

**Finding Friends for the Same Purpose**

**at Close Distance**

**Software Requirement Specification**

2020.04.25.

**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 |

CONTENTS

[**1. Introduction** 8](#_Toc70170786)

[1.1. Purpose 8](#_Toc70170787)

[1.2. Scope 8](#_Toc70170788)

[1.3. Definitions, Acronyms, and Abbreviation 9](#_Toc70170789)

[1.4. References 10](#_Toc70170790)

[1.5. Overview 10](#_Toc70170791)

[**2. Overall Description** 11](#_Toc70170792)

[2.1. Product perspective 11](#_Toc70170793)

[2.1.1. Market status 11](#_Toc70170794)

[2.1.2. Overall structure 12](#_Toc70170795)

[2.2. Product functions 12](#_Toc70170796)

[2.2.1. Create user account and log-in 12](#_Toc70170797)

[2.2.2. View the Map 12](#_Toc70170798)

[2.2.3. Choose the Category 13](#_Toc70170799)

[2.2.4. Icons 13](#_Toc70170800)

[2.2.5. Rating and Review 13](#_Toc70170801)

[2.2.6. Block other users 13](#_Toc70170802)

[2.3. User Classes and Characteristics 13](#_Toc70170803)

[2.3.1. User 13](#_Toc70170804)

[2.3.2. System Manager 14](#_Toc70170805)

[2.4. Operating Environment 14](#_Toc70170806)

[2.4.1. Hardware 14](#_Toc70170807)

[2.4.2. Software 14](#_Toc70170808)

[2.5. Design and Implementation Constraints 14](#_Toc70170809)

[2.6. User Documentation 15](#_Toc70170810)

[2.7. Assumptions and Dependencies 15](#_Toc70170811)

[**3. Specific Requirements** 16](#_Toc70170812)

[3.1. External Interface Requirements 16](#_Toc70170813)

[3.1.1. User Interfaces 16](#_Toc70170814)

[3.1.2. Hardware Interfaces 31](#_Toc70170815)

[3.1.3. Software Interfaces 31](#_Toc70170816)

[3.1.4. Communication Interfaces 32](#_Toc70170817)

[3.2. Functional Requirements 32](#_Toc70170818)

[3.2.1. Use Case 32](#_Toc70170819)

[3.2.2. Use Case Diagram 44](#_Toc70170820)

[3.2.3. Data Dictionary 45](#_Toc70170821)

[3.2.4. Data Flow Diagram 50](#_Toc70170822)

[3.3. Performance Requirements 50](#_Toc70170823)

[3.3.1. Static numerical requirement 50](#_Toc70170824)

[3.3.2. Dynamic numerical requirement 51](#_Toc70170825)

[3.4. Logical Database Requirements 51](#_Toc70170826)

[3.5. Design Constraints 51](#_Toc70170827)

[3.6. Standards compliance 51](#_Toc70170828)

[3.7. Software System Characteristics 52](#_Toc70170829)

[3.7.1. Product Requirements 52](#_Toc70170830)

[3.7.2. Organizational Requirements 53](#_Toc70170831)

[3.7.3. External Requirements 53](#_Toc70170832)

[3.8. Organizing the Specific Requirements 54](#_Toc70170833)

[3.8.1. Context Model 55](#_Toc70170834)

[3.8.2. Process Model 56](#_Toc70170835)

[3.8.3. Interaction Model 56](#_Toc70170836)

[3.8.4. Behavior Model 56](#_Toc70170837)

[3.9. System Architecture 57](#_Toc70170838)

[3.10. System Evolution 58](#_Toc70170839)

[3.10.1. Limitation and Assumption 58](#_Toc70170840)

[3.10.2. Evolutions of Map Application and Change of User Requirements 59](#_Toc70170841)

[**4. Supporting Information** 59](#_Toc70170842)

[4.1. Software Requirement Specification 59](#_Toc70170843)

[4.2. Document History 60](#_Toc70170844)

**LIST OF FIGURE**

[[Figure 1] System Structure 12](#_Toc70173255)

[[Figure 2] Use case diagram 45](#_Toc70173256)

[[Figure 3] Entity Relationship Diagram 49](#_Toc70173257)

[[Figure 4] Data flow diagram 50](#_Toc70173258)

[[Figure 5] Context Model 55](#_Toc70173259)

[[Figure 6] Overall Process Model 56](#_Toc70173260)

[[Figure 7] Sequence Diagram 57](#_Toc70173261)

[[Figure 8] System architecture of the system 58](#_Toc70173262)

**LIST OF TABLES**

[[Table 1] Table of acronyms and abbreviations 10](#_Toc70173294)

[[Table 2] Table of terms and definitions 11](#_Toc70173295)

[[Table 3] Starting Interface 18](#_Toc70173296)

[[Table 4] Main Interface 19](#_Toc70173297)

[[Table 5] Register Interface 20](#_Toc70173298)

[[Table 6] User profile Interface 21](#_Toc70173299)

[[Table 7] Friend lists Interface 23](#_Toc70173300)

[[Table 8] Chat room list Interface 24](#_Toc70173301)

[[Table 9] Chat room Interface 25](#_Toc70173302)

[[Table 10] Recommendation Interface 27](#_Toc70173303)

[[Table 11] Filtering user preference Interface 28](#_Toc70173304)

[[Table 12] Search Interface - Location of targets 29](#_Toc70173305)

[[Table 13] Information Interface of target place 30](#_Toc70173306)

[[Table 14] Review Interface 31](#_Toc70173307)

[[Table 15] Hardware Interface 33](#_Toc70173308)

[[Table 16] Software Interface 33](#_Toc70173309)

[[Table 17] Communication Interface 34](#_Toc70173310)

[[Table 18] Use case of register 35](#_Toc70173311)

[[Table 19] Use case of log-in/out 36](#_Toc70173312)

[[Table 20] Use case of find ID/password 36](#_Toc70173313)

[[Table 21] Use case of profile 37](#_Toc70173314)

[[Table 22] Use case of user filter 38](#_Toc70173315)

[[Table 23] Use case of friend request 38](#_Toc70173316)

[[Table 24] Use case of delete friend 39](#_Toc70173317)

[[Table 25] Use case of block the user 40](#_Toc70173318)

[[Table 26] Use case of report the user 40](#_Toc70173319)

[[Table 27] Use case of place review 41](#_Toc70173320)

[[Table 28] Use case of search location 41](#_Toc70173321)

[[Table 29] Use case of check information of the place 42](#_Toc70173322)

[[Table 30] Use case of open/close a chat room 43](#_Toc70173323)

[[Table 31] Use case of enter/leave the chat room 44](#_Toc70173324)

