HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
School of Information and communications technology

Test Plan
Version 1.0

EcoBikeRental(EBR)
Subject: ITSS Software Development

Group 03
Trịnh Thu Hải - 20184255
Nguyễn Huy Hoàng - 20184265
Bùi Thanh Tùng - 20184324

Hanoi, Dec 2021

<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

1 Introduction

1.1. Objective

The purpose of this document is to present the testing plan of EcoBikeRental(EBR), intended for the developers of the system.This Test Plan only covers the test functions for the Testing Phase of the project.

1.2 Scope

The main target of this product is to automatically manage bikes in stations,as well as their customer interaction. The system will make use of a scanner for the purpose of recognizing barcodes on each bike, and from that information handles the business process, as well as correctly prompt and record different fees for each bike instance, namely standard bike, twin bike, standard e-bike, and twin e-bike.

Firstly, customers register an account on EcoBikeRental application, fill in appropriate information entries, then allow permissions where needed of the application, and finally set up a working payment method to pay charges, either through linking with inter-bank or an e-wallet. Upon bootstrapping, the position of the user and the location of nearby docking stations will be visible on a map on the screen. By navigating on the map, as well as searching on the dock list, the customer can see available docks there currently, and information on each bike. If the customer finds a suitable bike for his/her need, by scanning the barcode on said bike, the system will proceed with the monetary procedure. To proceed with a payment, the application will first display the scanned bike information, then ask the customer to provide a payment method. The customer is required to deposit for at least 40% of the value of the bike. Upon agreeing to the payment details, the system will deduct the said amount from the customer's chosen monetary source, and finally open the bike's lock, and start counting used time.

During the renting period, the customer accesses relevant information of their renting on the system through the application, such as their renting bike type, current renting time, the amount to be paid, and the bike status. To return a bike, the user pushes the bike into an empty docking point of a dock, then closes the lock. The system will automatically calculate the fee, and return any remaining deposit if there is any.

1.3. Glossary

A.

Actor: a role played by a user or any other system that interacts with the subject.

B.

Barcode: a method of representing data in a visual, machine-readable form. Initially, barcodes represented data by varying the widths and spacings of parallel lines

C.

D.

Data: the quantities, characters, or symbols on which operations are performed by a computer, being stored and transmitted in the form of electrical signals and recorded on magnetic, optical, or mechanical recording media.

Docking station: Station that holds bikes.

Dock marker: mark of the map indicating a dock.

E.

F.

Flow of events: A sequence of events or actions within a use case.
G.
H.
I.
Information: is processed, organised and structured data.
Input: Data that is sent into a system or a program to be processed and executed.
Inter-bank: A bank to pay for transactions made by the customer.
J.
Java Runtime Environment (JRE): A set of software tools, which are included for development Java applications.
L.
Locker: A system that controls the opening and closing of physical locks and interacts with applications.
M.
N.
O.
P.
Q.
R.
S.
Software Development Kit (SDK): A set of software tools that allow some applications to be created for certain software packages or other similar development platforms
System: a hardware or software system, or combination, which has components as its structure and observable share data as its behavior
Scanner: A function of the software that takes barcode as input and checks bike information to display
Standard bike: a kind of bicycle that has 01 saddle, 01 pedal, and 01 rear seat in the back
Standard e-bike: a kind of bicycle that looks like a standard bike, but has an integrated electric motor for assisting propulsion and the rental fee costs 1.5 times that of a standard bike.
T.
Twin bike: a kind of bicycle that has 02 saddle, 02 pedal, and 01 rear seat with no integrated electric motor with rental fee costs 1.5 times that of a standard bike
U.
Use case: A sequence of actions a system performs that yields an observable result of value to a particular actor.
V.

