HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

School of Information and communications technology

Software Design Document

Version 1.0

**EcoBikeRental**

Subject: Software Development

Group 8:

Trần Thị Hằng - 20176748

Dương Thị Huê - 20176772

Đỗ Minh Thông - 20176881

Phạm Nhật Linh – 20184285

*Hanoi, November 2020*

Table of Contents

Table of Contents 1

1 Introduction 3

1.1 Objective 3

1.2 Scope 3

1.3 Glossary 3

1.4 References 3

2 Overall Description 4

2.1 General Overview 4

2.2 Assumptions/Constraints/Risks 4

2.2.1 Assumptions 4

2.2.2 Constraints 4

2.2.3 Risks 5

3 System Architecture and Architecture Design 6

3.1 Architectural Patterns 6

3.2 Interaction Diagrams 6

3.3 Analysis Class Diagrams 6

3.4 Unified Analysis Class Diagram 6

3.5 Security Software Architecture 6

4 Detailed Design 7

4.1 User Interface Design 7

4.1.1 Screen Configuration Standardization 7

4.1.2 Screen Transition Diagrams 7

4.1.3 Screen Specifications 7

4.2 Data Modeling 7

4.2.1 Conceptual Data Modeling 7

4.2.2 Database Design 7

4.3 Non-Database Management System Files 8

4.4 Class Design 8

4.4.1 General Class Diagram 8

4.4.2 Class Diagrams 8

4.4.3 Class Design 8

5 Design Considerations 10

5.1 Goals and Guidelines 10

5.2 Architectural Strategies 10

5.3 Coupling and Cohesion 11

5.4 Design Principles 11

5.5 Design Patterns 11

**List of Figures**

Fig 1. General Use Case Diagram 6

Fig 2. Communication Diagram for Deduct money from card 8

Fig 3. Communication Diagram for Return Bike 9

Fig 4. Communication Diagram for Rent Bike 9

Fig 5. Communication Diagram for Select Dock marker 10

Fig 6. Communication Diagram for View Bike or Station information 10

Fig 7. Sequence Diagram for Deduct money from card 11

Fig 8. Sequence Diagram for Return Bike 11

Fig 9. Sequence Diagram for Rent Bike 12

Fig 10. Sequence Diagram for Select Dock marker 12

Fig 11. Sequence Diagram for View Bike or Station information 13

Fig 12. Analysis Class Diagram for Deduct money from card 13

Fig 13. Analysis Class Diagram for Return Bike 14

Fig 14. Analysis Class Diagram for View Bike or Station information 14

Fig 15. Analysis Class Diagram for Rent Bike 15

Fig 16. Analysis Class Diagram for Select Dock marker 15

Fig 17. Unified Analysis Class Diagram 16

**List of Tables**

No table of figures entries found.

# Introduction

## Objective

This SDD is written for the purpose of giving the audience a clear view about the design of the software. The document’s intended audience is anyone who is interested in designing software.

## Scope

1.2.1 Product name: **EcoBikeRental Software (Eco-Bike-Rental is how we read)**

1.2.2 Explain:

The software is for users to rent and return bikes automatically. EcoBikeRental is a 24/7 platform-independent system which allows novice users to user without any training. User must have an account to use the system. The software allows user to enter barcode or directly choose bike to rent and choose any bike station to return bike, use credit card for payment, and show detailed information of station and bike.

1.2.3 Application:

Nowadays, the need for using bike is higher than ever. Using bike is not only environmentally friendly, but also a very effective way of exercising. The main drawback of this need is that not everyone has a bike, or has any intention of buying one. How about renting public bike for a relative cheap price? Introduce to our software. EcoBikeRental provides a quick and convenience way to rent bike. It helps to reduce employees, saves money and time, and very convenience. It is very easy to use. With a lot of stations and bikes available, it satisfies the need of bike rental service especially in Eco Park Township.

## Glossary

We assume that the reader of this document has relatively good base knowledge about computer/software in general. Still, the document will be written in general-audience-friendly way that most reader can understand. Scholarly terms, if any, in this document will be briefly explained after it has been used.

## References

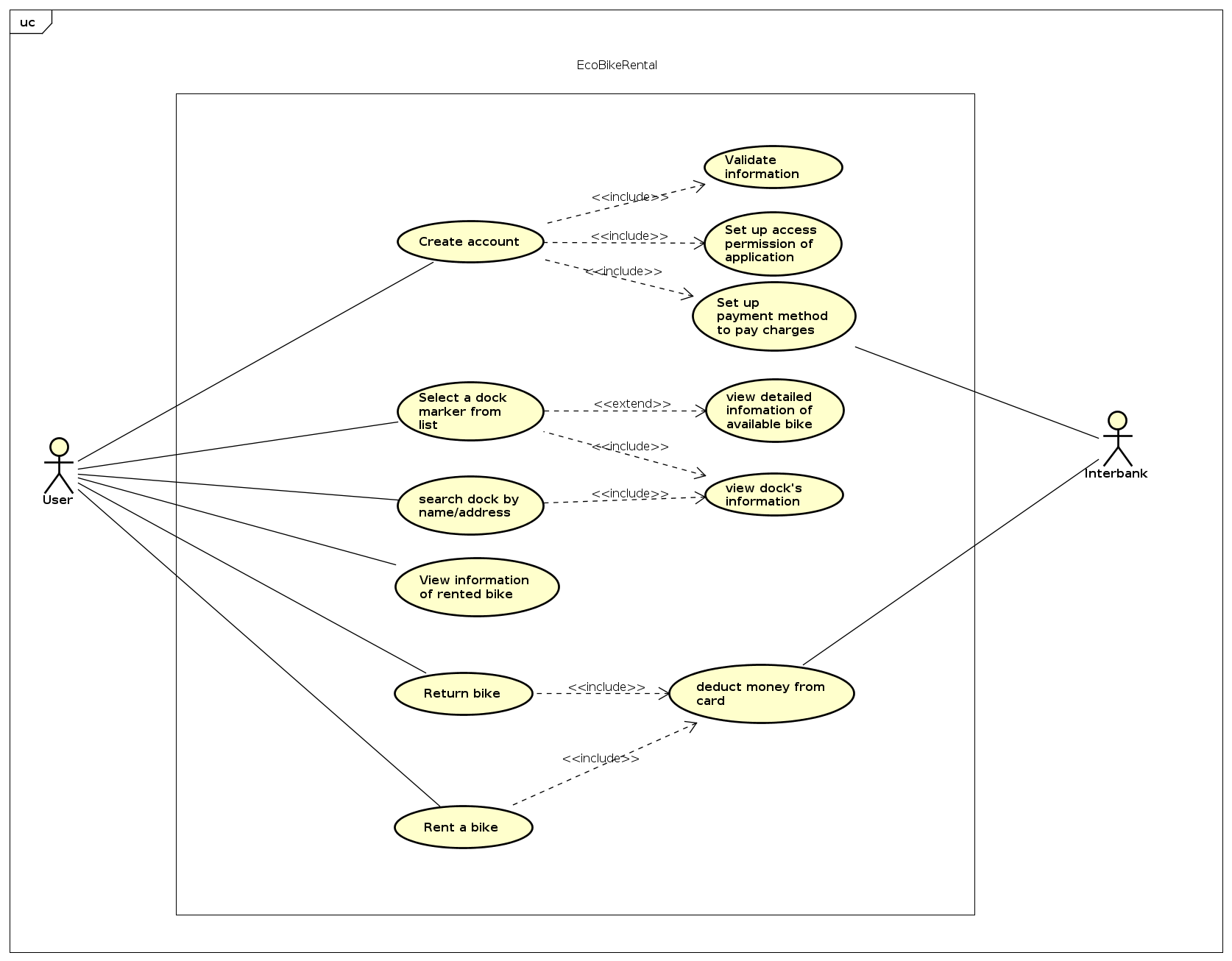
Centers for Medicare & Medicaid Services. (n.d.). *System Design Document Template.* Retrieved from Centers for Medicare & Medicaid Services: https://www.cms.gov/Research-Statistics-Data-and-Systems/CMS-Information-Technology/XLC/Downloads/SystemDesignDocument.docx

# Overall Description

## General Overview

About the system, we have some characteristics that make the apps resemble e-commercial website/software: An interface for interacting with user; user request by clicking on the interface and then the request is processed by system controller; we have a database (remote) to store any kind of data; any data-related request or change will be queried in the database; the change in database is then reflected in the UI (user interface). As you can see, there are three main components in the system: the UI, controller and data model. We choose this software to be a desktop application. We choose the three-layer architecture to be our design approach. The design architecture helps to separate different components and better organize the codebase.

