

**Finding Friends for the Same Purpose**

**at Close Distance**

**Software Test Specification**

2021.05.30.

**Introduction to Software Engineering**

**TEAM 7 (유생찾기)**

|  |  |
| --- | --- |
| Team Leader | Eunju Seok |
| Team Member | Eunji Gil |
| Team Member | Hyeyeong Kim |
| Team Member | Georyang Park |
| Team Member | Jiwon Seo |
| Team Member | Hyejoon Jang |

[**1. Introduction** 7](#_Toc73280682)

[1.1. Purpose 7](#_Toc73280683)

[1.2. Scope 8](#_Toc73280684)

[1.2.1. Exploratory test 8](#_Toc73280685)

[1.2.2. Unit test 8](#_Toc73280686)

[1.2.3. Interface test 9](#_Toc73280687)

[1.2.4. Test Acceptance Criteria 9](#_Toc73280688)

[1.3. Definitions, Acronyms, and Abbreviations 10](#_Toc73280689)

[1.4. References 11](#_Toc73280690)

[1.5. Overview 11](#_Toc73280691)

[1.6. Execution Strategy 12](#_Toc73280692)

[1.6.1. Test Cycles 12](#_Toc73280693)

[1.6.2. Validation and Defect Management 12](#_Toc73280694)

[1.6.3. Defect Tracking & Reporting 13](#_Toc73280695)

[1.7. Entry and Exit Criteria 14](#_Toc73280696)

[1.7.1. Entry Criteria 14](#_Toc73280697)

[1.7.2. Exit Criteria 14](#_Toc73280698)

[**2. Approach** 14](#_Toc73280699)

[2.1 Test method 15](#_Toc73280700)

[2.1.1 Software unit test methods 15](#_Toc73280701)

[2.1.2 Software interface test methods 18](#_Toc73280702)

[2.2 Test Assumptions 18](#_Toc73280703)

[2.2.1 Key Assumptions 18](#_Toc73280704)

[2.2.2 General Assumptions 18](#_Toc73280705)

[2.3 Testing tools 19](#_Toc73280706)

[**3. Unit-Test** 20](#_Toc73280707)

[3.1 Log-in page 20](#_Toc73280708)

[3.1.1 Register 20](#_Toc73280709)

[3.1.2 Login 22](#_Toc73280710)

[3.2 Profile page 26](#_Toc73280711)

[3.2.1 Set profile image 26](#_Toc73280712)

[3.2.2 Set nickname 27](#_Toc73280713)

[3.2.3 Set major 30](#_Toc73280714)

[3.2.4 Set sex 32](#_Toc73280715)

[3.2.5 Set introduction 33](#_Toc73280716)

[3.3 Main page 34](#_Toc73280717)

[3.3.1 Map 34](#_Toc73280718)

[3.3.2 Search 38](#_Toc73280719)

[3.4 Filtering page 40](#_Toc73280720)

[3.5 Review 42](#_Toc73280721)

[**4. Software Interface Test** 44](#_Toc73280722)

[4.1. Objectives 44](#_Toc73280723)

[4.2. Firebase 44](#_Toc73280724)

[4.2.1 Connection Test 44](#_Toc73280725)

[4.2.2 Stress Test 45](#_Toc73280726)

[4.2.3 Data Test 46](#_Toc73280727)

[4.3. Naver Map API 47](#_Toc73280728)

[4.3.1 Connection Test 47](#_Toc73280729)

[4.3.2 Data Test 48](#_Toc73280730)

[4.4. Apache 49](#_Toc73280731)

[4.4.1 Connection Test 49](#_Toc73280732)

[4.4.2 Stress Test 50](#_Toc73280733)

[**5. Supporting Information** 51](#_Toc73280734)

[5.1. Test Environment 51](#_Toc73280735)

[5.1.1. Tools 51](#_Toc73280736)

[5.1.2. Staffing 52](#_Toc73280737)

[5.2. Test Schedule 53](#_Toc73280738)

[5.2.1. Scheduling 53](#_Toc73280739)

[5.2.2. Timeline 55](#_Toc73280740)

[5.3. Document 55](#_Toc73280741)

[5.3.1. Document History 55](#_Toc73280742)

**LIST OF FIGURES**

[[Figure 1] Sign-off and readiness 9](#_Toc73281692)

[[Figure 2] Defect tracking process 13](#_Toc73281693)

[[Figure 3] Register 15](#_Toc73281694)

[[Figure 4] Login 15](#_Toc73281695)

[[Figure 5] Profile 16](#_Toc73281696)

[[Figure 6] Maps & Search 16](#_Toc73281697)

[[Figure 7] Filtering 17](#_Toc73281698)

[[Figure 8] Review 17](#_Toc73281699)

[[Figure 9] Test Scheduling 53](#_Toc73281700)

[[Figure 10] Test Scheduling2 54](#_Toc73281701)

[[Figure 11] Test Scheduling3 54](#_Toc73281702)

**LIST OF TABLES**

[[Table 1] Exploratory test 8](#_Toc73280504)

[[Table 2] Unit test 8](#_Toc73280505)

[[Table 3] Interface test 9](#_Toc73280506)

[[Table 4] Acronyms and abbreviations 10](#_Toc73280507)

[[Table 5] Terms and definitions 10](#_Toc73280508)

[[Table 6] Defect level 12](#_Toc73280509)

[[Table 7] Testing tools 19](#_Toc73280510)

[[Table 8] Firebase Test Method – Connection Test 45](#_Toc73280511)

[[Table 9] Firebase Test Method – Stress Test 45](#_Toc73280512)

[[Table 10] Firebase Test Method – Empty Input 46](#_Toc73280513)

[[Table 11] Firebase Test Method – Exceed Input 46](#_Toc73280514)

[[Table 12] Firebase Test Method – Request does not exist. 47](#_Toc73280515)

[[Table 13] Firebase Test Method – Normal Request 47](#_Toc73280516)

[[Table 14] API Test Method – Connection Test 48](#_Toc73280517)

[[Table 15] API Test Method – Empty Request 48](#_Toc73280518)

[[Table 16] API Test Method – Normal Request 49](#_Toc73280519)

[[Table 17] API Test Method – Request does not exist. 49](#_Toc73280520)

[[Table 18] Apache Test Method – Connection Test 50](#_Toc73280521)

[[Table 19] Apache Test Method – Stress Test 51](#_Toc73280522)

[[Table 20] Testing Tools 51](#_Toc73280523)

[[Table 21] Test Staffing 52](#_Toc73280524)

[[Table 22] Test Timeline 55](#_Toc73280525)

[[Table 23] Document History 55](#_Toc73280526)

# **1. Introduction**

## **1.1. Purpose**

This document is Software Test Specification (STS) for providing ‘유생 찾기’ service. With this service, users can find friends with the same purpose at close distance. The purpose is classified to eat, study, and rest in this service. The purpose of the service is to create a chat room according to the purpose based on a specific place, and to meet with the gathered people at that place to make friends. In this document, the test is designed and implemented so that this service can be tested in unit, component, and system.

