A CAMPUS SCOOTER SHARING SYSTEM

DEVELOPING THE SOFTWARE USING AGILE METHODS

Group 101 | EBU5304 – Software Engineering | 2019 Spring

Group Members

Name	QM ID	Mail
Xi XIA	161187914	x.xia@se16.qmul.ac.uk
Taizhou QING	161188645	t.qing@se16.qmul.ac.uk
Haoran CUI	161187729	h.cui@se16.qmul.ac.uk
Jiayi PANG	161188287	j.pang@se16.qmul.ac.uk
Yuanrong SHAO	161192664	y.shao@se16.qmul.ac.uk
Xi CUI	161187833	x.cui@se16.qmul.ac.uk

Table of Contents

Introduction

From March to June, our team developed a software of a campus scooter sharing system using Agile methods. We assigned every 3 weeks as an iteration, and Agile methods was used in all activities, from requirements, through to analysis/design, implementation and testing.

The Scooter Sharing System is divided into 2 parts:

1. Management System
   - Registration
   - Usage Report
   - Pay for Fine
2. Three Scooter Stations (A, B, C), each has
   - Eight slots
   - Two main functions: pick up and return

Agile processes are discussed in the following contents.

Agile Project Management

TEAMWORK IN SCRUM

The Scrum approach is a general agile method that we used to manage our iterative development. And we set the sprint a fixed length – 3 weeks.

Outline planning and architecture design

1. General objectives:

   We were going to develop a Scooter Sharing System containing both the Management System and Scooter Stations. The 3 Station should use the same framework. (Fig. 1)

```
                                  1...1   ┌──────────────┐
                                          │ Management   │
                                          │ System       │
         ┌──────────────┐                 └──────────────┘
         │ Scooter      │
         │ Sharing System│
         └──────────────┘
                                          ┌──────────────┐
                                  1...3   │ Stations     │
                                          └──────────────┘
```
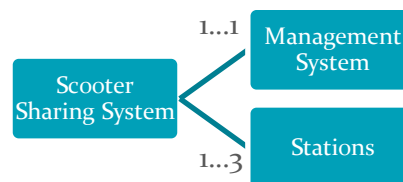
Figure 1. General objectives

2. Software architecture:
   - Two Entities: User and Station.
   - We use text files to record.
   - Each entity should have its control class to implement the connection(Insert/ Delete/ Modification) with text files: UserControl and StationControl
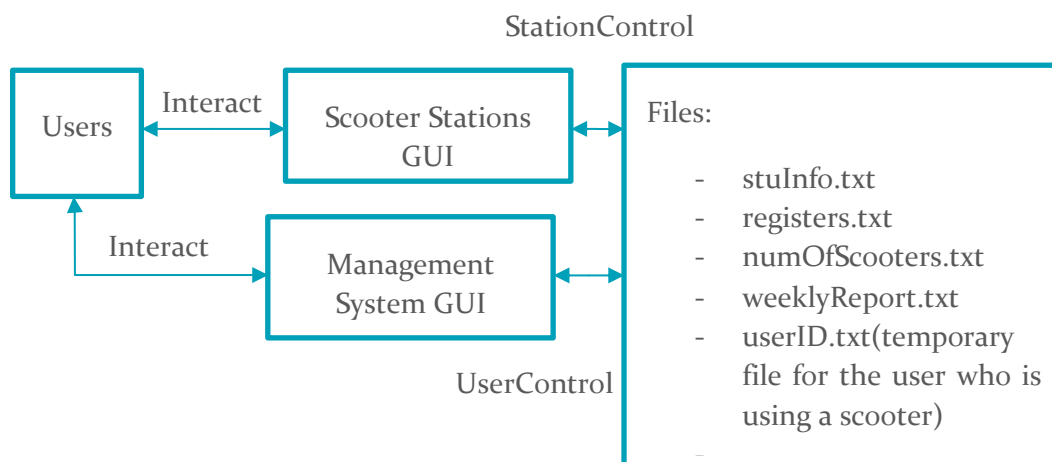   - We need GUIs to interact with users

