HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

School of Information and communications technology

Software Design Document

Version 1.3

EcobikeRental

Subject: Software Design and Construction

Group 17

Student Name StudentID

Tran Thi Hong Nhung 20183965

Vu Thi Ngoc Lan 20183939

Duong Hue Linh 20183942

*Hanoi, 10-2021*

Table of Contents

Table of Contents 1

1 Introduction 5

1.1 Objective 5

1.2 Scope 5

1.3 Glossary 5

1.4 References 5

2 Overall Description 7

2.1 General Overview 7

2.2 Assumptions/Constraints/Risks 7

2.2.1 Assumptions 7

2.2.2 Constraints 7

2.2.3 Risks 8

3 System Architecture and Architecture Design 9

3.1 Architectural Patterns 9

3.2 Interaction Diagrams 9

3.3 Analysis Class Diagrams 9

3.4 Unified Analysis Class Diagram 9

3.5 Security Software Architecture 9

4 Detailed Design 10

4.1 User Interface Design 10

4.1.1 Screen Configuration Standardization 10

4.1.2 Screen Transition Diagrams 10

4.1.3 Screen Specifications 10

4.2 Data Modeling 10

4.2.1 Conceptual Data Modeling 10

4.2.2 Database Design 10

4.3 Non-Database Management System Files 11

4.4 Class Design 11

4.4.1 General Class Diagram 11

4.4.2 Class Diagrams 11

4.4.3 Class Design 11

5 Design Considerations 13

5.1 Goals and Guidelines 13

5.2 Architectural Strategies 13

5.3 Coupling and Cohesion 14

5.4 Design Principles 14

5.5 Design Patterns 14

**List of Figures**

**No table of figures entries found.**

**List of Tables**

No table of figures entries found.

# Introduction

*<The following subsections of the Software Design Document (SDD) document should provide an overview of the entire SDD.>*

## Objective

<*Identify the purpose of this SDD and its intended audience. In this subsection, describe the purpose of the SDD and specify the intended audience for the SDD*>

This Software Design Document (SDD) provides the design details of the hourly bike rental service – EcobikeRental

The expected audience of this SDD is members of Group 17: Tran Thi Hong Nhung, Duong Hue Linh, Vu Thi Ngoc Lan, Mrs. Nguyen Thi Thu Trang – Lecture of the course Software Design and Construction and the people who will maintain EcobikeRental. It will also serve as a reference for SOICT students

## Scope

<*In this subsection:*

1. *Identify the software product(s) to be produced by name*
2. *Explain what the software product(s) will, and, if necessary, will not do*
3. *Describe the application of the software being specified, including relevant benefits, objectives, and goals*
4. *Be consistent with similar statements in higher-level specifications if they exist*

*This should be an executive-level summary. Do not enumerate the whole requirements list here*

*Note that this will be similar to what was written in the SRS.*

>

This document contains a complete description of the design of EcobikeRental

The EcobikeRental is an hourly bike rental service with lots of docking stations for users to rent or

return bike automatically. Users can scan the barcode on the lock to see the detailed information of

the bike (example: license plate, current battery percentage of electric bicycle, …) and choose rent

bike. After choosing rent bike, users must pay a deposit equal to 40% of the value of the bike. To

return a bike, users can search for an empty dock at a station (by station name) and choose return

bike. At this time, the system will return the deposit and deduct the amount of money corresponding to the rental period

The basic architecture is a Java application based on Model – View – Controller Pattern

## Glossary

*<Listing and explaining the terms appearing in the software’s profession and this document. Any assumption of the reader’s prior knowledge or experience on the subject is ill advised>*

## 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. |

*<Listing the referenced material used in this document, including the one related to the project>*

# Overall Description

<*This section describes the principles and strategies to be used as guidelines when designing and implementing the system.>*

## General Overview

*<Briefly introduce the system context and the basic design approach or organization. Provide a brief overview of the system and software architectures and the design goals. Include the high-level context diagram(s) for the system and subsystems provided in previous documents like SRS (e.g., general use case diagram, lower-level use case diagrams, activity diagrams), updated as necessary to reflect any changes that have been made based on more current information or understanding. If the high-level context diagram has been updated, identify the changes that were made and why>*

## Assumptions/Constraints/Risks

### Assumptions

*<Describe any assumptions or dependencies regarding the system, software and its use. These may concern such issues as: related software or hardware, operating systems, end-user characteristics, and possible and/or probable changes in functionality>*

### Constraints

*<Describe any global limitations or constraints that have a significant impact on the design of the system’s hardware, software and/or communications, and describe the associated impact. Such constraints may be imposed by any of the following (the list is not exhaustive):*

* *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*

*>*

### Risks

*<Describe any risks associated with the system design and proposed mitigation strategies.>*

# System Architecture and Architecture Design

<*Briefly describe the architectural design steps*>

## Architectural Patterns

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

MVC is an architectural pattern consisting of three parts: Model, View, Controller.

* Model: Handle data logic.

This part of the design pattern is the primary part and contains application information purely. It doesn’t contain any information on how to show the data to the user. It is independent of the user interface. It controls the logic and rules of application.

* View: It displays the information from the model to the user.

This part helps the user to see the model’s data. The main concern of this part is to access the model’s data. The view section uses a chart, table or diagrams to represent the information. It can also show similar data and use bar graphs and tables for different purposes. It is a visualization of information that the application contains.

* Controller: It controls the data flow into a model object and updates the view whenever data changes.

Most of the work is done by the controller. It provides the support for input and converts the input to commands for the application. It is used between the model and view part. The model and the view are interconnected, so the execution is reflected in the view part.

We should use MVC Architectural Pattern because:

-Multiple views can be made to models

-The partition of duties helps the developer in future developments and upgrades.

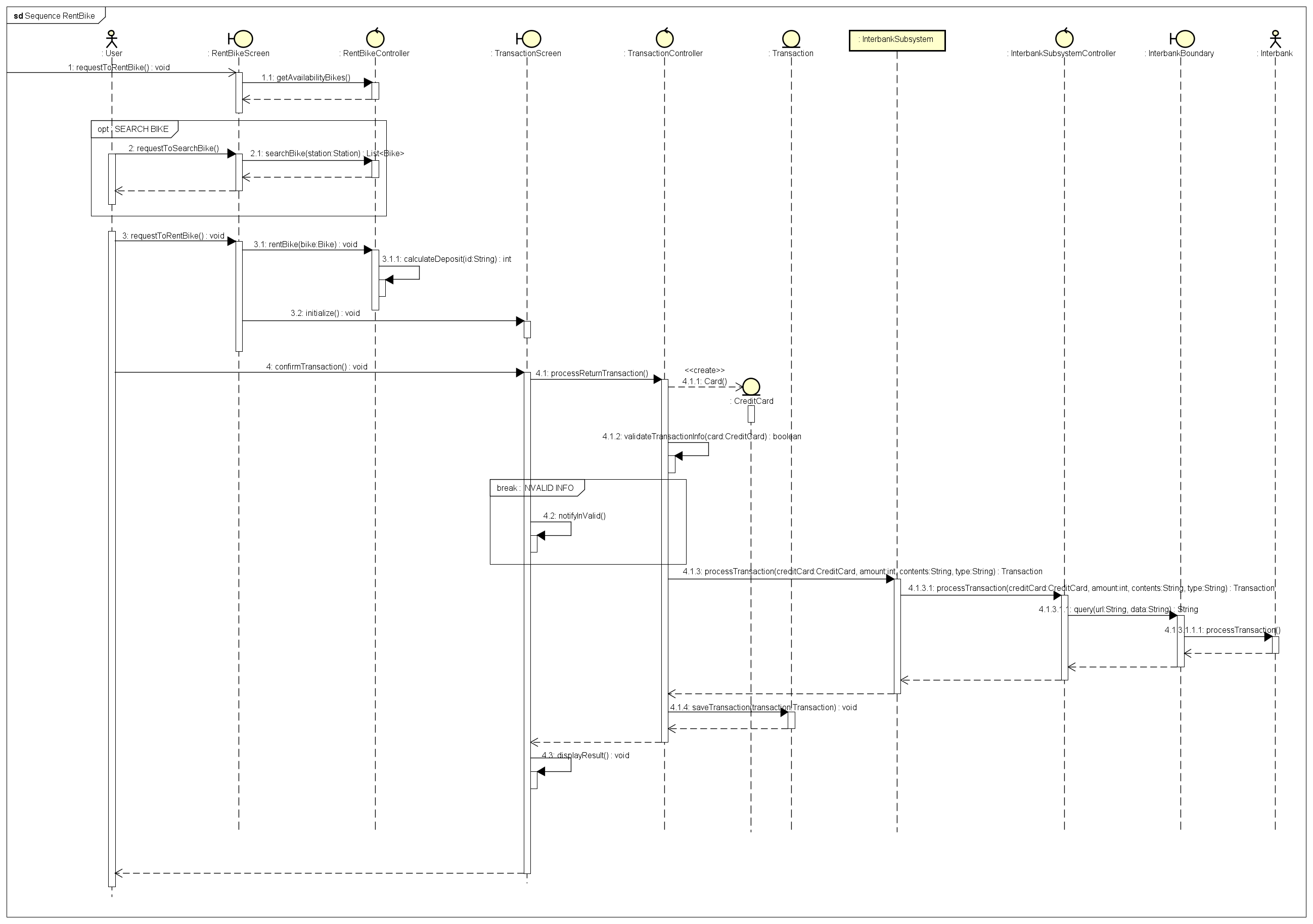
-The MVC theory works have low coupling behaviour among the models, views, and controllers.

-Multiple developers can work on models, views, and controllers at the same time.