In this document, Team7 is the main reader, and Team7 designs and implements tests according to the given environment, hardware specifications, and software specifications. In addition, professors, assistants, and team members in the introduction to software engineering class can be the main readers.

In this document, after introducing the test method and tool, the unit test method and the interface test method will be separately introduced. Each test method is written through pseudo code, and through these tests, you can check whether the service works correctly. If the expected output does not come out for the designed input, software refactoring will be necessary. We design a test for each detailed function such as register, log-in, and chatting. After that, a test is also designed for the external software interface used by the system. External software interfaces include databases, APIs, and AWS servers. At the end of the document, supporting information is explained and the test schedule is introduced before finishing.

## **1.2. Scope**

### **1.2.1. Exploratory test**

[Table 1] Exploratory test

|  |  |
| --- | --- |
| Purpose | The purpose of this test is to make sure critical defects are removed before the next levels of testing can start. |
| Scope | First level navigation, dealer and admin modules. |
| Testers | Testing team members. |
| Methods | This exploratory testing is carried out in the application without any test scripts and documentation. |
| Timing | At the beginning of the test stage. |

### **1.2.2. Unit test**

[Table 2] Unit test

|  |  |
| --- | --- |
| Purpose | Unit testing will be performed to check the functions of the individual units or components in the system. The unit testing is carried out by feeding the input and validates the output from the unit or components in the system. |
| Scope | Chapter three will detail the scope of the unit test. |
| Testers | Testing team members. |
| Methods | This unit testing is carried out in the unit or components in the system by breaking out each part of the source code into the unit and checking that each part works properly. |
| Timing | After the exploratory test is completed. |

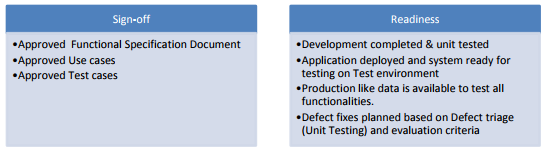
### **1.2.3. Interface test**

[Table 3] Interface test

|  |  |
| --- | --- |
| Purpose | The interface that a connection integrates two components is tested to ensure that end-users or customers should not encounter any problem when using the system. |
| Scope | Chapter four will detail the scope of the interface test. |
| Testers | Testing team members. |
| Methods | Interface test is performed by using abstract test cases and creating concrete instances of the test case for each implementation of interface testing strategy. |
| Timing | After the unit test is completed. |

### **1.2.4. Test Acceptance Criteria**

* Approved Functional Specification document, use case documents must be available prior to the start of the test design phase.
* Test cases approved and signed-off prior to start of Test execution.
* Development completed, unit tested with pass status and results shared to Testing team to avoid duplicate defects.
* Test environment with application installed, configured and ready to use state.



[Figure 1] Sign-off and readiness

## **1.3. Definitions, Acronyms, and Abbreviations**

The following table explains the acronyms and abbreviations used in this document.

[Table 4] Acronyms and abbreviations

|  |  |
| --- | --- |
| Acronyms& Abbreviations | Explanation |
| API | Application Program Interface |
| GUI | Graphical user interface |
| PM | Project manager |
| UAT | User acceptance testing |
| CM | Configuration Management |
| QA | Quality Assurance |

[Table 5] Terms and definitions

|  |  |
| --- | --- |
| Terms | Definitions |
| User | Someone who uses a system |
| Client (user  device) | A user device/user that connected to server |
| Server | A computer or computer program which manages access to a centralized resource or service in a network |
| Software | The programs and other operating information used by a computer |
| Unit | A unit is the smallest testable part of any software |
| Interface | A connection that integrates two components |

## **1.4. References**

* SoftwareTestingHelp.com Live Project Training - OrangeHRM, 02.05.2014, www.SoftwareTestingHelp.com
* Test Management Plan, Connecticut Department of Social Services
* IEEE 829-1998 Format, Test Plan Template, In IEEEXplore Digital Library, www.ecs.csum.edu

## **1.5. Overview**

The remainder of this Software Test Specification document includes four chapters inside. The second chapter describes an approach on how to run the test. Test methods and test tools used in '유생 찾기' are introduced, and test methods are divided into unit test methods and interface test methods. It also describes test assumptions, and tests can work well only in situations where all the assumptions are applicable.

The third chapter covers the entire software unit test. Since the '유생 찾기' service is a community-based service, there are various units and components such as log-in, member sign-up, chat, and review, and provides methods to test whether the function of the unit is working properly. Each test includes test input and expected result, and if the test fails, the developer has to fix the unit.

The fourth chapter covers software interface testing. Interface testing involves two main segments. The web server and application server interface and the application server and database server interface are included in the interface covered by interface testing. Interface testing inspects whether the server is operating properly, whether errors are properly managed, and outputs appropriate error messages when errors occur.

Finally, the fifth chapter provides supporting information such as test environment and test schedule. All members contributed equally to the production of this project. I hope readers enjoy this document.

## **1.6. Execution Strategy**

### **1.6.1. Test Cycles**

* There will be two cycles for testing. Each cycle will execute all the scripts.
* The objective of the first cycle is to identify any blocking, critical defects, and most of the high defects. It is expected to use some work-around in order to get to all the scripts.
* The objective of the second cycle is to identify remaining high and medium defects, remove the work-around from the first cycle, correct gaps in the scripts and obtain performance results.

### **1.6.2. Validation and Defect Management**

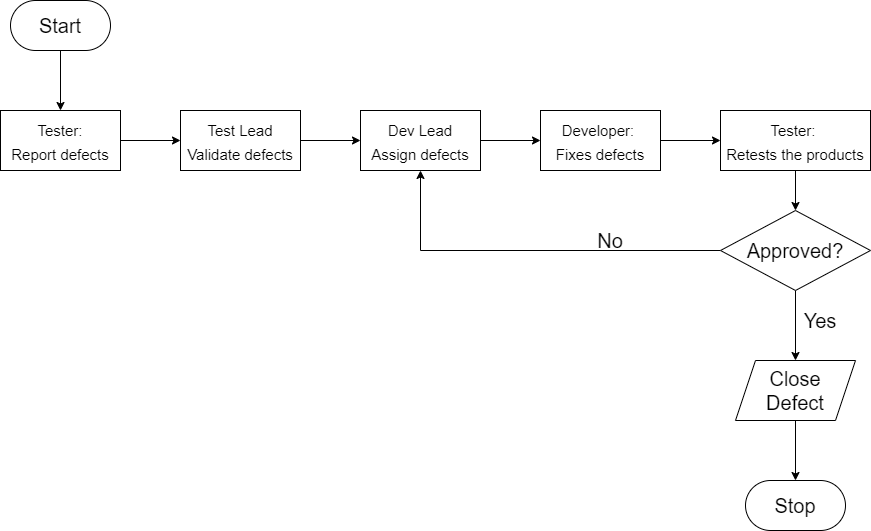
It is expected that the testers execute all the scripts in each of the cycles described above. However, it is recognized that the testers could also do additional testing if they identify a possible gap in the scripts.