StationControl

```
┌───────┐  Interact  ┌──────────────┐      ┌──────────────────────────┐
│ Users │◄─────────► │ Scooter      │◄───► │ Files:                   │
│       │            │ Stations GUI │      │                          │
└───────┘            └──────────────┘      │   -  stuInfo.txt         │
    ▲                                      │   -  registers.txt       │
    │      Interact                        │   -  numOfScooters.txt   │
    │                ┌──────────────┐      │   -  weeklyReport.txt    │
    └─────────────►  │ Management   │◄───► │   -  userID.txt(temporary│
                     │ System GUI   │      │      file for the user   │
                     └──────────────┘      │      who is using a      │
                           UserControl     │      scooter)            │
                                           └──────────────────────────┘
```

Figure 2. Software architecture

Spring cycles

We started from the story backlog in every spring cycle Then, we access the priorities and difficulties of every story. During the group meetings, we discussed and selected stories to be implemented in the corresponding spring. And each cycle develops an increment of the system.

As presented in the product backlog (see Appendix 3), we had 3 iterations:

1. 2019.3.20 – 2019.4.10
   - Finished the development of user registration, picking up and returning functions
   - Modified the priority of user verification as 1, finish it before user registration
   - Recorded the dates of Spring 1
2. 2019.4.11 – 2019.5.2
   - Finished the development of report and payment functions
   - Wrote test cases and generate unit tests
   - JUnit test done
   - Recorded dates of Spring2
3. 2019.5.3 – 2019. 5.24
   - Added an epic to check formats of inputs.
   - Reorganize all classes into 3 types: boundary, entity, control
   - Recorded dates of Spring3

Based on the agreement we gained, we organized ourselves well to develop the software. We review our work before every demo. And thanks to the advice from TAs, we made a lot of improvements. Then, we were able to continue software development cycle by cycle.

Project closure

All the Software developing tasks were finished before May 24, 2019. Then we wrote readme files and user manual for users. What's more, we revised the Agile software developing techniques we had used in this final report by May 31, 2019.

Requirements

REQUIREMENTS FINDING TECHNIQUES

Background reading

Since have between using the sharing-bikes in our daily life and they are similar to the scooters, we all know the basic rules of picking up and returning. However, we were not familiar with the scooter station. To find more facts about the scooter station, we searched on the Internet to do some background reading.

By looking reading the instructions on (Artlebedev.com, 2019), we found the fact that we need to implement the action of "taking out" and "locking" after the user choosing to pick up or return a scooter. So, we change the representation of scooter slots on GUI into buttons, so users can press the slot button to take out or lock the scooters.

Other fact-finding techniques

We did oral interviews to the students around us. (they are targeted customers of our system) And we ask questions like:

- Where would you like to register, check the report and pay?
- Is it necessary to aggregate the both the management system and the scooter station into one home page?

Then we observed that they prefer an online aggregated system. So, we set the application scenario of our system to be online. And we built a home page for users to choose whether they are going to manage or to operate on the scooter station.

CHANGES OF THE PRODUCT BACKLOG

Spring 1
- Generated the basic storiest and breaked them down to small epics
- Set the priorities and iteration length
- Record the start date of spring 1 -- 2019.3.20

Spring 2
- Changed the priority of E01 "Verify the identity of users" to 1
- Modified the requirements of weeklyReport.txt contents: don't record the pick up and return station
- Record the end date of spring 1 -- 2019.4.10, and start date of spring 2 -- 2019.4.11

Spring 3
- Added a requirement that the system should check the input formats: qmID is a 9-digit number and the mail address should contain the symbol '@'
- Record the end date of spring 2 -- 2019. 5. 2 and start date of spring 3 -- 2019. 5.03
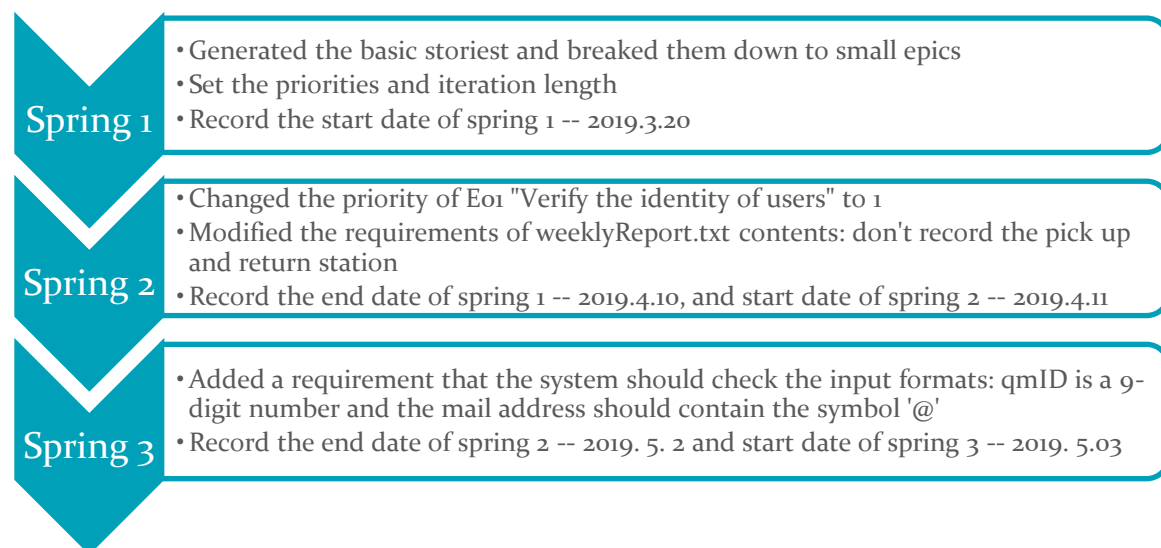
Figure 3. changes of the product backlog