[[Table 32] Use case of view a chat list 44](#_Toc70173325)

[[Table 33] Use case of student ID confirmation 45](#_Toc70173326)

[[Table 34] Use case of reported user confirmation 45](#_Toc70173327)

[[Table 35] User 48](#_Toc70173328)

[[Table 36] User’s filter 48](#_Toc70173329)

[[Table 37] User’s friend 48](#_Toc70173330)

[[Table 38] Chat room 49](#_Toc70173331)

[[Table 39] Chat 49](#_Toc70173332)

[[Table 40] Place Rating 49](#_Toc70173333)

[[Table 41] Place review 50](#_Toc70173334)

[[Table 42] Student id 50](#_Toc70173335)

[[Table 43] Document History 62](#_Toc70173336)

# **1. Introduction**

## **1.1. Purpose**

This document is a Software Requirements Specification (SRS) for providing ‘유생찾기’ services. It is an application that users can find friends for the same purpose at close distance. This service is designed and implemented by Team 7 of the Introduction to Software Engineering at Sungkyunkwan University. The requirements for this are summarized, analyzed, and the system is designed and implemented based on the contents described. In this document, Team 7 is the main reader, and Team 7 designs and implements the functions of this service according to this specification. Additionally, professor, TAs, and team members in the Introduction to Software Engineering class can be the main readers. The purpose of this document is to outline and publish the Requirement Specification for a new mobile application for helping to find friends with the same purpose. Unlike other communities in SKKU, ‘유생찾기’ allows users to build a social network by talking with friends comfortably and pleasantly and meeting friends for the same purpose. This app uses complex algorithms that recommend friends with similar purpose and open/exit chat rooms in places users want to go together using their own profile.

## **1.2. Scope**

The system is an application that helps users find friends with the same purpose and has many opportunities to make various friends. This system is a location-based chat service. We will support users to open chat rooms where they can chat with friends based on places they want to go together. Users can set profiles and get friends’ recommendation with the same purpose. Above all, we want users to have a user experience where they can make friends easily and pleasantly at SKKU.

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

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

[Table ] Table of acronyms and abbreviations

| **Acronyms& Abbreviations** | **Explanation** |
| --- | --- |
| RAM | Random Access Memory |
| HDD | Hard Disk Drive |
| CPU | Central Processing Unit |
| SSD | Solid-State Drive |
| OS | Operating System |
| GUI | Graphical User Interface |
| API | Application Programming Interface |
| UI | User Interface |
| HTTP | Hypertext Transfer Protocol |

The following table defines certain technical terms used in this document.

[Table ] Table of terms and definitions

| **Terms** | **Definitions** |
| --- | --- |
| User | Someone who uses a system |
| System administrator | Someone who quantify the keywords included in the reviews for each laptop and manage the system |
| Back-End | Application part that is not directly accessed by the user, such as the server and database |
| Front-End | The user interface, also known as the presentation layer of an application |
| Algorithm | A set of rules or procedures followed by a computer in problem-solving operations |
| 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 |
| Network | For connect devices together so that they can share information. In this system, it usually means internet |

## **1.4. References**

* IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications, In IEEEXplore Digital Library  
  http://ieeexplore.ieee.org/Xplore/guesthome.jsp
* Team 1. “Software Requirement Specification”. SKKU, Last Modified: May. 15, 2020.  
  https://github.com/skkuse/2020spring\_41class\_team1/blob/master/docs/SRS\_TEAM1.docx
* Multimedia Service Team. “Software Requirement Specification of Multimedia Contents-aware Intelligent Information Service System”. Kangwon National University. (2007)

## **1.5. Overview**

The remainder of this Software Requirement Specification document includes three chapters and an appendix. The second chapter provides an overall description of the product's perspective, including multiple interfaces, system features, and system interactions with other systems. This chapter also introduces interactions with different types of stakeholders and systems. This chapter also discusses item details, system constraints, assumptions, and product dependencies. The third chapter provides detailed requirement specifications, including a description of the various system interfaces and software system characteristics. Various specification techniques are used to more accurately specify requirements for different users. It also displays many use cases and data dictionaries. The fourth chapter mainly deals with the priorities of requirements. This includes a support document, a timeline for this SRS document for application ‘유생찾기’. All members contributed equally to the production of this project. I hope readers enjoy this document.

# **2. Overall Description**

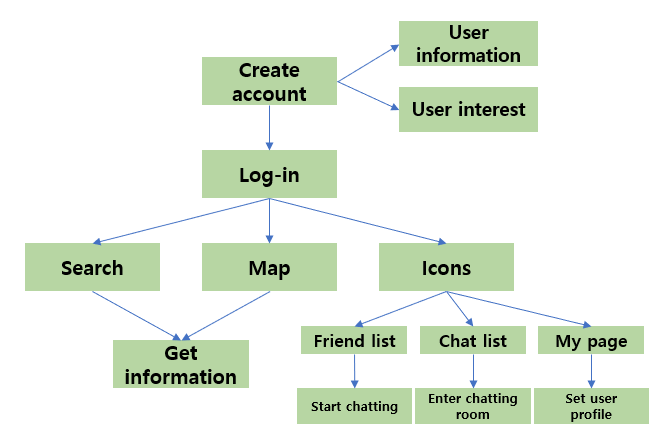
## **2.1. Product perspective**

This product is designed for college students who don’t have a lot of opportunities to make new friends in campus. This application will make it easier for those students to make friends that have same objective with them. This application will use GPS(Global Positioning System) of users, database of user information, students’ requirements and purpose of their usage.

### **2.1.1. Market status**

There are three kinds of communities to connect with students in SKKU. First, “Everytime” is a community accessible to the entire college students, and it has a lot of bulletins. Students can find some information about school life, but there is a limit to getting proper answer they want. Because there are so many questions or articles are posted in real time, so their posting is pushed back to new posting easily. Also, because of the anonymity and various topics, it is hard to promote friendship with friends that have the same purpose. Next, “Kakao open-chatting room” is a group chat composed of the same department students or club members. Compared to “Everytime”, it is easier to share needed information. Because the group size of this chatting room is smaller than “Everytime”. But, similar to “Everytime”, it is usually anonymous and students communicate with short question and answer rather than long-term conversation, so it is difficult to form a friendship with each other in the chatting room. Finally, “SKKUTER” is a community that accessible to both current students and graduates. It provides information sharing with graduates and current students but it has low accessibility due to web page access required. Also, it is hard to promote friendship with friends with the same purpose.

### **2.1.2. Overall structure**



[Figure ] System Structure

## **2.2. Product functions**

### **2.2.1. Create user account and log-in**

The first thing for user to do is downloading the application. After the user downloads this application, the user need account for enter and use the application functions. The first page of our application is login page which include the register button. User can enter the register page with this register button and fill the required information field to create their own account. Then user can log-in and enter to the system with their account.