Defects found during the Testing will be categorized according to the bug-reporting tool “HP Quality Center” and the categories are:

[Table 6] Defect level

|  |  |
| --- | --- |
| Severity | Impact |
| 1 (Critical) | * This bug is critical enough to crash the system, cause file corruption, or cause potential data loss. * It causes an abnormal return to the operating system (crash or a system failure message appears). * It causes the application to hang and requires rebooting the system. |
| 2 (High) | * It causes a lack of vital program functionality with workaround. |
| 3 (Medium) | * This Bug will degrade the quality of the service. However, there is an intelligent workaround for achieving the desired functionality - for example through another screen. * This bug prevents other areas of the product from being tested. However other areas can be independently tested. |
| 4 (Low) | * There is an insufficient or unclear error message, which has minimum impact on product use. |
| 5 (Cosmetic) | * There is an insufficient or unclear error message, which has minimum impact on product use. |

### **1.6.3. Defect Tracking & Reporting**

Following flowchart depicts Defect Tracking Process:

[Figure 2] Defect tracking process

## **1.7. Entry and Exit Criteria**

### **1.7.1. Entry Criteria**

* All test hardware platforms must have been successfully installed, configured, and functioning properly.
* All the necessary documentation, design, and requirements information should be available that will allow testers to operate the system and judge the correct behavior.
* All the standard software tools including the testing tools must have been successfully installed and functioning properly.
* Proper test data is available.
* The test environment such as, lab, hardware, software, and system administration support should be ready.
* QA resources have completely understood the requirements.
* QA resources have sound knowledge of functionality.
* Reviewed test scenarios, test cases and RTM.

### **1.7.2. Exit Criteria**

* A certain level of requirements coverage has been achieved.
* No high priority or severe bugs are left outstanding.
* All high-risk areas have been fully tested, with only minor residual risks left outstanding.
* Cost – when the budget has been spent.
* The schedule has been achieved.

# **2. Approach**

This part introduces the approach of testing. A brief introduction to software unit testing and software interface testing, what to assume before testing, and test tools are introduced. This part is a summary of the tests, since more detailed tests will be included in the documentation.

## **2.1 Test method**

### **2.1.1 Software unit test methods**

It expresses the contents of software unit test in pseudo code.

[Figure 3] Register

**Register**

# get user information u

if (u == non-existent)

if (u == valid)

success;

else

fail; --- existing user

else

fail; --- invalid student information

[Figure 4] Login

**Login**

# get user id i and user password p

if (i and p == valid)

success;

else

fail; --- invalid account

[Figure 5] Profile

**Profile**

# get user profile information p

if (p == valid)

success;

else

fail; --- Unable to set profile

[Figure 6] Maps & Search

**Map & Search**

# test map display

test1 = basic map display

test2 = map display with and without zoom

# test search result

test3 = timing about the search results come out

test4 = correct search results

[Figure 7] Filtering

**Filtering**

# test filtering

test = correct filtering results

# test button

previous button = p

next button = n

if (p == invalid with the first question)

success;

if (n == invalid with the last question)

success;

[Figure 8] Review

**Review**

# length of a review = l

if (0 <= l <= 200)

success;

if (l > 200)

if (l becomes 200)

success; --- The content is truncated to 200 characters

else

fail;

### **2.1.2 Software interface test methods**

The tests will be conducted on Firebase, Naver Map API, and Apache that will be connected to our application. For each, connection test, stress test, and data test are performed as needed. Description of Firebase, Naver Map API, Apache, and detailed description and method of test will be described later in Part 4.

## **2.2 Test Assumptions**

### **2.2.1 Key Assumptions**

* The data required for the system before each test if provided and available in advance.
* In each testing phase, cycles are performed up to 5 times, and if the defect rate in cycle 3 is high, 2 additional cycles are performed.

### **2.2.2 General Assumptions**

* Defects that occur during testing are saved by taking snapshots in JPEG format.
* Testing conducted in a mobile environment is performed through team members’ mobile devices.
* Testing conducted in the Internet environment is conducted through each team members’ Internet environment and VPN connection.
* When testing is conducted by external personnel, project team members provide support for test planning, test design, and test execution.
* When a defect is found, the test is stopped immediately, the defect is fixed through the project team member, and the test is performed from the beginning.
* Regardless of what project team members oversee, everyone should be familiar with the test information and share any defects or changes in the test schedule.
* After the end of the test, a review is performed by the project team members, and when everyone is satisfied with the test result, the project team leader performs the final review.
* After the project team leader’s review is over, the test may be terminated depending on the review result.

## **2.3 Testing tools**

[Table 7] Testing tools

|  |  |
| --- | --- |
| **Process** | **Tool** |
| Test case creation | Microsoft Excel |
| Test case tracking | Microsoft Excel |
| Test case execution | Java, Python, C++ |
| Test case management | Microsoft Excel |
| Defect management | Paint, Microsoft Word |
| Test reporting | PDF |
| Check list creating | Microsoft Excel |

* Microsoft Excel: It is a program mainly used to write documents for statistics and has the advantage of being easy to see the written contents at a glance. Microsoft Excel is most suitable for listing and organizing test cases and managing check list.
* Paint, Microsoft Word: If a defect is found, it is saved in JPEG format through Paint and documented in Microsoft Word.
* Java, Python, C++: Test cases will be performed through the most commonly used programming language, and as described in the proposal, the main work will be done through Java.
* PDF: Final test documentation is produced via PDF.

# **3. Unit-Test**

In this chapter, we will show scope of the unit test based on each page. We have categorized the functions to be tested on each page. There are several functions on each page, and we designed validation testing and defect testing for each function to test them.

From now on, we will show functions in the categorized page and its validation testing and defect testing.

## **3.1 Log-in page**

### **3.1.1 Register**

In the case of a user who has already registered, a new register should not be possible.

1. Click the register button.
2. Fill the user information text boxes with various cases of input.
3. Click the complete button.
4. Wait for the output message.

**<Test case>**

|  |  |
| --- | --- |
| (Success case) | |
| Input | Non-existent user information |
| Output message | “Success to register” |

|  |  |
| --- | --- |
| (Failure case) | |
| Input | Existing user information |
| Output message | “Existing user. Go to the log-in page.” |

An error message is sent for an invalid authentication method.

1. Click the register button.
2. Click the authentication button.
3. Take pictures of various authentications.
4. Wait for the output message.