1.4. 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

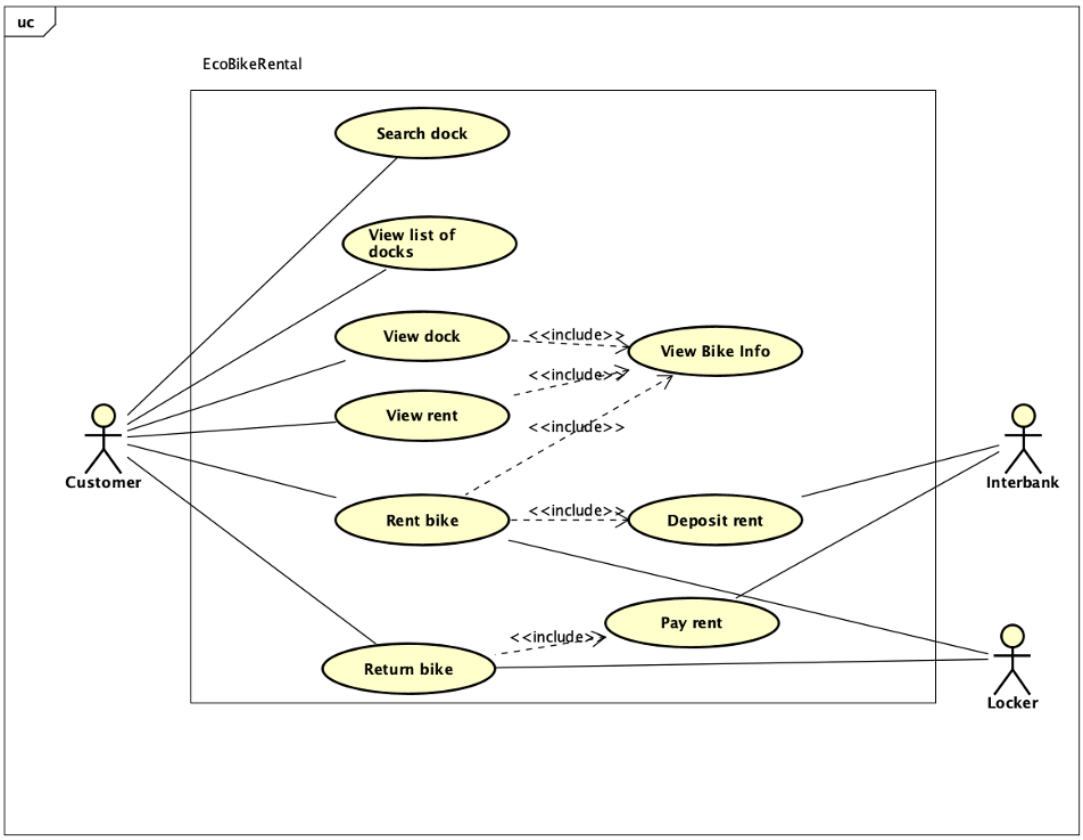
Trang, N. T. (2019, September). Problem Statement: Eco Bike Rental. Retrieved from: https://www.dropbox.com/sh/2llptvvm9atklen/AADGszPxE-IdkBrsRnaBYeQ6a/CapstoneProject2dl=0&preview=EcoBikeRental_ProblemStatement