### **2.2.2. View the Map**

After the logging in, user can view the map. The map only shows the range close to the user’s current location. Users can choose the interesting places in the map or search for the places using search button. Then user can get the information of places they want to go.

### **2.2.3. Choose the Category**

At the top of the map screen are category buttons where you can choose a purpose. Users can select a category to find other students with the same purpose.

### **2.2.4. Icons**

At the bottom of the map screen are three icons. One is a “friend list”, the other is a “chat list”, and the other is “my page”. User can check the list of friends he or she have added as friends by clicking the “friend list” icon. User can start the chatting clicking the friend’s profile. User can check the list of chatting room that he or she had participated by clicking the “chat list” icon. And user can change his or her profile such as, nickname, department, image, introduction by clicking the “my page” icon.

### **2.2.5. Rating and Review**

After the chatting room closing, users can leave ratings and reviews about the other's attitude or manners.

### **2.2.6. Block other users**

If there is a problem with other users, a user can block them that cannot start the chat again or find on the map.

## **2.3. User Classes and Characteristics**

### **2.3.1. User**

The user in this application is limited to “a specific college” students. For example, it can be the SKKU students. User is assumed that are interested in finding students with the same purpose or making friends. It is also assumed that user can read and understand basic Korean or English. User is assumed to have basic knowledge to use or install applications on their own devices.

### **2.3.2. System Manager**

The system manager in this application is limited to specialist of the system. They need to have sufficient knowledge of the system and can understand overall system circumstance. It is also assumed that they have competence to detect the system error or such kind of problems. And they should be able to deal with that error or problems. It is assumed that the system managers have ability to reflect the new system changes. To satisfy these things, the system administrator must be a computer engineering, network, or system-related major, or a person with equivalent knowledge and qualifications. And, of course, they must have a software ethics consciousness too.

## **2.4. Operating Environment**

### **2.4.1. Hardware**

The system is for Android mobile phone. The device, Android mobile phone need at least 1GB RAM and 1.0GHz single processor.

### **2.4.2. Software**

The system is for Android OS. And its version should be at least Android 6.0 (API 23). But the version Android 10 (API 29) is the best environment.

## **2.5. Design and Implementation Constraints**

The system will be designed and implemented by considering the following checklist. There are details of design and implementation directions.

* Showing the map should not take more than 2 seconds.
* The system needs to consider the users’ convenient and requirements.
* The system should run on mobile devices with least 1GB RAM.
* The system requires at least 600MB for installation and execution.
* Source code will be optimized properly to prevent memory waste.
* User device and server communicate with HTTP protocol.
* Consider both the system cost and its maintenance cost.
* Korean and basic English are required.
* Avoid using additional pays for royalty, using open-source software as much as possible.
* Develop with Windows 10 environment and Android Studio. And the build tools version is 29.0.3
* Develop with minimum Android version 6.0 (API 29)
* Emulate the system using Android version 10 (API 29)

## **2.6. User Documentation**

To assist end users to use the application and its service, some documentations are needed. First, minimum hardware and software requirements. And User manuals are also needed, the contents of user manuals are application installation guide to help users that not familiar with the device, description of how to start the system and how to use different features of the system. The screenshots explaining main features of the system and example inputs and outputs are attached in the user manual too and the explanation video about this document contents is also provided. And last, information to contact the developer of the system are given.

## **2.7. Assumptions and Dependencies**

The systems are designed and implemented based on Android devices. Thus, all function and contents are based on the Android OS. And it can not be applied to different kind of Operating Systems (such as, IOS).

# **3. Specific Requirements**

## **3.1. External Interface Requirements**

### **3.1.1. User Interfaces**

[Table ] Starting Interface

| **Name** | **Starting Interface** |
| --- | --- |
| Purpose/Description | Users choose whether login or register to use this program. |
| Input source/ Output destination | Client/Server |
| Range/  Accuracy/  Margin of error | N/A |
| Unit | Screen |
| Time/ Velocity | N/A |
| Relationship with other input/outputs | N/A |
| Format and configuration of screen | 1. User who has account press ID and Password before click Login button to access app.  2. User without account click register button to make new account  3. System will respond differently based on user’s request. |
| Format and configuration of window | N/A |
| Data type | Button, Text, Image |
| Instruction type | Instruction mapping according to the value of a button code |
| Exit message | N/A |

[Table ] Main Interface

| **Name** | **Basic Interaction at Main Interface Using Touchscreen** |
| --- | --- |
| Purpose/Description | Users transmit their instructions through a touchscreen of the device |
| Input source/ Output destination | User/ user device equipped with Android OS |
| Range/  Accuracy/  Margin of error | Range according to the number of buttons on the screen/  Accuracy according to the accuracy of touch from users/  Margin of error of touch sensitiveness |
| Unit | A click |
| Time/ Velocity | Asynchronous user input/ Instant execution of a user instruction |
| Relationship with other input/outputs | After receiving all the inputs, the user device transmits the input data to the server for processing the input data and request desired output data |
| Format and configuration of screen | 1. An activity screen connected to an XML file, mainly composed of TextViews and ImageVIews    2.A TextView and an ImageView provide a basis for choosing the following input, and several Buttons are ready to receive the inputs from users  3. Users are to click a desired button to interact with the system |
| Format and configuration of window | N/A |
| Data type | Screen |
| Instruction type | Instruction mapping according to the value of a button code |
| Exit message | N/A |

[Table ] Register Interface

| **Name** | **Customization Interface 1 - Register** |
| --- | --- |
| Purpose/Description | To use the service of the system, users have to register to the system, followed by Log-in process. During registration, they are asked to fill out the registration form. |
| Input source/ Output destination | Client/ Server |
| Range/  Accuracy/  Margin of error | N/A |
| Unit | Screen |
| Time/ Velocity | N/A |
| Relationship with other input/outputs | N/A |
| Format and configuration of screen | 1.Several empty slots for filling out necessary information of users in a vertical direction  2. At the bottom of the screen, there is a ‘Register’ button which returns all the information given by a user to the server for saving it  3.After successful registration process, the users are directed to starting interface. |
| Format and configuration of window | N/A |
| Data type | Query |
| Instruction type | Instruction mapped to the button |
| Exit message | “Register succeed!” |