**<Test case>**

|  |  |
| --- | --- |
| (Success case) | |
| Input | Student ID |
| Output message | “Authentication was successful” |

|  |  |
| --- | --- |
| (Failure case) | |
| Input | Registration card |
| Output message | “Invalid authentication. Try again with valid student ID” |

|  |  |
| --- | --- |
| (Failure case) | |
| Input | Driver’s license |
| Output message | “Invalid authentication. Try again with valid student ID” |

Proper length of student id number (10 for SKKU) should be set for “Student id number” text box.

1. Click the register button.
2. Fill the “student id number” text box.
3. Wait for the output message.

**<Test case>**

|  |  |
| --- | --- |
| (Success case) | |
| Input | 2020111111 |
| Output message | “Valid student ID number” |

|  |  |
| --- | --- |
| (Failure case) | |
| Input | 2020111111111 |
| Output message | “Invalid student ID number” |

|  |  |
| --- | --- |
| (Failure case) | |
| Input | 202011 |
| Output message | “Invalid student ID number” |

### **3.1.2 Login**

Minimum (0) and maximum (20) lengths should be set for all the text boxes.

1. Fill the user id text box.
2. Fill the user password text box.
3. Click the log-in button.
4. Check for the textbox result.

**<Test case>**

|  |  |
| --- | --- |
| (Success case) | |
| User id input (keyboard) | skkustudent1 |
| Textbox result | skkustudent1 |

|  |  |
| --- | --- |
| (Failure case) | |
| User id input (keyboard) | Skkudepartmentofsoftware |
| Textbox result | skkudepartmentofsoft |

Blank should not be set for all the text boxes.

1. Fill the user id text box.
2. Fill the user password text box.
3. Click the log-in button.
4. Check for the textbox result.

**<Test case>**

|  |  |
| --- | --- |
| (Success case) | |
| User id input (keyboard) | skkustudent1 |
| Textbox result | skkustudent1 |

|  |  |
| --- | --- |
| (Failure case) | |
| User id input (keyboard) | skku student 1 |
| Textbox result | skkustudent1 |

Password should be displayed in masked format.

1. Fill the user password text box.
2. Click the log-in button.
3. Check for the textbox result.

**<Test case>**

|  |  |
| --- | --- |
| (Success case) | |
| User id input (keyboard) | skku1234! |
| Textbox result | ●●●●●●●●● |

|  |  |
| --- | --- |
| (Failure case) | |
| User id input (keyboard) | skku1234! |
| Textbox result | skku1234! |

Validation message should be shown when invalid username and/or password is entered, and the fields are left blank.

1. Fill the user id text box.
2. Fill the user password text box.
3. Click the log-in button.
4. Check for the output message.

**<Test case>**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| (Success case) | | | | |
| Real user id | skkustudent1 | | Real user password | software1234! |
| Input user id | skkustudent1 | | Input user password | software1234! |
| Output message | | “Success to access the account” | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| (Failure case) | | | | |
| Real user id | skkustudent1 | | Real user password | software1234! |
| Input user id | skkustudent1 | | Input user password | skku1234! |
| User id textbox result | | (blank) | | |
| User password textbox result | | (blank) | | |
| Output message | | “Invalid user id or password. Try again!” | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| (Failure case) | | | | |
| Real user id | skkustudent1 | | Real user password | software1234! |
| Input user id | skku1 | | Input user password | software1234! |
| User id textbox result | | (blank) | | |
| User password textbox result | | (blank) | | |
| Output message | | “Invalid user id or password. Try again!” | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| (Failure case) | | | | |
| Real user id | skkustudent1 | | Real user password | software1234! |
| Input user id | skku1 | | Input user password | skku1234! |
| User id textbox result | | (blank) | | |
| User password textbox result | | (blank) | | |
| Output message | | “Invalid user id or password. Try again!” | | |

## **3.2 Profile page**

### **3.2.1 Set profile image**

User can update his profile image if at his profile page.

1. Click image file.
2. Click change button in enlarged image.
3. Click photo album/camera button.

3-1. (When click camera button) Take a picture to set as profile image.

3-2. (When click photo album button) Select a picture to set as profile image.

1. Click update button at profile page.
2. Wait for output.

**<Test case>**

|  |  |
| --- | --- |
| (Success case) | |
| Input | User’s own profile image |
| Output message | “Update complete” |
| Output | Profile page with selected profile image |

|  |  |
| --- | --- |
| (Failure case) | |
| Input | Someone else’s profile image |
| Output message | “You do not have permission to change profile” |

The type of image must be jpg type.

1. Click image file.
2. Click change button in enlarged image.
3. Click photo album/camera button.

3-1. (When click camera button) Take a picture to set as profile image.

3-2. (When click photo album button) Select a picture to set as profile image.

1. Wait for the output.

**<Test case>**

|  |  |
| --- | --- |
| (Success case) | |
| Input | jpg file |
| Output message | “Usable image” |
| Image | Selected jpg file |

|  |  |
| --- | --- |
| (Failure case) | |
| Input | Bmp file |
| Output message | “Your image is not suitable, please select jpg file” |
| Image | Image before change |

### **3.2.2 Set nickname**

User can update his nickname at his profile page.

1. Click nickname text box.
2. Type nickname to change.
3. Click update button at profile page.
4. Wait for output.

**<Test case>**

|  |  |
| --- | --- |
| (Success case) | |
| Input | User’s own nickname |
| Output message | “Update complete” |
| Output | Profile page with changed nickname |

|  |  |
| --- | --- |
| (Failure case) | |
| Input | Someone else’s nickname |
| Output message | “You do not have permission to change profile” |

The nickname must not be blank.

1. Click nickname text box.
2. Type nickname to change.
3. Click check button to check new nickname meets condition.

**<Test case>**

|  |  |
| --- | --- |
| (Success case) | |
| Input | Skku2017^^ |
| Output message | “Usable nickname” |
| Output | Skku2017^^ |

|  |  |
| --- | --- |
| (Failure case) | |
| Input | blank |
| Output message | “Your nickname must bot be blank” |
| Text box result | blank |

The nickname consists of numbers and English letters and cannot exceed 15 characters.

1. Click nickname text box.
2. Type nickname to change.
3. Click check button to check new nickname meets condition.

**<Test case>**

|  |  |
| --- | --- |
| (Success case) | |
| Input | Skku2017^^ |
| Output message | “Usable nickname” |
| Text box result | Skku2017^^ |

|  |  |
| --- | --- |
| (Failure case) | |
| Input | sasjdkkjzxbkcqwqbkasbckxz |
| Output message | “Your nickname exceeds 15 characters” |
| Text box result | blank |

Blank should not be set for nickname.

1. Click the nickname text box.
2. Type nickname to change.
3. Click check button to check new nickname meets condition.

**<Test case>**

|  |  |
| --- | --- |
| (Success case) | |
| Input | Skku2017^^ |
| Output message | “Usable nickname” |
| Text box result | Skku2017^^ |

|  |  |
| --- | --- |
| (Failure case) | |
| Input | Skku 2017 ^^ |
| Text box result | Skku2017^^ |

Nickname must not be duplicated with other users.