2. Overall Description

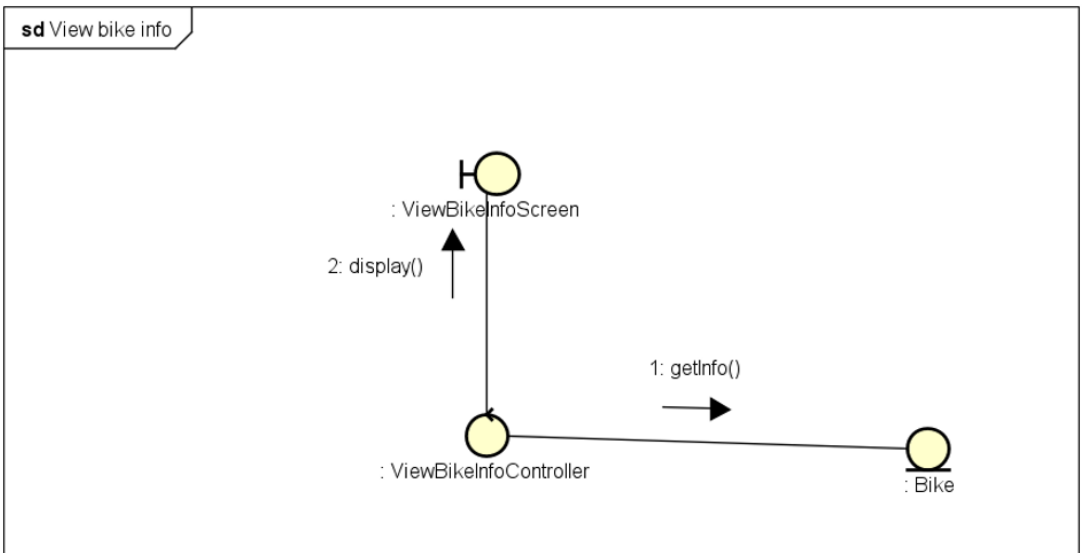
2.1. General Overview

The system is built with Java SDK to be used on multiple devices. The software will follow the client-server approach, where users will see information about bikes and docks on their devices. The requests will be sent to the server, where it will be proceeded and then sent back the result/error to the user interface. The system can also interact with outer devices and systems, namely the locker on dock stations for lock/unlocking the bikes, and the interbank API for payment processing.

