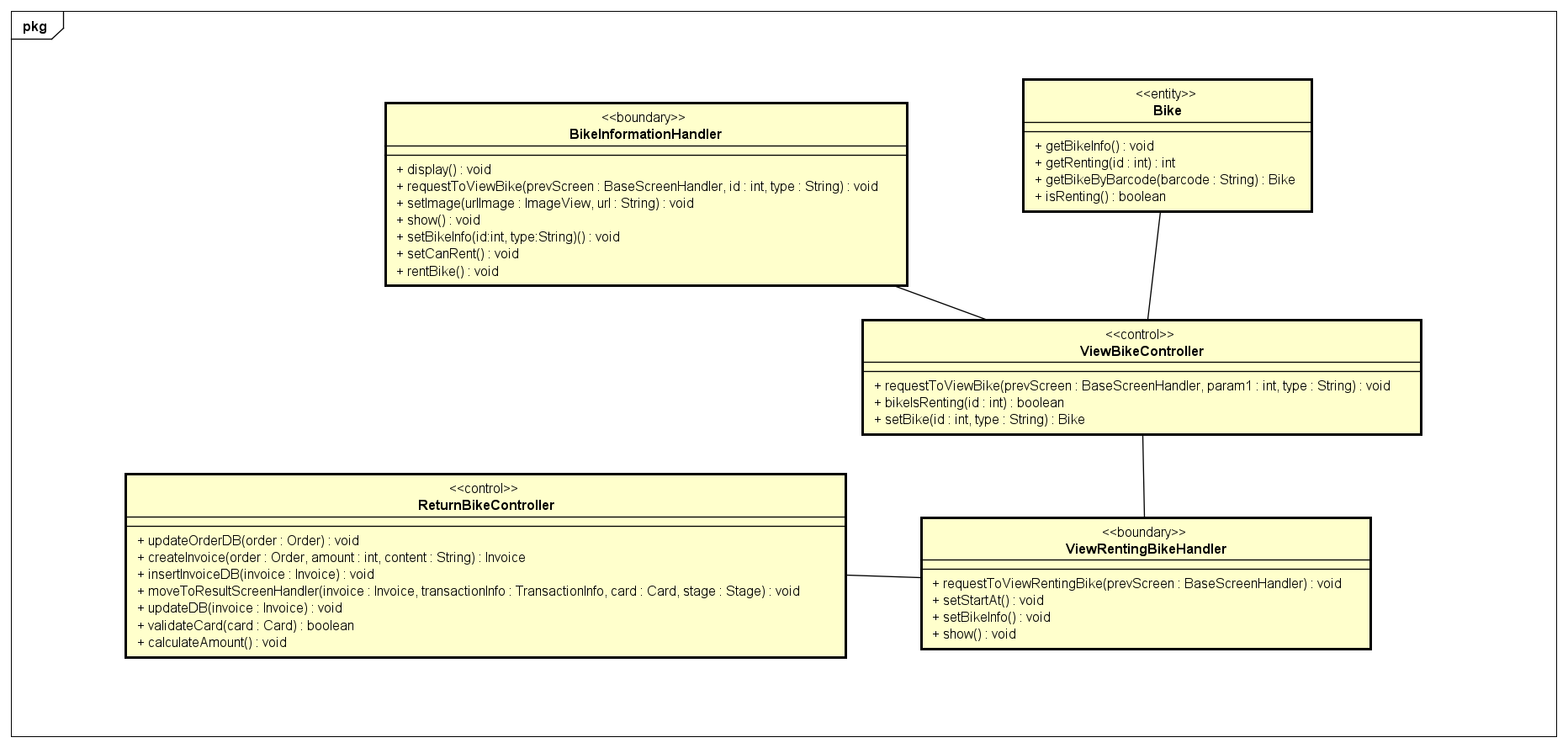
**

*Figure 3.11: Sequence Diagram for Return Deposit Use Case*

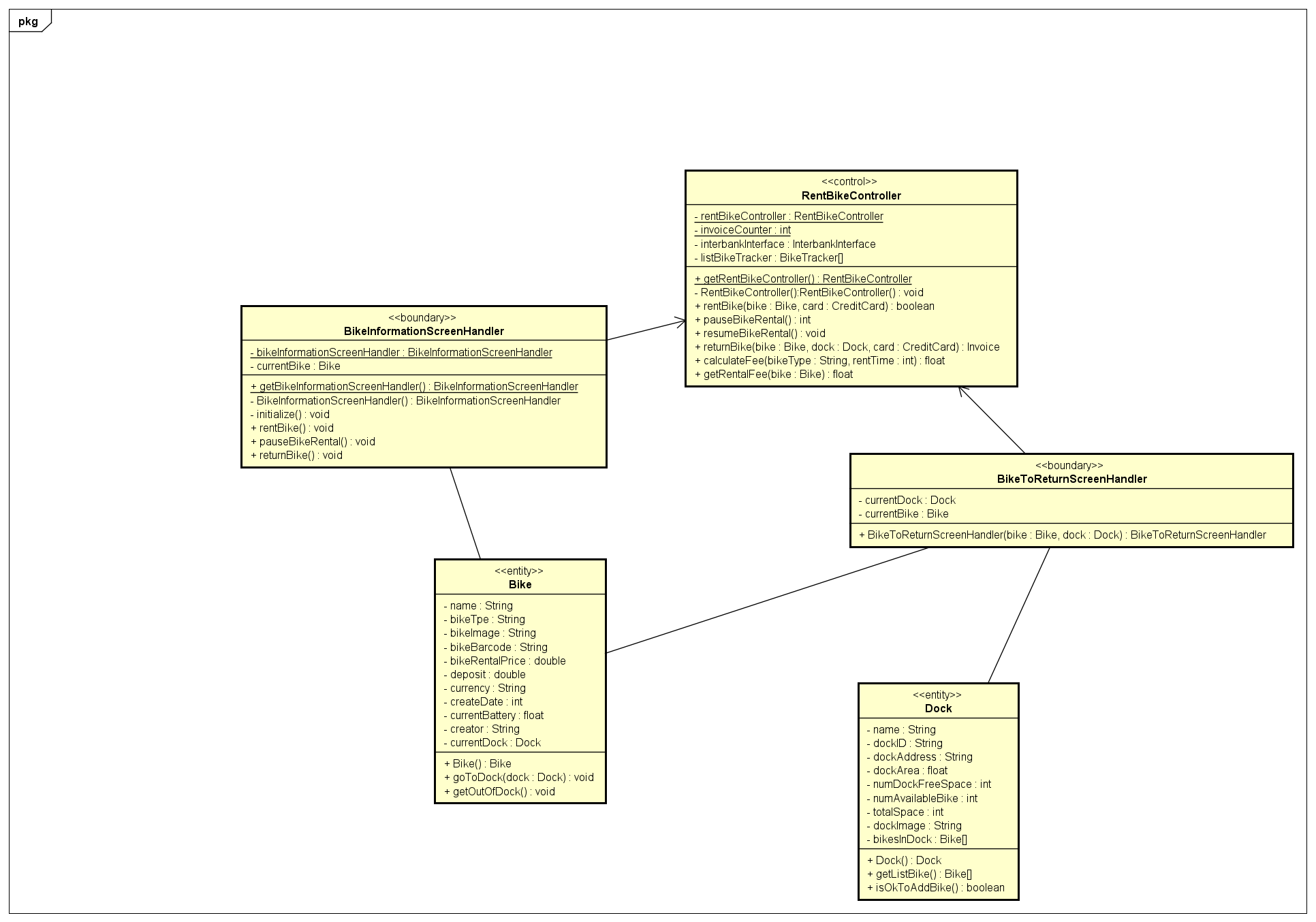
**

*Figure 3.12: Sequence Diagram for Pay For Rental Use Case*

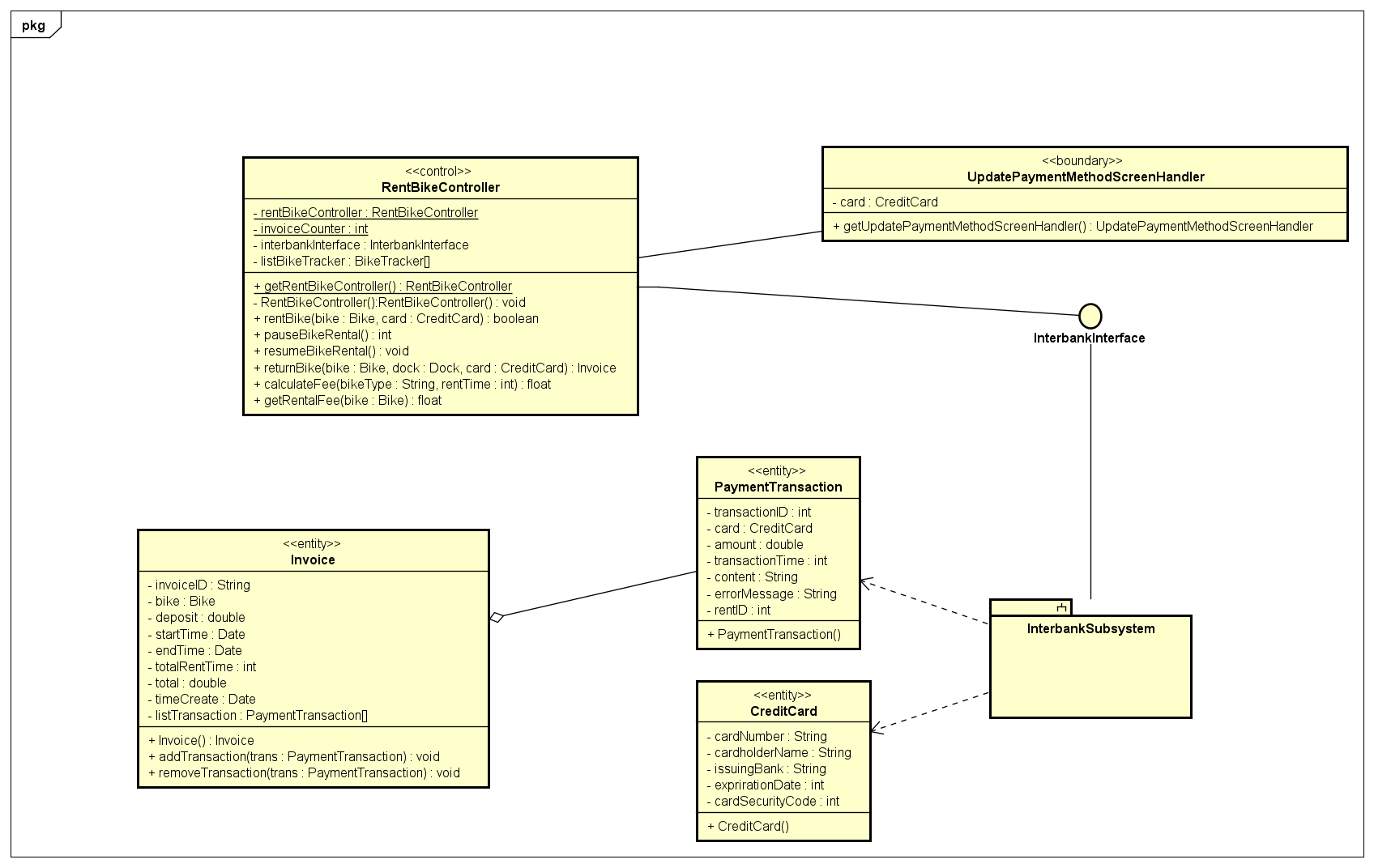
