HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

School of Information and communications technology

Software Requirement Specification

Version 1.0

<Project Name>

Subject: <Name of subject>

<Group Number>

<List of participants>

*Hanoi,* *<month, year>*

*<All notations inside the angle bracket are not part of this document, for its purpose is for extra instruction. When using this document, please erase all these notations and/or replace them with corresponding content as instructed>*

*<This document, written by Asst. Prof. NGUYEN Thi Thu Trang, is used as a case study for student with related courses. Any modifications and/or utilization without the consent of the author is strictly forbidden>*

Table of contents

Table of contents 1

1 Introduction 2

1.1 Objective 2

1.2 Scope 2

1.3 Glossary 2

1.4 References 2

2 Overall Description 3

2.1 Actors 3

2.2 Use case diagrams 3

2.3 Business processes 3

3 Detailed Requirements 4

3.1 Use case specification for “Use case 1” 5

3.2 Use case specification for “Use case 2” 6

4 Supplementary specification 8

4.1 Functionality 8

4.2 Usability 8

4.3 Reliability 8

4.4 Performance 8

4.5 Supportability 8

4.6 Other requirements 8

# Introduction

## Objective

This document is to provide information about EcobikeRental system, users and services that system provided

## Scope

### Product name: EcoBikeRental software

### 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. Users must have account to enter to system. Software allow user to enter barcode to rent bike and return bike, use creadit card for payment, show information of dock and bike.

### Software helps to reduce emloyees, saves money and time. It satisfies needs on bike rental service specially in Ecopark township. It is expected to serve 100 users at the same time without noticeable loss of performance and to operate in an average of 200 hours without failure. The system also can be repaired within 2 hours after any typical failure. The response time for the system is 1 second or 2 seconds during a peak load if it is not explicitly stated

## Glossary

## References

# Overall Description

< *Describe the general factors that affect the product and its requirements. This section does not state specific requirements. Instead, it provides a background for those requirements, which are defined in section 3, and makes them easier to understand*. *In a sense, this section tells the requirements in plain English for the consumption of the customer. Section 3 will contain a specification written for the developers*>

## Actors

## Use case diagrams

*<Both general and lower-level use case diagrams>*

## Business processes

*<If the software has business processes, please use activity diagrams to visualize them. Please note that they are different from the activity diagrams for the use case flow of events>*

# Detailed Requirements

***<Please provide use case specifications here. You can refer the below template>***

<*This section contains all the software requirements at a level of detail sufficient to enable designers to design a system to satisfy those requirements, and testers to test that the system satisfies those requirements. Throughout this section, every stated requirement should be externally perceivable by users, operators, or other external systems. These requirements should include at a minimum a description of every input (stimulus) into the system, every output (response) from the system and all functions performed by the system in response to an input or in support of an output. The following principles apply:*

1. *Specific requirements should be stated with all the characteristics of a good SRS*
2. *correct*
3. *unambiguous*
4. *complete*
5. *consistent*
6. *ranked for importance and/or stability*
7. *verifiable*
8. *modifiable*
9. *traceable*
10. *Specific requirements should be cross-referenced to earlier documents that relate*
11. *All requirements should be uniquely identifiable (usually via numbering like 3.1.2.3)*
12. *Careful attention should be given to organizing the requirements to maximize readability (Several alternative organizations are given at end of document)*

*Before examining specific ways of organizing the requirements it is helpful to understand the various items that comprise requirements as described in the following subclasses. This section reiterates section 2, but is for developers not the customer. The customer buys in with section 2, the designers use section 3 to design and build the actual application.*

*Remember this is not design. Do not require specific software packages, etc unless the customer specifically requires them. Avoid over-constraining your design. Use proper terminology:*

*The system shall… A required, must have feature*

*The system should… A desired feature, but may be deferred til later*

*The system may… An optional, nice-to-have feature that may never make it to implementation.*

*Each requirement should be uniquely identified for traceability. Usually, they are numbered 3.1, 3.1.1, 3.1.2.1 etc. Each requirement should also be testable. Avoid imprecise statements like, “The system shall be easy to use” Well no kidding, what does that mean? Avoid “motherhood and apple pie” type statements, “The system shall be developed using good software engineering practice”*

*Avoid examples, this is a specification, a designer should be able to read this spec and build the system without bothering the customer again. Don’t say things like, “The system shall accept configuration information such as name and address”. The designer doesn’t know if that is the only two data elements or if there are 200. List every piece of information that is required so the designers can build the right UI and data tables*>