Here is a high-level diagram to help you understand the core of our design:



*Fig 1. General Use Case Diagram*

## Assumptions/Constraints/Risks

### Assumptions

User that use the software should have a good connection to the Internet. Also, our software is a desktop application, so the user also must have a laptop/desktop with an OS (we recommended 64 Bit Microsoft Windows 8 or later; macOS 10.13 or later; or any Linux distribution that supports running application) to run the apps. About the system requirement, we would say 2 GB RAM minimum, 8 GB RAM recommended; for storage 2.5 GB and another 1 GB for caches minimum, solid-state drive with at least 5 GB of free space recommended; require latest version of JRE; 1024×768 minimum screen resolution, 1920×1080 is a recommended screen resolution.

### Constraints

* Less than 2GB RAM; JRE version<8; Low storage may cause the software to run incorrectly, or cannot start running at all.
* Implicitly stated, ideally, the response time for any tasks, with a moderate load, within the system is 1 second. But in case of peak load, a response time in the interval of 2 seconds is admissible.
* Weak or no internet connection may cause the software to run improperly.

### Risks

* No risks to be discussed, yet.

# System Architecture and Architecture Design

Architecture Design steps:

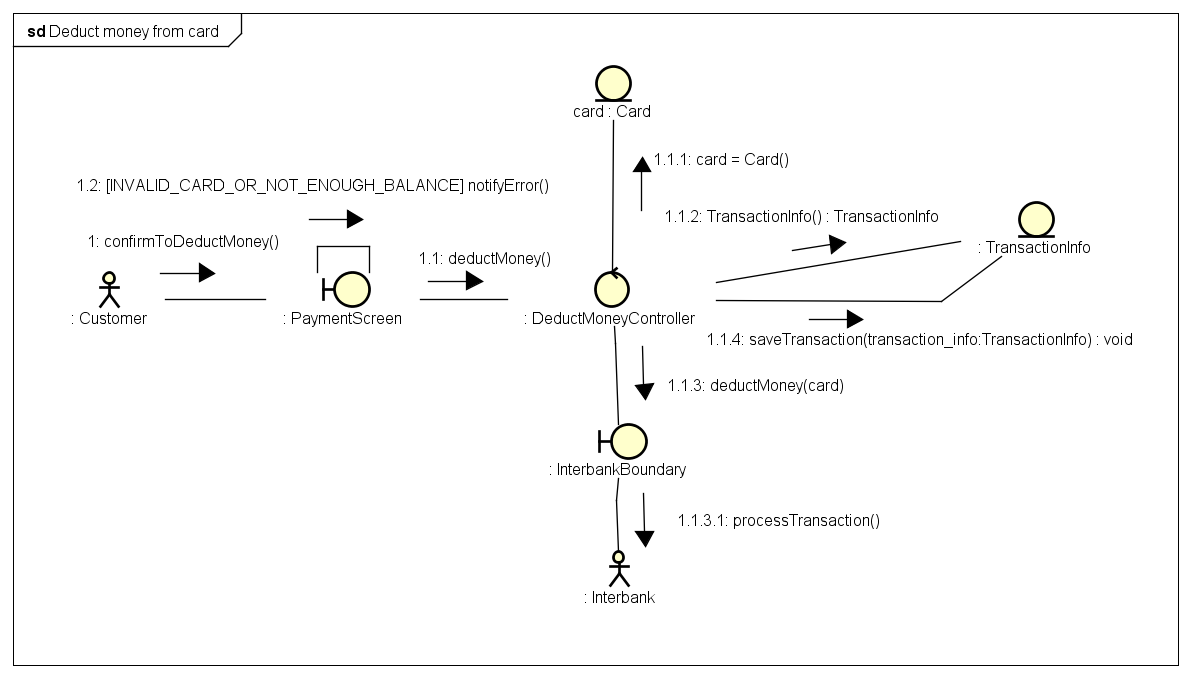
1. Find out software components -> use cases
2. Find out Interaction between use cases
3. Find out Relationship between use cases
4. Draw UML Diagram includes: interaction diagram and analysis class diagram

## Architectural Patterns

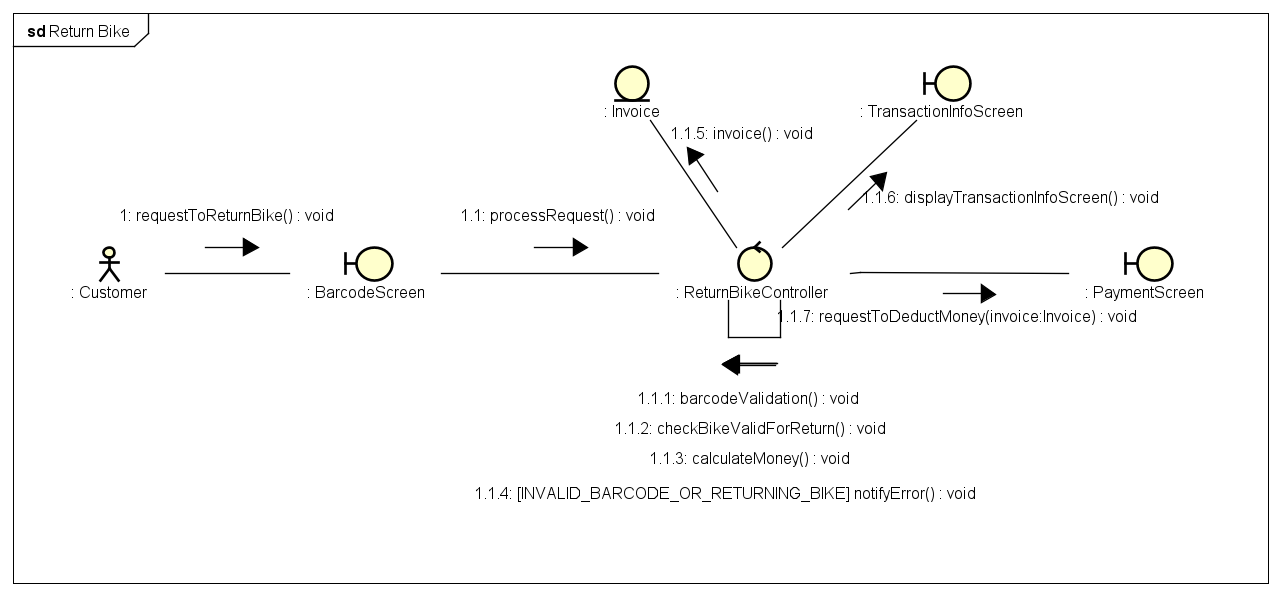
In our project, we use 3-tier architectural pattern. There are many benefits of separating an application into tiers and the most important thing is it allows us to update a specific part of an application independently of the other parts