ITERATION AND ESTIMATION OF STORIES

For the estimation of stories, we adopted the x^2 sequence numbers (1, 4, 9, 16, 25 …) rather than the linear scale (1,2,3,4,5,6,7…).This is because the difference between 1 and 2 can seem insignificant. However, the difference between 1 and 4 is obvious. (Medium, 2019)

1. Create a Matrix for Estimation
   We generally have 6 large epics

E01	Verify the identity of users
E02	Register New Account
E03	Pick up the scooter
E04	Return the scooter
E05	Fine
E06	Usage report

Table 1. Epics

2. Story Estimation
   Our group assigned two estimators: Xi XIA and Yuanrong SHAO. After discussing carefully, we came out an original estimation of the stories

Story ID	Story	Priority	Story points (x^2)
E01	Verify the identity of users	2	16
E02	Register New Account	2	16
E03	Pick up the scooter	1	25
E04	Return the scooter	1	25
E05	Fine	3	9
E06	Usage report	4	4

Table 2. Story Estimation

3. Iterations
   We have three iterations, but we mainly finish the software development in the first two weeks. This is because we want to set aside time for the new features found in the later updates of product backlog, as well as in debugging and testing.
   We added the stories (1) check the format of inputs, and (2) Instructions of usage to our product backlog. Please see the Appendix 3 Product Backlog (Final Version) for details.

Analysis and Design

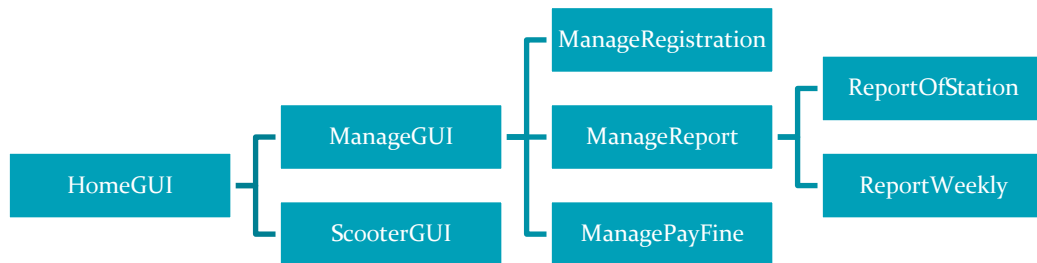## A DESIGN CLASS DIAGRAM

Software classes

1. Boundary Classes

```
                                    ManageRegistration
                                                            ReportOfStation
                    ManageGUI           ManageReport
    HomeGUI                                                 ReportWeekly
                    ScooterGUI          ManagePayFine
```

Figure 4. Boundary Class Relationship

2. Entity Classes

Entity	Main Attributes	
User	name qmID mailAddress fine(boolean)	Name of the user Queen Mary ID of the user Mail address of the user "true" for fined, "false" for no fine
Station	stationName scooterNum	"A" or "B" or "C" Number of scooters in the station

Table 3. Entity Classes

3. Control Classes

Control	Files Involved
UserControl	stdInfo.txt registers.txt weeklyReport.txt
StationControl	numOfScooters.txt stdInfo.txt registers.txt userID.txt(temporary file for the user who is using a scooter) weeklyReport.txt