## 3.3. Analysis Class Diagrams



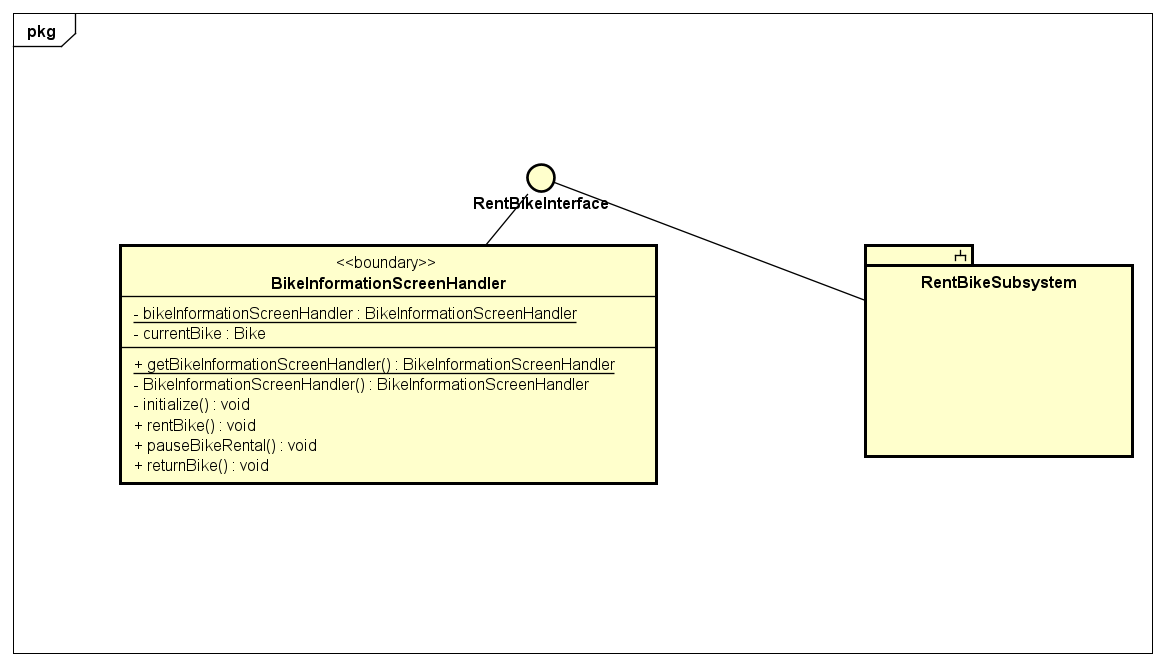
*Figure 3.13: Class Diagram for View Bike Use Case*

**

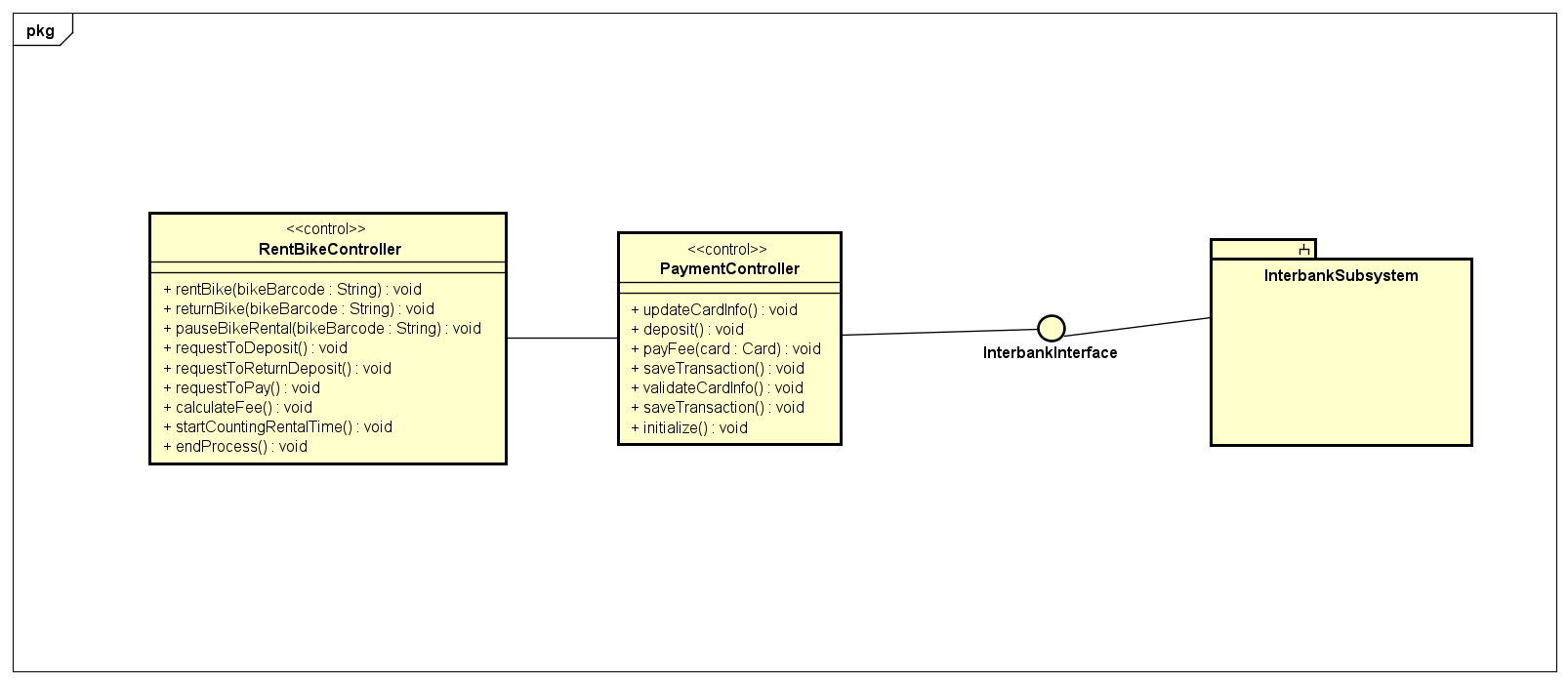
*Figure 3.14: Class Diagram for Rent Bike Use Case*