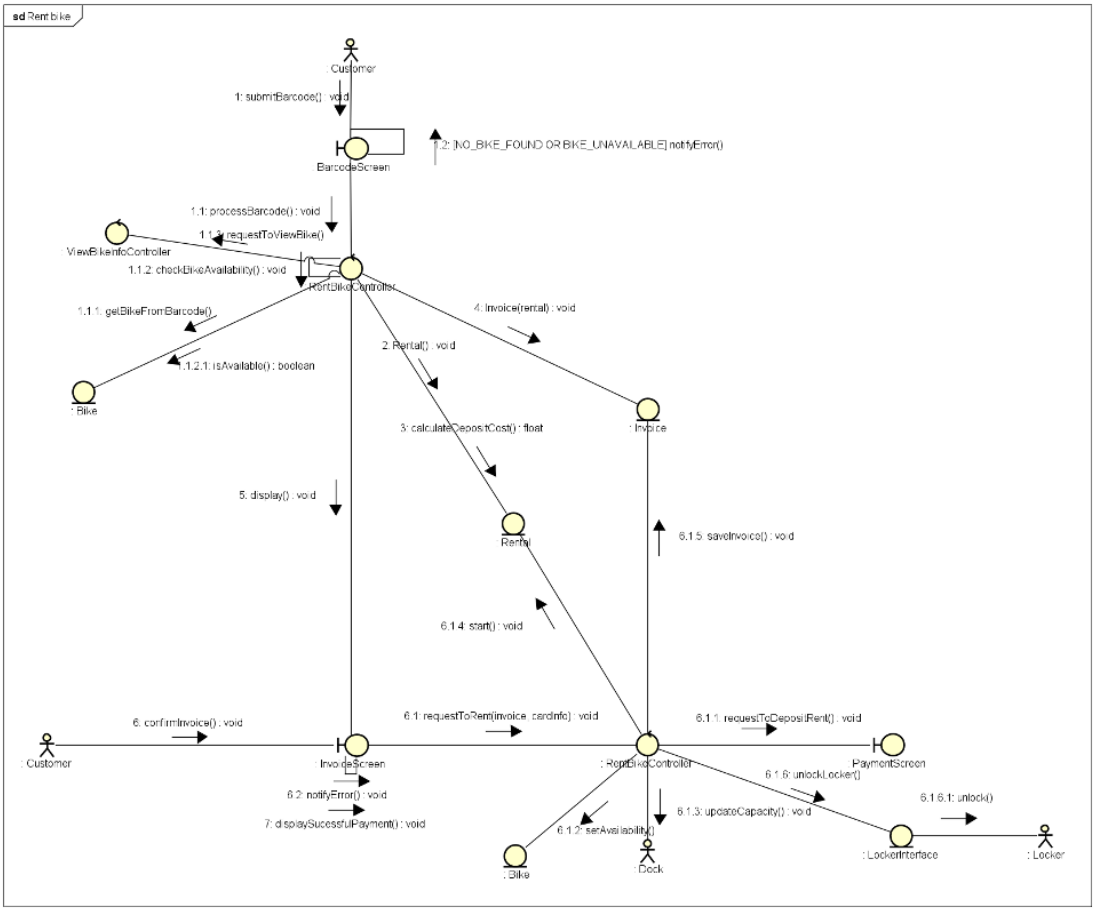
2.2.1. Usecase diagrams



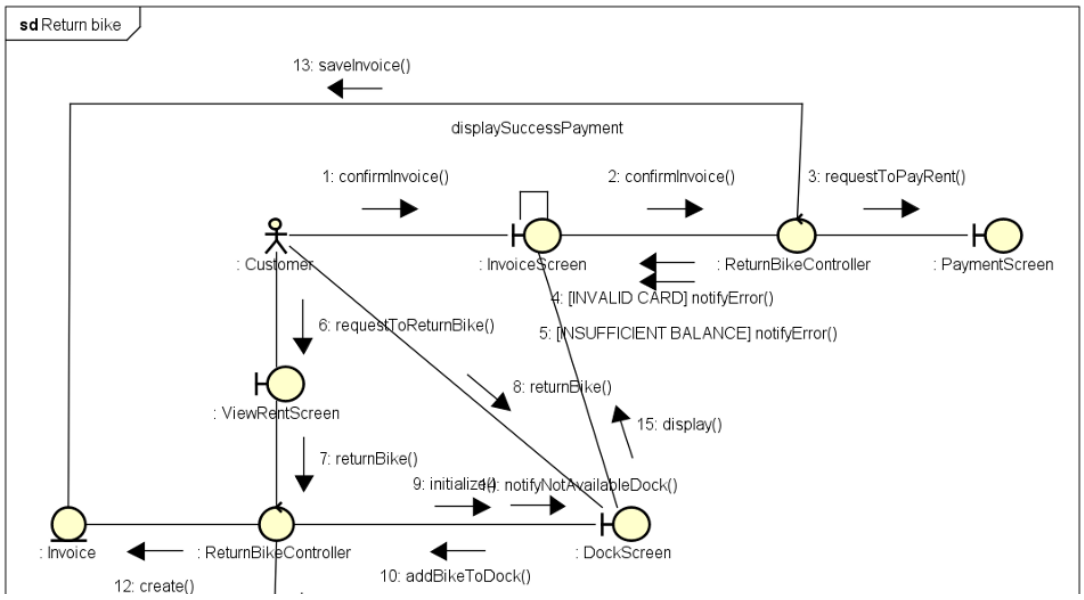
2.2.1.1. Usecase UC001-"View Bike Info"