1. Click the nickname text box.
2. Type nickname to change.
3. Click check button to check new nickname meets condition.

**<Test case>**

|  |  |
| --- | --- |
| (Success case) | |
| Input | Nickname that doesn’t exist in database (Skku2017^^) |
| Output message | “Usable nickname” |
| Text box result | Skku2017^^ |

|  |  |
| --- | --- |
| (Failure case) | |
| Input | Nickname that already exists in database (Superpower123) |
| Output message | “This nickname is already exists” |
| Text box result | blank |

### **3.2.3 Set major**

User can update his major at his profile page.

1. Click major text box.
2. Type major to change.
3. Click update button at profile page.
4. Wait for output.

**<Test case>**

|  |  |
| --- | --- |
| (Success case) | |
| Input | User’s own major |
| Output message | “Update complete” |
| Output | Profile page with changed major |

|  |  |
| --- | --- |
| (Failure case) | |
| Input | Someone else’s major |
| Output message | “You do not have permission to change profile” |

Major must not be blank.

1. Click major text button.
2. Type major to change.
3. Wait for message.

**<Test case>**

|  |  |
| --- | --- |
| (Success case) | |
| Input | Software engineering |
| Output message | “Valid major” |
| Text box result | Software engineering |

|  |  |
| --- | --- |
| (Failure case) | |
| Input | blank |
| Output message | “Major must not be blank” |
| Text box result | blank |

Major user typed in text box must be exist in database.

1. Click major text box.
2. Type major to change.
3. Wait for message.

**<Test case>**

|  |  |
| --- | --- |
| (Success case) | |
| Input | Software engineering |
| Output message | “Valid major” |
| Text box result | Software engineering |

|  |  |
| --- | --- |
| (Failure case) | |
| Input | Hotel culinary arts |
| Output message | “Invalid major” |
| Text box result | blank |

### **3.2.4 Set sex**

User can update his sex at his profile page.

1. Click sex button.
2. Select sex among male and female to change.
3. Click update button at profile page.
4. Wait for output.

**<Test case>**

|  |  |
| --- | --- |
| (Success case) | |
| Input | User’s own sex |
| Output message | “Update complete” |
| Output | Profile page with selected sex |

|  |  |
| --- | --- |
| (Failure case) | |
| Input | Someone else’s sex |
| Output message | “You do not have permission to change profile” |

### **3.2.5 Set introduction**

User can update his introduction at his profile page.

1. Click introduction text box.
2. Type introduction to update.
3. Click update button at profile page.
4. Wait for output.

**<Test case>**

|  |  |
| --- | --- |
| (Success case) | |
| Input | User’s own introduction |
| Output message | “Update complete” |
| Output | Profile page with changed introduction |

|  |  |
| --- | --- |
| (Failure case) | |
| Input | Someone else’s introduction |
| Output message | “You do not have permission to change profile” |

The number of characters in the introduction must not exceed 200 characters.

1. Click introduction text box.
2. Type introduction to update.
3. Wait for output.

**<Test case>**

|  |  |
| --- | --- |
| (Success case) | |
| Input | 150 characters |
| Text box result | 150 characters |

|  |  |
| --- | --- |
| (Failure case) | |
| Input | 250 characters |
| Output message | “Introduction should not exceed 200 characters” |
| Output | 250 characters |

## **3.3 Main page**

### **3.3.1 Map**

The map only displays the current user's location as red dot and information within a 1km radius.

1. Login to the application.
2. Turn on the GPS system.
3. Wait for the map to appear on the screen.

**<Test case>**

|  |  |
| --- | --- |
| (Success case) | |
| Motion | Login to the application and wait for the main page. |
| Map result | Current user’s location displayed with red dot. |
| Only displays within a 1km radius with user’s location. |

|  |  |
| --- | --- |
| (Failure case) | |
| Motion | Login to the application and wait for the main page. |
| Map result | Current user’s location displayed with red dot. |
| Only displays more than a 1km radius with user’s location. |

|  |  |
| --- | --- |
| (Success case) | |
| Motion | Login to the application and wait for the main page. |
| Map result | Current user’s location displayed with blue dot. |
| Only displays within a 1km radius with user’s location. |

|  |  |
| --- | --- |
| (Success case) | |
| Motion | Login to the application and wait for the main page. |
| Map result | Current user’s location displayed with blue dot. |
| Only displays more than a 1km radius with user’s location. |

The map should be able to zoom in and out appropriately.

1. Login to the application.
2. Turn on the GPS system.
3. Wait for the map to appear on the screen.
4. Zoom in with two fingers and check it works properly to desired direction.
5. Zoom out with two fingers and check it works properly to desired direction.

**<Test case>**

|  |  |
| --- | --- |
| (Success case) | |
| Motion 1 | Zoom in the map. |
| Motion 2 | Zoom out the map. |
| Motion 1 result | Works properly to desired direction. |
| Motion 2 result | Works properly to desired direction. |

|  |  |
| --- | --- |
| (Failure case) | |
| Motion 1 | Zoom in the map. |
| Motion 2 | Zoom out the map. |
| Motion 1 result | Does not move. |
| Motion 2 result | Works properly to desired direction. |

|  |  |
| --- | --- |
| (Failure case) | |
| Motion 1 | Zoom in the map. |
| Motion 2 | Zoom out the map. |
| Motion 1 result | Works properly to desired direction. |
| Motion 2 result | Does not move. |

|  |  |
| --- | --- |
| (Failure case) | |
| Motion 1 | Zoom in the map. |
| Motion 2 | Zoom out the map. |
| Motion 1 result | Does not move. |
| Motion 2 result | Does not move. |

When the map is zoomed in, certain popular places are displayed with their information, but when it zoomed out, that places are displayed as blue dots only.

1. Login to the application.
2. Turn on the GPS system.
3. Wait for the map to appear on the screen.
4. Zoom in with two fingers and check it displays places properly.
5. Zoom out with two fingers and check it displays places properly.

**<Test case>**

|  |  |
| --- | --- |
| (Success case) | |
| Motion 1 | Zoom in the map. |
| Motion 2 | Zoom out the map. |
| Motion 1 result | Places are displayed as blue dots. |
| Places have their information in their right side. |
| Motion 2 result | Places are displayed as blue dots. |
| Places do not have any information in their right side. |

|  |  |
| --- | --- |
| (Failure case) | |
| Motion 1 | Zoom in the map. |
| Motion 2 | Zoom out the map. |
| Motion 1 result | Places are displayed as red/no dots. |
| Places do not have any information in their right side. |
| Motion 2 result | Places are displayed as blue dots. |
| Places do not have any information in their right side. |

|  |  |
| --- | --- |
| (Failure case) | |
| Motion 1 | Zoom in the map. |
| Motion 2 | Zoom out the map. |
| Motion 1 result | Places are displayed as blue dots. |
| Places have their information in their right side. |
| Motion 2 result | Places are displayed as red/no dots. |
| Places have their information in their right side. |

|  |  |
| --- | --- |
| (Failure case) | |
| Motion 1 | Zoom in the map. |
| Motion 2 | Zoom out the map. |
| Motion 1 result | Places are displayed as red/no dots. |
| Places do not have any information in their right side. |
| Motion 2 result | Places are displayed as red/no dots. |
| Places have their information in their right side. |

### **3.3.2 Search**

Search result should be displayed when user click the search button or press the enter key.