**

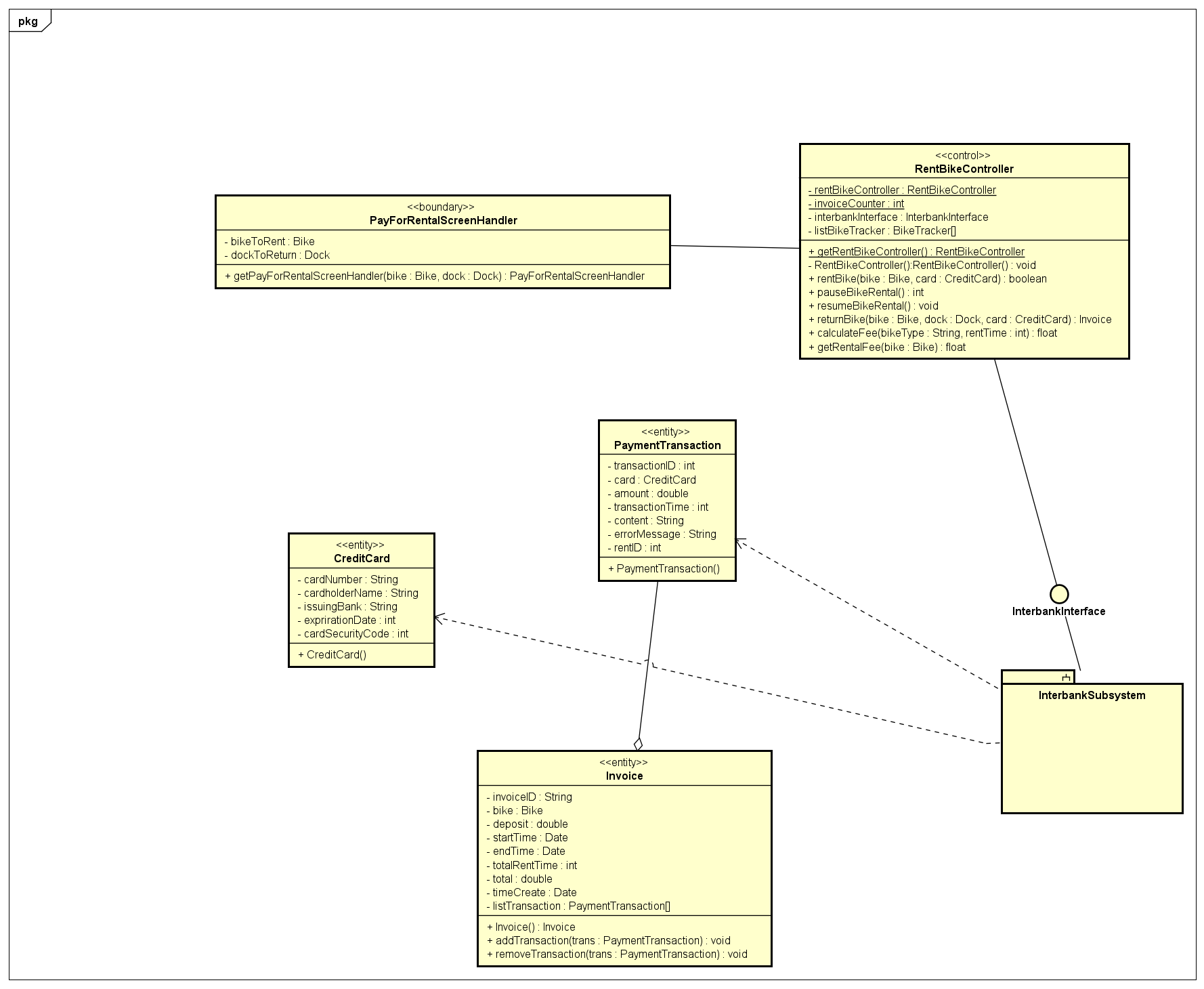
*Figure 3.15: Class Diagram for Deposit Use Case*

**

*Figure 3.16: Class Diagram for Return Bike Use Case*

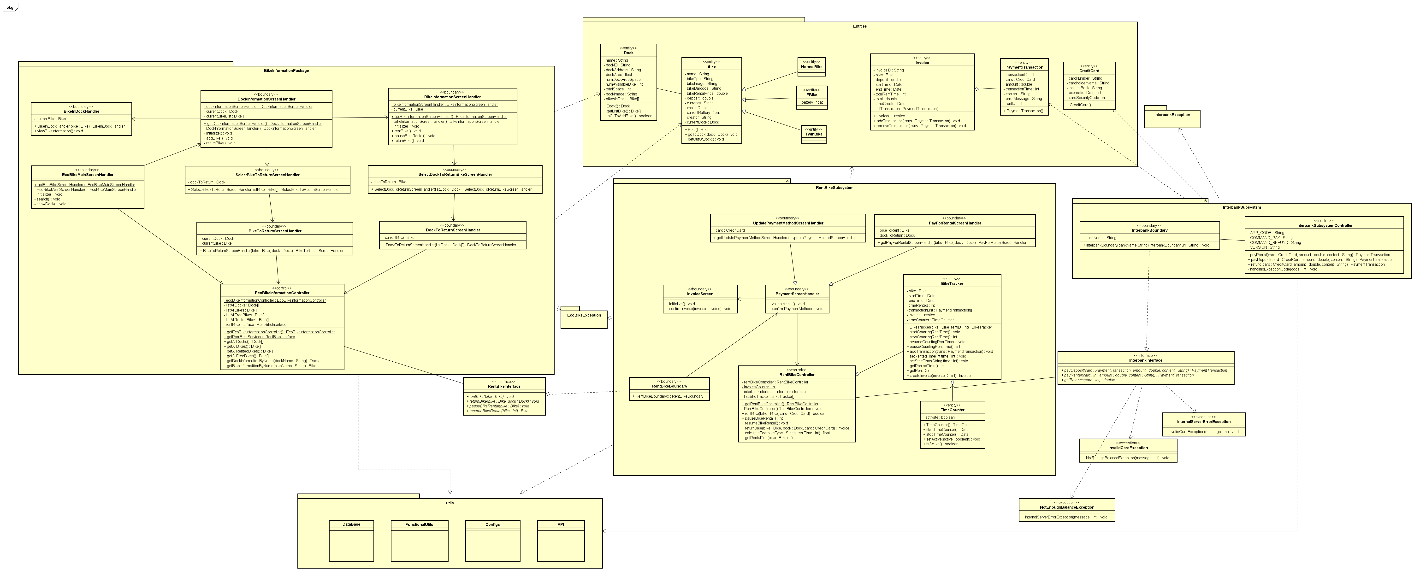
**

*Figure 3.17: Class Diagram for Return Deposit Use Case*

**

*Figure 3.18: Class Diagram for Pay Rental Use Case*

## 3.4. Unified Analysis Class Diagram



*Figure 3.19: Unified Class Diagram for EcoBike Application*

## 3.5. Security Software Architecture

In this project, we will not consider features such as user authentication (e.g., sign up, sign in, sign out), we only focus on features related to rent and return bikes.