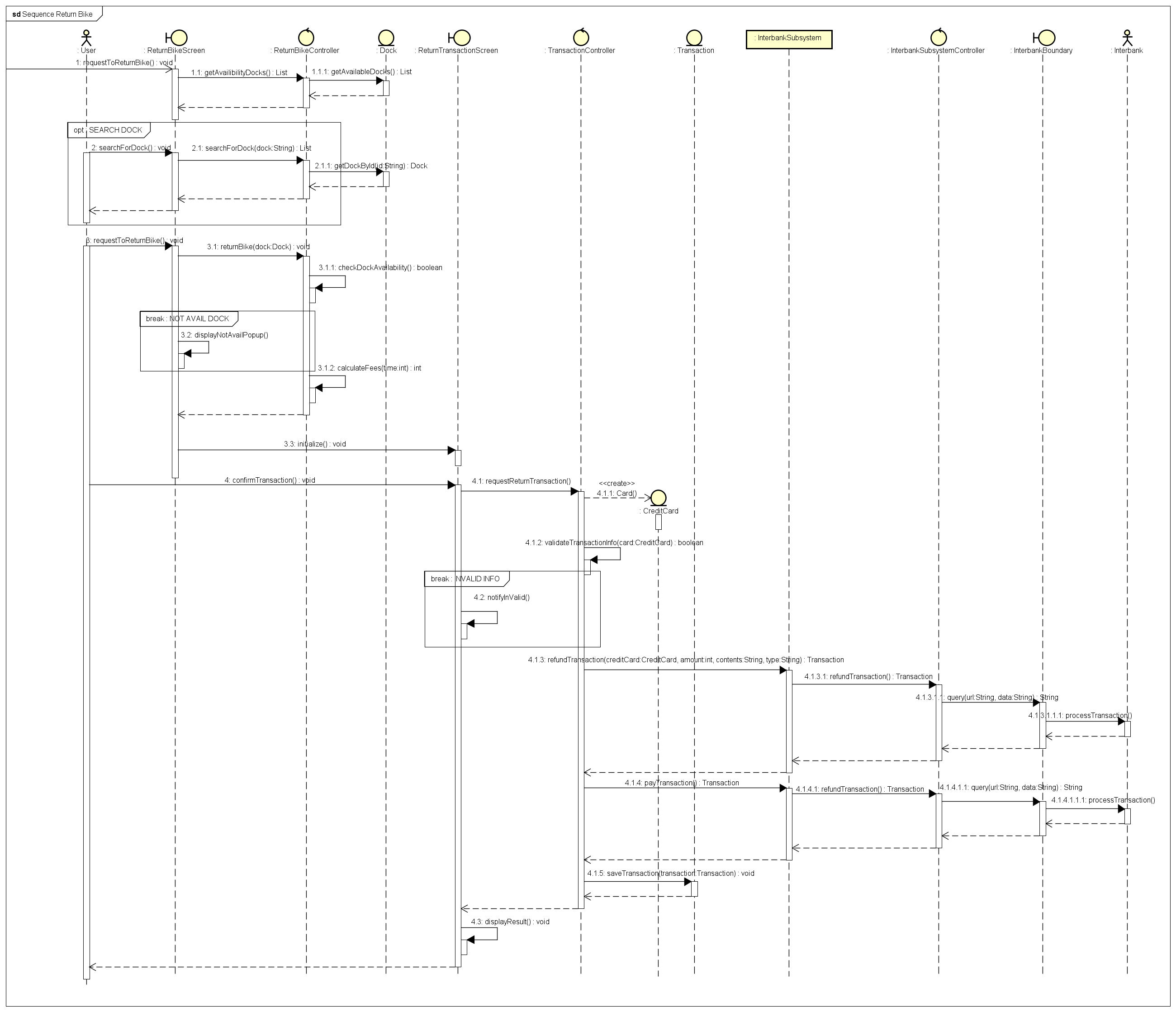
-The views for a required model are grouped together.

## Interaction Diagrams

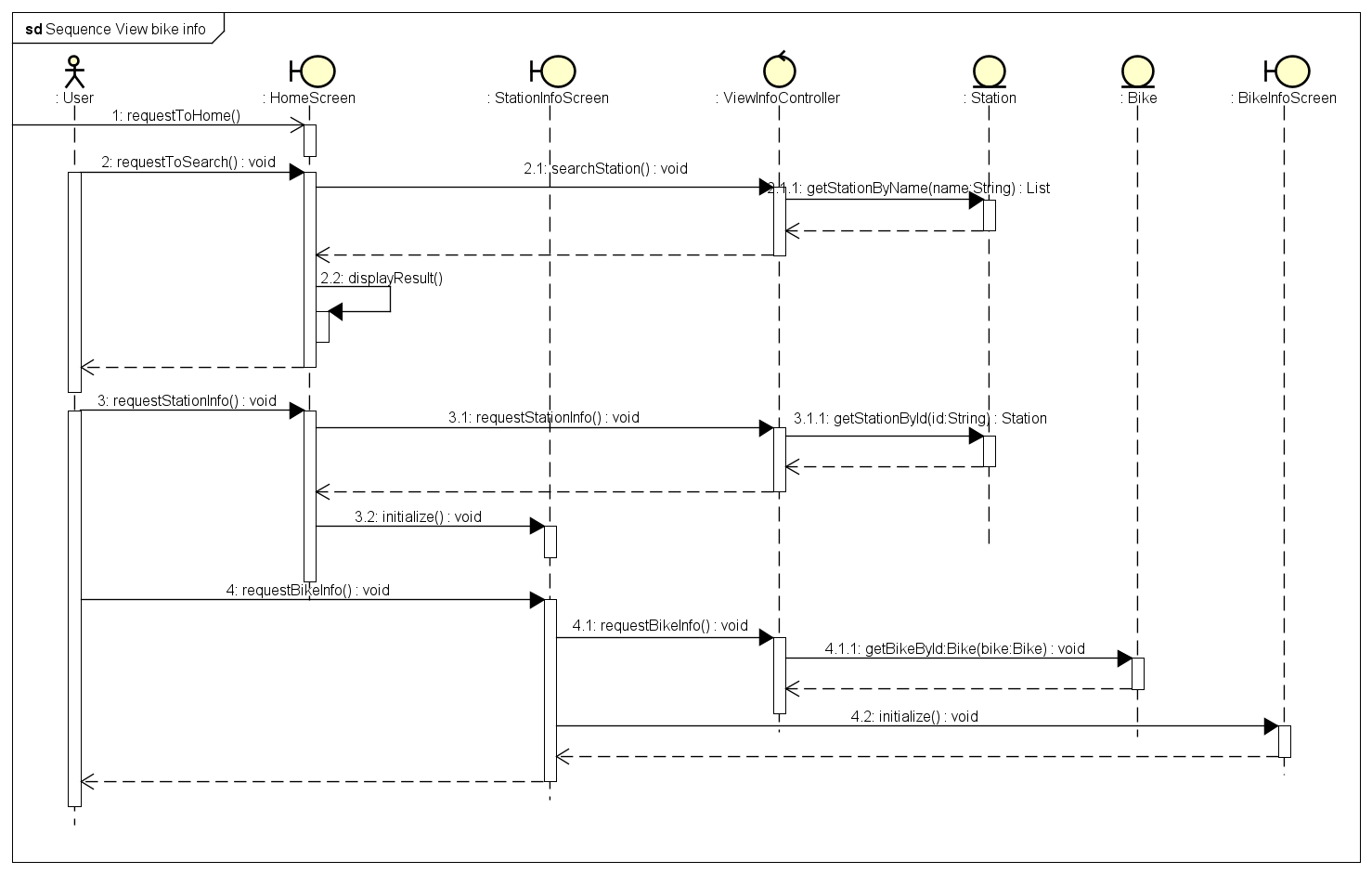
* Sequence Diagrams:



Hình 1 Sequence Rent Bike

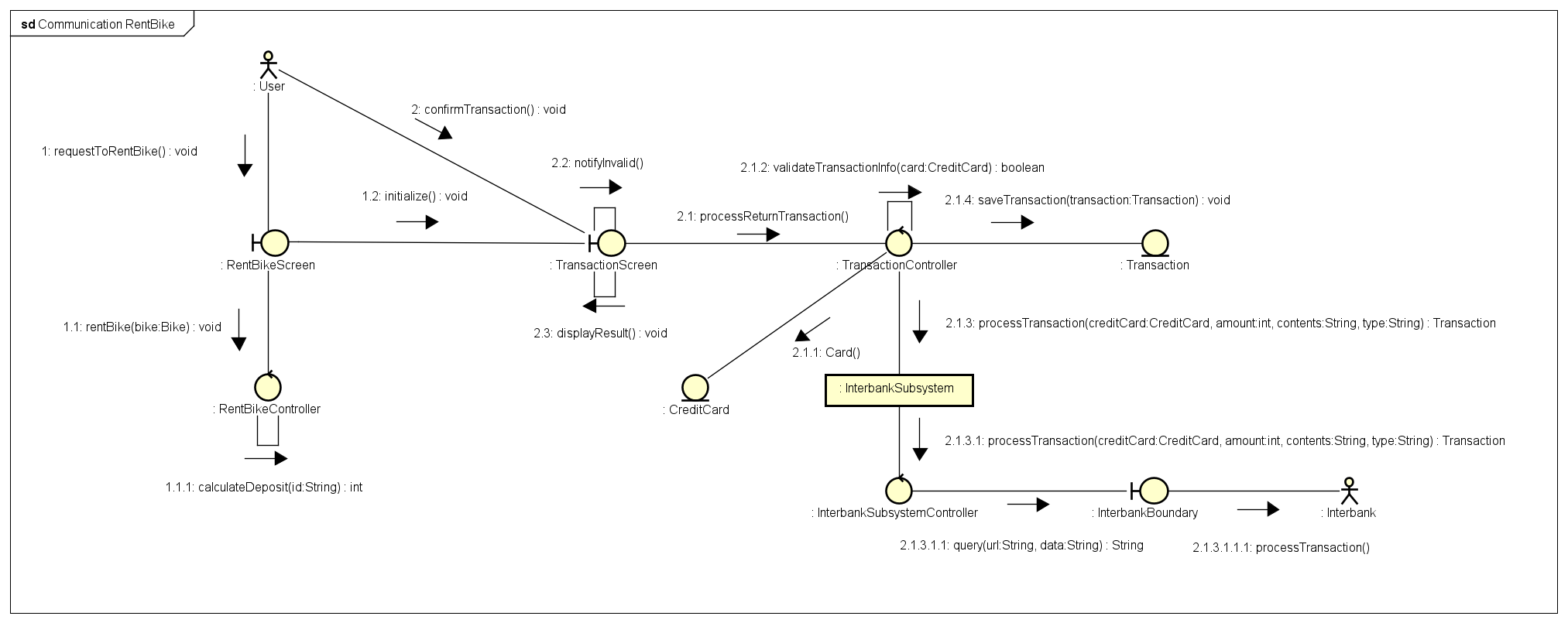


Hình 1 Sequence Return Bike

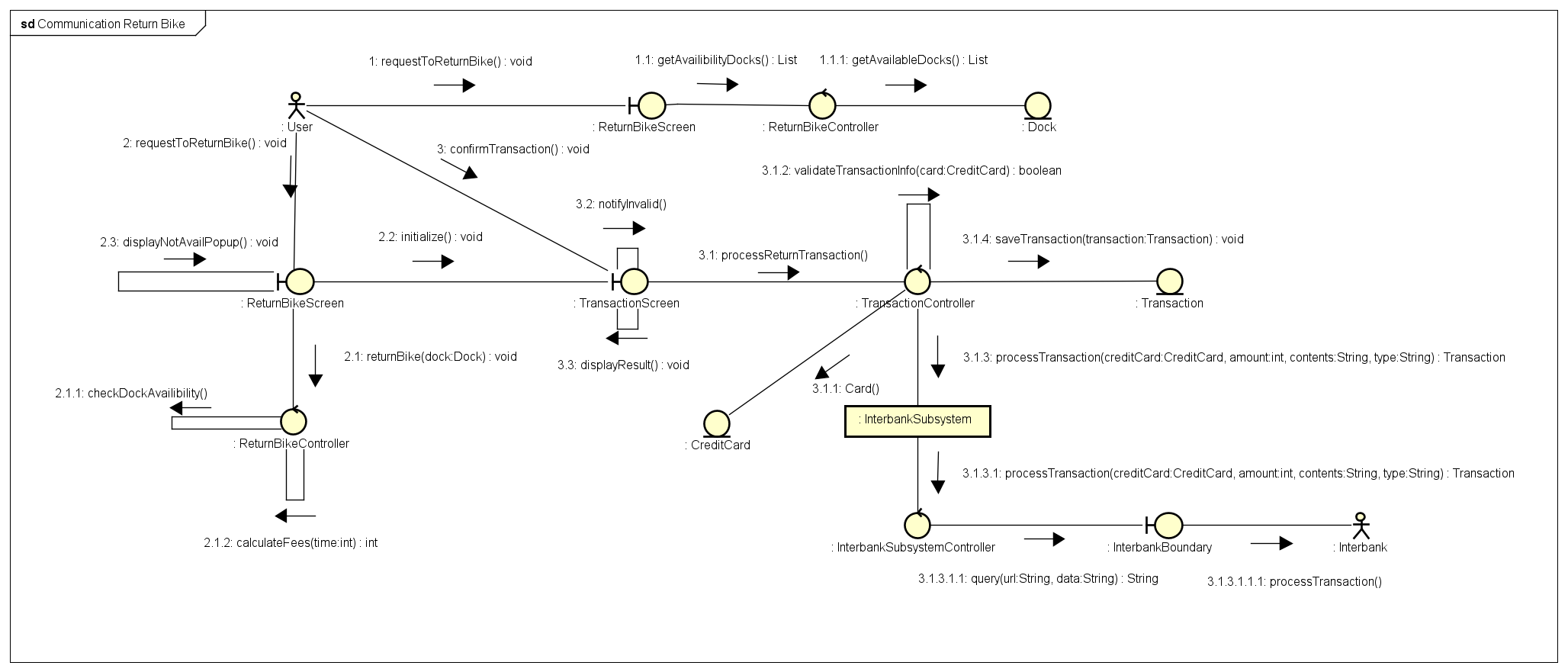


Hình 2 Sequence View Bike/Station Info

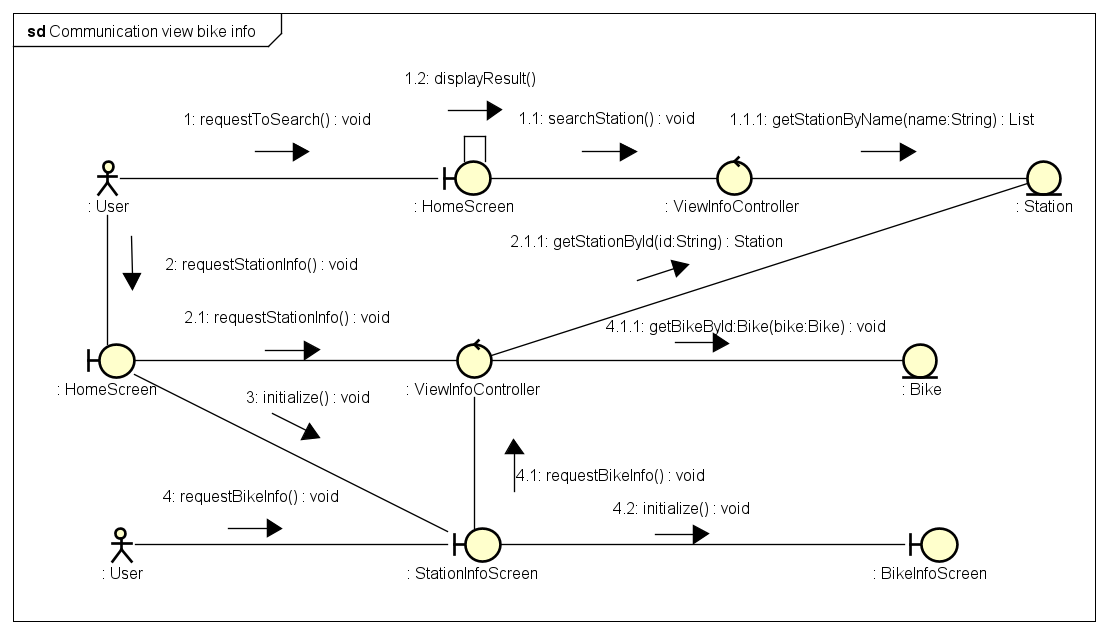
* Communication Diagrams:



Hình 3 Communication Rent Bike

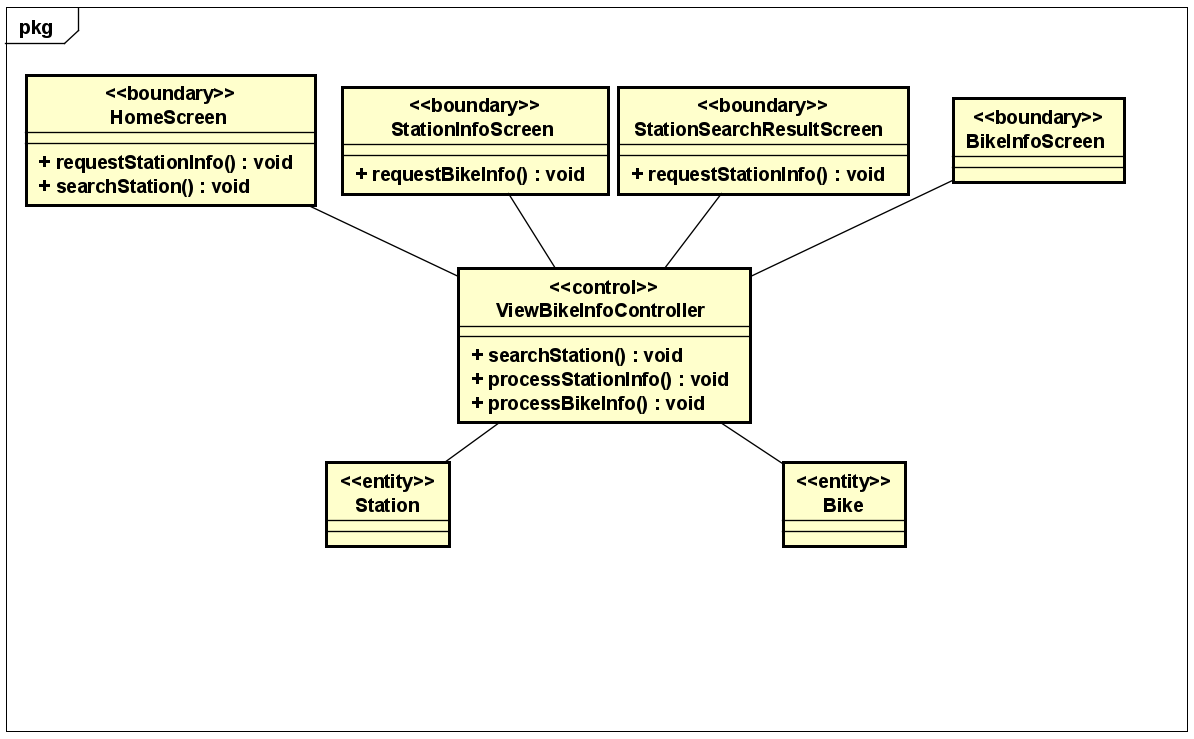


Hình 4 Communication Return Bike

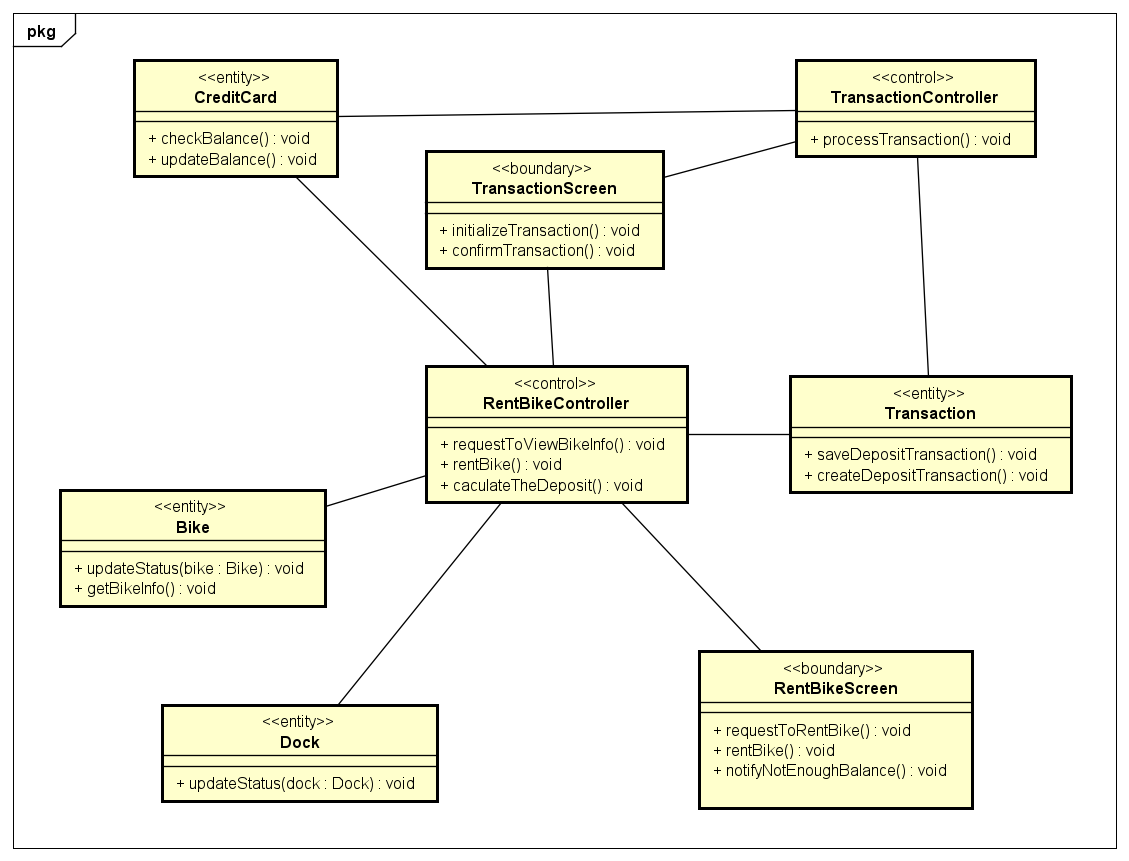


Hình 5 Communication View Bike/Station Info

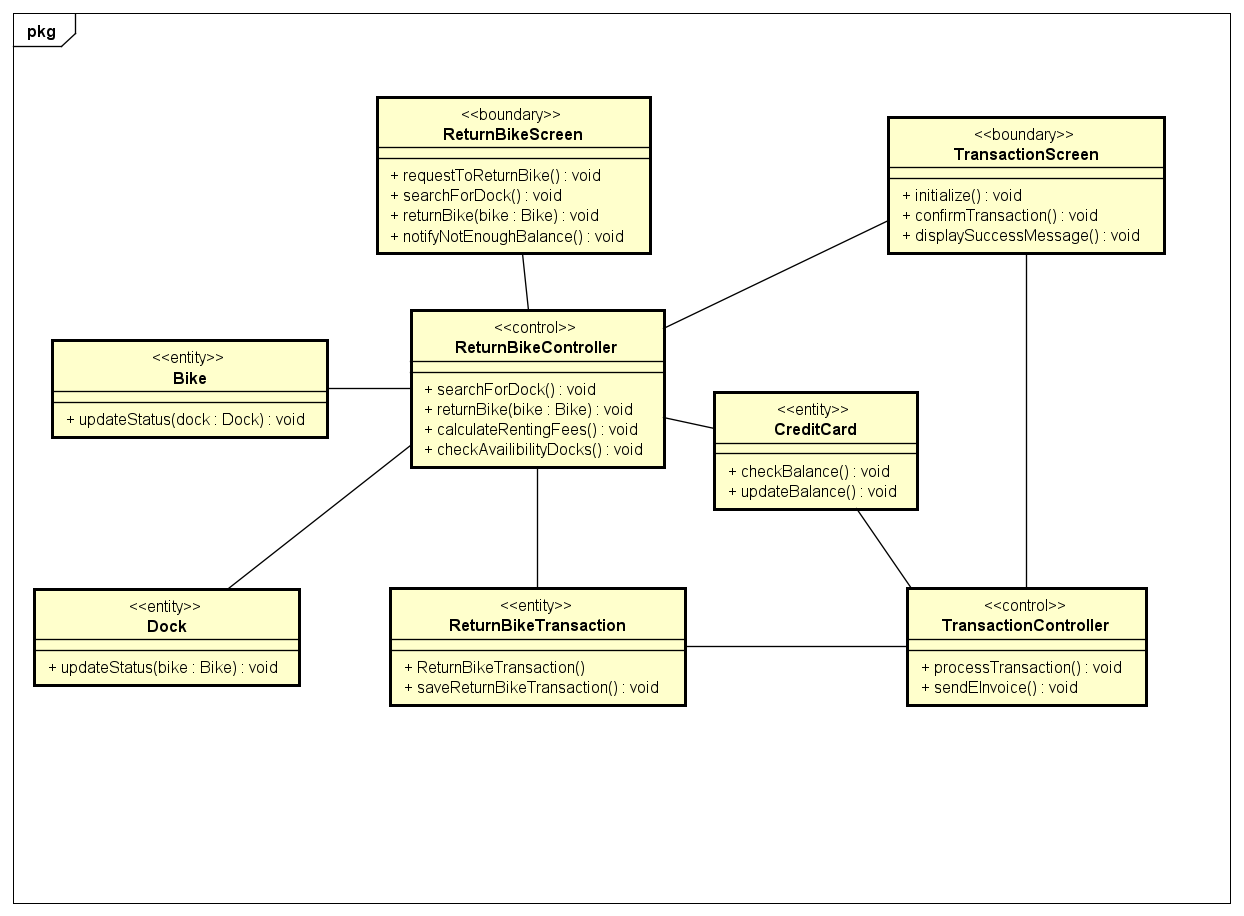
## Analysis Class Diagrams



Hình 6 View Bike /Station Info

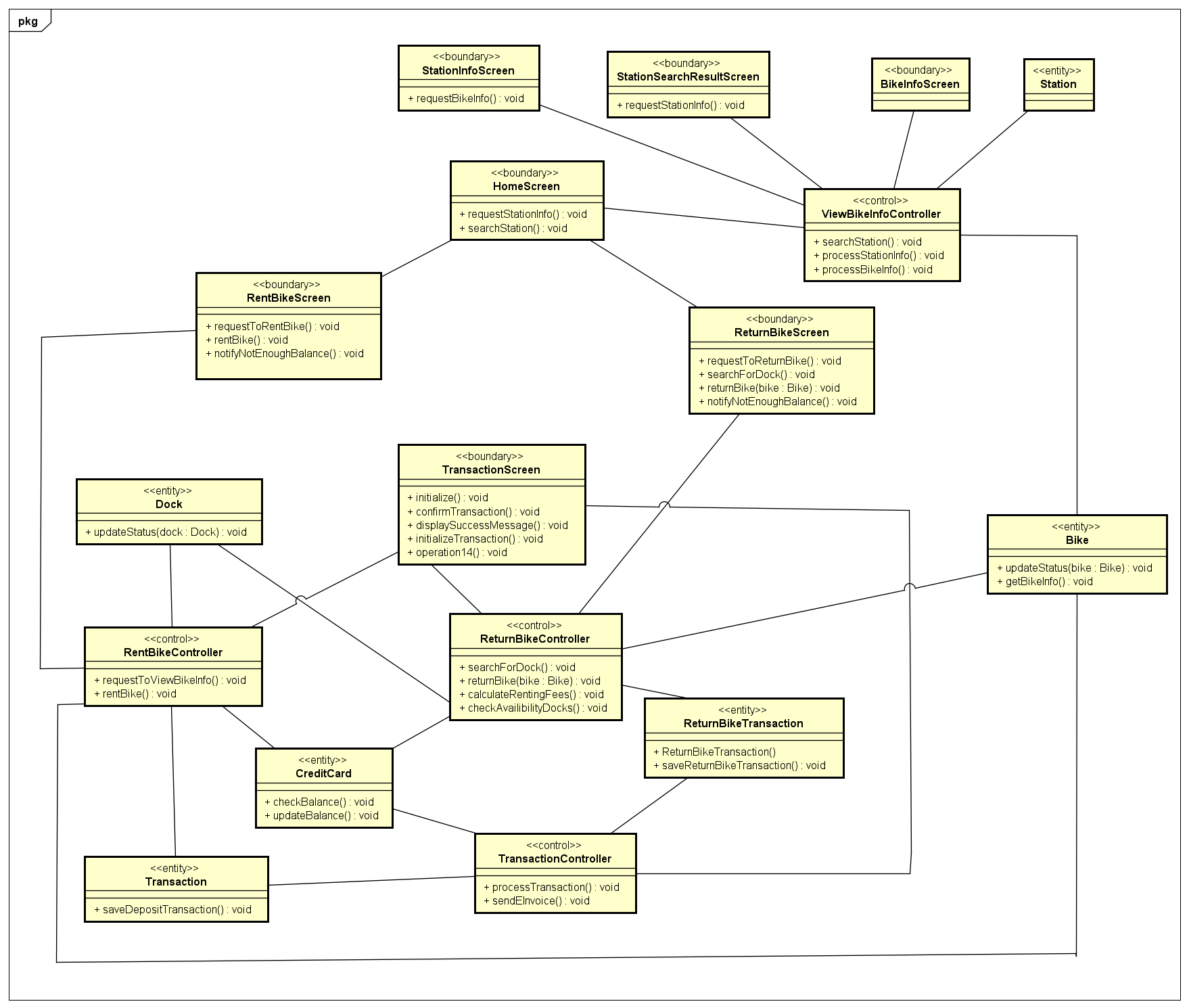


Hình 7 RentBike



Hình 8 Return Bike

## Unified Analysis Class Diagram



## Security Software Architecture

*<Describe the software components and configuration supporting the security and privacy of the system. Specify the architecture for (1) authentication to validate user identity before allowing access to the system;(2) authorization of users to perform functional activity once logged into the system, (3) encryption protocol to support the business risks and the nature of information, and (4) logging and auditing design, if required.>*

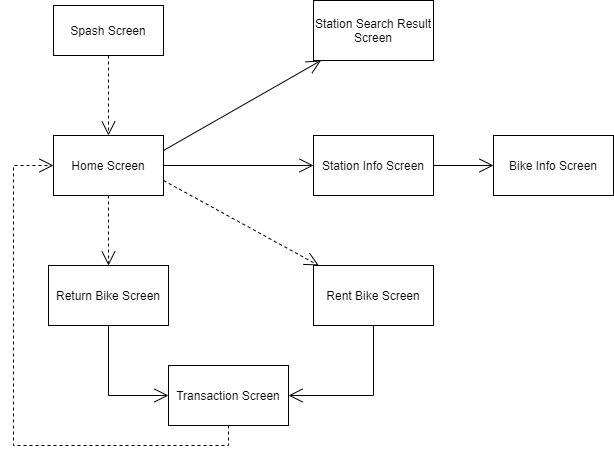
# Detailed Design

## User Interface Design

*<Suppose that you design a Graphical User Interface (GUI)>*

### Screen Configuration Standardization

### Screen Transition Diagrams



### Screen Specifications

*<Screen images should be included in the screen specifications>*

* **Home Screen**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| EcobikeRental Software | | Date of creation | Approved by | Reviewed by | Person in charge |
| Screen Specification | Home Screen | 15/11/2021 |  |  | Duong Hue Linh |
|  | | Control | Operation | Function | |
| Rent Button | Click | Display Rent Bike Screen | |
| Return Button | Click | Display Return Bike Screen | |
| Search Button | Click | Display Station Search Result Screen | |
| Area for displaying Staton | Initial | Display information of Station | |
| INFO Button | Click | Display Station Info Screen | |

|  |  |
| --- | --- |
| Screen name | Home Screen |
| Item name | Number of digits(bytes) | Type | Field attribute | Remark |
| Station | 20 | Text | Black | Left-justified |
| Addr | 50 | Text | Black | Left-justified |
| Avail | 20 | Numeral | Black | Left-justified |

* **Station Info Screen**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| * EcobikeRental Software | | Date of creation | Approved by | Reviewed by | Person in charge |
| Screen Specification | Station Info Screen | 15/11/2021 |  |  | Duong Hue Linh |
|  | | Control | Operation | Function | |
| Cancel Button | Click | Display Home Screen | |
| INFO Button | Click | Display Bike Info Screen | |
| Area for displaying Station Info | Initial | Display information of station | |

|  |  |
| --- | --- |
| Screen name | Station Info Screen |
| Item name | Number of digits(bytes) | Type | Field attribute | Remark |
| ID | 20 | Text | Black | Left-justified |
| Name | 20 | Text | Black | Left-justified |
| Address | 50 | Text | Black | Left-justified |
| Area | 20 | Numeral | Black | Left-justified |
| Available Bikes | 20 | Numeral | Black | Left-justified |
| Available Empty Docks | 20 | Numeral | Black | Left-justified |
| Distance | 20 | Numeral | Black | Left-justified |
| Walking Time | 20 | Numeral | Black | Left-justified |

* **Station Search Screen**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| * EcobikeRental Software | | Date of creation | Approved by | Reviewed by | Person in charge |
| Screen Specification | Station Search Result Screen | 15/11/2021 |  |  | Duong Hue Linh |
|  | | Control | Operation | Function | |
| Rent Button | Click | Display Rent Bike Screen | |
| Return Button | Click | Display Return Bike Screen | |
| Search Button | Click | Display Station Search Result Screen | |
| Area for displaying Staton | Initial | Display information of Station | |
| INFO Button | Click | Display Station Info Screen | |

|  |  |
| --- | --- |
| Screen name | Station Search Result Screen |
| Item name | Number of digits(bytes) | Type | Field attribute | Remark |
| Station | 20 | Text | Black | Left-justified |
| Addr | 50 | Text | Black | Left-justified |
| Avail | 20 | Numeral | Black | Left-justified |