1. Click the search textbox.
2. Write the search keyword in textbox.
3. Wait for the result.
4. Click the search button or press the enter key.
5. Wait for the result.

**<Test case>**

|  |  |
| --- | --- |
| (Success case) | |
| Motion | Write “Alchon” and click search button |
| Search result | Information of “Alchon” |

|  |  |
| --- | --- |
| (Success case) | |
| Motion | Write “Alchon” and press enter key |
| Search result | Information of “Alchon” |

|  |  |
| --- | --- |
| (Failure case) | |
| Motion | Write “Alchon” |
| Search result | Information of “Alchon” |

When user start typing word in text box it should suggest words that matches typed keyword.

1. Click the search textbox.
2. Write the search keyword in textbox.
3. Check the changed textbox.

**<Test case>**

|  |  |
| --- | --- |
| (Success case) | |
| Motion | Write the keyword in textbox. |
| Input content | “Alchon” |
| Textbox result | Display “Alchon” simultaneously |

|  |  |
| --- | --- |
| (Failure case) | |
| Motion | Write the keyword in textbox. |
| Input content | “Alchon” |
| Textbox result | Display “Alchon” not simultaneously |

Search result displayed should be relevant to search keyword.

1. Click the search textbox.
2. Write the search keyword in textbox.
3. Wait for the search result.

**<Test case>**

|  |  |
| --- | --- |
| (Success case) | |
| Motion | Write the keyword in textbox. |
| Input content | “Alchon” |
| Search result | Information of “Alchon” |

|  |  |
| --- | --- |
| (Failure case) | |
| Motion | Write the keyword in textbox. |
| Input content | “Alchon” |
| Search result | Information of “Subway” |

## **3.4 Filtering page**

User can update recommendation friend list by choosing one item when receive some questions. User must select at least one of the items corresponding to the question.

1. Click the Next button until the question is finished.
2. Click Finish button when question is finished.
3. Wait for output.

**<Test case>**

|  |  |
| --- | --- |
| (Success case) | |
| Input | Data with selecting for all items per question |
| Output page | Updated recommendation friend list page |

|  |  |
| --- | --- |
| (Failure case) | |
| Input | Data without selecting one or more of the questions |
| Output message | “You have to select at least one of the items corresponding question” |
| Output page | First Pages with questions with no item selected |

User can move to next question by clicking Next button until last question.

1. Click Next button.
2. Wait for output.

**<Test case>**

|  |  |
| --- | --- |
| (Success case) | |
| Input | Page that is not the last question |
| Output page | Page with the following question |

|  |  |
| --- | --- |
| (Failure case) | |
| Input | Page with last question |
| Output message | “This is the last question” |
| Output page | Page with last question |

User can move to previous question by clicking Previous button if current question is not first question.

1. Click Previous button.
2. Wait for output.

**<Test case>**

|  |  |
| --- | --- |
| (Success case) | |
| Input | Page that is not the first question |
| Output page | Page with previous question |

|  |  |
| --- | --- |
| (Failure case) | |
| Input | Page with first question |
|  | “This is the first question” |
| Output page | Page with first question |

## **3.5 Review**

The length of a review is longer than 0 and shorter than 200.

1. Close the chatting room.
2. Write the review.
3. Click the submit button.
4. Check the review textbox.

**<Test case>**

|  |  |
| --- | --- |
| (Success case) | |
| Textbox input | “Alchon is a great restraint that deserves an excellent rating in its atmosphere, but it is certainly not a great restaurant.” |
| Textbox result | “Alchon is a great restraint that deserves an excellent rating in its atmosphere, but it is certainly not a great restaurant.” |

|  |  |
| --- | --- |
| (Success case) | |
| Motion | “Alchon is a great restraint that deserves an excellent rating in its atmosphere, but it is certainly not a great restaurant. It is worth paying a visit and I definitely suggest! Service was friendly and the food was fantastic.” |
| Search result | “Alchon is a great restraint that deserves an excellent rating in its atmosphere, but it is certainly not a great restaurant. It is worth paying a visit and I definitely suggest! Service was friendly an” |

|  |  |
| --- | --- |
| (Failure case) | |
| Motion | “Alchon is a great restraint that deserves an excellent rating in its atmosphere but it is certainly not a great restaurant. It is worth paying a visit and I definitely suggest! Service was friendly and the food was fantastic.” |
| Search result | “Alchon is a great restraint that deserves an excellent rating in its atmosphere but it is certainly not a great restaurant. It is worth paying a visit and I definitely suggest! Service was friendly and the food was fantastic.” |

# **4. Software Interface Test**

## **4.1. Objectives**

In this chapter, we describe the test plan of the interface between our application and the database, API, and web server. We use Realtime Database provided by Firebase, Naver Map API for the map searching process, and the Apache web server for our ‘유생찾기’ project. While doing the interface test, we will verify that the server is working properly, that a problem between the application and the server is handled normally and returns an error message, and that it produces the correct result when the connection between the server is reset. Through testing, we can check that there are no problems when using our application, that communication within the system is smooth, and that errors can be handled in the event of a network failure.

## **4.2. Firebase**

We can test the connection and stress testing using the Firebase Performance Monitoring service. Using the Firebase Performance Monitoring service, we can collect the performance data from applications using the Performance Monitoring SDK and review and analyze the data in the Firebase Console.

## **4.2.1 Connection Test**

In the connection test, we check that the database server is connected to our application and what the connection state is. The connection test will be done using the database profiler tool which is provided by the Firebase.

**4.2.1.1 Test Method**

[Table 8] Firebase Test Method – Connection Test

|  |  |
| --- | --- |
| **Input** | Database profiler tool which provided by Firebase  firebase database: profile |
| **Expected results** | Speed, bandwidth, and unindexed queries. |
| **Error handling** | If speed report return permission denied property, show the communication error information message. |
| **Condition** | N/A |

### **4.2.2 Stress Test**

In the stress test, we measure the database performance on its robustness and error handling capabilities under the heavy load conditions. The stress test on the Firebase realtime database will be done by using the Monitor Database Usage in Cloud Monitoring service.