# 4. Detailed Design

## 4.1. User Interface Design

### 4.1.1. Screen Configuration Standardization

Display

***Screen resolution:*** 1366x768px

***Number of colors supported:*** 16,177,216 colors

Screen

***Size:*** 1200 x 600px

***Main background color:*** #e5ffcc (R: 229, G: 255, B: 204)

***Location of buttons:*** Bottom center of the frame

***Logo:*** 100x100 px

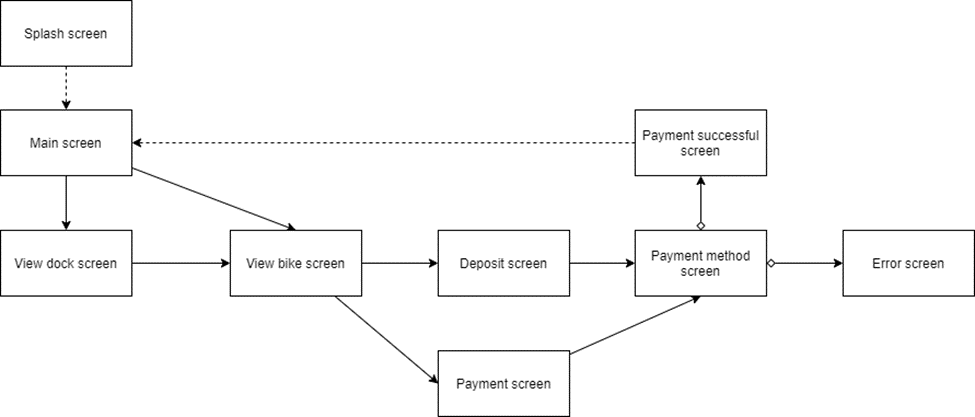
***Header logo:*** 100x100 px, located top left of the screen

***Header/Screen title:*** Segoe UI, Bold, 24px, Green

***Numbers:*** comma for thousand separation, dot for decimal separation

***Text:*** Segoe UI, size at most 24px

### 4.1.2. Screen Transition Diagrams



*Figure 4.1: Screen Transition Diagram for EcoBike Application*

### 4.1.3. Screen Specifications

#### 4.1.3.1. Splash Screen

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **EcoBike Software** | | **Date of creation** | **Approved by** | **Reviewed by** | **Person in charge** |
| **Screen specification** | **Splash screen** | 01/01/2023 |  |  | Bien |
|  | | **Control** | **Operation** | **Function** | |
| Main area | None | Introduce the application | |

*Table 4.1. Splash Screen Specification*

#### 4.1.3.2. Main Screen

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **EcoBike Software** | | **Date of creation** | **Approved by** | **Reviewed by** | **Person in charge** |
| **Screen specification** | **Main screen** | 01/01/2023 |  |  | Bien |
|  | | **Control** | **Operation** | **Function** | |
| Header logo | Click | Return immediately to main screen | |
| Search bar | Type, select & click | Type in information and select search type to search for docks or bikes | |
| Main area | Initial | Display map at current location of users and nearby docks in term. The pins of docks can be clicked to see docks details | |

*Table 4.2. Main Screen Specification*

#### 4.1.3.3. Dock Screen

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **EcoBike Software** | | **Date of creation** | **Approved by** | **Reviewed by** | **Person in charge** |
| **Screen specification** | **View Dock screen** | 01/01/2023 |  |  | Bien |
|  | | **Control** | **Operation** | **Function** | |
| Logo | Click | Return to the main screen immediately | |
| Header | Initial | Display title of screen | |
| Dock information | Initial | Display dock information | |
| Return bike | Click | Allow user to start return bike process at the dock | |
| Bike list | Click | Display brief details about bikes available in current dock. Allow choosing each bike to see detailed information | |

*Table 4.3. View Dock Screen Specification*

#### 4.1.3.4. Bike Screen

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **EcoBike Software** | | **Date of creation** | **Approved by** | **Reviewed by** | **Person in charge** |
| **Screen specification** | **View bike screen** | 01/01/2023 |  |  | Bien |
|  | | **Control** | **Operation** | **Function** | |
| Logo | Click | Return to the main screen immediately | |
| Header | Initial | Display title of screen | |
| Bike information | Initial | Display bike information | |
| Option pane | Click | Allow customer to perform renting, pause or return bike | |

*Table 4.4. View Bike Screen Specification*

#### 4.1.3.5. Payment Method Screen

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **EcoBike Software** | | **Date of creation** | **Approved by** | **Reviewed by** | **Person in charge** |
| **Screen specification** | **Paying method screen** | 01/01/2023 |  |  | Bien |
|  | | **Control** | **Operation** | **Function** | |
| Logo | Click | Return to the main screen immediately | |
| Header | Initial | Display title of screen | |
| Payment | Initial | Display information of paying method | |
| Button | Click | Allow customer confirm to the paying method | |

*Table 4.5. Payment Method Screen Specification*

#### 4.1.3.6. Deposit screen

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **EcoBike Software** | | **Date of creation** | **Approved by** | **Reviewed by** | **Person in charge** |
| **Screen specification** | **Payment screen** | 01/01/2023 |  |  | Bien |
|  | | **Control** | **Operation** | **Function** | |
| Logo | Click | Return to the main screen immediately | |
| Header | Initial | Display title of screen | |
| Information of payment | Initial | Display information of payment | |
| Button | Click | Allow customer confirm to deposit the bike | |

*Table 4.6. Deposit Screen Specification*

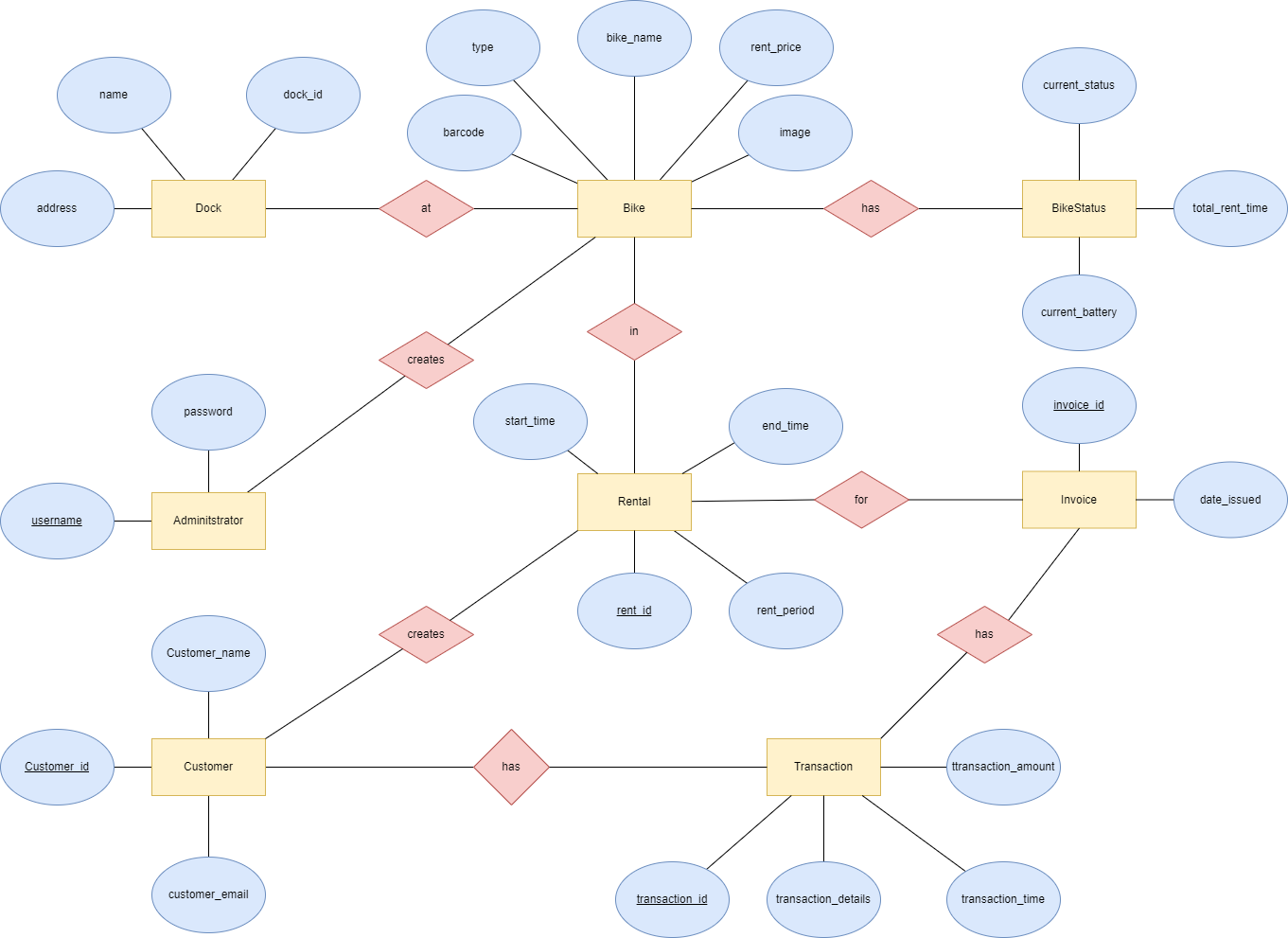
#### 4.1.3.7. Payment screen

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **EcoBike Software** | | **Date of creation** | **Approved by** | **Reviewed by** | **Person in charge** |
| **Screen specification** | **Payment screen** | 01/01/2023 |  |  | Bien |
|  | | **Control** | **Operation** | **Function** | |
| Logo | Click | Return to the main screen immediately | |
| Header | Initial | Display title of screen | |
| Information of payment | Initial | Display information of payment | |
| Buttons | Click | Allow customer confirm to pay or update card info | |