Table 4. Control Classes

Reusability of software components

All the stations are using the same framework, so we can add stations whenever it is required. And the GUI settings are almost the same (please see Appendix 2: Main Screenshots of the System), so we can add GUI pages quickly.
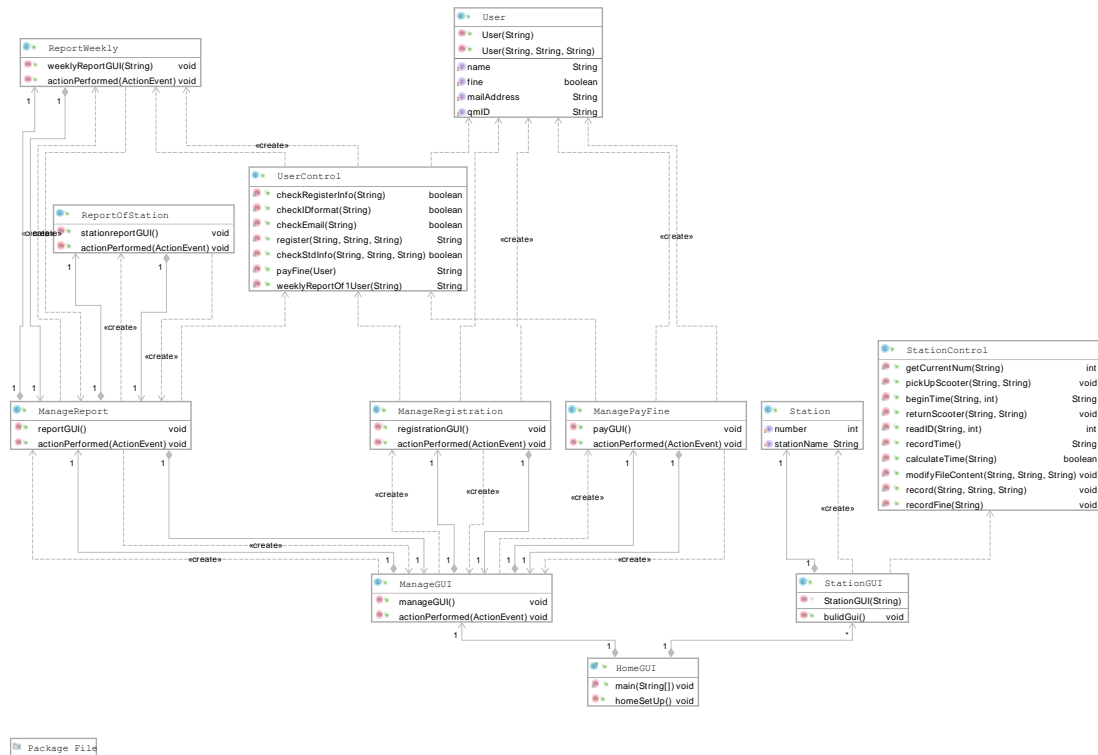
Class UML and Code

Figure 5. Class UML

We deigned the UML to present all the methods to implement our stories. (Fig. 5.)

Our code satisfy the base object-oriented principles of JAVA.

GUI Design

We used paper protype to design GUIs. This has be shown to the TA in the first demonstration.

Implementation and Testing

IMPLEMENTATION
Strategy

1. Determine the functions to be tested

Management	Station
Registration	Pick up a Scooter
Usage Weekly Report	Return Scooters
Pay for Fine	Station Report

Table 5. Tested Functions

2. Determine why to test

The test is for the entire system to work in all aspects properly and give correct feedback in different situations.

Iteration/built plan
Test Case Grouping

We used distributed test cases into different groups to list all the situations that need to be tested. According to the Test Case, we wrote the test code.