**4.2.2.1 Test Method**

[Table 9] Firebase Test Method – Stress Test

|  |  |
| --- | --- |
| **Input** | Monitor Database Usage in Cloud Monitoring |
| **Expected results** | Connections, storage, downloads, load |
| **Error handling** | If load approaches 100%, show the error information message. |
| **Condition** | The limitation supported by Firebase is 200000 of simultaneous connections, and ~100000/second simultaneous responses sent from a single database.  Database should be connected first. |

### **4.2.3 Data Test**

In the data test, we test if the database is working properly while communicating with our application and handling the data request properly.

**4.2.3.1 Test Method**

[Table 10] Firebase Test Method – Empty Input

|  |  |
| --- | --- |
| **Input** | Empty input to store in the database |
| **Expected results** | Return the error message as it has not null attributes. |

[Table 11] Firebase Test Method – Exceed Input

|  |  |
| --- | --- |
| **Input** | Input value that exceeds the limit of the attribute of the database table. |
| **Expected results** | Return the error information message to request again. |

[Table 12] Firebase Test Method – Request does not exist.

|  |  |
| --- | --- |
| **Input** | Request data which is not in the database |
| **Expected results** | Return the no result message. |

[Table 13] Firebase Test Method – Normal Request

|  |  |
| --- | --- |
| **Input** | Request data which is in the database |
| **Expected results** | Return the data corresponding to the request. |

## **4.3. Naver Map API**

### **4.3.1 Connection Test**

In the connection test, we check that the API is working properly with our application and what the connection state is. The connection test on the Naver Map API will be done by using the Naver Cloud Platform Gov Client for connect request.

**4.3.1.1 Test Method**

[Table 14] API Test Method – Connection Test

|  |  |
| --- | --- |
| **Input** | Connect request using Naver Cloud Platform Gov Client |
| **Expected results** | Successfully connected message. |
| **Error handling** | If it returns 401 error code, it is exception of unauthorized client, so re-assign the client.  If it returns 429 error code, it is exception of exceeded limit, so shows the exceeded information message.  If it returns 800 error code, it is exception of unspecified client, so re-assign the client. |
| **Condition** | N/A |

### **4.3.2 Data Test**

In the data test, we test if the Naver Map API is working properly with our application and handling the information request properly.

**4.3.2.1 Test Method**

[Table 15] API Test Method – Empty Request

|  |  |
| --- | --- |
| **Input** | Request empty input |
| **Expected results** | Return the no result message. |

[Table 16] API Test Method – Normal Request

|  |  |
| --- | --- |
| **Input** | Request search location |
| **Expected results** | Return data information for response |

[Table 17] API Test Method – Request does not exist.

|  |  |
| --- | --- |
| **Input** | Request search location which has no corresponding result |
| **Expected results** | Return the no result message. |

## **4.4. Apache**

We can test the connection and stress testing using the ab (Apache HTTP server benchmarking tool). Using the Apache ab, we can collect the performance data of Apache server by inundating web server with HTTP requests and recording metrics about latency and success. It helps to determine how much traffic can sustain before performance degrades and set a baseline for typical response times.

### **4.4.1 Connection Test**

In the connection test, we check that the Apache server is connected to our application and what the connection state is. The connection test on the Apache web server will be done by using the Apache HTTP server benchmarking tool.

**4.4.1.1 Test Method**

[Table 18] Apache Test Method – Connection Test

|  |  |
| --- | --- |
| **Input** | Apache ab  ab -v [additional option] [domain name] |
| **Expected results** | Time taken for tests, complete requests, failed requests, requests per second, time per request. |
| **Error handling** | For the failed request with non-2xx responses, shows the error information message. |
| **Condition** | N/A |

### **4.4.2 Stress Test**

In the stress test, we measure the web server on its robustness and error handling capabilities under the heavy load conditions. The stress test on Apache web server will be done by using the Apache HTTP server benchmarking tool.

**4.4.2.1 Test Method**

[Table 19] Apache Test Method – Stress Test

|  |  |
| --- | --- |
| **Input** | Apache ab  ab -n 1000 -c 20 -v [additional option] [domain name] |
| **Expected results** | Time taken for tests, complete requests, failed requests, requests per second, time per request. |
| **Error handling** | For the failed request, shows the error information message. |
| **Condition** | Web server should be connected first. |

# **5. Supporting Information**

## **5.1. Test Environment**

**5.1.1. Tools**

[Table 20] Testing Tools

|  |  |
| --- | --- |
| Testing Tools | |
| Support Level 1  (browsers) | * Windows 10: Edge, Chrome(latest), Firefox(latest), Safari(latest) * Mac OS X: Chrome(latest), Firefox(latest), Safari(latest) * Linus Ubuntu: Chrome(latest), Firefox(latest) |
| Support Level 1  (devices) | * iphone 11/12, ipad 8, Samsung Galaxy S21, Galaxy Tab S6 |
| Support Level 2 | * Windows 10: IE 11, Chrome(latest), Firefox(latest), Safari(latest) |
| Support Level 3 | * Anything else |