*Table 4.7. Payment Screen Specification*

## 4.2. Data Modeling

### 4.2.1. Conceptual Data Modeling

**

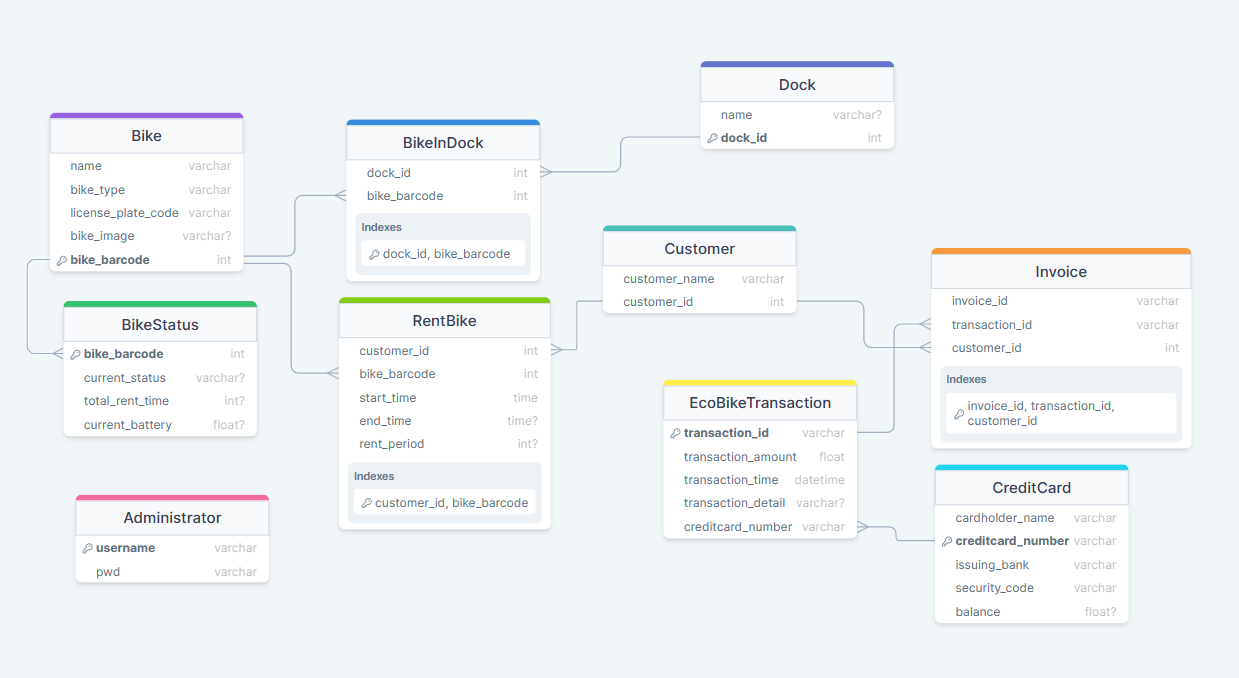
*Figure 4.2. ER Diagram for EcoBike Application*

### 4.2.2. Database Design

#### 4.2.2.1. Database Management System

Database Management System: SQLite

#### 4.2.2.2. Database Diagram



*Figure 4.3. Database Diagram for EcoBike Application*

#### 4.2.2.3. Database Detail Design

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***No.*** | ***PK*** | ***FK*** | ***Name*** | ***Data type*** | ***Mandatory*** | ***Description*** |
| *1* | *x* | *x* | *customer\_id* | *int* | *X* | *ID of customer* |
| *2* |  |  | *customer\_name* | *varchar(256)* | *x* | *Name of customer renting bike* |
| *3* |  |  | *customer\_email* | *varchar(256)* | *x* | *Email of customer renting bike for sending invoice* |

*Table 4.8. Customer table design*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***No.*** | ***PK*** | ***FK*** | ***Name*** | ***Data type*** | ***Mandatory*** | ***Description*** |
| *1* | *x* | *x* | *username* | *varchar(256)* | *x* | *Username of the administrator* |
| *2* |  |  | *pwd* | *varchar(256)* | *x* | *Password of the administrator used to login* |

*Table 4.9. Administrator table design*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***No.*** | ***PK*** | ***FK*** | ***Name*** | ***Data type*** | ***Mandatory*** | ***Description*** |
| *1* | *x* | *x* | *dock\_id* | *int* | *x* | *ID of dock* |
| *2* |  |  | *dock\_name* | *varchar(256)* | *x* | *Name of the dock* |
| *3* |  |  | *dock\_address* | *varchar(256)* | *x* | *Address of the dock* |
| *4* |  |  | *dock\_area* | *float* |  | *Area of the dock* |
| *5* |  |  | *num\_available\_bike* | *int* | *x* | *Number of current available bike in dock* |
| *6* |  |  | *num\_free\_dock* | *int* | *x* | *Number of current available bike slot in dock for returning bike* |

*Table 4.10. Dock table design*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***No.*** | ***PK*** | ***FK*** | ***Name*** | ***Data type*** | ***Mandatory*** | ***Description*** |
| *1* |  |  | *bike\_name* | *varchar(256)* | *x* | *Name of the bike* |
| *2* |  |  | *bike\_type* | *varchar(16)* | *x* | *Type of bike* |
| *3* |  |  | *license\_plate\_code* | *varchar(32)* | *x* | *Code of the license plate of the bike* |
| *4* |  |  | *bike\_image* | *varchar(256)* |  | *Path to image of the bike* |
| *5* | *x* | *x* | *bike\_barcode* | *int* | *x* | *Barcode of the bike* |
| *6* |  |  | *bike\_rental\_price* | *float* | *x* | *Price to rent the bike* |
| *7* |  |  | *deposit\_price* | *float* | *x* | *Deposit cost to rent the bike* |
| *8* |  |  | *currency\_unit* | *varchar(3)* | *x* | *Currency unit used to calculate rental fee and deposit fee* |
| *9* |  |  | *create\_date* | *date* | *x* | *Day imported bike data* |
| *10* |  |  | *creator* | *varchar(256)* | *x* | *The administrator who create data for the bike* |

*Table 4.11. Bike table design*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***No.*** | ***PK*** | ***FK*** | ***Name*** | ***Data type*** | ***Mandatory*** | ***Description*** |
| *1* |  |  | *dock\_id* | *int* | *x* | *Id of the dock* |
| *2* |  |  | *bike\_barcode* | *int* | *x* | *Barcode of the bike in dock* |

*Table 4.12. Bike In Dock table design*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***No.*** | ***PK*** | ***FK*** | ***Name*** | ***Data type*** | ***Mandatory*** | ***Description*** |
| *1* |  |  | *bike\_barcode* | *int* | *x* | *Barcode of the bike* |
| *2* |  |  | *current\_status* | *varchar(4)* | *x* | *‘free’/’rent’* |
| *3* |  |  | *total\_rent\_time* | *int* | *x* | *Total time that the bike is rented (in minute)* |
| *4* |  |  | *current battery* | *float* | *x* | *Current battery status of the bike* |

*Table 4.13. Bike Status table design*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***No.*** | ***PK*** | ***FK*** | ***Name*** | ***Data type*** | ***Mandatory*** | ***Description*** |
| *1* | *x* |  | *invoice\_id* | *int* | *X* | *ID of the invoice* |
| *2* |  | *x* | *transaction\_id* | *int* | *x* | *ID of the transaction* |
| *3* |  | *x* | *customer\_id* | *int* | *x* | *ID of the customer* |

*Table 4.14. Invoice table design*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***No.*** | ***PK*** | ***FK*** | ***Name*** | ***Data type*** | ***Mandatory*** | ***Description*** |
| *1* | *x* |  | *transaction\_id* | *int* | *x* | *ID of transaction* |
| *2* |  |  | *transaction\_amount* | *int* | *x* | *The amount of money for the transaction* |
| *3* |  |  | *transaction\_time* | *DATETIME* | *x* | *Time the transaction is made* |
| *4* |  |  | *transaction\_detail* | *varchar(256)* |  | *The content of the transaction* |
| *5* |  | *x* | *creditcard\_number* | *int* | *x* | *The number of the credit card* |