[Table ] User profile Interface

| **Name** | **Customization Interface 2 - Profile** |
| --- | --- |
| Purpose/Description | After log-in, users can enter the ‘Profile’ page by clicking a button ‘profile’ in order to edit their information.  Users can also enter the other users’ profile in order to see information about them. |
| Input source/ Output destination | User/ Servers |
| Range/  Accuracy/  Margin of error | N/A |
| Unit | Page |
| Time/ Velocity | N/A |
| Relationship with other input/outputs | N/A |
| Format and configuration of screen | 1. User can edit their profile information and picture if they enter their own profile page.  2. User can change information by click Edit button if they enter their own profile page.    3. User can exit profile page by click x button located at left above without change anything.  4. System will toast message if user who change his information click x button instead of edit button.  5. User can report other users to host by click siren button.  6. User can block other users by click ‘Block’ button. Blocked users cannot see user profile and chat with user. |
| Format and configuration of window | N/A |
| Data type | Query |
| Instruction type | Instruction mapped according to the value of a button code |
| Exit message | “Profile Was Updated!”/  “Are you sure to exit without change information?”/  “Your report is successfully received” |

[Table ] Friend lists Interface

| **Name** | **Customization 3 – Friends List** |
| --- | --- |
| Purpose/Description | User can enter their friends list page by click friends’ button from main page. |
| Input source/ Output destination | Server / Client |
| Range/  Accuracy/  Margin of error | N/A |
| Unit | Screen |
| Time/ Velocity | N/A |
| Relationship with other input/outputs | N/A |
| Format and configuration of screen | 1. Linear layout that shows friend lists to user with friends’ profile image, name, and information set by friends themselves.    2. User can enter friends’ profile by click their information.  3. User can see friends’ image by click image.  4.When a user clicks the button ‘Recommendation’, user is directed to the recommendation page. |
| Format and configuration of window | N/A |
| Data type | Text, Image, Widget |
| Instruction type | N/A |
| Exit message | N/A |

[Table ] Chat room list Interface

| **Name** | **Customization Interface 4 – Chat room list** |
| --- | --- |
| Purpose/Description | User can enter to chat room list by click ‘chat’ button from main page. |
| Input source/ Output destination | Client/ Server  Server/ Client |
| Range/  Accuracy/  Margin of error | N/A |
| Unit | Screen |
| Time/ Velocity | N/A |
| Relationship with other input/outputs | N/A |
| Format and configuration of screen | 1. Linear layout that shows chat room lists to user with friends’ profile image, and last chat content with other users.    2. When user click chat room, user enter the chat room.  3. When user click chat room long time, system shows buttons.  4. User can add selected friends by click ‘Add friends’ button.  5. User can delete chat room by click ‘delete’ button.  6. User can blow up chat room by click ‘Blow the room’ button. |
| Format and configuration of window | N/A |
| Data type | Image, Text, Widget |
| Instruction type | N/A |
| Exit message | N/A |

[Table ] Chat room Interface

| **Name** | **Customization Interface 5 – Chat room** |
| --- | --- |
| Purpose/Description | User can enter to chat room by click one of chat room list from chat room list page. |
| Input source/ Output destination | Client/ Server  Server/Client |
| Range/  Accuracy/  Margin of error | N/A |
| Unit | Screen |
| Time/ Velocity | N/A |
| Relationship with other input/outputs | N/A |
| Format and configuration of screen | 1. TextView that show name and information of chat room.  2. User can see other users’ profile by click image of other users.  3. User can send message to users in chat room by type text.  4. User can see chat content.  5. User can exit and go to chat room list by click button by left bottom. |
| Format and configuration of window | N/A |
| Data type | Image, Text, Widget |
| Instruction type | N/A |
| Exit message | N/A |

[Table ] Recommendation Interface

| **Name** | **Recommendation Interface 1 – Recommendation friends lists** |
| --- | --- |
| Purpose/Description | User can enter to recommendation friends lists by click recommendation button from friend lists page. |
| Input source/ Output destination | Server/Client |
| Range/  Accuracy/  Margin of error | N/A |
| Unit | Screen |
| Time/ Velocity | N/A |
| Relationship with other input/outputs | N/A |
| Format and configuration of screen | 1. Linear layout that shows friends list recommended by system algorithm.  2. User can see other users’ profile by click image of other users.  3. User can filter recommended friends by click ‘preference’ button.  4. User can go to friend lists page by click ‘friend list’ button. |
| Format and configuration of window | N/A |
| Data type | Image, Text, Widget |
| Instruction type | N/A |
| Exit message | N/A |

[Table ] Filtering user preference Interface

| **Name** | **Recommendation Interface 2 – Filtering friends** |
| --- | --- |
| Purpose/Description | User are asked numerous questions on the usage of potential friends.  Users answer the questions by choosing a button representing answer regarding their preference on friends.  When users answer all the questions, the data is transmitted to the server from the user device. |
| Input source/ Output destination | Server/Client  Client/Server |
| Range/  Accuracy/  Margin of error | N/A |
| Unit | Screen |
| Time/ Velocity | After all the input data were received to the user device/ Communication time between the server and the user device |
| Relationship with other input/outputs | Each answer by user affects the result of filtering. |
| Format and configuration of screen | 1. Question and lists of enable answers made by TextView.    2. User can answer the questions by select one of options.  3. User go next or previous question by click buttons located at bottom.  4. Server will show new recommendation friend lists reflected by user’s answer after user answers all questions. |
| Format and configuration of window | Activity with XML document |
| Data type | Image, Text, Widget, Query |
| Instruction type | N/A |
| Exit message | “Your recommendation friend lists are renewed!” |

[Table ] Search Interface - Location of targets

| **Name** | **Search Interface 1 – Showing user requested targets on the map** |
| --- | --- |
| Purpose/Description | User can get location and information by searching what he wants and clicking buttons.  Server will show location of targets related with user request. |
| Input source/ Output destination | Server/Client  Client/Server |
| Range/  Accuracy/  Margin of error | Range according to the number of target on the screen/  Accuracy according to the accuracy of target location/  Margin of error of real location of target |
| Unit | Screen |
| Time/ Velocity | Asynchronous user input/ Communication time of a user instruction |
| Relationship with other input/outputs | After receiving user input, server processing the input data and request desired output data. |
| Format and configuration of screen | 1. User can click button above the screen or type text in search widget to get location of targets.    2. Server shows targets location based on location-based service  3. When user click one of targets on the map, server shows name and information of target.  4. When user click double one of the targets, user go to the place information page. |
| Format and configuration of window | Activity with XML document |
| Data type | Image, Text, Widget, Query |
| Instruction type | N/A |
| Exit message | N/A |