## Interaction Diagrams

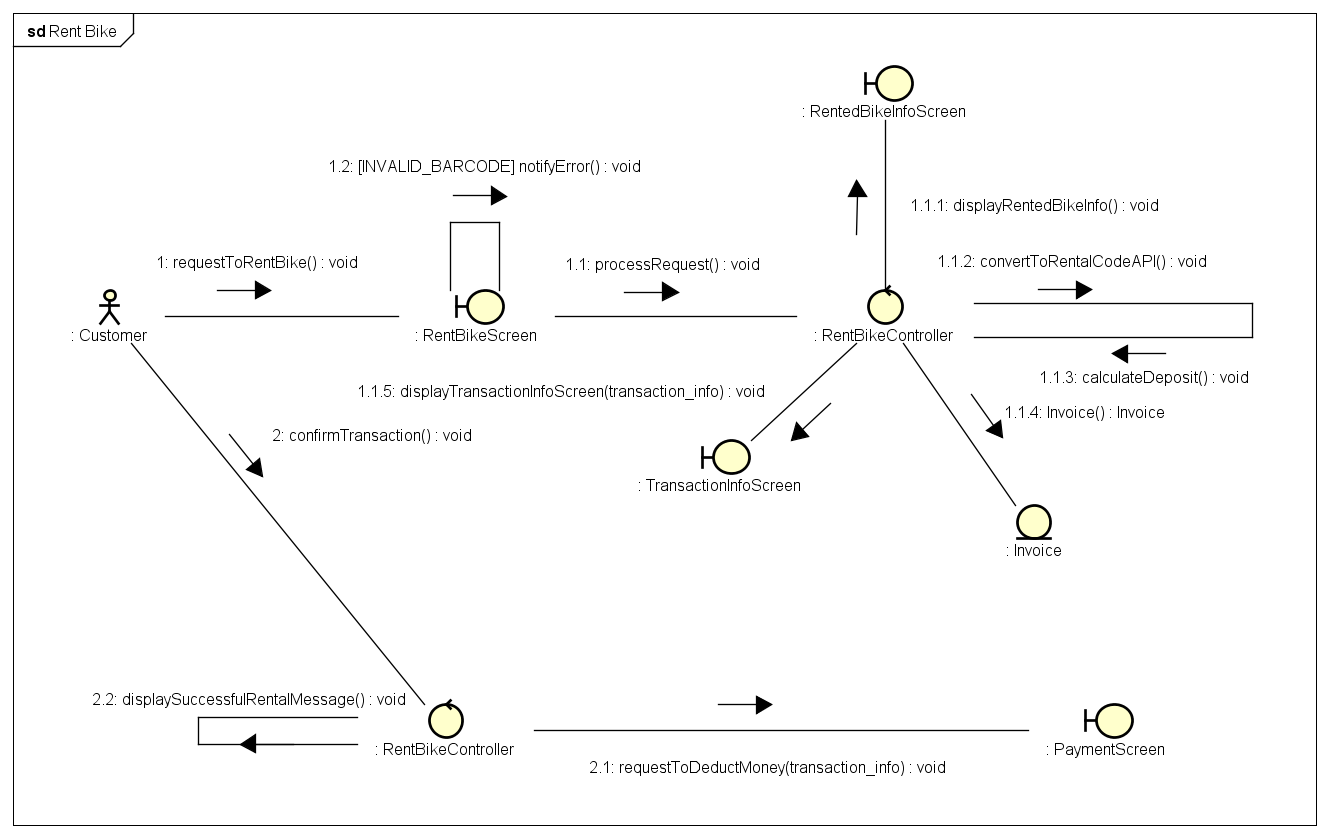
**3.2.1. Communication Diagrams**



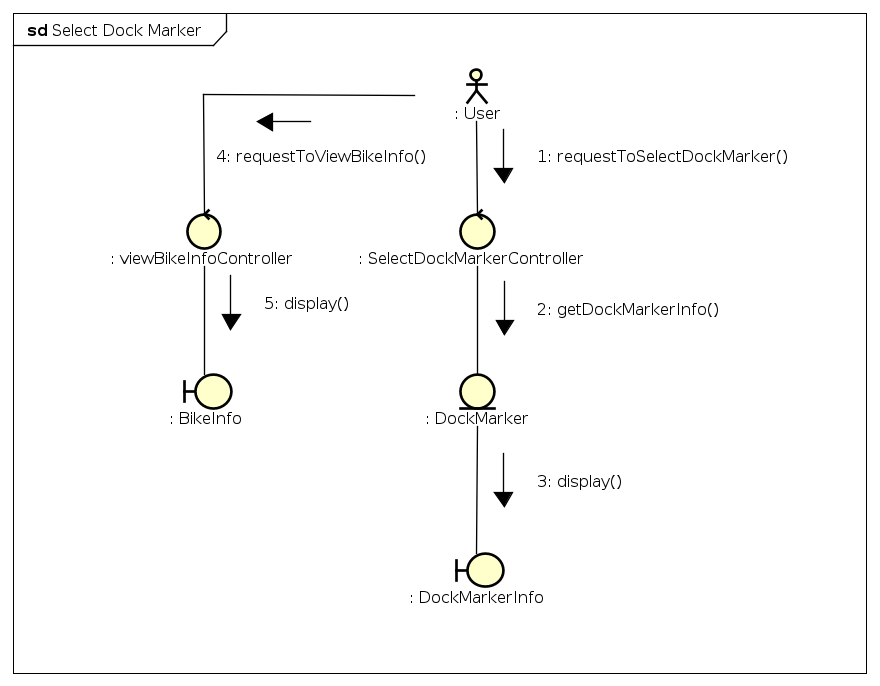
*Fig 2. Communication Diagram for Deduct money from card*