*Table 4.15. Transaction table design*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***No.*** | ***PK*** | ***FK*** | ***Name*** | ***Data type*** | ***Mandatory*** | ***Description*** |
| *1* | *x* | *x* | *customer\_id* | *int* | *X* | *ID of customer* |
| *2* | *x* |  | *bike\_barcode* | *int* | *x* | *Barcode of the bike being rented* |
| *3* |  |  | *start\_time* | *time* | *x* | *Time start renting* |
| *4* |  |  | *end\_time* | *time* |  | *Time end renting (null if the bike is currently being rented)* |
| *5* |  |  | *rent\_period* | *int* |  | *Total time renting the bike, in terms of minutes (null if the bike is currently being rented)* |

*Table 4.16. Rent Bike table design*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***No.*** | ***PK*** | ***FK*** | ***Name*** | ***Data type*** | ***Mandatory*** | ***Description*** |
| *1* | *x* | *x* | *card\_number* | *int* | *X* | *Number of the credit card* |
| *2* |  |  | *cardholder\_name* | *varchar(256)* | *x* | *Name of the cardholder* |
| *3* |  |  | *issuing\_bank* | *varchar(256)* | *x* | *Bank in charge of the card* |
| *4* |  |  | *security\_code* | *varchar(16)* | *x* | *Security code on the credit card for transaction* |
| *5* |  |  | *balance* | *float* | *x* | *Current balance of the credit card* |

*Table 4.17. Credit Card table design*

Database script:

create table Administrator(username varchar(256) not null primary key,

pwd varchar(256) not null);

create table Bike(name varchar(256) not null,

bike\_type varchar(16) not null,

license\_plate\_code varchar(32) not null,

bike\_image varchar(256),

bike\_barcode int not null primary key identity(1,1),

bike\_rental\_price float not null,

currency\_unit varchar(3) not null,

create\_date date,

creator varchar(256),

constraint FK\_Bike\_Creator foreign key (creator) references Administrator(username));

create table Dock(name varchar(256),

dock\_id int not null primary key identity(1,1),

dock\_address varchar(256),

dock\_area float,

num\_available\_bike int,

num\_free\_dock int);

create table BikeInDock(dock\_id int not null,

bike\_barcode int not null,

constraint PK\_Bike\_In\_Dock primary key (dock\_id, bike\_barcode),

constraint FK\_BikeInDock\_Dock foreign key (dock\_id) references Dock(dock\_id),

constraint FK\_BikeInDock\_Bike foreign key (bike\_barcode) references Bike(bike\_barcode));

create table Customer(customer\_name varchar(256) not null,

customer\_id int not null identity(1,1) primary key,

customer\_email varchar(128) not null);

create table CreditCard(cardholder\_name varchar(256) not null,

creditcard\_number varchar(25) not null primary key,

issuing\_bank varchar(128) not null,

security\_code varchar(8) not null,

balance float,

constraint Check\_CardBalance check (balance >=0));

create table RentBike(customer\_id int not null,

bike\_barcode int not null,

start\_time time not null,

end\_time time,

rent\_period int,

constraint PK\_Rent\_Bike primary key (customer\_id, bike\_barcode),

constraint FK\_RentBike\_Bike foreign key (bike\_barcode) references Bike(bike\_barcode),

constraint FK\_RenBike\_Customer foreign key (customer\_id) references Customer(customer\_id),

constraint Check\_RenBike\_Time check (end\_time > start\_time));

create table BikeStatus(bike\_barcode int not null primary key,

current\_status varchar(4),

total\_rent\_time int,

current\_battery float,

constraint FK\_BikeStatus\_Barcode foreign key (bike\_barcode) references Bike(bike\_barcode),

constraint Check\_BikeStatus\_Total\_Rent\_Time check (total\_rent\_time >=0),

constraint Check\_BikeStatus\_Battery check (current\_battery >=0),

constraint Check\_BikeStatus\_Status check (current\_status = 'free' or current\_status = 'rent'));

create table EcoBikeTransaction(transaction\_id varchar(32) not null primary key,

transaction\_amount float not null,

transaction\_time datetime not null,

transaction\_detail varchar(256),

creditcard\_number varchar(25) not null,

constraint FK\_Transaction\_CreditCard foreign key (creditcard\_number) references CreditCard(creditcard\_number),

constraint Check\_Amount check (transaction\_amount > 0));

create table Invoice(invoice\_id varchar(256) not null,

transaction\_id varchar(32) not null,

customer\_id int not null,

constraint PK\_InvoiceTransactionCustomer primary key (invoice\_id, transaction\_id, customer\_id),

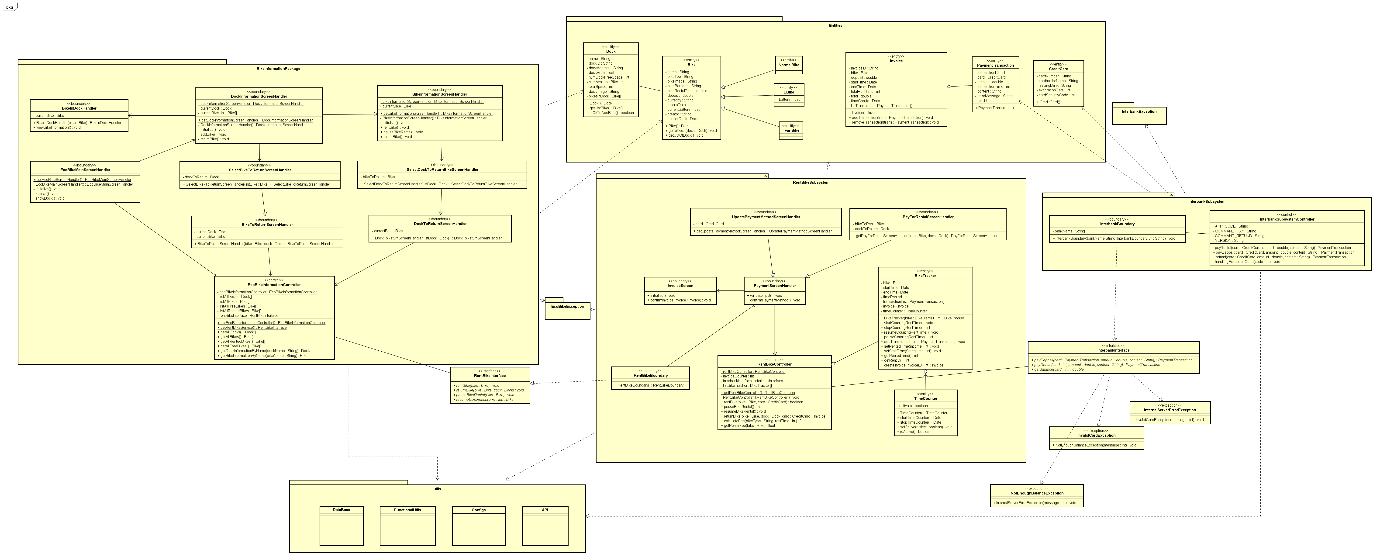
constraint FK\_Invoice\_Transaction foreign key (transaction\_id) references EcoBikeTransaction(transaction\_id),

constraint FK\_Invoice\_Customer foreign key (customer\_id) references Customer(customer\_id));