## Use case specification for “View bike information”

**Use Case “View bike information”**

1. **Use case code**

UC005

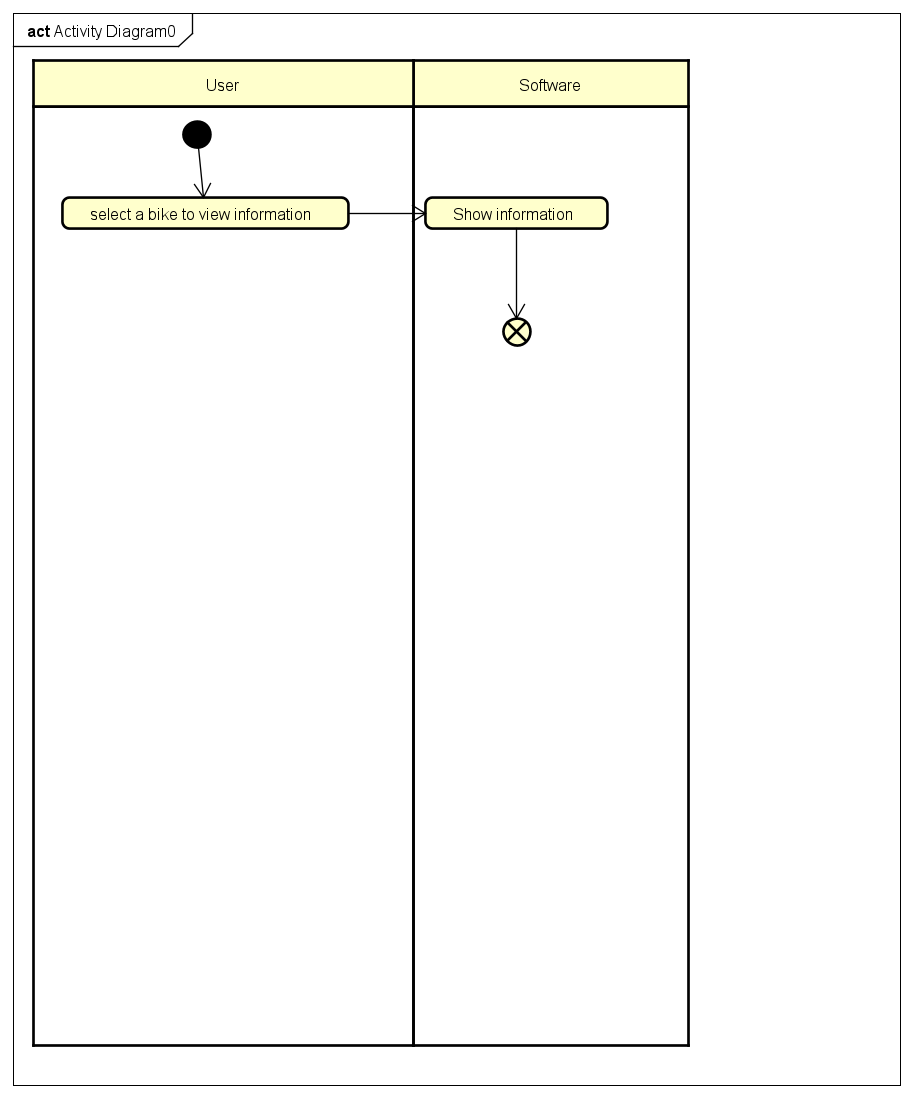
1. **Brief Description**

This use case describes the interaction between user and EcoRentalbike software when user wishes to view bike ‘s information

1. **Actors**
   1. **User**
   2. **Software**
2. **Preconditions**

User selected a dock

1. **Basic Flow of Events**
2. User select bike to view information
3. Software show information of that bike and end use case
4. **Alternative flows**
5. **Activity diagrams**



1. **Input data**

**Table A-Input data of View station information**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Data fields** | **Description** | **Mandatory** | **Valid condition** | **Example** |
| **1.** | **bike id** | **Id of the bike in database** | **yes** | **Id is in database** | **STB01** |

1. **Output data**

Table B-Output data of View bike information

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Data fields** | **Description** | **Display format** | **Example** |
|  | Bike ‘s id |  |  | STB01 |
|  | Bike type |  |  | Standard bike |
|  | Bike ‘s coeficient price |  |  | 1 |
|  | Electric motor ‘s battery percentage | If bike type is e-bike show this information |  | 50% |
|  | Estimating time | How much time is left if this bike is e-bike |  | 1 hour 15 minutes |