[Table ] Information Interface of target place

| **Name** | **Search Interface 2 – Information of target** |
| --- | --- |
| Purpose/Description | User can see reviews and chat room lists related with the target place. |
| Input source/ Output destination | Server/Client  Client/Server |
| Range/  Accuracy/  Margin of error | N/A |
| Unit | Screen |
| Time/ Velocity | N/A |
| Relationship with other input/outputs | N/A |
| Format and configuration of screen | 1. ImageView and TextView represent information of the target place.    2. Linear layout shows review lists of the target place written by users who already used this place.  3. Linear layout shows chat rooms with information related with this place.  4. User can enter the chat room by click one of the chat room lists.  5. User can exit place information page by click ‘x’ button on the right above at the page. |
| Format and configuration of window | N/A |
| Data type | Image, Text, Widget |
| Instruction type | N/A |
| Exit message | N/A |

[Table ] Review Interface

| **Name** | **Review Interface** |
| --- | --- |
| Purpose/Description | User can give score and write review of place they visited already. |
| Input source/ Output destination | Client/Server |
| Range/  Accuracy/  Margin of error | N/A |
| Unit | Screen |
| Time/ Velocity | N/A |
| Relationship with other input/outputs | N/A |
| Format and configuration of screen | 1. ImageView of the place.    2. Star widget shows score of the place.  3. User can write review of the place on the text box.  4. User can give score of the place by click ‘-‘ or ‘+’ button.  5. User can post review by click ‘complete’ button.  6. User can exit without posting review by click ‘x’ button. |
| Format and configuration of window | N/A |
| Data type | Image, Text, Widget |
| Instruction type | N/A |
| Exit message | “Review was successfully posted!”/  “Are you sure to exit this page without posting review?” |

### **3.1.2. Hardware Interfaces**

[Table ] Hardware Interface

| **Name** | **Applicable device for the system** |
| --- | --- |
| Purpose/Description | Enable users to take advantage of the service provided by the system/Android OS Enabled Smart Phone. (At least Android 6.0) |

### **3.1.3. Software Interfaces**

[Table ] Software Interface

| **Name** | **Firebase Real-time Database** |
| --- | --- |
| Purpose/Description | Query input/output for managing multimedia/meta data |
| Input source/ Output destination | Host server/ User, User/Host server, User/User |
| Range/  Accuracy/  Margin of error | Depends on the performance of the Firebase |
| Unit | Query |
| Time/ Velocity | Instant reaction |
| Relationship with other input/outputs | Related to all inputs/outputs from server |
| Format and configuration of screen | N/A |
| Format and configuration of window | N/A |
| Data type | Query |
| Instruction type | Query statement |
| Exit message | N/A |

### **3.1.4. Communication Interfaces**

[Table ] Communication Interface

| **Name** | **Client and Host** |
| --- | --- |
| Purpose/Description | Each client requests the connection to the host, requesting list of results of friends based on preference, map based on location-based service, chat room list related with the user-selected place, and chat content with other users.  Host provides a friend list, map, and chat room list, and chat content to user. |
| Input source/  Output destination | User/Host server |
| Unit | packet |
| Time/ Velocity | At least 10Mbps |
| Relationship with other input/outputs | Related to all inputs/outputs from server |
| Format and configuration of screen | N/A |
| Format and configuration of window | N/A |
| Data type | Query |
| Instruction type | Query statement |
| Exit message | N/A |

**3.2. Functional Requirements**

**3.2.1. Use Case**

**<Account Management>**

[Table ] Use case of register

|  |  |
| --- | --- |
| **Use case name** | **Register** |
| Actor | Unregistered user |
| Description | When an unregistered user tries to register in our app, this process will happen. |
| Normal Course | 1. Every user except admin encounters a log-in page after executing the application. 2. If the user isn’t registered yet, the user will touch a register button under a log-in button. 3. The user is redirected to a register page. 4. In the register page, the user should provide several information.    1. ID(This should be unique)    2. Email address    3. Nickname    4. Password    5. Age    6. Gender    7. Major    8. Entrance year 5. The system sends a verification code to the given email address to verify whether the email address is correct and to prepare the situation of finding the password. 6. Then the user set his/her search filter for other users. For example, if the user wants to meet a man, the user can uncheck a woman option in the filter setting page. In the filter,    1. Entrance year    2. Gender    3. Major 7. After setting the user filter, the user should provide his/her student ID card image to authenticate himself/herself. 8. If the system administrator confirms the user, the system sends an email notifying that the user has been approved. This takes several days. |
| Pre-condition | The user is not registered to the system yet.  The user enters correct information.  The same student id card should not be overlapped with that of other users.  In case of incorrect inputs or overlapped student id cards, the system sends a rejection email to the user. |
| Post-condition | The following information should be encrypted and saved to system DB to manage users.   1. User ID 2. Password 3. Student ID |
| Assumptions | N/A |

[Table ] Use case of log-in/out

|  |  |
| --- | --- |
| **Use case name** | **Log-in/out** |
| Actor | Registered user |
| Description | Log-in is the process that a registered user tries to log-in to the system.  Log-out is the process that the user who has logged-in tries to get out of the system. |
| Normal Course | <Log-in>   1. A registered user wants to use the services of the system. 2. The user enters id and password that were the user entered in registration. 3. If the information is correct, the user can log-in successfully and use the system.   <Log-out>   1. If the user wants to get out of the system, the user can touch a log-out button. This button is available in the user profile page. 2. Then if next time the user opens the application, the user should log-in again. 3. If the user closed the application without logging-out, the system arbitrarily closed the session. |
| Pre-condition | <Log-in>  The user is already registered into the system.  <Log-out>  The user is logged in the system. |
| Post-condition | The user should be connected to the network. |
| Assumptions | N/A |

[Table ] Use case of find ID/password

|  |  |
| --- | --- |
| **Use case name** | **Find ID/password** |
| Actor | Registered user |
| Description | Find ID is the process that a registered user tries to find id for log-in.  Find password is the process that a registered user tries to find a password for log-in. |
| Normal Course | <Find ID>   1. When a user tries to log-in, the user enters the wrong id. And the user gets to know that the id is wrong. 2. Touch the ‘find ID/password’ button in the log-in page and the user redirected to the ‘find ID/password’ page. 3. When the user enters the correct email address, the system sends the user's ID to that email. 4. If the user forgot the password too, go to the step of finding the password.   <Find password>   1. When a user tries to log-in, the user enters the correct ID but wrong password. And the user gets to know that the password is wrong. 2. Users know their ID and don't know their password. So, touch the ‘find ID/password’ button to find the password and the user redirected to the ‘find ID/password’ page. 3. When the user enters the correct id and email address, the system sends a link to set a new password. 4. If the user touches the link, the user can set a new password for log-in. |
| Pre-condition | <Find ID>  The user forgets his/her ID for log-in.  <Find password>  The user knows his/her ID for log-in but forgets his/her password for log-in. |
| Post-condition | The user gets his/her ID/password information. |
| Assumptions | The user forgets his/her ID or password. |