Test Scenario	Test Case File	Test Case
check register information	UserControlTest .java	qmID is not 9 digits
check register information	UserControlTest .java	Mail address doesn't include "@"
check register information	UserControlTest .java	Mail address doesn't include "@" and qmID is not 9 digits
check register information	UserControlTest .java	qmID has been registered.
check student information	UserControlTest .java	User is not registered and the information of the user exists in the database.
check student information	UserControlTest .java	The information of the user doesn't exist in the database.
manage pay fine	UserControlTest .java	qmID doesn't exist
pay fine	UserControlTest .java	qmID exists and user doesn't have fine.
pay fine	UserControlTest .java	User has fine.
manage report	UserControlTest .java	User has used scooters.
manage report	UserControlTest .java	qmID exists and user doesn't use scooters.
manage report	UserControlTest .java	qmID doesn't exist.
station	StationControlTest.java	qmID doesn't exist
station	StationControlTest.java	User hasn't paid a fine

station	StationControlTest.java	User hasn't returned the scooter
station	StationControlTest.java	User hasn't registered
station	StationControlTest.java	User picks up a scooter successfully.
station	StationControlTest.java	User hasn't picked up a scooter before returning.
station	StationControlTest.java	User returns a scooter successfully

Table 6. Test Cases

Please see Appendix 4 – Test Cases for details.

TESTING

Strategy and techniques that we have used

Regression Testing

For each build stage, we create according test cases to test our functions. When a new function is added, the test cases will be updated.

Black-box Testing

For more detailed testing, we use black-box testing to check whether there is a missing function or any incorrect function.

1. Equivalent Class Partitioning

For example, we just checked the user whether she or he is in the database. However, we didn't check the format of the qmID. Then, we added this functionality into our system and checked the correctness.

We spited the range of qmID input into 3 parts.
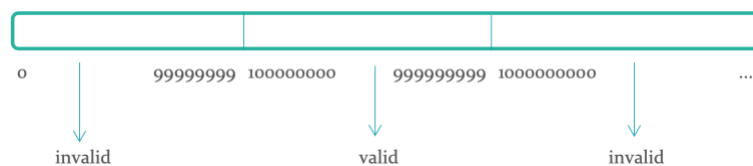
Figure 6. Equivalent Class Partitioning

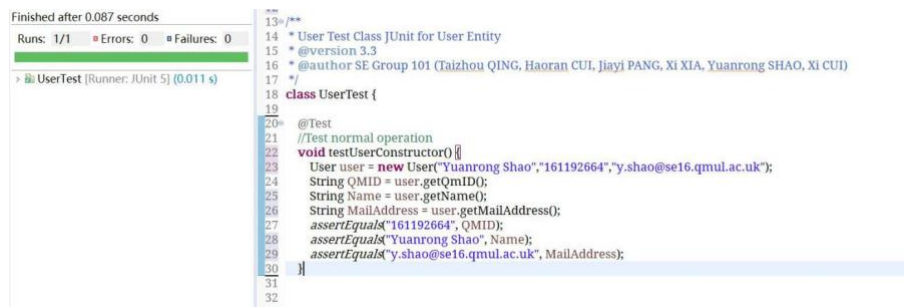2. Boundary Value Analysis

We tested 5 values which includes:

Minimum	0
Just above the minimum	99999999
A nominal value	100000000
Just below the maximum	999999999
Maximum	1000000000

Then we input these 5 values separately and got the corresponding feedbacks from the system.

JUnit4

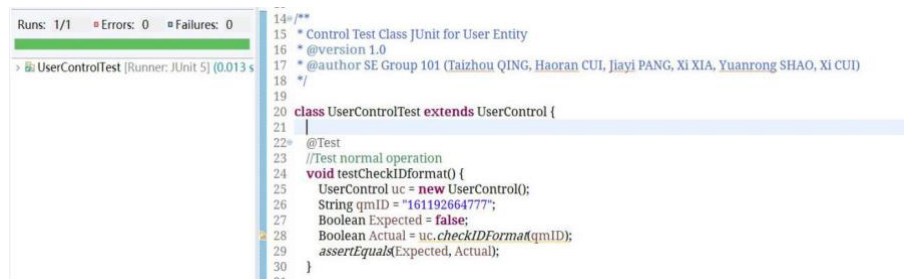We used Junit as a simple unit-testing framework for supporting TDD.

1. User.java

Figure 7-1. JUnit Test

It is obviously that the method getQmID( ),getName( ),getMailAddress( ) in User.java passed the test.

2. UserControl.java

Figure 7-2. JUnit Test

It is obviously that checkIDFormat( ) in UserControl.java passed the test.

TDD

TDD is a simple mechanism which starts with designing and developing tests for every small functionality of an applications. In TDD approach, the test is developed which specifies and validates what the code will do. And We did not apply TDD on all parts but on some fundamental functions. Like, UserControl class and StationControl class.