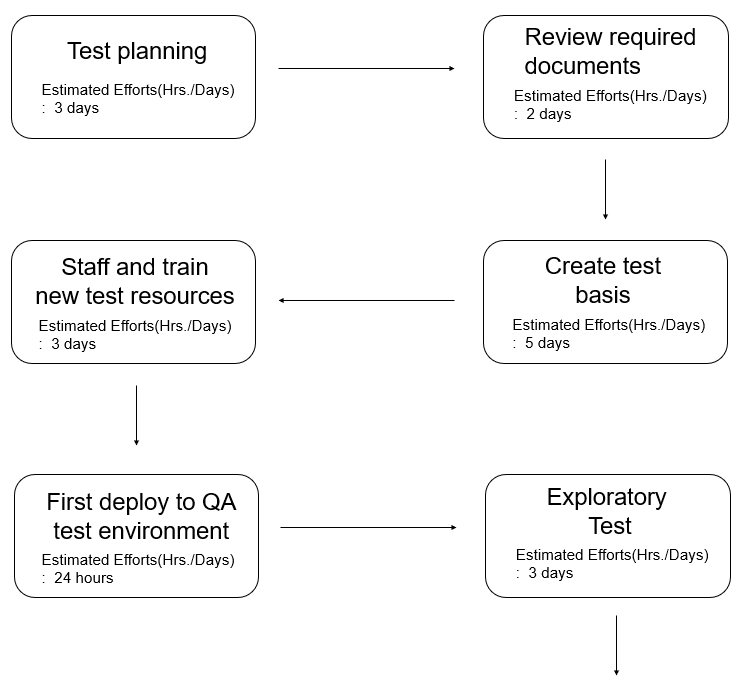
### **5.1.2. Staffing**

[Table 21] Test Staffing

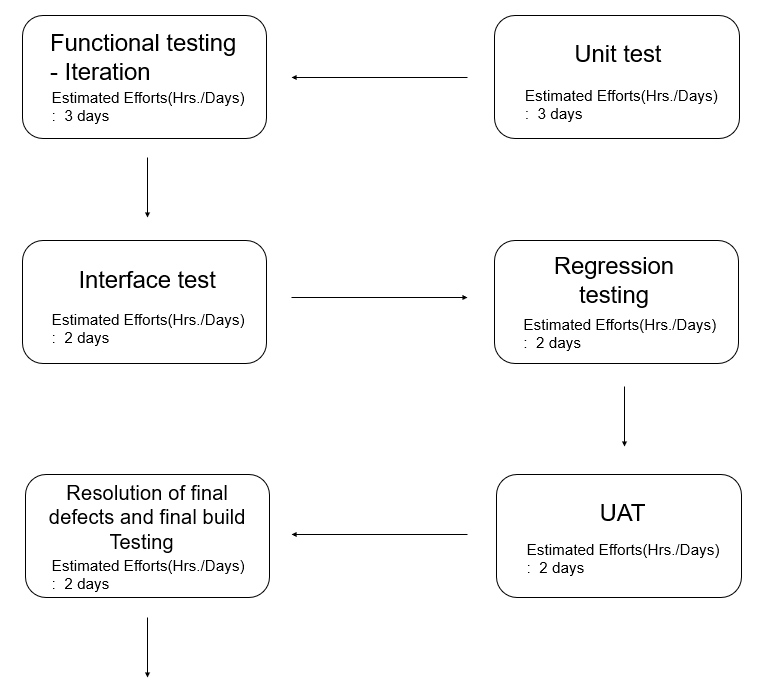
|  |  |
| --- | --- |
| Responsibilities | Team member |
| Test Plan | Eunju Seok, Hyejoon Jang |
| Test Case | Jiwon Seo, Hyeyeong Kim |
| Manual Testing | Georyang Park, Eunji Gil |

## **5.2. Test Schedule**

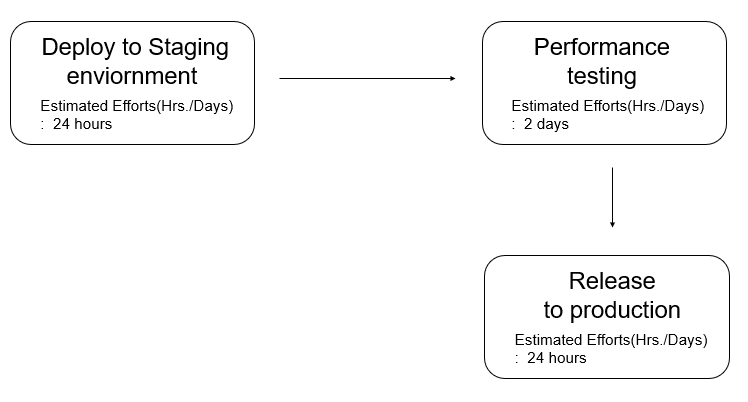
### **5.2.1. Scheduling**



[Figure 9] Test Scheduling



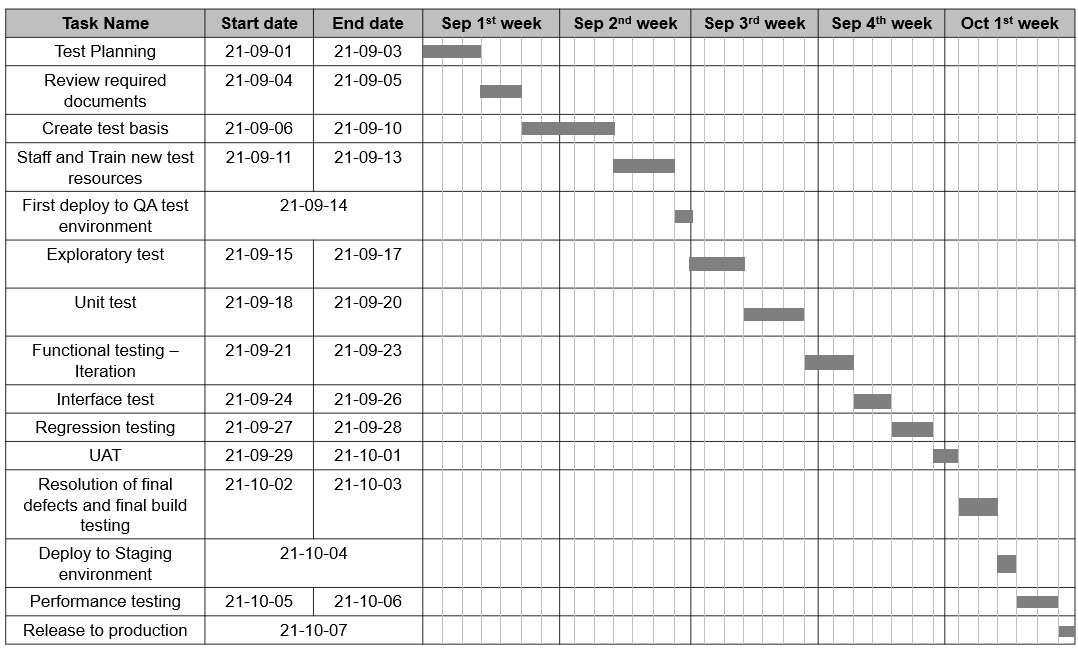
[Figure 10] Test Scheduling2



[Figure 11] Test Scheduling3

### **5.2.2. Timeline**

[Table 22] Test Timeline



## **5.3. Document**

This test plan specification was written in accordance with format of attached file that the professor provided to i-campus.

### **5.3.1. Document History**

[Table 23] Document History

| Date | Version | Description | Writer |
| --- | --- | --- | --- |
| 2021/05/19 | 0.1 | Style and overview | Eunju Seok |
| 2021/05/20 | 1.0 | Addition of 4 | Hyejoon Jang |
| 2021/05/24 | 1.1 | Addition of 1 | Eunju Seok |
| 2021/05/25 | 1.2 | Revision of 4 | Hyejoon Jang |
| 2021/05/25 | 1.3 | Addition of 3.1, 3.2 | Jiwon Seo |
| 2021/05/26 | 1.4 | Addition of 3.3, 3.4 | Hyeyeong Kim |
| 2021/05/26 | 1.5 | Addition of 3 | Jiwon Seo |
| 2021/05/27 | 1.6 | Revision of 3 | Hyeyeong Kim |
| 2021/05/27 | 1.7 | Addition of 2 | Georyang Park |
| 2021/05/28 | 1.8 | Addition of 5.1, 5.2 | Eunji Gil |
| 2021/05/28 | 1.9 | Addion of 5.3 | Eunji Gil |
| 2021/05/29 | 2.0 | Revision of 2 | Georyang Park |
| 2021/05/29 | 2.1 | Integration | Eunju Seok |