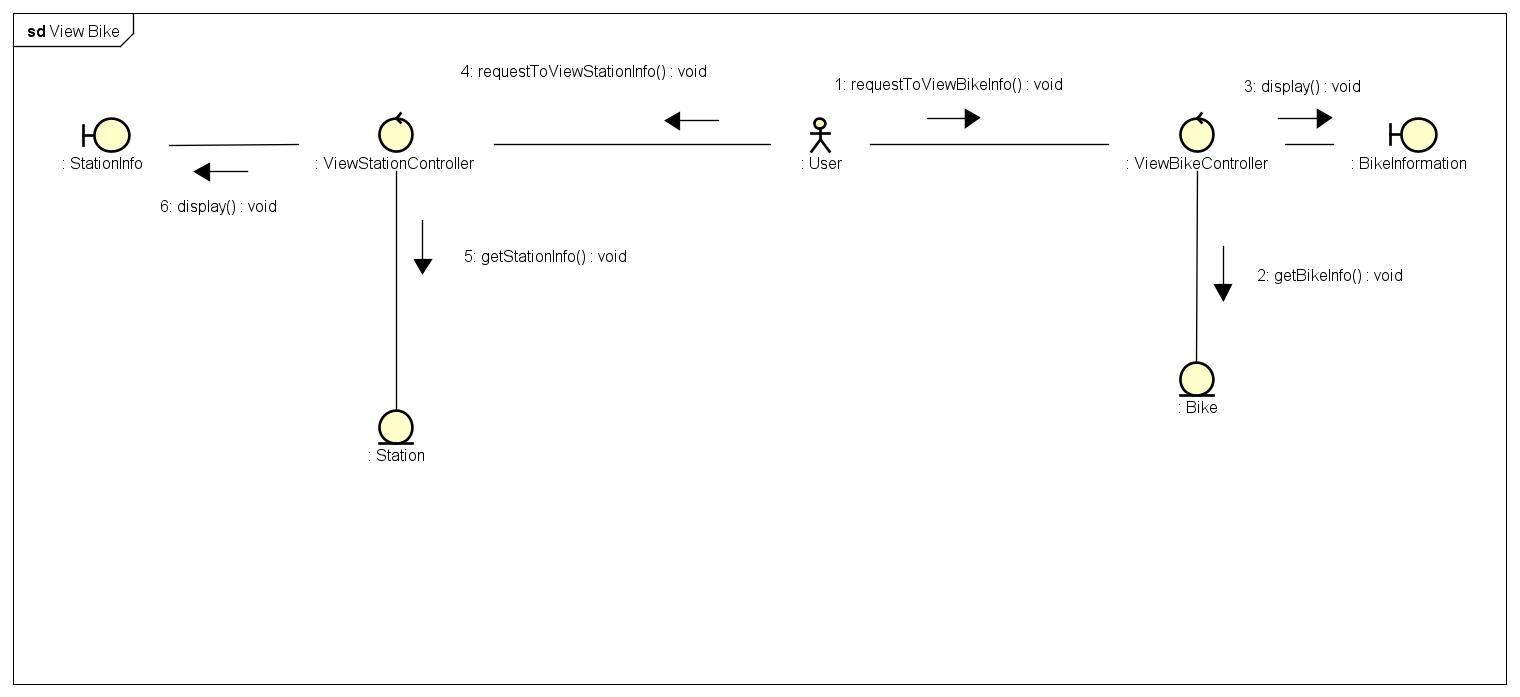
*Fig 3. Communication Diagram for Return Bike*