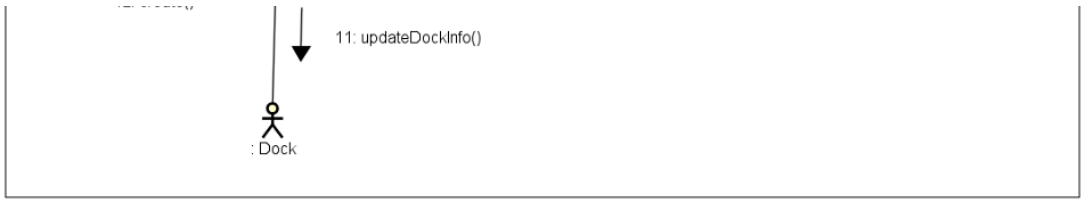


2.2.1.2. Usecase UC002-"Rent Bike"

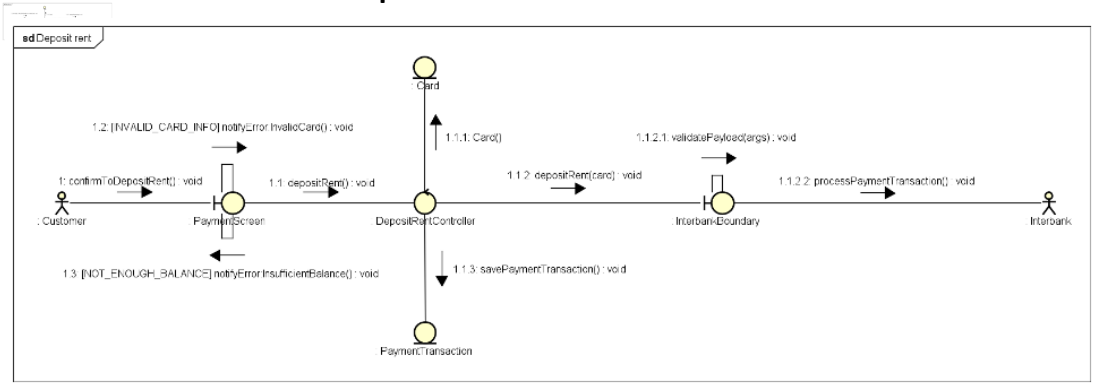


2.2.1.3. Usecase UC003-"Renturn Bike"

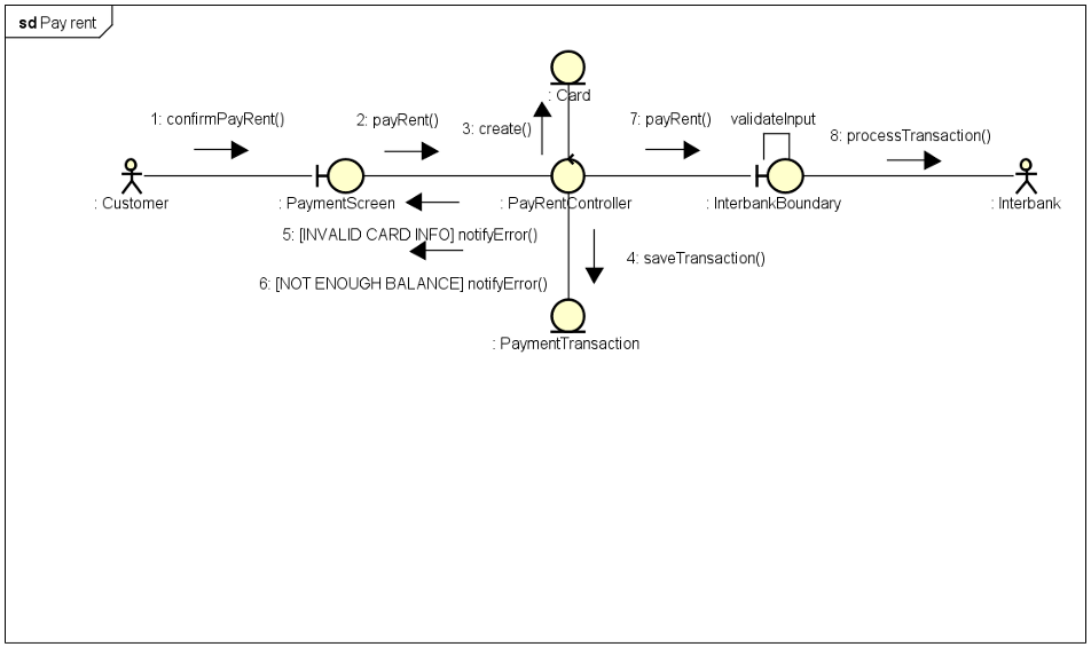




2.2.1.4. Usecase UC004-"Deposit Rent"



2.2.1.5. Usecase UC005-"Pay Rent"



2.2.1. Assumptions

The system pre-suppose that the users have internet connection on their devices, as well as a working payment method (through interbank or e-wallet) if they want to actually rent a bike.
The device that the software will be installed in is required to have Java SDK, have at least 50MB of memory. The dock station also requires an internet connection, as well as electricity on proceeding the rent.
The user should know the basic flow of how to rent and return a bike.
Possible changes in the future will mostly be related to the display of the software to the user interface.