Appendix 1: References

1. Artlebedev.com. (2019). [online] Available at:
   https://www.artlebedev.com/samocat-sharing/ [Accessed 20 May 2019].
2. Medium. (2019). How we rediscovered the joys of Story Estimation. [online]
   Available at:
   https://medium.com/zendesk-engineering/the-joys-of-story-estimation-cda0cd807903 [Accessed 26 May 2019].

Appendix 2: Main Screenshots of the System

1. Home page


```
SCOOTER SHARING SYSTEM
Please choose the function you need!

        Management

        Station  A

        Station  B

        Station  C
```

2. Scooter Station (e.g. Station A)


```
                          A
                       Welcome

Please have your id card scanned for picking up or returning a scooter.

  scooter    scooter    scooter    scooter

  scooter    Empty      Empty      Empty

                            Pick up        Return
```

3. Management System


```
Management System
Please choose the service you need!

        Registration

        Usage Report

        Pay for fine
```



```
New User Registration
        Please enter your information below.
QM ID


FULL NAME


MAIL ADDRESS


        Back              Register
```



```
USAGE REPORT

Weekly Report
Please enter the id below.


Weekly Report!


Stock Report
Press if you want to check the stock status.

Station Report


        Back
```



```
Pay for fine

Please enter your id below


        Back                  Pay
```

Appendix 3 Product Backlog (Final Version)

Story ID	Story Name	Description	Priority	Iteration (Sprint) number	Acceptance Criteria	Notes	Date started (actual date)	Date finished (actual date)
E01	Verify the identity of users	As an administration office, I want to get details of the unregisted user and compare with the directory. So that I can verifiy the identity of the user.	2	1	If the user is not QMUL student or staff member,he/she will not pass the verification.		2019/3/20	2019/4/10
E02	Register New Account	As an administration office, I want to get the users' ID, email address and full name. So that I can create a new account.	2	1	The user should have a valid identification.		2019/3/20	2019/4/10
E02a	Provide New Account	As a manager, I want to enter information of the user and make verification. So that I can provide a new account for the user.	1	1	The user should have a valid identification. Registed user should not register again.	The unrehisted user need to provide their QM number,full name and QM email address.	2019/3/20	2019/4/10
E02b	Get New Account	As an unregistered user, I want to provide my personal information. So that I can get new account and use the scooter.	1	1	The ID should be 9 digits, and the mail address should contain a symbol '@'		2019/3/20	2019/4/10
E02c	Check the format of inputs	As a manager, I want to make sure the customers enter a 9-digit QMID and a mail address with '@' So that customers can be noticed of these simple mistakes.	5	3	The ID should be 9 digits, and the mail address should contain a symbol '@'		2019/5/3	2019/5/24
E03	Pick up the scooter	As a user, I want to go to the docking station which has available scooters. So that I can pick up the scooter from the docking station.	1	1	If scooter hasn't been picked up after 1 min, it will be locked atuomatically.Meanwhile,the screen will display the information to the user.	The scooter will be allocated by the system.	2019/3/20	2019/4/10
E04	Return the scooter	As a user, I want to go to the docking station which has empty slots. So that I can I can return the scooter to the docking station.	1	1	If scoooter hasn't been returned back after 1 min, it will unable to be put into the slot. Meanwhile, the screen need to show the message to inform to the user.	The slot will be allocated by the system.	2019/3/20	2019/4/10
E05	Fine	As an administration office, I want to know who obey the rules. So that I can record the fine.	3	2	If user rides scooter more than 30 mins each time or more than 2 hours in a day,he/she will be recorded for a "true" fine. Otherwise,they can ride scooter free all the time.		2019/4/11	2019/5/2
E05a	Pay a fine	As a user, I want to check the status of my fine.(whether I have obey the rules). So that I can pay a fine.	3	2	If user with the provided ID doesn't have any fine, he will get a message"You don't have a fine". If the user has a record of fine, pressing the button "Pay" will clear his/her fine.		2019/4/11	2019/5/2
E05b	Impose a fine	As a manager, I want to know who should be fined. So that I can impose a fine.	3	2	Users won't be able to operate in the scooter station.		2019/4/11	2019/5/2
E05c	Check fine	As a manager, I want to check whether the user has a fine before the user pay for fine	3	2	If the user has no fine, he/she should not be allowed to pay		2019/4/11	2019/5/2
E06	Usage report	As a scooter sharing system, I want to get details of every usage. So that I can provide an overall usage report.	4	2	User usage should contain User ID,Scooter ID,Pick up time.		2019/4/11	2019/5/2
E06a	Send weekly usage report	As a manager, I want to acquire usage detail of a single user. So that I can send the weekly usage report via email to the user.	4	2	Report should contain User ID,Scooter ID,Pick up time.		2019/4/11	2019/5/2
E06b	Monitor the docking station	As a manager, I want to acquire usage detail of every scooter. So that I can monitor the situation of the docking station and usage of scooters.	4	2	The reprot should provide the status of each docking station. e.g. how many scooters are currently available at each station.		2019/4/11	2019/5/2
E07	Instructions for usage	As a user, I want to be instructed, so that I can use the system correctly	5	3	The isnructions should be lear and easy to understand		2019/5/3	2019/5/24