*Fig 4. Communication Diagram for Rent Bike*

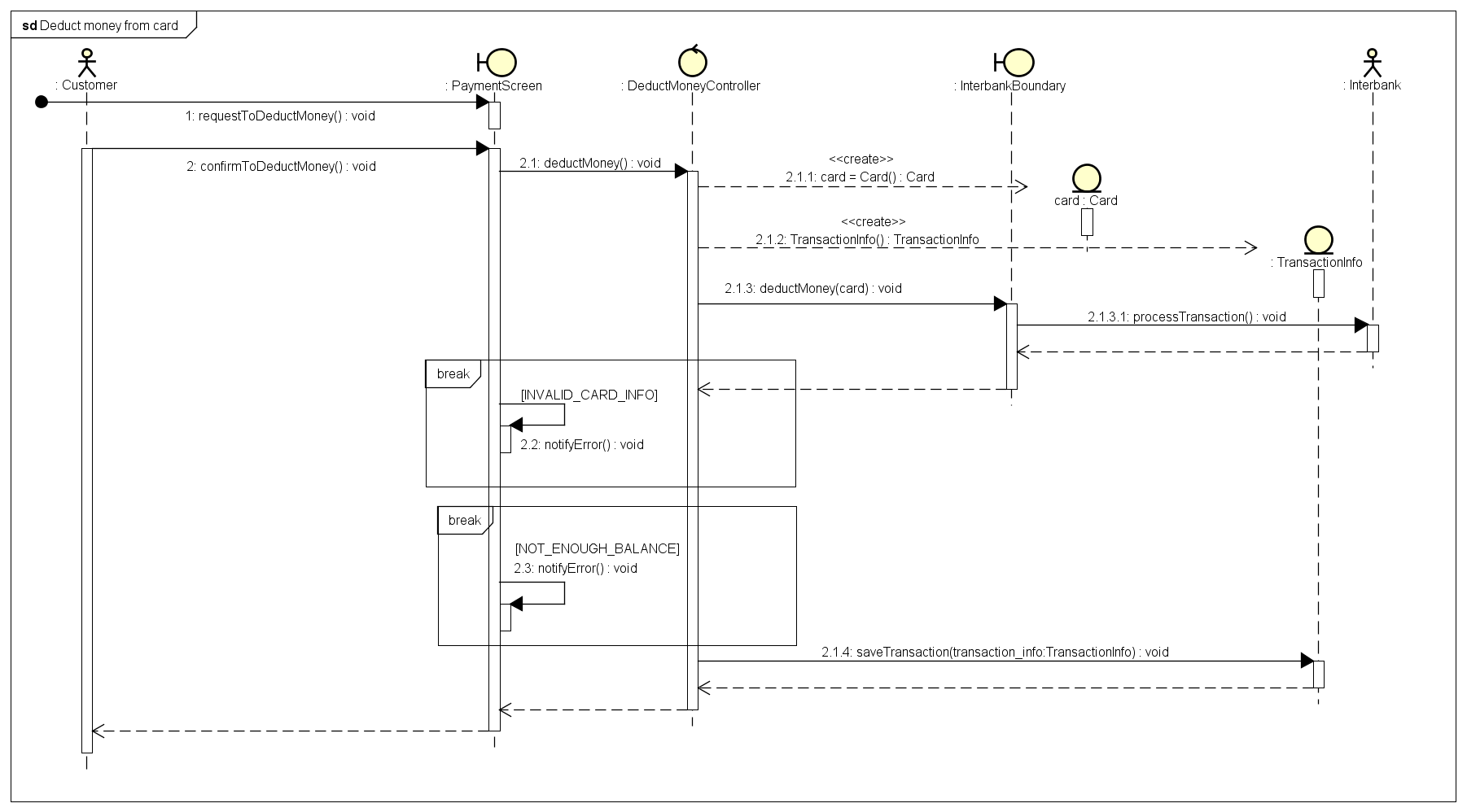


*Fig 5. Communication Diagram for Select Dock marker*

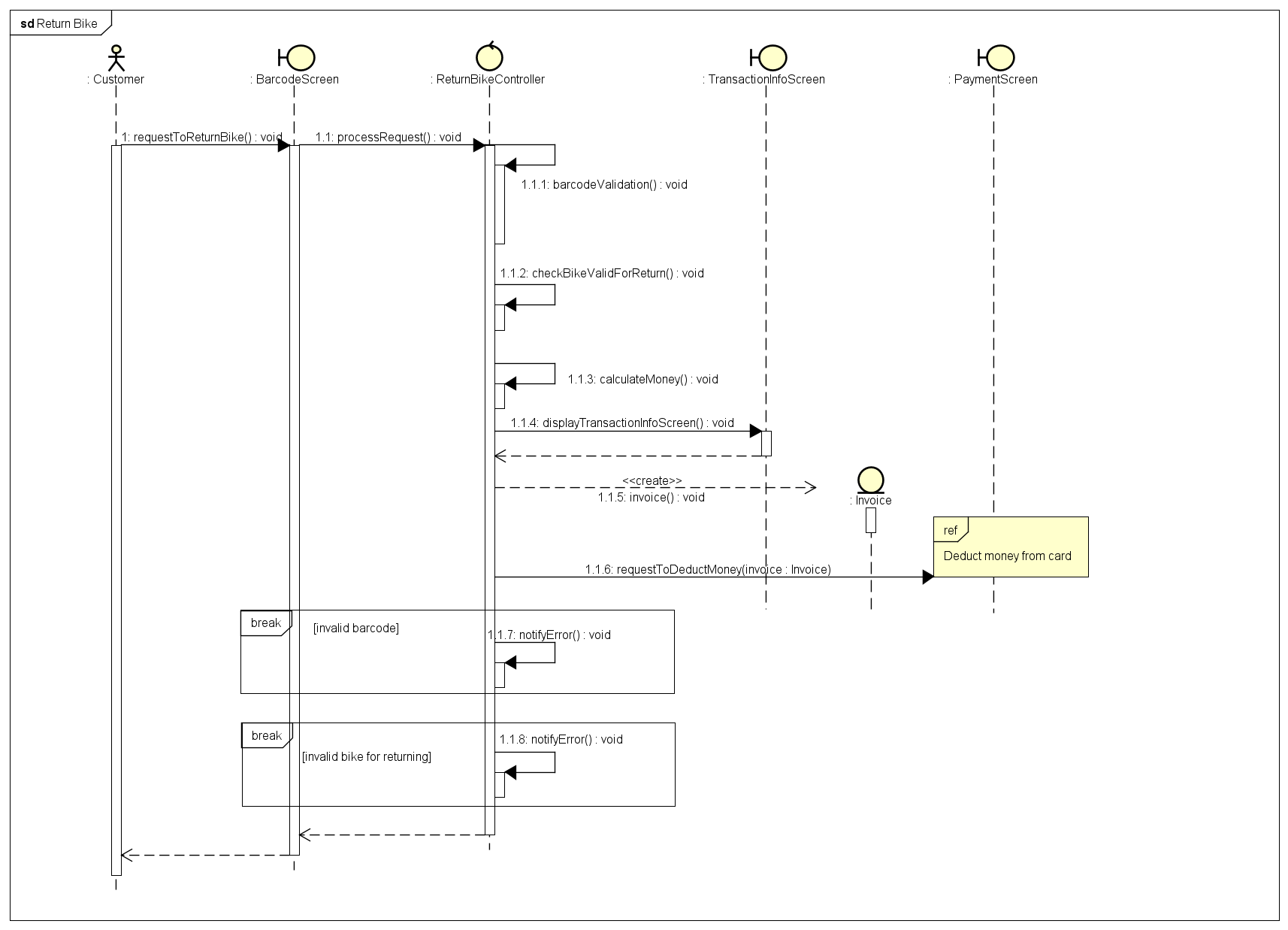


*Fig 6. Communication Diagram for View Bike or Station information*

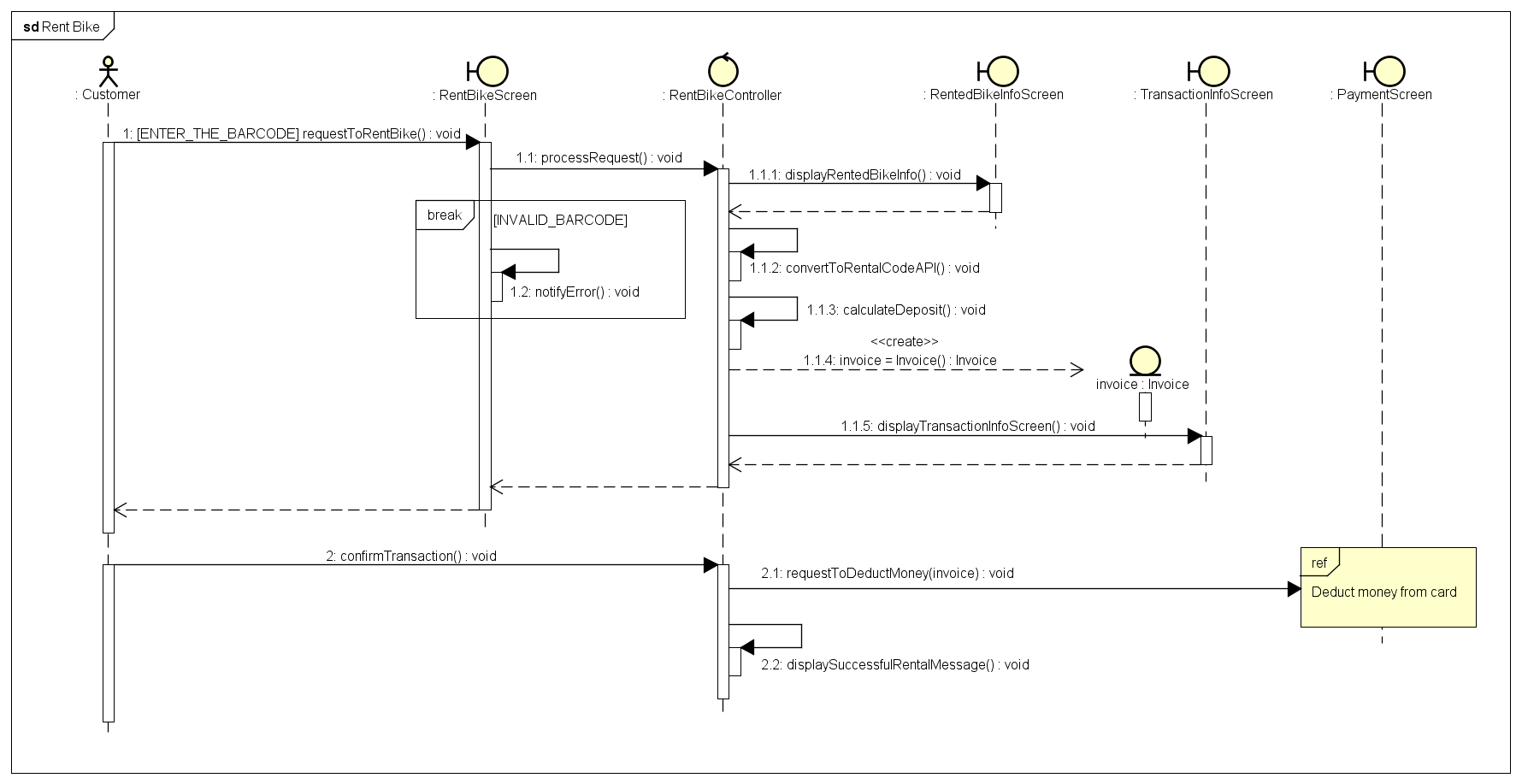
* + 1. **Sequence Diagrams**

****

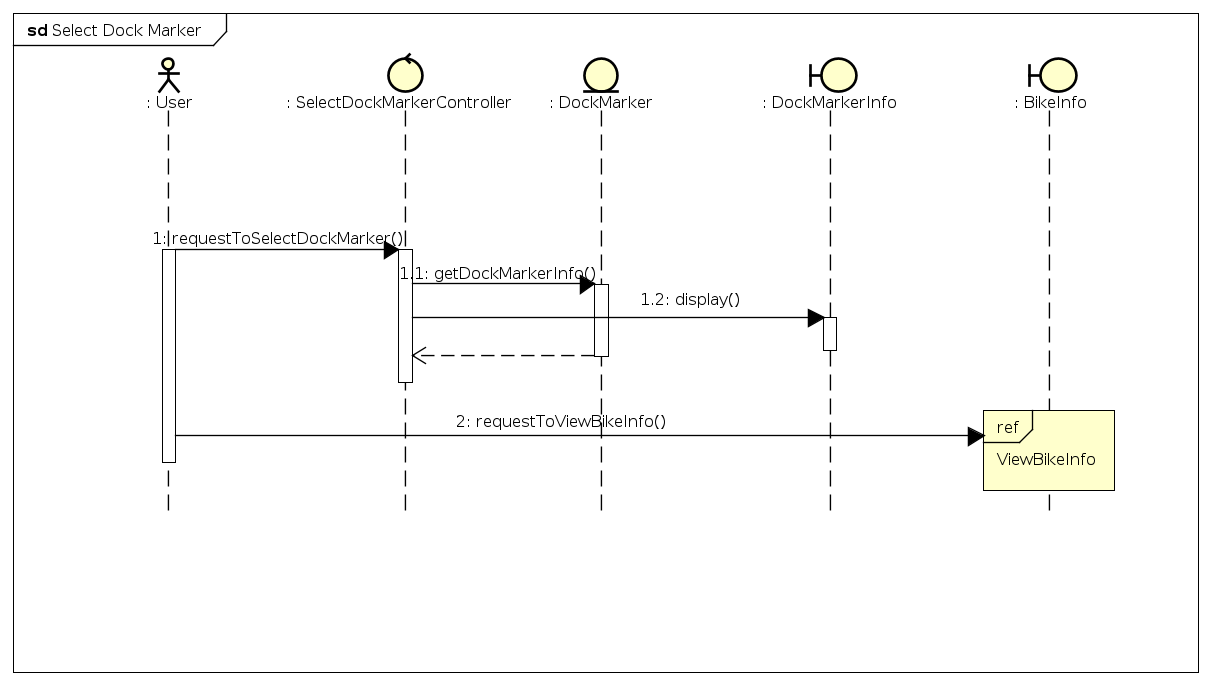
*Fig 7. Sequence Diagram for Deduct money from card*