1. **Postconditions**

User has viewed bike information

## Use case specification for “View station information”

**Use Case “View station information”**

1. **Use case code**

UC004

1. **Brief Description**

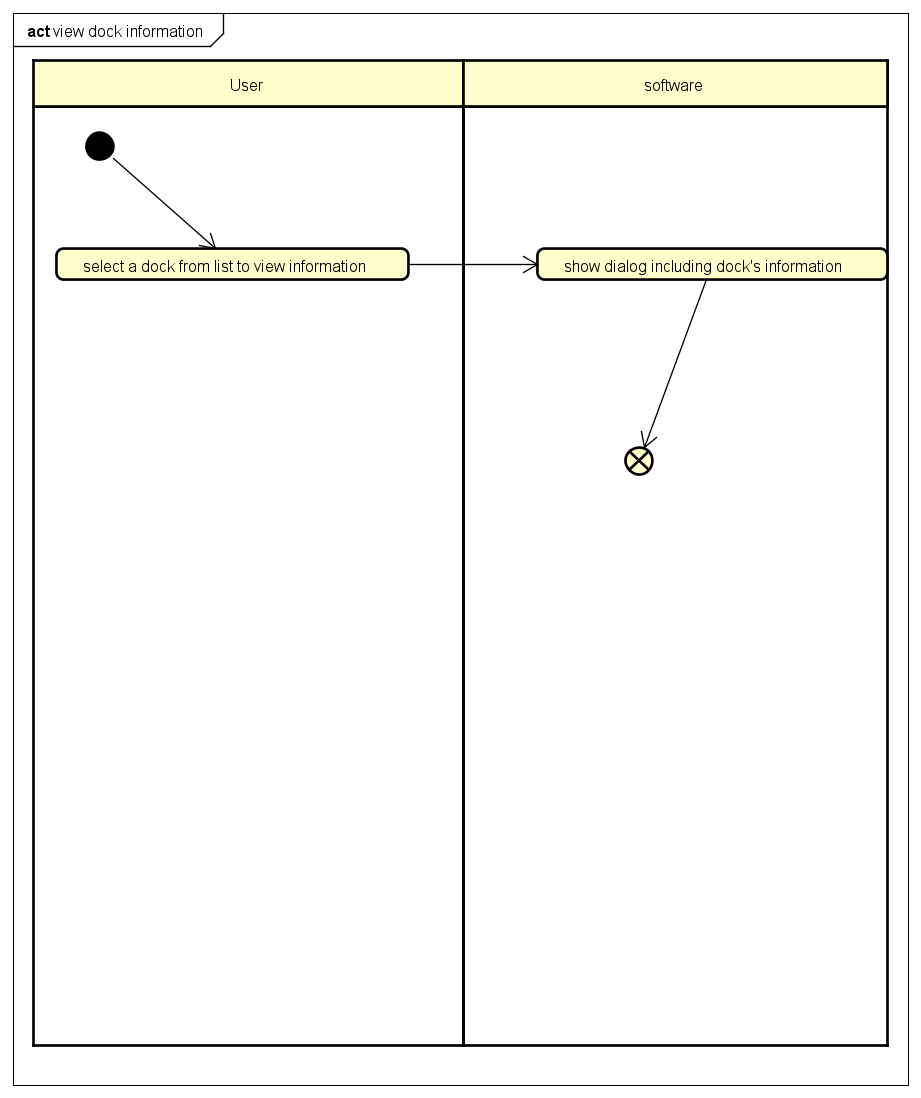
This use case describes the interaction between user and EcobikeRental software when user wishes to view information of the dock in detail

1. **Actors**
   1. **User**
   2. **Software**
2. **Preconditions: User signed in software**
3. **Basic Flow of Events**

1.User select a dock from list to view information

2.Software show information of the dock and end use case

1. **Alternative flows**
2. **Activity diagrams**



1. **Input data**

**Table A-Input data of View station information**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Data fields** | **Description** | **Mandatory** | **Valid condition** | **Example** |
| **1.** | **Dock id** | **Id of the dock in database** | **yes** | **Id is in database** | **DOCK01** |

1. **Output data**

Table B-Output data of View dock ‘s information

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Data fields** | **Description** | **Display format** | **Example** |
|  | Dock ‘s name |  | Bold text  Middle alignment | **Trương Định** |
|  | Address | Address of the dock |  | 1 Trương Định street, Hoang Mai district |
|  | Dock area |  |  | 700m2 |
|  | Number of available bikes |  |  | 60 |
|  | Number of empty docking |  |  | 50 |
|  | Distance | Distance from current user ‘s location to dock |  | 500m |
|  | Walking time | Walking time from user ‘s location to this dock |  | 10 minutes |
|  | List available bike | List available bikes: type and quantity coresponding to these bike |  | Standard bike: 10 |

1. **Postconditions**

User has viewed dock ‘s information

# Supplementary specification

*<Presenting other requirements if necessary, including non-functional requirements such as performance, reliability, usability, and supportability; or other technical requirements such as database system, used technology…>*

## Functionality

*<Functional requirements that are general to many use cases>*

## Usability

<*Requirements that relate to, or affect, the usability of the system. Examples include ease-of-use requirements or training requirements that specify how readily the system can be used by its actors*>

## Reliability

*<Any requirements concerning the reliability of the system. Quantitative measures such as mean time between failure or defects per thousand lines of code should be stated>*

## Performance

*<The performance characteristics of the system. Include specific response times. Reference related use cases by name>*

## Supportability

*<Any requirements that will enhance the supportability or maintainability of the system being built>*

## Other requirements

*<Descriptions of other requirements are located here>*