HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

School of Information and communications technology

Software Requirement Specification

Version 1.2

Eco Bike Rental

Subject: ITSS Software Development

Group 10

Nguyễn Thị Linh

Phạm Khánh Linh

Vũ Thị Thùy Linh

Bùi Thành Long

*Hanoi,* *July 2023*

*<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 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 3

2 Overall Description 4

2.1 Survey 4

2.2 Overall requirements 4

2.3 Business process 4

3 Detailed Requirements 5

3.1 Use case 1 6

3.2 Use case 2 7

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

The objective of this Software Requirements Specification (SRS) document is to provide a comprehensive understanding of the requirements for the development of the EcoBikeRental System. The intended audience for this SRS includes the developers and testers, who will be involved in the development, testing and implementation of the system.

The SRS aims to serve as a reference guide and communication tool between the project team and the stakeholders, ensuring a clear understanding of the system's objectives, functionalities, and constraints. It will provide a basis for system design, development, testing, and validation activities.

## Scope

The software product to be produced is the EcoBikeRental System, which is a web-based platform for bike rental services. It will facilitate the process of renting and returning bicycles to multiple docking stations. The system will support various features and functionalities to provide a seamless user experience.

The EcoBikeRental System have following functionalities:

* See a list of docking stations and bikes on each dock
* Search a dock by name or address
* Rent bikes by enter the barcode of the bike in the feature of bike renting
* Calculate rental fee based on the selected bike type and renting duration
* Process payment transactions using credit cards
* Allow users to choose an available dock station for returning bikes

The development of the EcoBikeRental System seeks to achieve the following benefits, objectives, and goals:

* Provide a convenient and user-friendly platform for bike rental services
* Enable efficient management and tracking of bike rentals and returns
* Improve customer experience by providing real-time availability of bikes and docking stations
* Automate payment processing for secure and seamless transactions
* Support scalability and accommodate a large number of concurrent users
* Enhance operational efficiency and reduce manual efforts in managing the rental process

In summary, the EcoBikeRental System is intended to be used by bike rental service providers who wish to automate their operations and provide a user-friendly interface for their customers. The system aims to streamline the bike rental process, enhance customer satisfaction, and improve operational efficiency for the rental service providers.

## Glossary

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

| ***No*** | ***Term*** | ***Explanation*** | ***Example*** | ***Note*** |
| --- | --- | --- | --- | --- |
| 1 | token | A piece of data created by server, and contains the user's information, as well as a special token code that user can pass to the server with every method that supports authentication, instead of passing a username and password directly. | JSON Web Token (JWT) | Compact, URL-safe and usable especially in web browser single sign-on (SSO) context. |
| 2 | … |  |  |  |

## References

[*Problem Statement*](https://www.dropbox.com/sh/evtc0mz0ps1kzgo/AADq20c3QMo7JCmY_YYYrnWca/CapstoneProject?dl=0&preview=EcoBikeRental-ProblemStatement-EN.pdf&subfolder_nav_tracking=1)

# Overall Description

## Survey

The EcoBikeRental System involves two main actors who interact with the system:

* User: The user represents individuals who want to rent bikes from the EcoBikeRental service. They interact with the system to choose a bikes, rent bikes, and return them after use. Users provide necessary personal information, select payment methods, and adhere to the rental terms and conditions.
* Interbank: The interbank actor represents the banking system or financial institutions that handle the payment transactions between the user and the EcoBikeRental service. It ensures secure and reliable payment processing by facilitating the transfer of funds from the user's account to the service provider.

## Overall requirements

Use case diagram:

A picture containing text, screenshot, diagram, line

Description automatically generated

## Business process

*<Sequence of use cases, e.g. Use case 1 then use case 2, if a condition matched, do the use case 3… You should visualize the process with activity diagrams>*

# Detailed Requirements

<*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 “View Dock”

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Use Case “View Dock”**   1. **Use case code**   UC001   1. **Brief Description**   This use case describes the interaction between user and EcoBikeRental System when user wishes to see information about a certain docking station   1. **Actors**   User   1. **Preconditions**   None   1. **Basic Flow of Events** 2. The software display a list of dock stations on screen 3. The user select a certain dock 4. The software process to get information about the selected dock (include detailed information of available bikes) 5. The software displays information of the dock 6. **Alternative flows**   None   1. **Input data**   Table A-Input   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **No** | **Data fields** | **Description** | **Mandatory** | **Valid condition** | **Example** | |  | Name | Name of docking station | Yes |  | HUST Station |  1. **Output data**   Table B-Output   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **No** | **Data fields** | **Description** | **Display format** | **Example** | |  | Name | Name of docking station | A string | HUST station | |  | Address | Address of docking station | A string | 1st Dai Co Viet, Hai Ba Trung | |  | Dock Area | Area of docking station | - Float number  - Unit: m2 | 500 m2 | |  | Number of bikes | Number of available bikes in station | Non-negative integer | 10 | |  | Number of empty docks | Number of empty docking points | Non-negative integer | 5 | |  | Distance | Distance from user location to this dock | - Float number  - Unit: km | 0.5 km | |  | Time | Walking time from user location | - String display time  - Time format: hh:mm | 00:08 minutes |  1. **Postconditions**   None |

## Use case 2

<Use case specification of use case 2>

# 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>*