* **Bike Info Screen**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| * EcobikeRental Software | | Date of creation | Approved by | Reviewed by | Person in charge |
| Screen Specification | Bike Info Screen | 15/11/2021 |  |  | Duong Hue Linh |
|  | | Control | Operation | Function | |
| Cancel Button | Click | Display Station Info Screen | |
| Area for displaying Bike Info | Initial | Display information of Bike | |

|  |  |
| --- | --- |
| Screen name | Bike Info Screen |
| Item name | Number of digits(bytes) | Type | Field attribute | Remark |
| ID | 20 | Text | Black | Left-justified |
| BikeType | 20 | Text | Black | Left-justified |
| Dock | 20 | Text | Black | Left-justified |
| Station | 20 | Text | Black | Left-justified |
| Battery Percentage | 20 | Numeral | Black | Left-justified |
| Time Limit | 20 | Numeral | Black | Left-justified |
| Deposit | 20 | Numeral | Black | Left-justified |

* **Rent Bike Screen**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| EcobikeRental Software | | Date of creation | Approved by | Reviewed by | Person in charge |
| Screen Specification | Return Bike Screen | 17/11/2021 |  |  | Vũ Thị Ngọc Lan |
|  | | Control | Operation | Function | |
| Scan Button | Click | Scan the barcode in the lock and display the information of the bike below | |
| Rent Button | Click | * Choose a bike to rent * Display Deposit Transaction Screen | |

|  |  |
| --- | --- |
| Screen name | Bike Info Screen |
| Item name | Number of digits(bytes) | Type | Field attribute | Remark |
| ID | 20 | Text | Black | Left-justified |
| BikeType | 20 | Text | Black | Left-justified |
| Dock | 20 | Text | Black | Left-justified |
| Station | 20 | Text | Black | Left-justified |
| Battery Percentage | 20 | Numeral | Black | Left-justified |
| Time Limit | 20 | Numeral | Black | Left-justified |
| Deposit | 20 | Numeral | Black | Left-justified |

* **Return Bike Screen**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| EcobikeRental Software | | Date of creation | Approved by | Reviewed by | Person in charge |
| Screen Specification | Return Bike Screen | 12/11/2021 |  |  | Tran Thi Hong Nhung |
|  | | Control | Operation | Function | |
| Search button | click | Display docks related to keyword | |
| Area for displaying docks | Initial | Display information of docks available | |
| Return Button | Click | * Choose a dock to return bike * Display Transaction Screen | |

|  |  |
| --- | --- |
| Screen name | Return Bike Screen |
| Item name | Number of digits(bytes) | Type | Field attribute | Remark |
| Dock Name | 50 | Text | Black | Left-justified |
| Station | 20 | Text | Black | Left-justified |
| Addr | 50 | Text | Black, italic | Left-justified |
| Search | 20 | Text | Black | Left-justified |
| Username | 50 | Text | Green | Right-justified |
| Bike | 20 | Text | Green | Right-justified |

* **Transaction Screen**

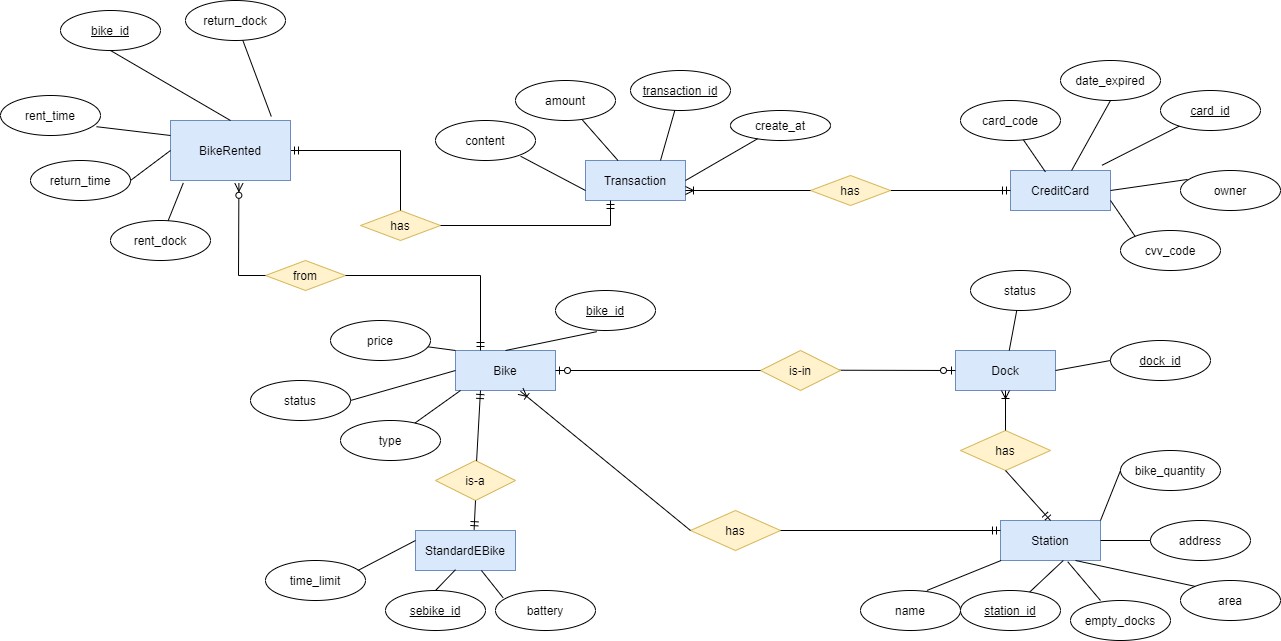
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| EcobikeRental Software | | Date of creation | Approved by | Reviewed by | Person in charge |
| Screen Specification | Transaction Screen | 12/11/2021 |  |  | Tran Thi Hong Nhung |
|  | | Control | Operation | Function | |
| Confirm Button | Click | Display Dialog | |
| Cancel Button | Click | Display Return Bike Screen | |
| OK Button | Click | * Close Dialog * Display Home Screen | |
| Area for displaying Transaction Info | Initial | * Display information of transaction | |
| Area for payment info | Initial | * Display text field for entering payment information | |
| Area for displaying Dialog | Initial | * Display result of transaction | |

|  |  |
| --- | --- |
| Screen name | Transaction Screen |
| Item name | Number of digits(bytes) | Type | Field attribute | Remark |
| Station | 20 | Text | Black | Left-justified |
| Username | 50 | Text | Green | Left-justified |
| Bike | 20 | Text | Green | Left-justified |
| TransactionType | 20 | Text | Black | Left-justified |
| Dock Name | 50 | Text | Black | Left-justified |
| Rent Time | 20 | Numeral | Black | Left-justified |
| Rent fees | 20 | Numeral | Black | Left-justified |
| Deposit | 20 | Numeral | Black | Left-justified |
| Note | 50 | Text | Black | Left-justified |
| Card Number | 20 | Numeral | Black | Right-justified |
| Card Holder Name | 50 | Text | Black | Right-justified |
| Issuing Bank | 50 | Text | Black | Right-justified |

## Data Modeling

### Conceptual Data Modeling

*<E-R Diagram image and description of entities and relationships>*



### Database Design

#### Database Management System

*<Specify what is the decision of Database Management System (DBMS) and give some description of the DBMS>*

Hệ thống sử dụng hệ quản trị cơ sở dữ liệu MySQL. Hệ quản trị cơ sở dữ liệu MySQL là một trong những phần mềm quản trị CSDL dạng server based. MySQL là phần mềm quản lý dữ liệu thông qua CSDL. Và mỗi một CSDL đều có bảng quan hệ chứa dữ liệu riêng biệt.

MySQL có cơ chế quản lý sử dụng riêng giúp cho mỗi người sử dụng đều có thể quản lý cùng lúc một hay nhiều CSDL khác nhau. Và mỗi người dùng đều có 1 username và password để truy nhập và truy xuất đến CSDL. Khi truy vấn đến CSDL của MySQL, bạn phải cung cấp tài khoản và mật khẩu có quyền sử dụng cơ sở dữ liệu đó*.*

MySQL có các ưu điểm: hiệu năng cao, tốc độ nhanh, hỗ trợ ngôn ngữ truy vấn, mã nguồn mở, dễ sử dụng.

#### Database Diagram

#### Database Detail Design

<

*Give a detail design of each element in the DB diagram. For instance, in a Relational DBMS, give a detail design for each Table and their constraints, illustrated in below table (PK: Primary Key, FK: Foreign Key).*

* 1. “bike” Table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| STT | Tên trường | Kiểu dữ liệu | Nullable | Ràng buộc | Mô tả |
| 1 | bike\_id | varchar | no | PK | mã số xe đạp |
| 2 | type | varchar | no |  | loại xe đạp |
| 3 | price | int | no |  | giá trị xe đạp |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 4 | status | tinyint | no |  | trạng thái mượn/trả của xe  đạp |
| 5 | station\_id | varchar | yes | FK | mã số bãi để xe |

* 1. “station” Table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| STT | Tên trường | Kiểu dữ liệu | Nullable | Ràng  buộc | Mô tả |
| 1 | station\_id | varchar | no | PK | mã số bãi gửi xe |
| 2 | name | varchar | no |  | tên bãi gửi xe |
| 3 | address | varchar | no |  | địa chỉ bãi gửi xe |
| 4 | area | int | no |  | diện tích bãi gửi xe |
| 5 | bike\_quantity | int | no |  | số xe hiện đang  có trong bãi |
| 6 | empty\_docks | int | no |  | số lượng chỗ để  xe còn trống  trong bãi |

* 1. “dock” Table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| STT | Tên trường | Kiểu dữ liệu | Nullable | Ràng  buộc | Mô tả |
| 1 | dock\_id | varchar | no | PK | mã số chỗ để xe |
| 2 | status | tinyint | no |  | trạng thái của chỗ để xe (đã có xe để/trống) |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 3 | station\_id | varchar | no | FK | mã số bãi gửi xe |
| 4 | bike\_id | varchar | yes | FK | mã số xe đạp  trong chỗ để |

* 1. “standard\_ebike” Table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| STT | Tên trường | Kiểu dữ liệu | Nullable | Ràng  buộc | Mô tả |
| 1 | sebike\_id | varchar | no | PK | mã số xe đạp điện |
| 2 | time\_limit | int | no |  | thời gian tối đa có thể sử dụng được  xe |
| 3 | battery | int | no |  | lượng pin còn lại của xe điện |

* 1. “transaction” Table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| STT | Tên trường | Kiểu dữ liệu | Nullable | Ràng  buộc | Mô tả |
| 1 | transaction\_id | int | no | PK | mã số giao dịch |
| 2 | content | varchar | no |  | nội dung giao dịch |
| 3 | amount | int | no |  | số tiền giao dịch |
| 4 | create\_at | date | no |  | thời điểm thực hiện giao dịch |
| 5 | bike\_id | varchar | no | FK | mã số xe đạp |
| 6 | card\_id | varchar | no | FK | mã số thẻ thực  hiện giao dịch |

* 1. “creditcard” Table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| STT | Tên trường | Kiểu dữ liệu | Nullable | Ràng  buộc | Mô tả |
| 1 | card\_id | varchar | no | PK | mã số thẻ |
| 2 | owner | varchar | no |  | tên chủ sở hữu  thẻ |
| 3 | card\_code | varchar | no |  | mã thẻ |
| 4 | cvv\_code | varchar | no |  | mã xác thực thẻ |
| 5 | date\_expired | varchar | no |  | ngày hết hạn của  thẻ |