[Table ] Use case of profile

|  |  |
| --- | --- |
| **Use case name** | **Profile** |
| Actor | Registered user |
| Description | This is the process by which users edit their profile picture and introduction. Through the profile page, users can advertise themselves. |
| Normal Course | 1. The user clicks a profile button in the bottom of the main page for redirection to the profile page. 2. The user can show his/her information and modify the information. 3. After modifying the information, the user touches the ‘save’ button to save modified information. |
| Pre-condition | The user is registered, and logged-in the system. |
| Post-condition | When the user touches the ‘save’ button, the modified information is delivered to the server database and updated. |
| Assumptions | N/A |

[Table ] Use case of user filter

|  |  |
| --- | --- |
| **Use case name** | **User filter** |
| Actor | Registered user |
| Description | In this process, the user can filter other users by their entrance year, gender, major. |
| Normal Course | 1. The user clicks a profile button in the bottom of the main page for redirection to the profile page. 2. The user can show the existing filter setting and modify the filter. Default filter is set at registration. 3. After modifying the information, the user touches the ‘save’ button to save the modified filter. |
| Pre-condition | The user is registered, and logged-in the system. |
| Post-condition | When the user touches the ‘save’ button, the modified filter information is delivered to the server database and updated in real time. |
| Assumptions | N/A |

**<Friends Management>**

[Table ] Use case of friend request

|  |  |
| --- | --- |
| **Use case name** | **Friend request** |
| Actor | Registered user |
| Description | The user can send friend requests to other users, and other users also can do so. Then the user accepts or refuses a friend request in his/her list of friend requests received. |
| Normal Course | 1. The user can watch other user’s profiles by touching their profile picture. The user can watch other users in,    1. Chatting room    2. Review page    3. Place chatting list(Before join in, only can watch chat owner) 2. In other user’s profiles, the user can send friend requests to that user. 3. The user who got a friend request can accept or refuse a friend request in his/her list of friend requests received. 4. If a user views the profile of another user who has sent him/her a friend request, a friend request button is diabled. 5. After accepting a friend request, an accepted user is visible in the friend list. 6. After refusing a friend request, a refused user is removed from the list of friend requests received. But refused users can send friend requests again. |
| Pre-condition | The user is registered, and not a suspended user in the system.  <Accept/refuse a friend request>  Someone should send a friend request to the user.  <Send a friend request>  The user must not have already sent the same friend request before.  The user must not be blocked by that user. |
| Post-condition | <Accept/refuse a friend request>  The user is visible/invisible in the friend list. |
| Assumptions | The user wants to be friends with others. |

[Table ] Use case of delete friend

|  |  |
| --- | --- |
| **Use case name** | **Delete friend** |
| Actor | Registered user |
| Description | When the user wants to delete a friend in the friend list, the user can delete the other user. |
| Normal Course | 1. The user can watch the friend user in their friends list. 2. Then the user touches and holds the friend user to delete him/her. 3. The system asks if the user really wants to remove the friend, and the user can choose yes or no. 4. After deleting, the deleted user is invisible in the friends list. |
| Pre-condition | To delete from the friend list, they should be friends already. |
| Post-condition | The user’s id is removed from the friend database. |
| Assumptions | The user wants to delete a friend user. |

**<User/Place Review>**

[Table ] Use case of block the user

|  |  |
| --- | --- |
| **Use case name** | **Block the user** |
| Actor | Registered user |
| Description | This is the process that users can block users who don’t want to see or to be recommended. |
| Normal Course | 1. If the user does not want to see a particular user, the user clicks the user’s profile who wants to block. 2. The user clicks the block button which is in the right bottom of the user’s profile page. 3. The system once again confirms to the user if the user really wants to block that user. 4. When the user confirms to block, the user and blocked user can’t see each other anymore. |
| Pre-condition | The user is registered, and logged-in the system. |
| Post-condition | When the user clicks the confirm button, modified friend status information is delivered to the server database and updated.  Blocked users will no longer appear in the friend list or be recommended. |
| Assumptions | N/A |

[Table ] Use case of report the user

|  |  |
| --- | --- |
| **Use case name** | **Report the user** |
| Actor | Registered user |
| Description | This is the process that users can report the user who did unpleasant behavior, such as abusive language, or if there is something to report. |
| Normal Course | 1. If the user has something to report about the particular user, enter that user’s profile. 2. The user clicks the report button which is in the right bottom of the profile page. 3. The report page is displayed where the user can write the reasons for reporting. 4. After the user has finished writing, he/she clicks the report button to send a report message to the administrator. 5. The system once again confirms to the user if the user really wants to report that user. 6. When the user confirms to report, the account of the reported user will be suspended after the system administrator checks the reported message. |
| Pre-condition | The user is registered, and logged-in the system. |
| Post-condition | The report message is sent to the system administrator. |
| Assumptions | N/A |

**<Place Search>**

[Table ] Use case of place review

|  |  |
| --- | --- |
| **Use case name** | **Place review** |
| Actor | Registered user |
| Description | This is the process by which the user can write a review and give a rating of the places where the chat room he/she has joined has been created. |
| Normal Course | 1. The user checks the closed chat room in the user’s chat list page. 2. The user clicks the review button to write a review and give a rating to the place. 3. When the user clicks the review button, the page where the user can write a review and rating is displayed. 4. The user writes a review and rating and clicks the submit button. 5. The system confirms once again to the user if the user really wants to submit the review. 6. When the user confirms to submit, review and rating are posted. |
| Pre-condition | The user is registered, and logged-in the system.  The chat room of the place is closed. |
| Post-condition | When the user clicks the confirm button, review and rating is delivered to the server database and updated. Other users can see the reviews and ratings by searching the place. |
| Assumptions | N/A |

[Table ] Use case of search location

|  |  |
| --- | --- |
| **Use case name** | **Search location (Search filter)** |
| Actor | Registered user |
| Description | This is the process that the user searches for the desired location with desired purpose by using the map search function. |
| Normal Course | 1. The user searches for the place name in the search bar at the top of the main page. 2. When the searched place is displayed on a map, the user can see marks of chat rooms created at that location. 3. The users can set filters and view chat rooms for the desired purposes by clicking the filter buttons under the search bar. 4. Study 5. Eat 6. Rest 7. The user can select the chat room that fits his/her desired purpose. 8. The user can check the location in the map through the searching. |
| Pre-condition | The user is registered, and logged-in the system.  The user should be connected to the network. |
| Post-condition | The user can enter the chat room. |
| Assumptions | N/A |