2.2.2. Constraints

Hardware or software environment: the software needs to run on JAVA Virtual

Machine, ideally takes small enough memory and network bandwidth to be usable on mobile devices.
End-user environment: Users are not required to know too much about underlying system
Availability or volatility of resources: The software should run relatively stable, and calculate the rent time correctly.
Standards compliance: The system should not collect customer information without consent
Interoperability requirements: The system can read user input in ascii
Interface/protocol requirements: The network request should take at most 1 second
Licensing requirements: None
Data repository and distribution requirements: Include business flow and basic instruction on how to use and install the system.
Security requirements: The software should have validation on user input.
Memory or other capacity limitations: Should take less than 50MB to fit on a mobile device.
Performance requirements: The system is quick to navigate.
Network communications: Require regular network communication.

2.2.3. Risks

The system design is quite simple, and might not scale well if business flow changes, or there is a change in fee calculation. With current flow in mind, we should keep the current design and renovate it later if such a need arise.

3. Testing Approach/Strategy

< Describe the overall approach that will be used to test all functions, features, and requirements of the automated system, application, or situation for which the Test Plan applies. As applicable to this Test Plan, describe the measures to be taken to ensure all aspects of the system are successfully tested and can be implemented. Document key aspects of the testing approach, such as content, methodology, prioritization, and progression of development, validation, implementation, and operational testing activities to be performed during the corresponding lifecycle phases
For example, will some unit and application integration testing be done, and then some more development, and so on? Is a prototype being built that will be usability tested before the releasable software is developed? Also include plans for testing related documentation (e.g., installation instructions, User Manual, Operations & Maintenance (O&M) Manual, Training Artifacts, etc.) and for conducting applicable readiness reviews. Also, if applicable, describe how reuse will be applied to the testing effort to make testing more efficient and less costly.>

4. Unit Testing Summary

4.1. Traceability from Test Cases to Use Cases

<Mark the cell with an "x" where the Test Case tests.>

USE CASE ID			*UC001*	*UC002*	*UC003*	*UC004*	*UC005*	*UC006*
Test Case	*Test CaseTitle*	*Totals*	*1*	*2*	*1*	*2*	*2*	*1*
TC001	*RentBikeControllerTest*	*2*	*x*	*x*				
TC002	*PayRentControllerTest*	*2*				*x*	*x*	
TC003	*RentalTest*	*2*				*x*	*x*	
TC004	*SearchStationControllerTest*	*1*						*x*
TC005	*StationTest*	*2*		*x*	*x*			

Table: Test Case-to-Use Case Traceability Matrix

4.1.1. Test Suite for UC002-"Use Case Name"

Test Suite	Test Suite Title	Description	Test Cases
TS001	*<Test Suite Title>*	<Briefly describe the test suite>	*<Test Case ID, Test*

4.2. Traceability from Test Cases to Requirements

<Mark the cell with an "x" where the Test Case tests.>

REQUIREMENT ID			*RQ001*	*RQ002*	*RQ003*	*RQ004*		*RQ005*
Test Case	*Test CaseTitle*	*Totals*	*1*	*1*	*1*	*0*		*1*
TC001		*1*	*x*					
TC002		*3*		*x*	*x*			*x*
TC003		*1*	*x*					
TC004		*1*		*x*				
TC005		*1*						*x*
...		*1*			*x*			

Table: Test Case-to-Requirement Traceability Matrix

4.2.1. Test Suite for RQ001-"Requirement Title"

Test Suite	Test Suite Title	Description	Test Cases
TS004	*<Test Suite Title>*	<Briefly describe the test suite>	*<Test Case ID, Test Case ID>*

5. Test Case Details

5.1. Test Case Specification for "RentBikeControllerTest"

Test Case ID	TC001	**Test Case Descript**	Test the validation of the barcode of the bike		
Created By	Trinh Thu Hai	**Reviewed By**	Nguyen Huy Hoa	**Version**	1.0

QA Tester's Log

Tester's Name	Trinh Thu Hai	**Date Tested**	5/12/2021	**Test Case**	Pass

#	Prerequisites:
1	Access to the Internet

#	Test Data	Value	Value	Value	Value
1	Barcode	123456	abc123	123	null

Test Scenario

- Verify on entering valid barcode of the bike
- Verify on entering invalid barcodes of the bike

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Run test class RentBikeControllerTes	All tests should pass	As Expected	Pass