****

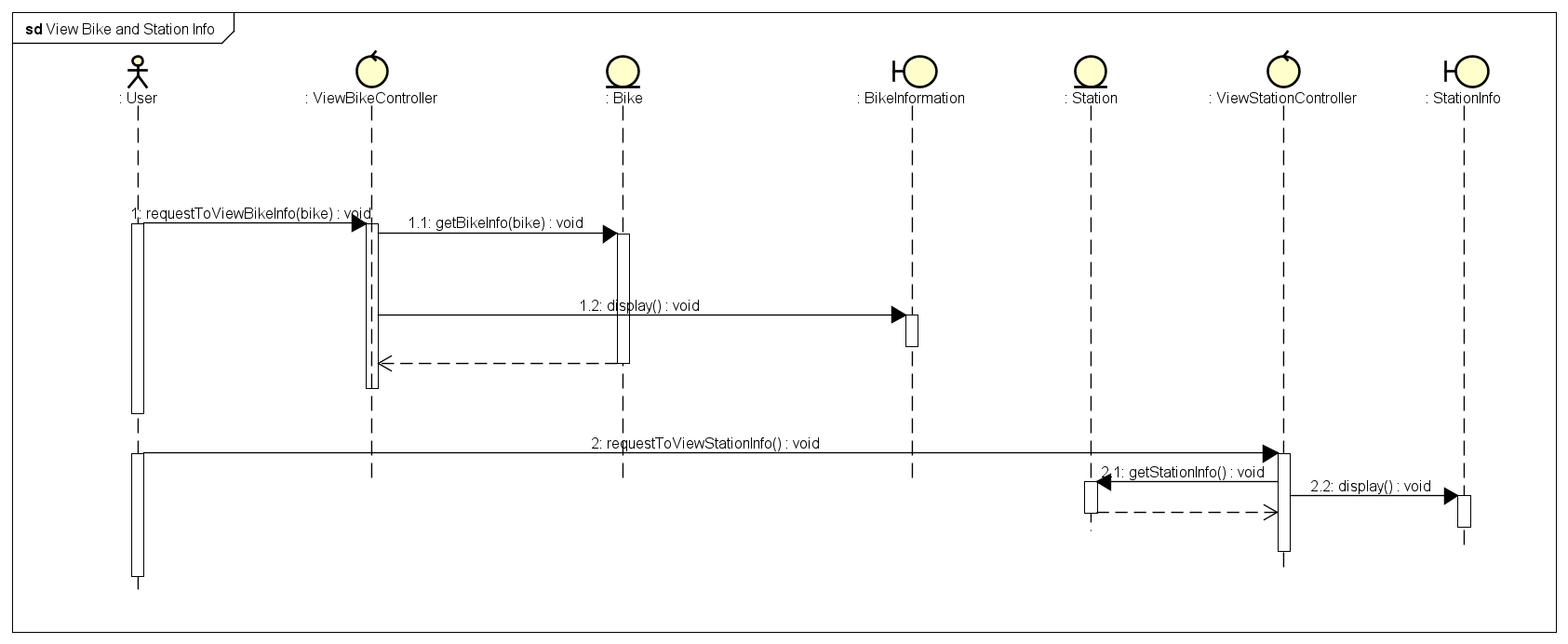
*Fig 8. Sequence Diagram for Return bike*