* 1. “bike\_rent” Table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| STT | Tên trường | Kiểu dữ liệu | Nullable | Ràng  buộc | Mô tả |
| 1 | bike\_id | varchar | no | PK | mã số xe đạp |
| 2 | rent\_time | date | no |  | thời điểm mượn  xe |
| 3 | return\_time | date | no |  | thời điểm trả xe |
| 4 | Rent\_dock | varchar | no |  | mã số chỗ để xe  của xe lúc mượn |
| 5 | date\_expired | varchar | no |  | mã số chỗ để xe  lúc trả |

## Non-Database Management System Files

*<Provide the detailed description of all non-DBMS files if any and include a narrative description of the usage of each file that identifies if the file is used for input, output, or both, and if the file is a temporary file. Also provide an indication of which modules read and write the file and include file structures (refer to the data dictionary). As appropriate, the file structure information should include the following:*

*• Record structures, record keys or indexes, and data elements referenced within the records*

*• Record length (fixed or maximum variable length) and blocking factors*

*• Access method (e.g., index sequential, virtual sequential, random access, etc.)*

*• Estimate of the file size or volume of data within the file, including overhead resulting from file access methods*

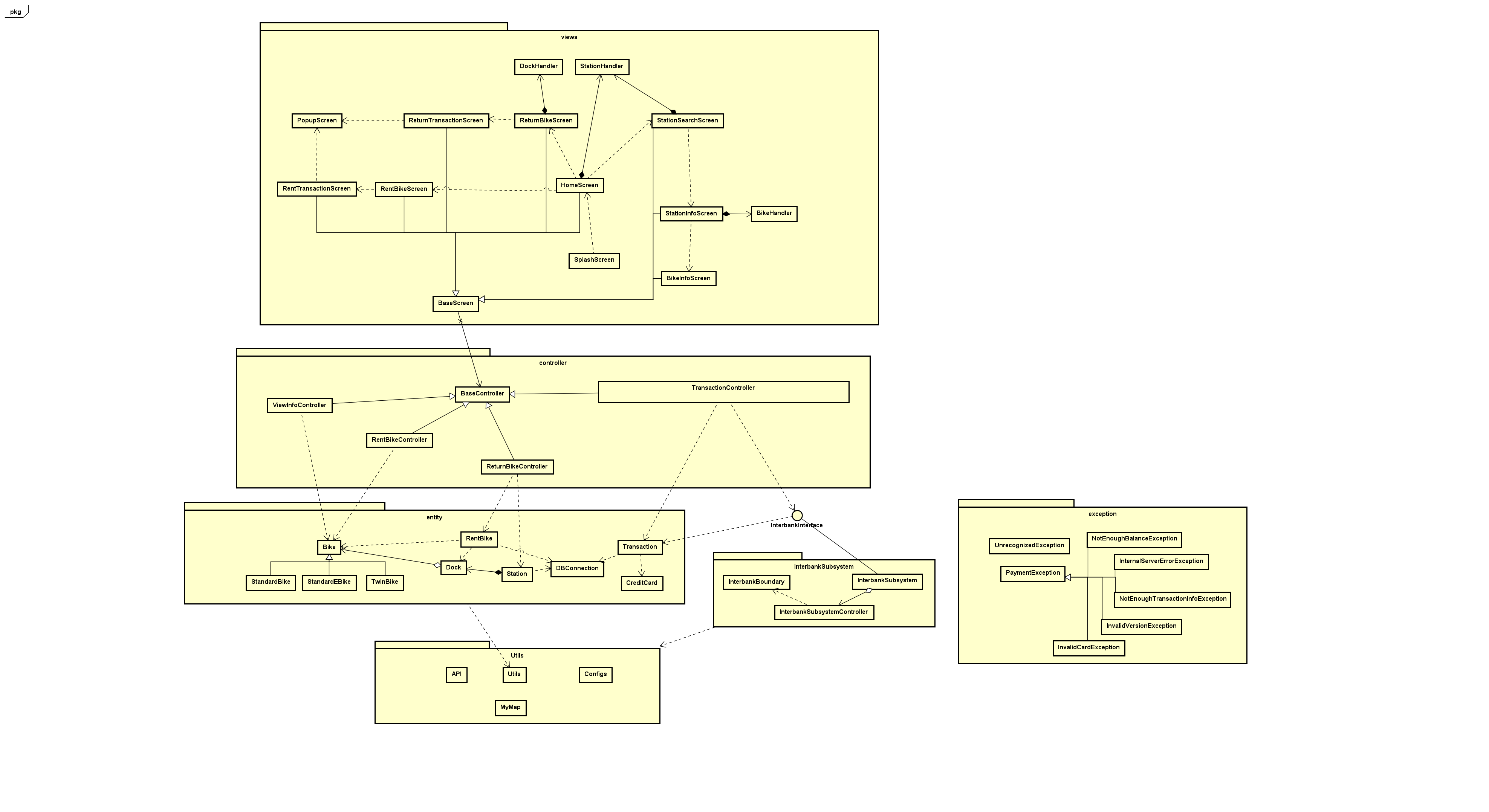
*• Definition of the update frequency of the file (If the file is part of an online transaction-based system, provide the estimated number of transactions per unit of time, and the statistical mean, mode, and distribution of those transactions.)*

*• Backup and recovery specifications>*

## Class Design

### General Class Diagram

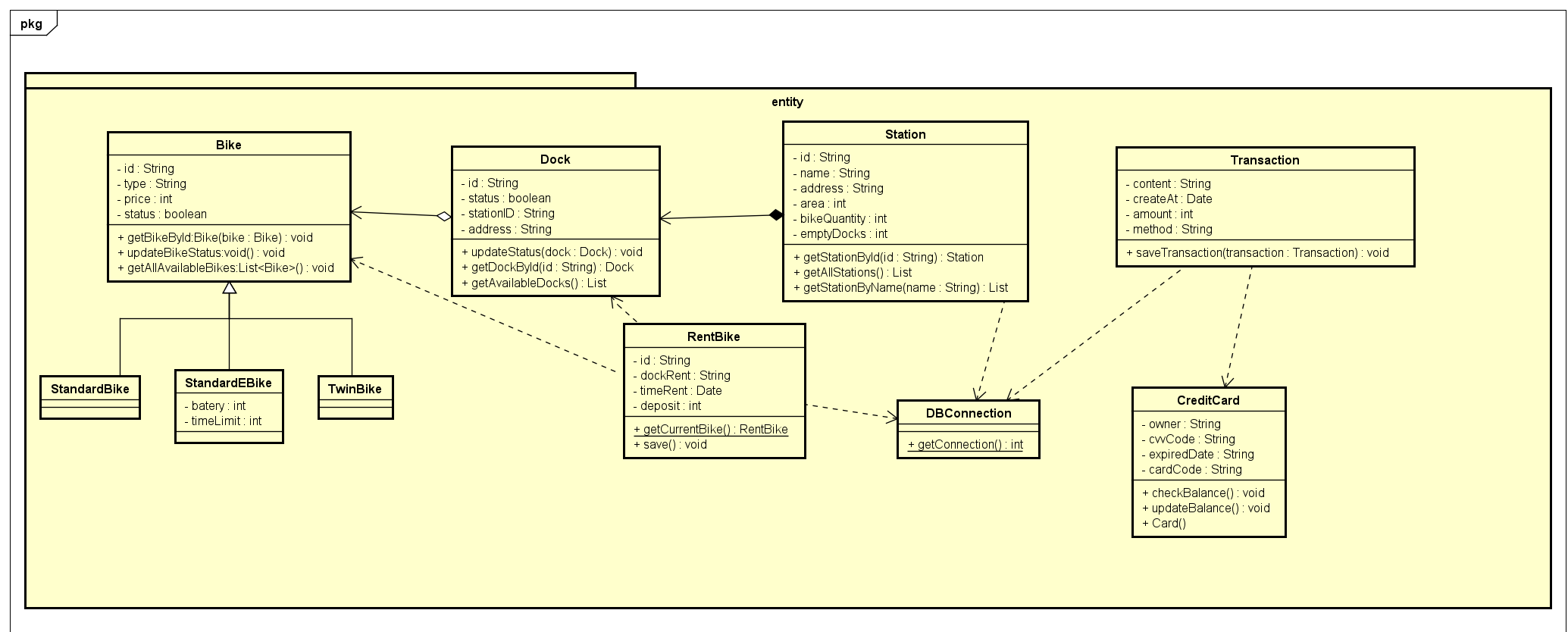
<General class diagram which shows the whole class diagram of the software. This diagram may have packages, subsystems and classes. Classes in this diagram may not have all attributes and operations>