Appendix 4 Test Cases

TEST CASE						
Test Scenario	**Test Case File**	**Test Case**	**Test Data**	**Expected Result**	**Actual Results**	**Pass/Fail**
UserTest						
check register information	UserControlTest .java	qmID is not 9 digits	QM ID: 1611879144	The QMID doesn't satisfy the requirement.	The QMID doesn't satisfy the requirement.	Pass
check register information	UserControlTest .java	Mail address doesn't include "@"	Mail Address: y.shaose16.qmul.ac.uk	The Email doesn't satisfy the requirement.	The Email doesn't satisfy the requirement.	Pass
check register information	UserControlTest .java	Mail address doesn't include "@" and qmID is not 9 digits	Mail Address: y.shaose16.qmul.ac.uk QM ID: 1611879144	The QM ID and Email don't satisfy the requirement.	The QM ID and Email don't satisfy the requirement.	Pass
check register information	UserControlTest .java	qmID has been registered.	QM ID:161187914 Full Name: Xi Xia Mail Address: x.xia@se16.qmul.ac.u k	You have already registered.	You have already registered.	Pass
check student information	UserControlTest .java	User is not registered and the information of the user exists in the database.	QM ID:161187833 Full Name:Xi Cui Mail Address: x.cui@se16.qmul.ac.u k	Succeed in registration!	Succeed in registration!	Pass
check student information	UserControlTest .java	The information of the user doesn't exist in the database.	QM ID:161187844 Full Name: Hanwen Xu Mail Address: h.xu@se16.qmul.ac.uk	Your information is invalid	Your information is invalid	Pass
manage pay fine	UserControlTest .java	qmID doesn't exist	QM ID:161187844	You haven't registered or you provided a invaild ID	You haven't registered or you provided a invaild ID	Pass
pay fine	UserControlTest .java	qmID exists and user doesn't have fine.	QM ID:161192664	You don't have a fine.	You don't have a fine.	Pass
pay fine	UserControlTest .java	User has fine.	QM ID:161188645	Your fine is cleared.	Your fine is cleared.	Pass
manage report	UserControlTest .java	User has used scooters.	QM ID:161192664	Your weekly ManageReport	Your weekly ManageReport	Pass
manage report	UserControlTest .java	qmID exists and user doesn't use scooters.	QM ID:161187833	You didn't use our service this week.	You didn't use our service this week.	Pass
manage report	UserControlTest .java	qmID doesn't exist.	QM ID:161187844	You haven't registered or the input is invalid	You haven't registered or the input is invalid	Pass
StationTest						
station	StationControlTest.ja va	qmID doesn't exist	QM ID:161187844	Please input right ID	Please input right ID	Pass
station	StationControlTest.ja va	User hasn't paid a fine	QM ID:161188645	You haven't paid the fine	You haven't paid the fine	Pass
station	StationControlTest.ja va	User hasn't returned the scooter	QM ID:161187914	You haven't returned a scooter	You haven't returned a scooter	Pass
station	StationControlTest.ja va	User hasn't registered	QM ID:161187833	You haven't registered	You haven't registered	Pass
station	StationControlTest.ja va	User picks up a scooter successfully.	QM ID:161192664	You have picked up successfully.	You have picked up successfully.	Pass
station	StationControlTest.ja va	User hasn't picked up a scooter before returning.	QM ID:161187729	Please pick up first	Please pick up first	Pass
station	StationControlTest.ja va	User returns a scooter successfully	QM ID:161187914	You have returned successfully	You have returned successfully	Pass