[Table ] Use case of check information of the place

|  |  |
| --- | --- |
| **Use case name** | **Check information of the place** |
| Actor | Registered user |
| Description | When the user searches for the location on a map, the user can see the representative picture of the place, the total rating, and reviews left by other users and the opened chat room list. |
| Normal Course | 1. The user searches for the place in the search bar on the main page. 2. When the searched place is displayed on the map, the user clicks the mark of the place. 3. The information page of the place is displayed, and the user can view the information of the place. 4. Name 5. Representative image 6. Total rating 7. Reviews 8. Opened chat rooms in that place |
| Pre-condition | The user is registered, and logged-in the system. The user should search the place in the search bar. The user should be connected to the network. |
| Post-condition | The user can get the information of the place. |
| Assumptions | N/A |

**<Chat>**

[Table ] Use case of open/close a chat room

|  |  |
| --- | --- |
| **Use case name** | **Open/Close a chat room** |
| Actor | Registered user (Chat room owner) |
| Description | This is the process that the user opens a chat room in the place he/she wants. Then the user becomes a chat room owner, who can finish the chat room. |
| Normal Course | <Open a chat room>   1. The user searches the place where he/she wants to open the chat room. 2. The user clicks the create button to create a new chat room. 3. The user set the conditions of the chat room. 4. Name 5. Purpose 6. Number of people 7. Short description of the chat room   <Close the chat room>   1. The chat room owner can close the chat room in the chat room page by clicking the close button. |
| Pre-condition | The user is registered, and logged-in the system.  <Open a chat room>  The user set the place to open a chat room.  <Close the chat room>  To close the chat room, the user should have the authority of the chat room owner. |
| Post-condition | <Open a chat room>  The chat room has been created, and other users can see the chat room by searching.  <Close the chat room>  Chat room is closed, and users can check that the chat room has been closed. Then users can write reviews and ratings about the place, friend request, and delete the chat room. |
| Assumptions | N/A |

[Table ] Use case of enter/leave the chat room

|  |  |
| --- | --- |
| **Use case name** | **Enter/Leave the chat room** |
| Actor | Registered user |
| Description | This is the process by which the user selects a desired chat room from the list of opened chat rooms in the place, enters and leaves the chat room. |
| Normal Course | <Enter the chat room>   1. The user clicks the place mark and sees the list of opened chat rooms in the place. 2. The user selects the chat room to join where the number of people is not exceeded. 3. The user clicks the enter button to enter the chat room. 4. The user can send messages in the chat room.   <Leave the chat room>   1. The user can leave the chat room in the chat room page by clicking the leave button. |
| Pre-condition | The user is registered, and logged-in the system.  The user searched the place where he/she wants. |
| Post-condition | <Enter the chat room>  The user can chat in the entered chat room. The chat room has been added to the user’s chat list.  <Leave the chat room>  The chat room has been deleted from the user’s chat list. |
| Assumptions | N/A |

[Table ] Use case of view a chat list

|  |  |
| --- | --- |
| **Use case name** | **View a chat list** |
| Actor | Registered user |
| Description | This is the process by which the user can see a list of the chat rooms he/she is participating in. |
| Normal Course | 1. The user clicks the chat room button in the bottom of the main page. 2. The user can check the list of chat rooms he/she is participating in. 3. The user can check the new message notifications. 4. The user can check the closed chat rooms and review the place, send friend requests or delete the closed chat room. |
| Pre-condition | The user is registered, and logged-in the system. |
| Post-condition | The user can enter the chat room page. |
| Assumptions | N/A |

**<System Management>**

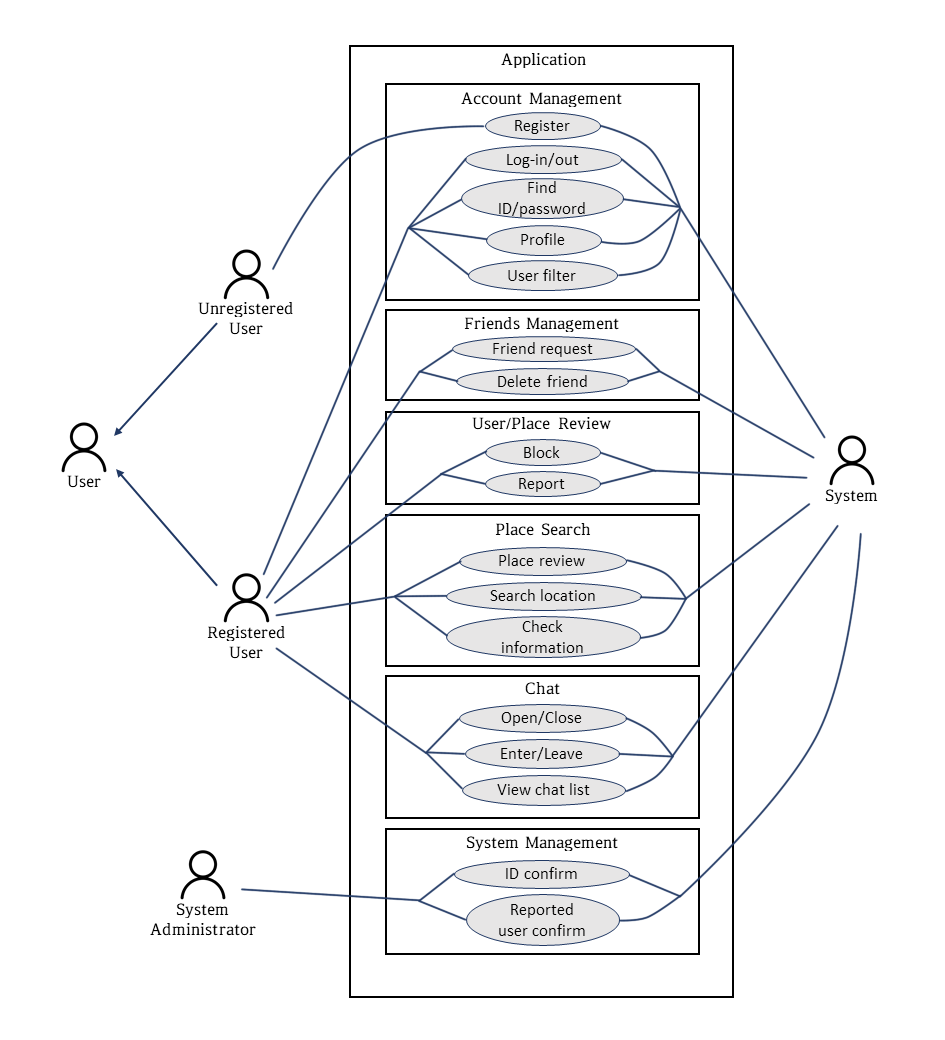
[Table ] Use case of student ID confirmation

|  |  |
| --- | --- |
| **Use case name** | **Student ID confirmation** |
| Actor | System administrator |
| Description | When a new user tries to sign up to the system, the user should submit his/her student id card. Then the system administrator must check it. |
| Normal Course | 1. A new user submits his/her student id card to the system. 2. The system administrator compares the id card and user’s information(name, age, etc). Also check if there is a user with the same information in the past. 3. Then if there is no problem, the administrator approves the application for admission. 4. The system saves the user id and user’s student id for overlapping checks. |
| Pre-condition | The new user completes the sign-up process and submit the correct and non overlapped student id card. |
| Post-condition | The user is approved and the user can log-in to the system. |
| Assumptions | The user submits the correct student id card. |