## 4.3. Non-Database Management System Files

## 4.4. Class Design

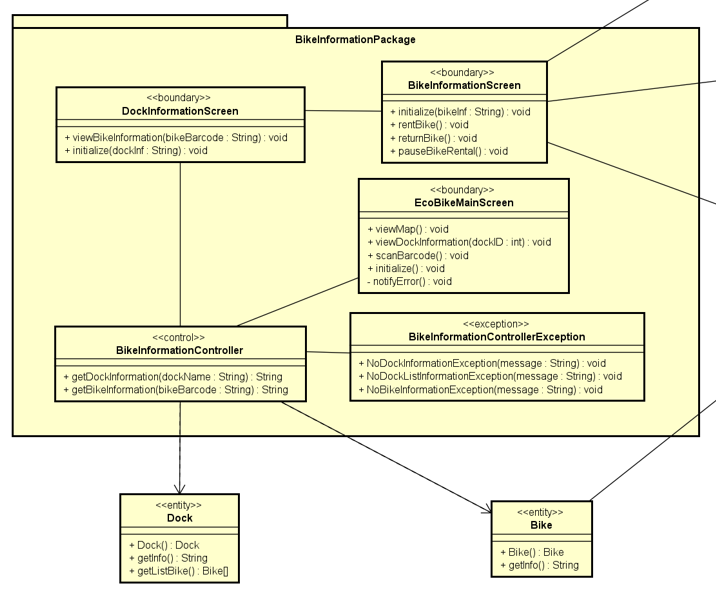
### 4.4.1. General Class Diagram



*Figure 4.4. General Class Diagram for EcoBike Application*

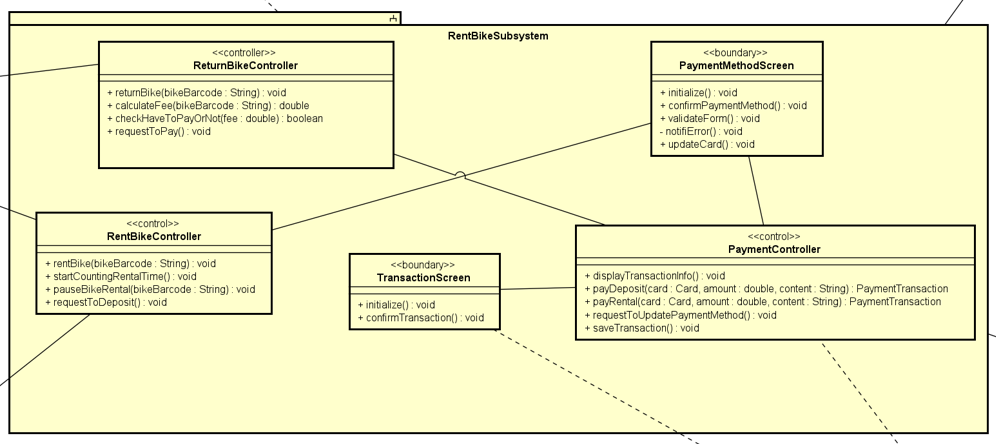
### 4.4.2. Class Diagrams

#### 4.4.2.1. Class Diagram for Package BikeInformation



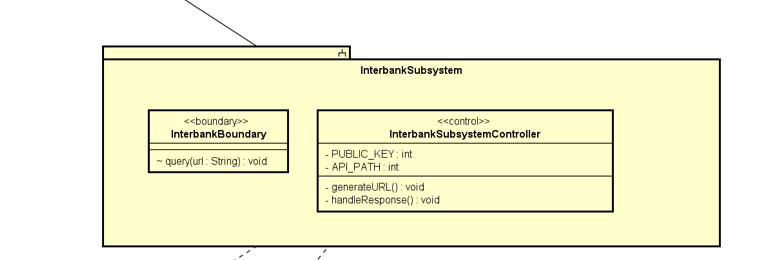
*Figure 4.5. Class Diagram for Package BikeInformation*

#### 4.4.2.2. Class Diagram for Subsystem RentBike



*Figure 4.6. Class Diagram for Subsystem RentBike*

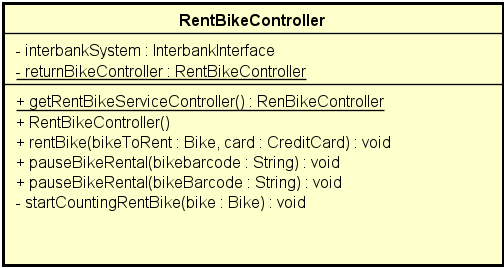
#### 4.4.2.3. Class Diagram for Subsystem InterBank



*Figure 4.7. Class Diagram for Subsystem InterBank*

### 4.4.3. Class Design

#### 4.4.3.1. Class RentBikeController



*Figure 4.8. RentBikeController Class Diagram*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Name* | *Data type* | *Default value* | *Description* |
| 1 | interbankSystem | InterbankInterface | InterbankInterface | Interbank to proceed transaction |
| 2 | rentBikeController | RentBikeController | null | static instance of the RentBikeController |

*Table 4.18. RentBikeController attributes*

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | getRentBikeServiceController | RentBikeController | return static instance rentBikeController of class RentBikeController |
| 2 | rentBike | void | start renting proceed |
| 3 | pauseBikeRental | void | pause counting rental time |
| 4 | startCountingRentBike | void | start counting renal time |

*Table 4.19. RentBikeController operations*

*Parameter*:

* bikeBarcode: bar code of the bike to rent
* card: the credit card to perform transaction (deposit)
* bikeToRent: the bike entities represent the bike to be rented

*Exception*:

* IOException
* RentBikeException If the bike is not currently available, the barcode is not valid
* EcoBikeUndefinedException If there is an unexpected error occurs during the renting process

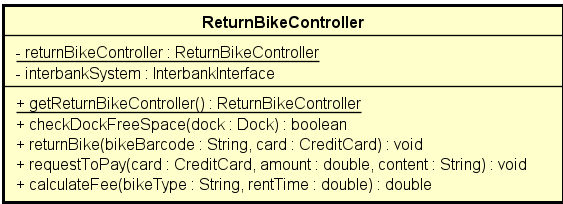
*Method:*

None

*State*

None

#### 4.4.3.2. Class ReturnBikeController



*Figure 4.9. ReturnBikeController Class Diagram*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Name* | *Data type* | *Default value* | *Description* |
| 1 | returnBikeController | ReturnBikeController | null | the static instance of ReturnBikeController |
| 2 | interbankSystem | InterbankInterface | InterbankInterface | the interbank to proceed transaction |

*Table 4.20. ReturnBikeController attributes*

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | getReturnBikeController | ReturnBikeController | get the static instance of ReturnBikeController |
| 2 | checkDockSpace | boolean | check whether the dock have empty slots |
| 3 | returnBike | void | proceed return bike process |
| 4 | requestToPay | void | proceed pay for rental process |
| 5 | calculateFee | double | return the fee of the rental process |

*Table 4.21. ReturnBikeController operations*

*Parameter*:

* dock: the entity represent a dock
* bikeBarcode: bike bar code represent as a string
* card: the entity represent a card
* bikeType: type of current bike

*Exception*:

* RentBikeException If the bike is not currently available, the barcode is not valid
* EcoBikeUndefinedException If there is an unexpected error occurs during the renting process

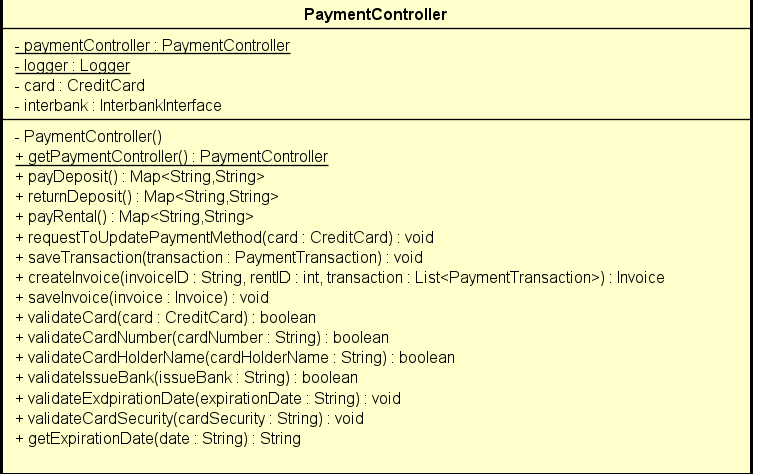
*Method*

None

*State*

None

#### 4.4.3.3. Class PaymentController



*Figure 4.10. PaymentControllerClass Diagram*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Name* | *Data type* | *Default value* | *Description* |
| 1 | paymentController | PaymentController | null | static instance of PaymentController |
| 2 | logger | Logger | Logger | logger to log interaction |
| 3 | card | CreditCard | null | credit card to perform transaction |
| 4 | interbank | Interbank | Interbank | interbank to perform transaction |

*Table 4.22. PaymentController attributes*

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | payDeposit | Map<String, String> | start pay deposit process |
| 2 | returnDeposit | Map<String, String> | start return deposit process |
| 3 | payRental | Map<String, String> | start pay rental process |
| 4 | requestToUpdatePaymentMethod | void | start request to update payment method process |
| 5 | saveTransaction | void | save current transaction |
| 6 | createInvoice | Invoice | create invoice for current transaction |
| 7 | saveInvoice | void | save current invoice |