****

*Fig 9. Sequence Diagram for Rent bike*

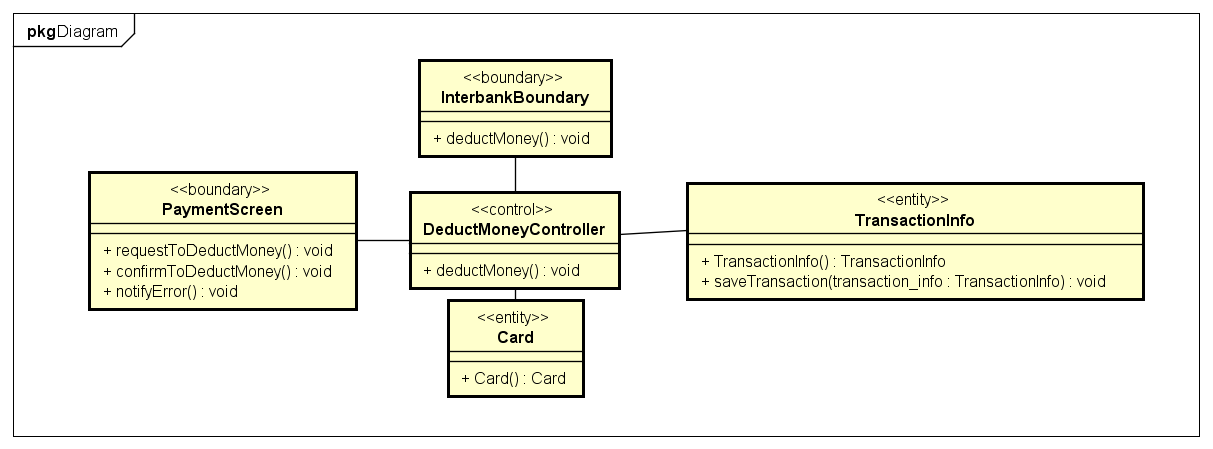
****

*Fig 10. Sequence Diagram for Select Dock marker*

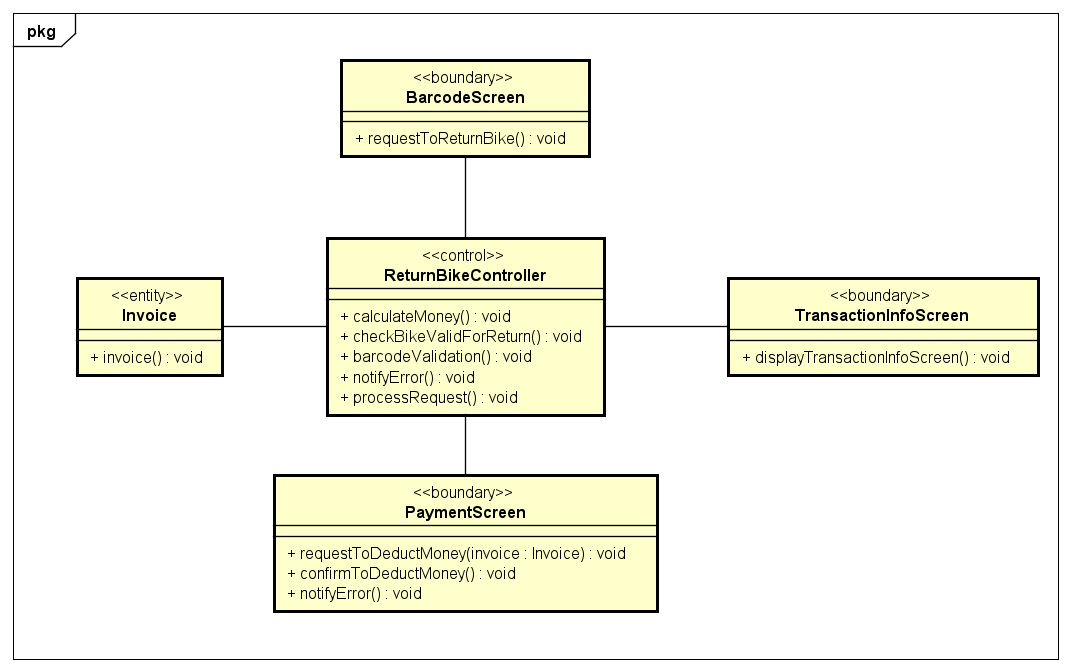
****

*Fig 11. Sequence Diagram for View Bike or Station information*

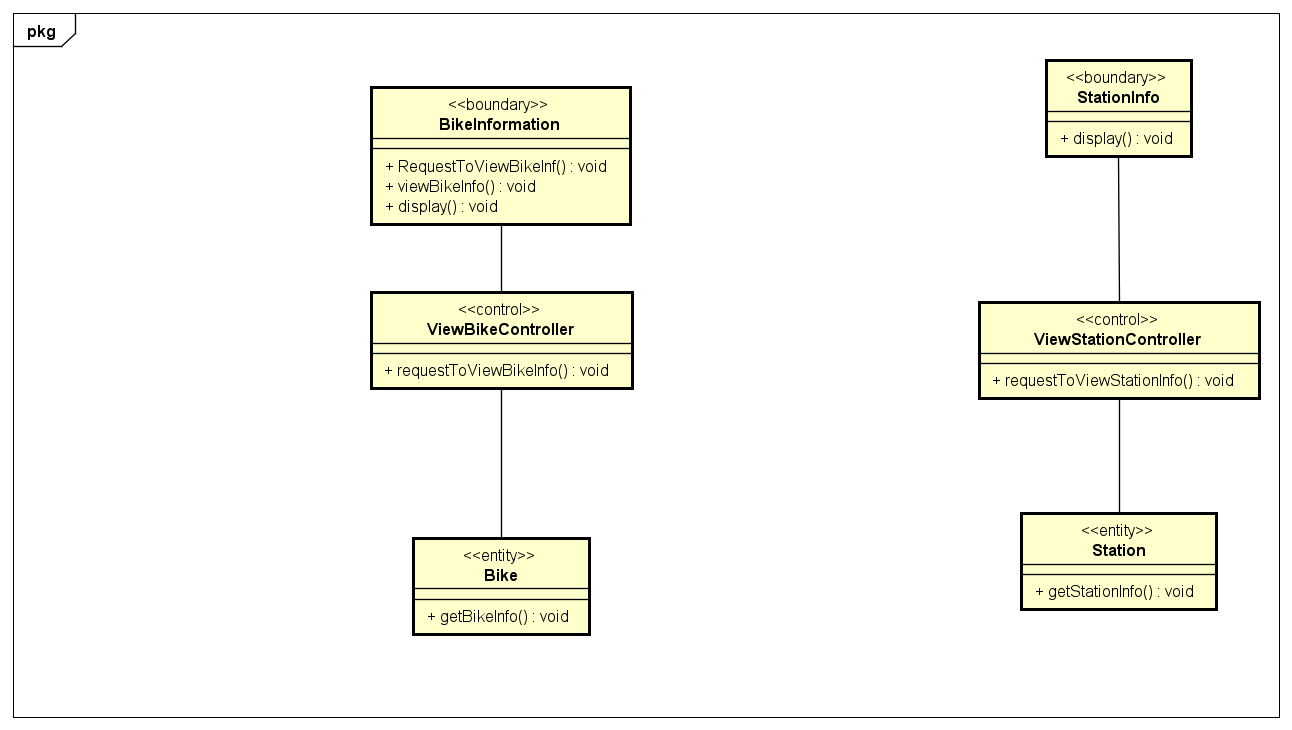
## Analysis Class Diagrams

****

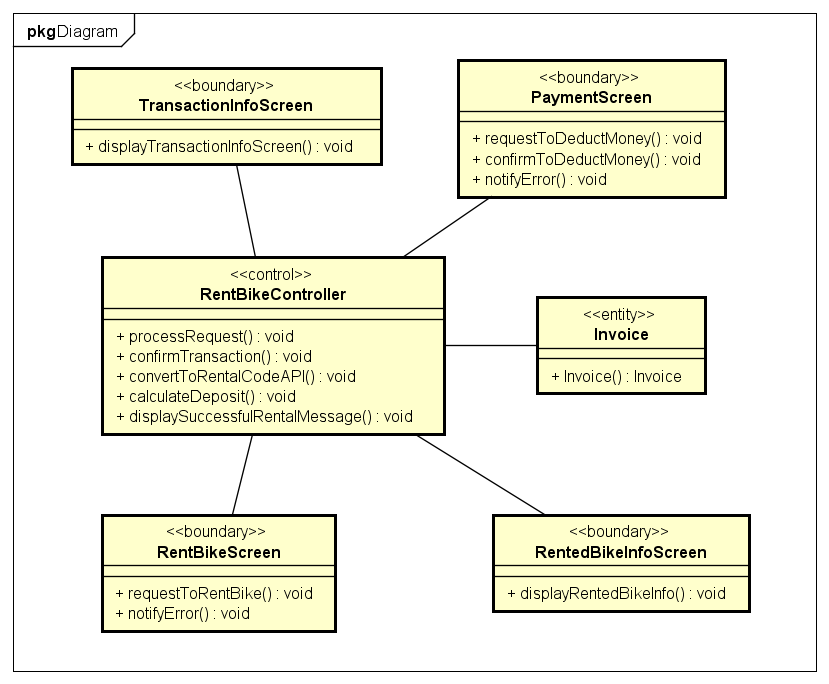
*Fig 12. Analysis Class Diagram for Deduct money from card*