### Class Diagrams

<Detail class diagram with full attributes and operations>

#### Class Diagram for Package “enitity”

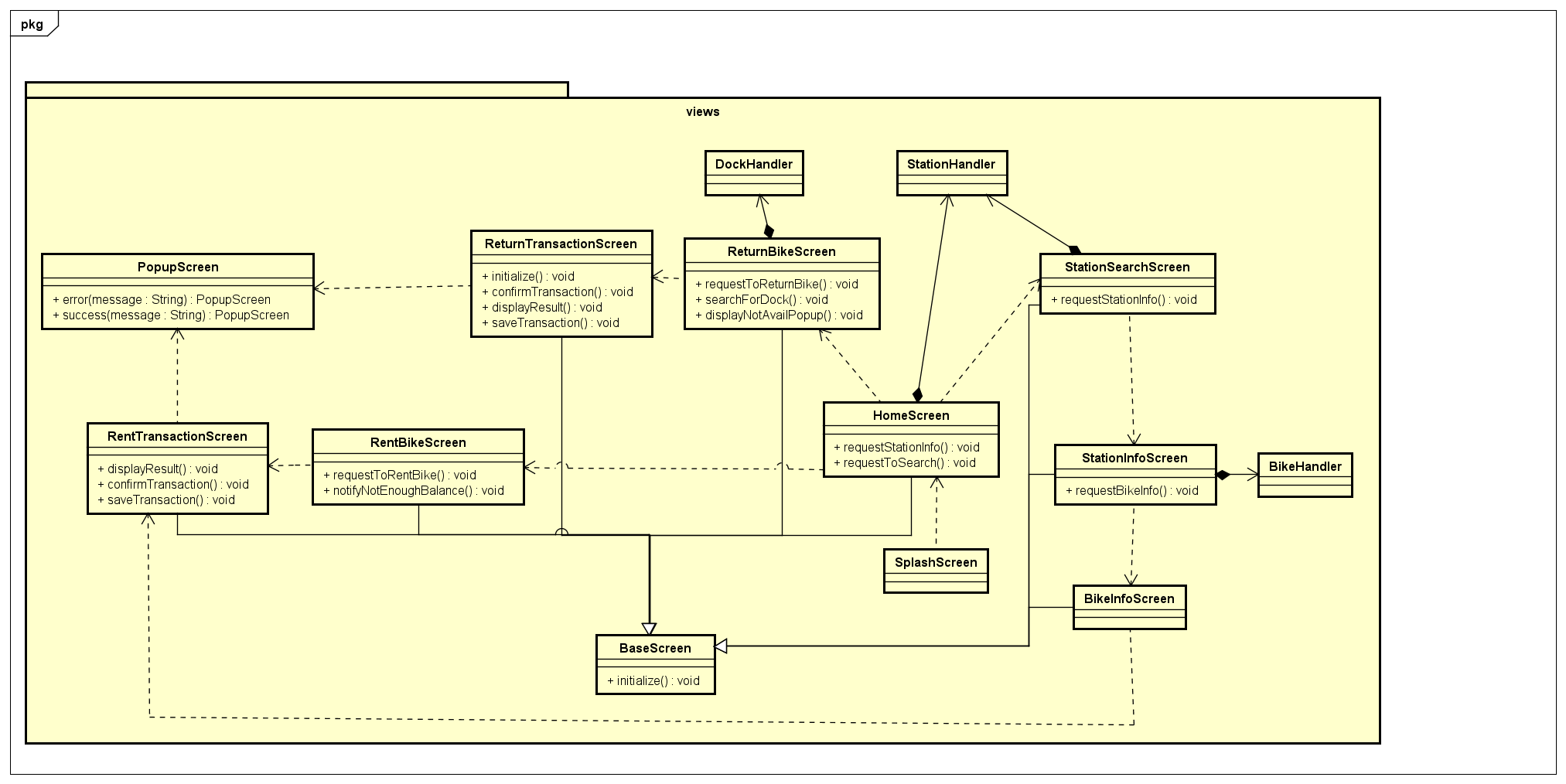


#### Class Diagram for Package “controller”

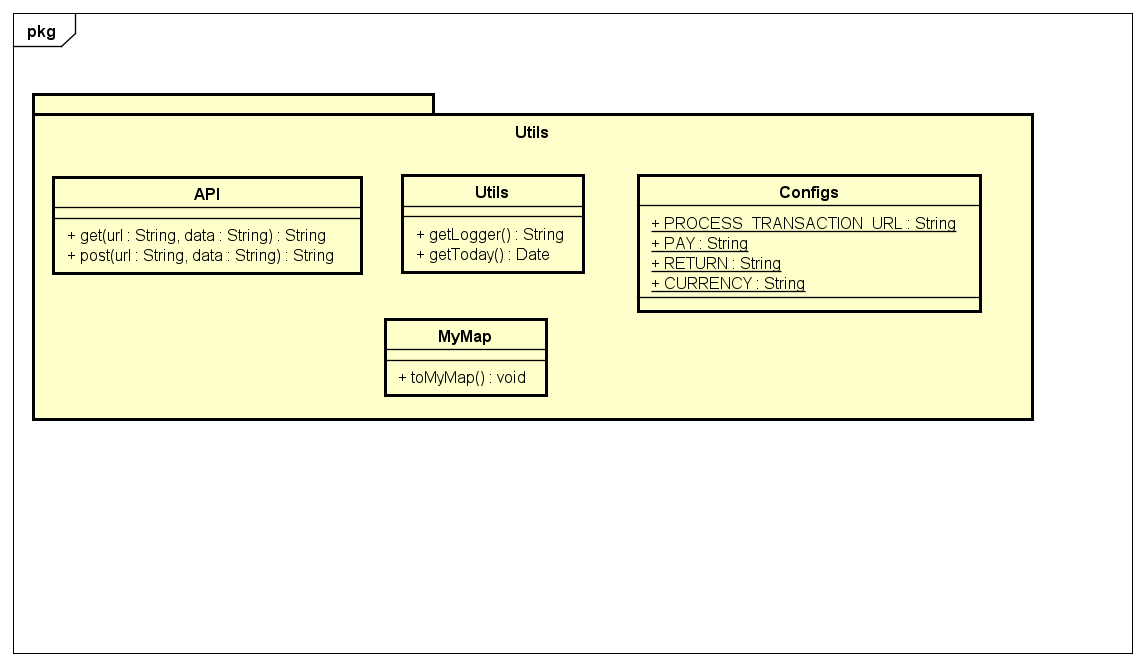
Ảnh có chứa bàn

Mô tả được tạo tự động

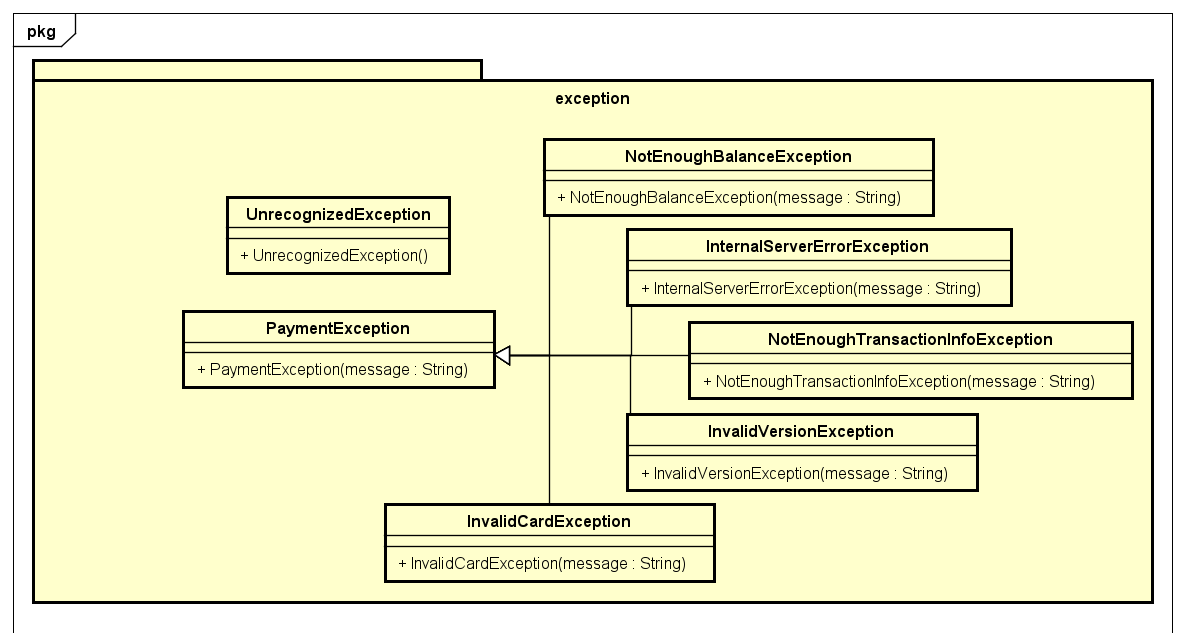
#### Class Diagram for Package “views”



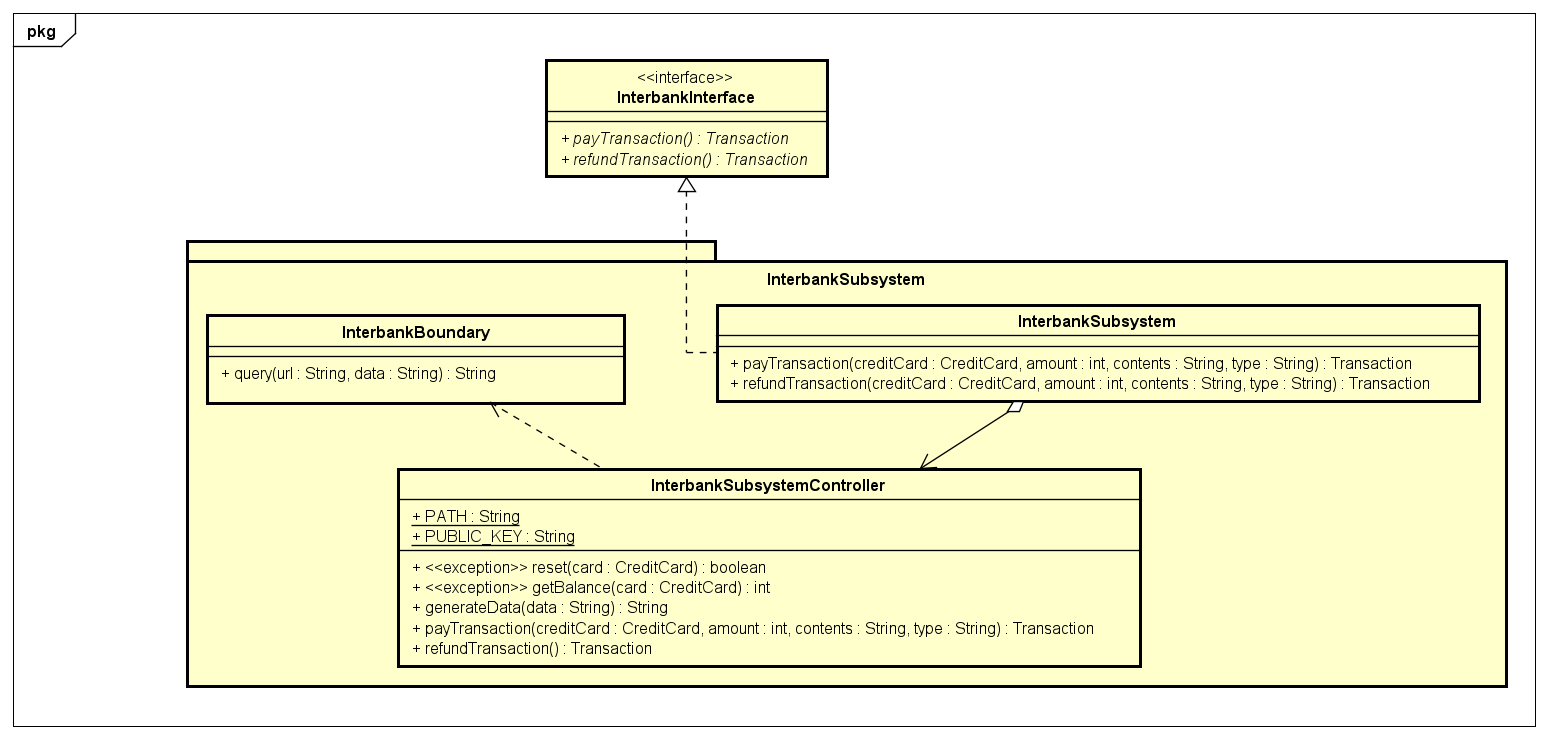
#### Class Diagram for Package “utils”



#### Class Diagram for Package “exception”



#### Class Diagram for Subsystem “InterbankSubsystem”



### Class Design

**Controller**

1. Class “TransactionController”

**Attribute**

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

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | returnTransaction | Map<String, String> | pay amount and return transaction |
| 2 | rentTransaction | Map<String, String> | pay deposit and return transaction |

*Parameter*:

* amount – số tiền giao dịch
* contents – nội dung giao dịch
* cardNumber – số thẻ
* cardHolderName – tên chủ sở hữu
* expirationDate – ngày hết hạn theo định dạng "mm/yy"
* securityCode - mã bảo mật cvv/cvc
* type – loại giao dịch
* transaction: giao dịch

*Exception*:

* Không

**Method**

* getExpirationDate: Chuyển dữ liệu ngày từ định dạng “mm/yy” sang “mmyy”
* validateTransactionInfo: Kiểm tra các thông tin đầu vào

**State**

Không

1. Class “RentBikeController”

**Attribute**

Không

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | requestToViewBikeInfo | void | view bike information |
| 2 | rentBike | void | request to rent a bike |

*Parameter*:

* bike – Xe đạp

*Exception*:

* Không

**Method**

* getListBikes: Lấy ra tất cả các xe hiện đang chưa được mượn trong 1 station
* getListsStation: Lấy ra danh sách station
* checkBikeAvailable: kiểm tra xe có đang sẵn có không
* calculateDeposit: tính tiền đặt cọc

**State**

Không

1. Class “ReturnBikeController”

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Name* | *Data type* | *Default value* | *Description* |
| 1 | bike | Bike | NULL | represent the bike rented |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | returnBike | void | return a bike |

*Parameter*:

* bike: Xe đang mượn
* dock: điểm trả xe

*Exception*:

* Không

**Method**

* calculateRentingFees: tính số tiền phí mượn xe
* searchForDock: tìm kiếm các điểm trả xe đang trống
* checkAvailabilityDocks: kiểm tra xem còn điểm trả đang trống hay không

**State**

Không

1. Class “ViewBikeController”

**Attribute**

Không

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | searchStation | void | search for a station dock by its name/address |
| 2 | processStationInfo | void | display station info |
| 2 | processBikeInfo | void | display bike info |

*Parameter*:

* station: tên/địa chỉ bãi xe cần tìm

*Exception*:

* Không

**Method**

Không

**State**

Không

**Entity**

1. Class “Station”

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Name* | *Data type* | *Default value* | *Description* |
| 1 | id | String | NULL | represent the id of station |
| 2 | name | String | NULL | represent the name of station |
| 3 | area | String | NULL | represent the area of station |
| 4 | emptyDocks | int | NULL | represent the number of available bikes |
| 5 | bikeQuantity | int | NULL | represent the number of empty docking points |
| 6 | address | String | NULL | represent the address of station |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | getAllStation | List<Station> | get all stations |
| 2 | getStationById | Station | get station by id |

*Parameter*:

* id – id của bãi xe

*Exception*:

* SQLException – nếu lỗi trả về khi thao tác với CSDL

**Method**

Không

**State**

Không

1. Class “Bike”

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Name* | *Data type* | *Default value* | *Description* |
| 1 | id | String | NULL | represent the id of bike |
| 2 | price | int | NULL | represent the price of bike |
| 3 | status | boolean | NULL | represent the status of bike |
| 4 | type | String | NULL | represent the type of bike |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | getAllBike | List<Bike> | get all bikes |
| 2 | getBikeById | Bike | get bike by id |

*Parameter*:

* id – id của xe

*Exception*:

* SQLException – nếu lỗi trả về khi thao tác với CSDL

**Method**

Không

**State**

Không

1. Class “Dock”

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Name* | *Data type* | *Default value* | *Description* |
| 1 | id | String | NULL | represent the id of dock |
| 2 | status | boolean | NULL | represent status of dock |
| 3 | stationId | String | NULL | represent id of station for dock |
| 4 | bikeId | String | NULL | represent id of bike in dock |
| 5 | address | String | NULL | represent the address of dock |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | getAvailableDocks | List<Dock> | get all docks available |
| 2 | getDockById | Dock | get dock by id |

*Parameter*:

* id – id của bãi xe

*Exception*:

* SQLException – nếu lỗi trả về khi thao tác với CSDL

**Method**

Không

**State**

Không

1. Class “Transaction”

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Name* | *Data type* | *Default value* | *Description* |
|  | errorcode | String | NULL | error code of transaction |
|  | content | String | NULL | content for transaction |
|  | method | String | NULL | method payment |
|  | createAt | Date | NULL | time create transaction |
|  | amount | int | NULL | amount for payment |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
|  | savePaymentTransaction | void | lưu giao dịch vào CSDL |

*Parameter*:

* card: credit card dùng để thanh toán
* order: đơn hàng tương ứng với giao dịch

*Exception*:

* SQLException – nếu lỗi trả về khi thao tác với CSDL

**Method**

* getErrorCode: lấy mã lỗi của giao dịch thanh toán

**State**

Không

Class “DBConnection”

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Name* | *Data type* | *Default value* | *Description* |
|  | connection | Connection | NULL | connection to database |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
|  | getConnection | Connection | tạo kết nối tới CSDL |

*Parameter*:

* không

*Exception*:

* Exception – nếu trả về lỗi trong quá trình kết nối CSDL

**Method**

* Không

**State**

Không

**Subsystem**

Class “InterbankBoundary”

**Attribute**

Không

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | query | String | truy vấn yêu cầu |

*Parameter*:

* url: url của API
* data: dữ liệu truyền vào truy vấn

*Exception*:

* Không

**Method**

* Không

**State**

Không

Class “InterbankSubsystemController”

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Name* | *Data type* | *Default value* | *Description* |
|  | path | String | NULL | path |
|  | public\_key | String | NULL | public\_key |
|  | secret\_key | String | NULL | secret key |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | refundTransaction | Transaction | hoàn lại tiền |
| 2 | payTransaction | Transaction | thanh toán |

*Parameter*:

* card: credit card
* amount: số tiền cần thanh toán
* content: nội dung giao dịch

*Exception*:

* Không

**Method**

* generateData: chuyển dữ liệu về dạng JSON

**State**

Không

**Utils**

Class “Utils”

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Name* | *Data type* | *Default value* | *Description* |
|  | logger | Logger | NULL | logger |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | getLogger | String | get logger |
| 2 | getCurrentDate | Date | tìm ngày hiện tại |
| 3 | encryptKey | String | mã hóa thông tin |

*Parameter*:

* message: thông điệp

*Exception*:

* Không

**Method**

* Không

**State**

Không

Class “API”

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Name* | *Data type* | *Default value* | *Description* |
| 1 | logger | Logger | NULL | logger |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | get | String | get request |
| 2 | post | String | post request |

*Parameter*:

* url : API path
* data: dữ liệu trong request

*Exception*:

* Không

**Method**

* Không

**State**

Không

# Design Considerations

***<Describe issues which need to be addressed or resolved before attempting to devise a complete design solution. Remember that, you have to refactor your source code to strictly follow the final design>***

## Goals and Guidelines

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

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

*Guidelines include coding guidelines and conventions.*

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

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

Goals:

- An emphasis on speed versus memory use

-Working properly

Guidelines:

Installation

- Java 11 or later: Install in https://www.oracle.com/java/technologies/downloads/

- JUnit5 for unit test

- Install JavaFX SDK

-mysql-connector-java.jar downnload in https://mvnrepository.com/artifact/mysql/mysql-connector-java

- Database: MySQL

script in: TKXDPM.KSCQ.20211-17\Construction\EcobikeRental\assets\db\ecobike.sql

## Architectural Strategies

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

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

*• Use of a particular type of product (programming language, database, library, commercial off-the-shelf (COTS) product, etc.)*

*• Reuse of existing software components to implement various parts/features of the system*

*• Future plans for extending or enhancing the software*

*• User interface paradigms (or system input and output models)*

*• Hardware and/or software interface paradigms*

*• Error detection and recovery*

*• Memory management policies*

*• External databases and/or data storage management and persistence*

*• Distributed data or control over a network*

*• Generalized approaches to control*

*• Concurrency and synchronization*

*• Communication mechanisms*

*• Management of other resources*

## Coupling and Cohesion

**COUPLING**

* Control Coupling

|  |  |  |
| --- | --- | --- |
| Related modules | Description | Improvement |
| controller | Trong phương thức processTransaction() của controller truyền vào tham số điều khiển cho giao dịch rent và return | Tách ra thành 2 phương thức riêng cho 2 trường hợp rent và return |

* Stamp Coupling

|  |  |  |
| --- | --- | --- |
| Related modules | Description | Improvement |
| controller | Phương thức calculateFee() truyền vào tham số RentBike nhưng không sử dụng toàn bộ dữ liệu của nó | Chỉ truyền vào tham số amount được sử dụng trong phương thức |

**COHESION**

* Coincidental cohesion

|  |  |  |
| --- | --- | --- |
| Related modules | Description | Improvement |
| utils | Các phương thức getToday(), md5() là các phương thức có mục đích riêng không liên quan, được sử dụng ở nhiều nơi |  |

* Temporal cohesion:

|  |  |  |
| --- | --- | --- |
| Related modules | Description | Improvement |
| views.screen | các phương thức khởi tạo trong ReturnTransactionScreenHandler khởi tạo tất cả các thành phần hiển thị thông tin thẻ và nội dung giao dịch | tách riêng các khởi tạo cho từng thành phần ra thành method setTransactionInfo(), setCardInfo() và gọi đến trong hàm khởi tạo |

## Design Principles

1. **Single Responsibilty**

|  |  |  |
| --- | --- | --- |
| Related modules | Description | Improvement |
| subsystem.interbank | Lớp InterbankSubsystemController chịu trách nhiệm: điều khiển luồng dữ liệu, chuyển đổi dữ liệu. Khi dữ liệu thay đổi thì lớp cũng thay đổi | Tách ra làm 2 class |
| views | Lớp TransactionHandler đang chịu trách nhiệm hiển thị cả 2 loại giao dịch cho mượn và trả xe | Tách ra làm 2 class riêng cho Rent và Return |

1. **Open/Closed**

|  |  |  |
| --- | --- | --- |
| Related modules | Description | Improvement |
| controller | Lớp ReturnBikeController, RentBikeController khi muốn tính phí mượn/trả xe theo kiểu khác thì cần sửa lại code trong calculateFees() | Tạo interface FeesCalculator có các method calculateFees(). ReturnBikeController chỉ cần khởi tạo đối tượng là interface này với instance là loại tính phí mong muốn |
|  |  |  |

1. **Liskov Substitution**

|  |  |  |
| --- | --- | --- |
| Related modules | Description | Improvement |
| entity.media | Các lớp con kế thừa từ BaseController không ghi đè phương thức đã có trong lớp cha, có thể thay thế lớp BaseController trong mọi tình huống |  |

1. **Interface Segregation**

|  |  |  |
| --- | --- | --- |
| Related modules | Description | Improvement |
| subsystem | InterbankInterface có 2 phương thức cho pay, refund. Trong tương lai, nếu xuất hiện hệ thống ngân hàng không cho phép refund, cho phép Nếu cần thì tách nhỏ Interface getBalance thì refund này sẽ bị dư thừa khi implement | Khi phát sinh yêu cầu thì cần tách nhỏ Interface |

1. **Dependency Inversion**

|  |  |  |
| --- | --- | --- |
| Related modules | Description | Improvement |
| entity.transaction | Lớp Transaction đang phụ thuộc chặt chẽ vào CreditCard. CreditCard là một thành phần cụ thể, trong tương lai có thể xuất hiện loại thẻ khác | Tạo một abstract class cho các hình thức thanh toán. CreditCard kế thừa abstract class này |

## Design Patterns

*N/A*