*Table 4.23. PaymentController operations*

*Parameter*:

* transaction: the entity represent a transaction
* invoice: the entity represent a invoice
* card: the entity represent a card

*Exception*:

* EcoBikeUndefinedException If there is an unexpected error occurs during the renting process
* SQLException if there is an unexpected error with the database

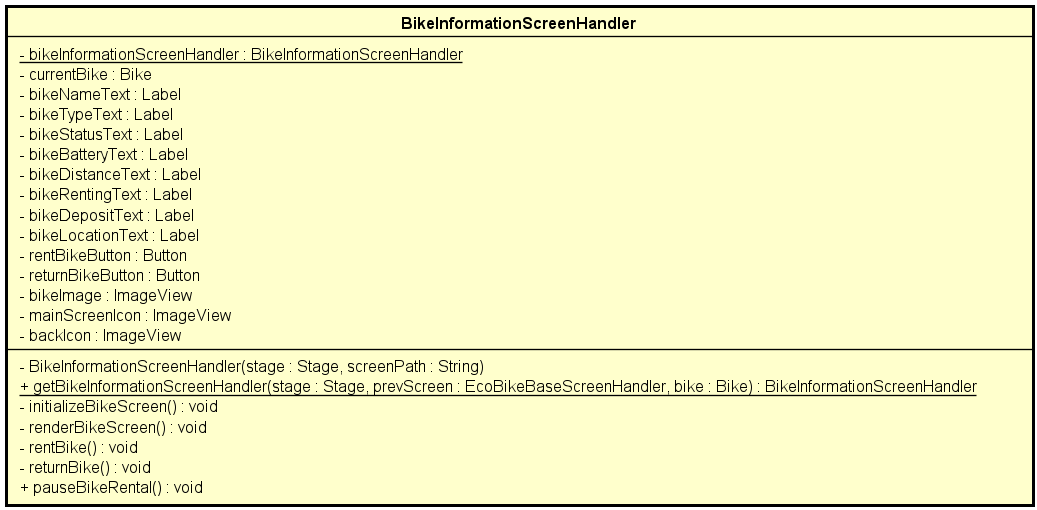
*Method*

None

*State*

None

#### 4.4.3.4. Class BikeInformationScreenHandler



*Figure 4.11. BikeInformationScreenHandler Class Diagram*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Name* | *Data type* | *Default value* | *Description* |
| 1 | currentBike | Bike | null | the bike the screen is showing |
| 2 | bikeNameText | Label | null | label of the name of the bike |
| 3 | bikeTypeText | Label | null | label of the type of the bike |
| 4 | bikeStatusText | Label | null | label of the status of the bike |
| 5 | bikeBatteryText | Label | null | label of the battery of the bike |
| 6 | bikeDistanceText | Label | null | label of the estimation distance that the bike can travel |
| 7 | bikeRentingText | Label | null | label of the renting price per hour of the bike |
| 8 | bikeDepositText | Label | null | label of the amount of money needed to deposit to rent the bike |
| 9 | bikeLocationText | Label | null | label of the location of the bike |
| 10 | returnBikeButton | Button | null | button to return this bike |
| 11 | rentBikeButton | Button | null | button to rent this bike |

*Table 4.24.* BikeInformationScreenHandler *operations*

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | renderBikeScreen | void | render the screen with information of the bike |
| 2 | rentBikeS | void | start rent bike process |
| 3 | returnBike | void | start return bike process |

*Table 4.25.* BikeInformationScreenHandler *operations*

*Parameter*:

None

*Exception*:

* IOException if there is unexpected error with the IO

*Method*

None

*State*

None

# 5. Design Considerations

## 5.1. Goals and Guidelines

Goals:

* Provide a user-friendly application
* Provide an eye-catching interface and convenient experience for users
* The response time for the system is 1 second at normal and 2 seconds during a peak load

Guidelines:

* Obligate the coding convention in Java, and OOP principles.
* Avoid hard-coding
* Write comments for codes
* Structure the doc for maintenance

## 5.2. Architectural Strategies

Our intention is to reuse components

* Programming Language: Java
* Database: MySQL
* UML: Astah
* GUI: Scene Builder

We’re always looking toward minimizing the memory and space usage; reduce the complexity to speed up the response time, and improve the performance. We’re also concerned about the maintenance. For the future, we’re looking forward to updating the system, integrating new features such as admin to manage the crud, the statistics, the profit.

## 5.3. Coupling and Cohesion

### 5.3.1. Coupling

#### 5.3.1.1. Content coupling

|  |  |  |
| --- | --- | --- |
| Related modules | Description | Improvement |
| No related module | Our modules are self-contained and don’t rely on other modules to operate | No improvement |

#### 5.3.1.2. Common coupling

|  |  |  |
| --- | --- | --- |
| Related modules | Description | Improvement |
| No related module | We only use static with Singleton pattern to share the controller instance between boundaries to control the flow of the programs.  Some constants exist in the system, but only with careful usage shared between the related modules | No improvement |

#### 5.3.1.3. Control coupling

|  |  |  |
| --- | --- | --- |
| Related modules | Description | Improvement |
| No related module | Our methods are designed to carry out only one specific task, so no control coupling existed | No improvement |

#### 5.3.1.4. Stamp coupling

|  |  |  |
| --- | --- | --- |
| Related modules | Description | Improvement |
| RentBikeServiceController | In module RentBikeServiceController, the Bike entities was used as an argument for the calculateFee method, which only need bikeType and totalRentTime as arguments | Fix the method to accept only needed arguments instead of the accepting Bike entities as the argument |

#### 5.3.1.5. Data coupling

|  |  |  |
| --- | --- | --- |
| Related modules | Description | Improvement |
| Controllers and Boundaries modules | Boundaries need data to render GUI, which is acceptable | No improvement |

### 5.3.2. Cohesion

#### 5.3.2.1. Coincidental cohesion

|  |  |  |
| --- | --- | --- |
| Related modules | Description | Improvement |
| No module | The only visible coincidental cohesion in our project might be the class Configs, which contains some constant share between some controllers and entities | No improvement |

#### 5.3.2.2. Logical cohesion

|  |  |  |
| --- | --- | --- |
| Related modules | Description | Improvement |
| No module |  | No improvement |

#### 5.3.2.3. Temporal cohesion

|  |  |  |
| --- | --- | --- |
| Related modules | Description | Improvement |
| Controller  View, | In our project, we put all controllers into a Controller package, screen handlers into a View package, which might be considered temporal cohesion | No improvement |

#### 5.3.2.4. Procedure cohesion

|  |  |  |
| --- | --- | --- |
| Related modules | Description | Improvement |
| RentBikeServiceCon-troller | Consist of validating methods | No improvement |

#### 5.3.2.5. Communicational cohesion

|  |  |  |
| --- | --- | --- |
| Related modules | Description | Improvement |
| No module |  | No improvement |

#### 5.3.2.6. Sequential cohesion

|  |  |  |
| --- | --- | --- |
| Related modules | Description | Improvement |
| No module |  | No improvement |

#### 5.3.2.7. Information cohesion

|  |  |  |
| --- | --- | --- |
| Related modules | Description | Improvement |
| DBUtils  JSONUtils | All methods are to perform database queries or manipulate json string | No improvement |

#### 5.3.2.8. Functional cohesion

|  |  |  |
| --- | --- | --- |
| Related modules | Description | Improvement |
| Most of the modules |  | No improvement |

In our software design, we detect that there are still some components that have Control Coupling and Communicational Cohesion problems.

We are trying our best to resolve these problems, decrease Coupling level and increase Cohesion level. However, due to lack of time, we might not be able to fix this before the announced deadline.

## 5.4. Design Principles

We design simple classes follow SOLID principles that means:

* A class should have only one job, one responsibility.
* Software entities are open for extension but close for modification.
* We also use interfaces, abstract classes. So, subclasses should be substitutable for their base classes.
* Use specific interfaces if necessary instead of using general purpose interfaces which do not use.
* We put all classes with the same properties into one package to manage easily. Therefore, we can reuse source code, adapt to any changing requirements.

## 5.5. Design Patterns

Facade pattern:

* We use InterbankInterface for communication between software and interbank subsystem. It decrease the overall complexity of our application and provides an easier interface for communication

Singleton pattern:

* We use Singleton pattern for screen handler so that we do not need to create a new instance of screen handler each time we change the screen

Observer pattern:

* The observer pattern is used to observe the change in the dock so each time a bike is rented  
  or returned, the GUI can be updated accordingly without having to query the database. We  
  only need to store the status back to the database.