5.2. Test Case Specification for "PayRentControllerTest"

Test Case ID	TC002	**Test Case Descript**	Test the validation of the card info of the credit card		
Created By	Trinh Thu Hai	**Reviewed By**	Nguyen Huy	**Version**	1.0

QA Tester's Log

Tester's Name	Trinh Thu Hai	**Date Tested**	5/12/2021	**Test Case**	Pass

#	Prerequisites:
1	Access to the Internet

#	Test Data	Value	Value	Value	Value		Value
1	Card holder's name	TRINH THU HAI	TRINH HAI	Group 3	!@HAI	null	
2	Card number	ict_group3_2021	ict group03 202	ict_groupXYZ_2021	ict_group05_abcd	null	

3	Card secuity code	123	1234	12	1ab	null	
4	Expiration date	1125	1/25/2021	112025	1325	1301	null

Test Scenario
- Verify on valid information
- Verify on invalid information

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Run test class PayRentControllerTest	All tests should pass	As expected	Pass

5.3. Test Case Specification for "RentalTest.validateRent"

Test Case ID	TC003	**Test Case Descrip**	Test the rent calculation		
Created By	Nguyen Huy Hoang	**Reviewed By**	Trinh Thu Hai	**Version**	1.0

QA Tester's Log

Tester's Name	Nguyen Huy Hoang	**Date Tested**	5/12/2021	**Test Case**	Pass

#	Prerequisites:

#	Test Data	Value	Value	Value	Value	Value	Value
1	Start date	1986-04-08 1:30	##########	1986-04-08 1:30	1986-04-08 1:30	#########	#########
2	End date	##########	##########	1986-04-08 1:30	##########	#########	#########
3	Rent amount	200000	190000	80000	200000	202000	202000

Test Scenario
- Case less than 12 hours
- Case more than 12 hours

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Run test class Rental, method validateRent	All tests should pass	As expected	Pass

5.4. Test Case Specification for "RentalTest.validateRefund"

Test Case ID	TC004	**Test Case Descript**	Test the refund calculation		
Created By	Nguyen Huy Hoang	**Reviewed By**	Trinh Thu Hai	**Version**	1.0

QA Tester's Log

Tester's Name	Nguyen Huy Hoang	**Date Tested**	5/12/2021	**Test Case**	Pass

#	Prerequisites:

#	Test Data	Value	Value	Value
1	Start date	1986-04-08 1:30	##########	1986-04-08 1:30
2	End date	##########	##########	1986-04-08 1:30
3	Deposit amount	200000	200001	199999
4	Rent amount	0	1	0

Test Scenario

- Case less than 12 hours
- Case more than 12 hours

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Run test class Rental, method	All tests should pass	As expected	Pass

5.5. Test Case Specification for "SearchStationControllerTest"

Test Case ID	TC004	**Test Case Descript**	Test the search station by name function		
Created By	Bui Thanh Tung	**Reviewed By**	Trinh Thu Hai	**Version**	1.0

QA Tester's Log

Tester's Name	Bui Thanh Tung	**Date Tested**	5/12/2021	**Test Case**	Pass

#	Prerequisites:
1	Access to the Internet

#	Test Data	Value	Value	Value
1	Station name	HUST	NEU	FTU

Test Scenario	- Search results when entering valid names - No results when entering unavailable names

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not
1	Run test class SearchStationControll	All tests should pass	As Expected	Pass

5.6. Test Case Specification for "StationTest"

Test Case ID	TC005	**Test Case Descrip**	Test the add and remove bike functions of station entity		
Created By	Bui Thanh Tung	**Reviewed By**	Trinh Thu Hai	**Version**	1.0

QA Tester's Log

Tester's Name	Bui Thanh Tung	**Date Tested**	5/12/2021	**Test Case**	Pass

#	Prerequisites:
1	Access to the Internet

#	Test Data	Value	Value	Value
1	Bike Code	1234	1235	1236

Test Scenario	- Add legit bikes to station - Remove legit bikes from station - Add more bikes to station than max capacity - Remove non-existent bike from station

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Run test class StationTest	All tests should pass	As Expected	Pass