****

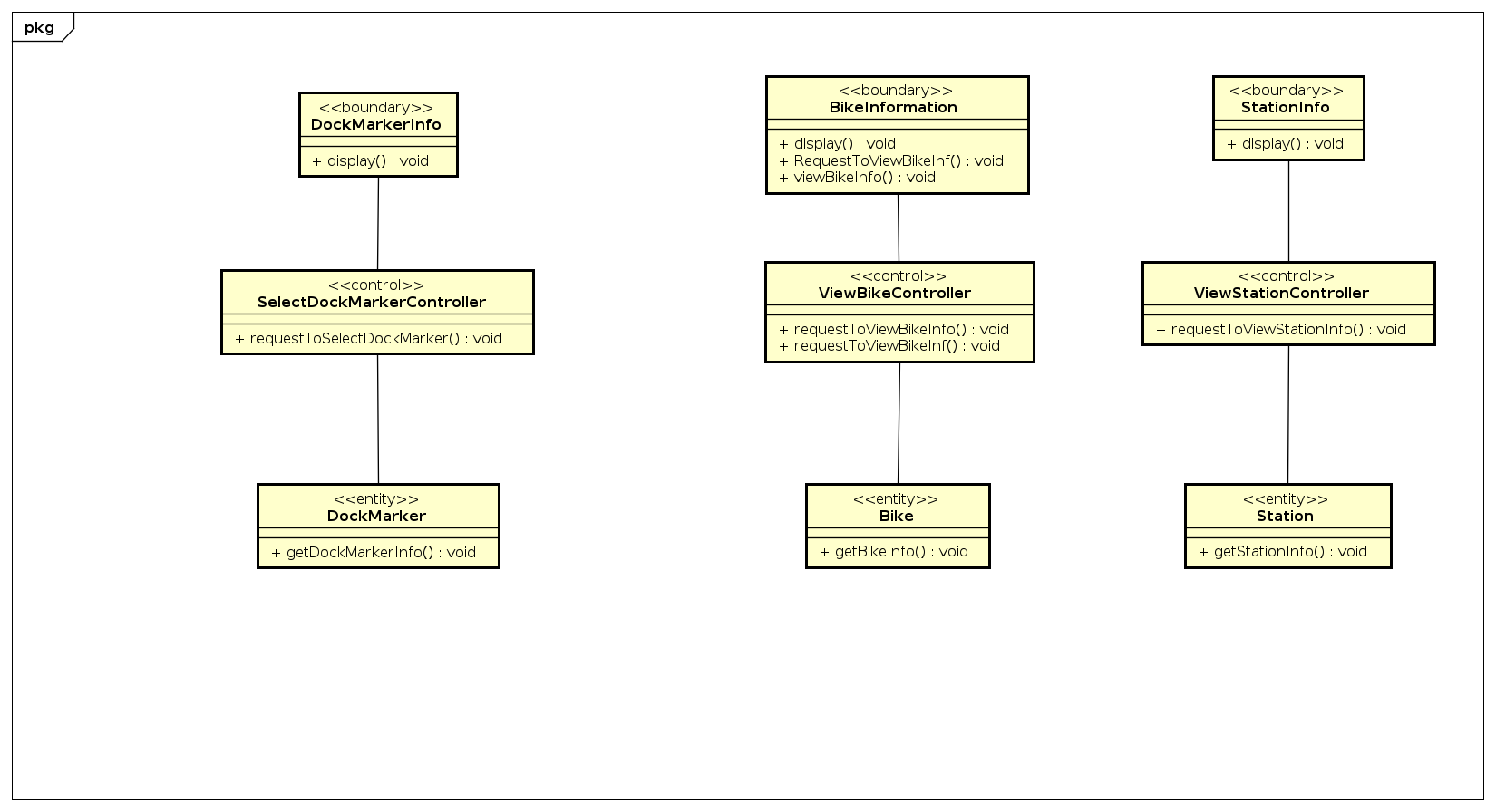
*Fig 13. Analysis Class Diagram for Return bike*

****

*Fig 14. Analysis Class Diagram for View Bike or Station information*

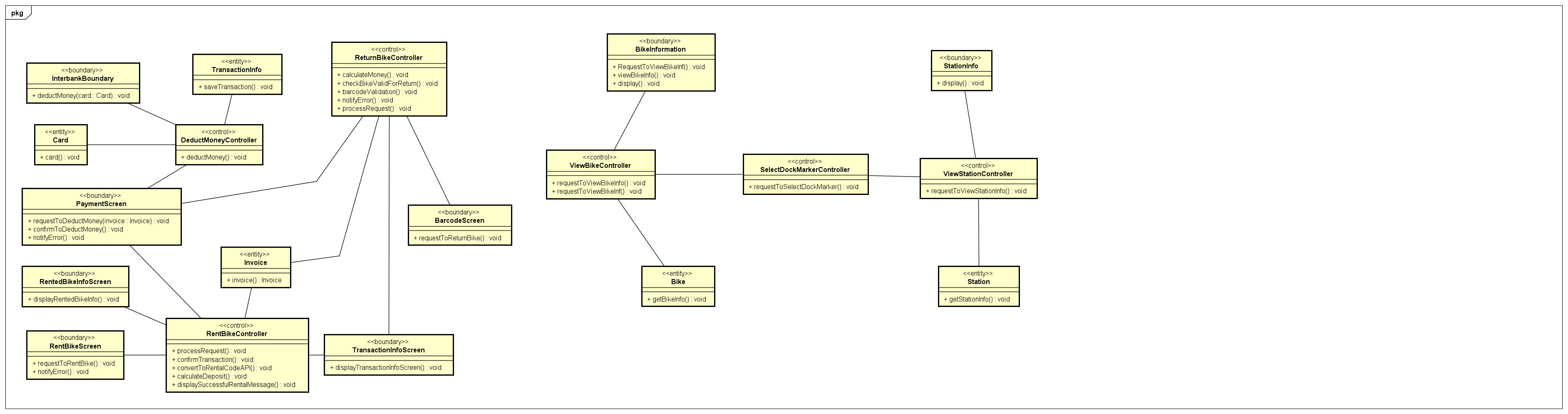
****

*Fig 15. Analysis Class Diagram for Rent bike*

****

*Fig 16. Analysis Class Diagram for Select Dock marker*

## Unified Analysis Class Diagram



*Fig 17. Unified Analysis Class Diagram*

## Security Software Architecture

In this project, we will not consider features such as user authentication (e.g., sign up, sign in, sign out), but we focus on features related to bike renting and returning.

# 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>*

## Data Modeling

### Conceptual Data Modeling

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

### Database Design

#### Database Management Systems

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

#### Logical Data Model

<

* *Show the process to design database from E-R diagram*
* *Show the diagram of DB design*

*>*

#### Physical Data Model

<

*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).*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *#* | *PK* | *FK* | *Column name* | *Data type* | *Default value* | *Mandatory* | *Description* |
| 1 | x |  | ProductID |  |  |  |  |
| 2 |  | x | CategoryID |  |  |  |  |

*You may add indexing, trigger, view, etc.*

*Give the database script*>

## 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

### Class Diagrams

#### Class Diagram for Package A

#### Class Diagram for Subsystem B

…

### Class Design

#### Class “SampleClass1”

<SampleClass1 class image in UML>

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Name* | *Data type* | *Default value* | *Description* |
| 1 |  |  |  |  |
| 2 |  |  |  |  |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 |  |  |  |
| 2 |  |  |  |

*Parameter*:

* x: Default value, description
* y: Default value, description

*Exception*:

* AException if …
* BException if …

**Method**

How to use parameters / attributes

Flowchart / Sequence diagram if the method has a complex/special algorithm

**State**

State diagram if any

#### Class “SampleClass2”

…

# Design Considerations

## Goals and Guidelines

**Goals:**

* Bring a good looking and good experience for users
* The response time for the system is 1 second at normal and 2 seconds during a peak load

**Guidelines**

* Observe java convention in coding, OOP principles
* Avoid hash code
* Explain code, write java doc for maintenance

## Architectural Strategies

Our design decisions focus on reusing components, unified system following

+ Programing Language: java

+ Database: MySQL

+ Unified on error detection and recovery

We always toward save memory and spaces, also speed up response time and nice looking. In the future, we plan to extend software: have site for admin to add, delete bike, statistics, business strategies. These targets make us concentrate totally on architectural design.

## Coupling and Cohesion

## Design Principles

We design simple classes that means a class should have only one job, one responsibility. Object or entities are open for extension but close for modification. We also use interfaces, abstract classes. We put all class with same properties into one package to manage easily. Therefore, we can reuse source code, adapt any changing requirements.

## Design Patterns

We don’t use any design patterns .