[Table ] Use case of reported user confirmation

|  |  |
| --- | --- |
| **Use case name** | **Reported user confirmation** |
| Actor | System administrator |
| Description | When one user reports the other user, reported user’s chatting logs are sent to the system administrator to confirm the bad user. |
| Normal Course | 1. When a user reports the other user for malicious reason(Use of unhealthy language, Suspicious of hacking, Impersonating a student), related information is sent to the system administrator. 2. The system administrator checks chatting logs, or other information of the reported user. 3. If the system administrator confirms the reported user as a malicious user, that user will be suspended from the system. |
| Pre-condition | The user reports the other user. |
| Post-condition | If the system administrator confirmed, the reported user will be suspended. |
| Assumptions | There is a malicious user. |

**3.2.2. Use Case Diagram**



[Figure ] Use case diagram

**3.2.3. Data Dictionary**

[Table ] User

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Key | Constraint | Description |
| id | PK | Not Null | User id |
| email |  | Not Null | User email |
| password |  | Not Null | User password |
| nickname |  | Not Null | User’s nickname |
| gender |  | Not Null | User’s gender |
| major |  | Not Null | User’s major |
| entrance\_year |  | Not Null | User’s entrance year |
| introduction |  |  | User’s introduction |
| suspended |  |  | Whether user suspended in the system |

[Table ] User’s filter

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Key | Constraint | Description |
| user\_id | PK, FK | Not Null | Id of user who has this table |
| filter\_entrance\_year |  |  | The user wants to watch others whose entrance year is in this range |
| filter\_gender |  |  | The user wants to watch others whose gender is this |
| filter\_major |  |  | The user wants to watch others whose major is this |

[Table ] User’s friend

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Key | Constraint | Description |
| user\_id | PK, FK | Not Null | The user’s id |
| friend\_id | FK | Not Null | The user’s friend id |
| status |  | Not Null | 0: Received, 1: Send, 2: Friends, 3: Blocked |

[Table ] Chat room

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Key | Constraint | Description |
| room\_id | PK | Not Null |  |
| user\_id | FK | Not Null | User who made a room |
| member\_id |  | Not Null | User who joined a room |
| name |  | Not Null | Room’s name |
| date |  | Not Null | Created date |
| introduction |  |  | Room’s introduction, Word limit |
| purpose |  | Not Null | 0: Study, 1: Eat, 2: Rest |

[Table ] Chat

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Key | Constraint | Description |
| user\_id | PK, FK | Not Null |  |
| date |  | Not Null | Send date |
| content |  | Not Null | Chatting content |
| reported |  |  | Whether the chat is reported |
| reported\_id | FK |  | Id of who reported the chat |

[Table ] Place Rating

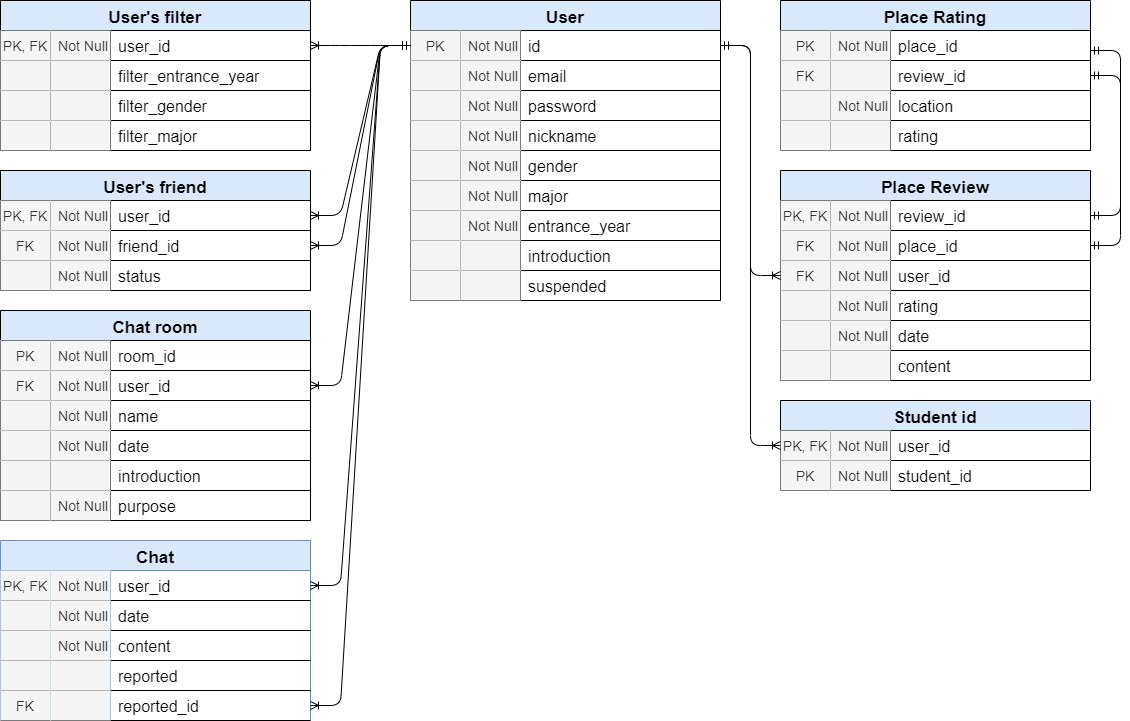
|  |  |  |  |
| --- | --- | --- | --- |
| Field | Key | Constraint | Description |
| place\_id | PK | Not Null | Place’s id |
| review\_id | FK |  | Review’s ids in place |
| location |  | Not Null | Place’s location. |
| rating |  |  | Place’s rating |

[Table ] Place review

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Key | Constraint | Description |
| review\_id | PK, FK | Not Null | Review’s id |
| place\_id | FK | Not Null | Id of the review place |
| user\_id | FK | Not Null | User’s id |
| rating |  | Not Null | Rating |
| date |  | Not Null | Created date |
| content |  |  | Review’s content |

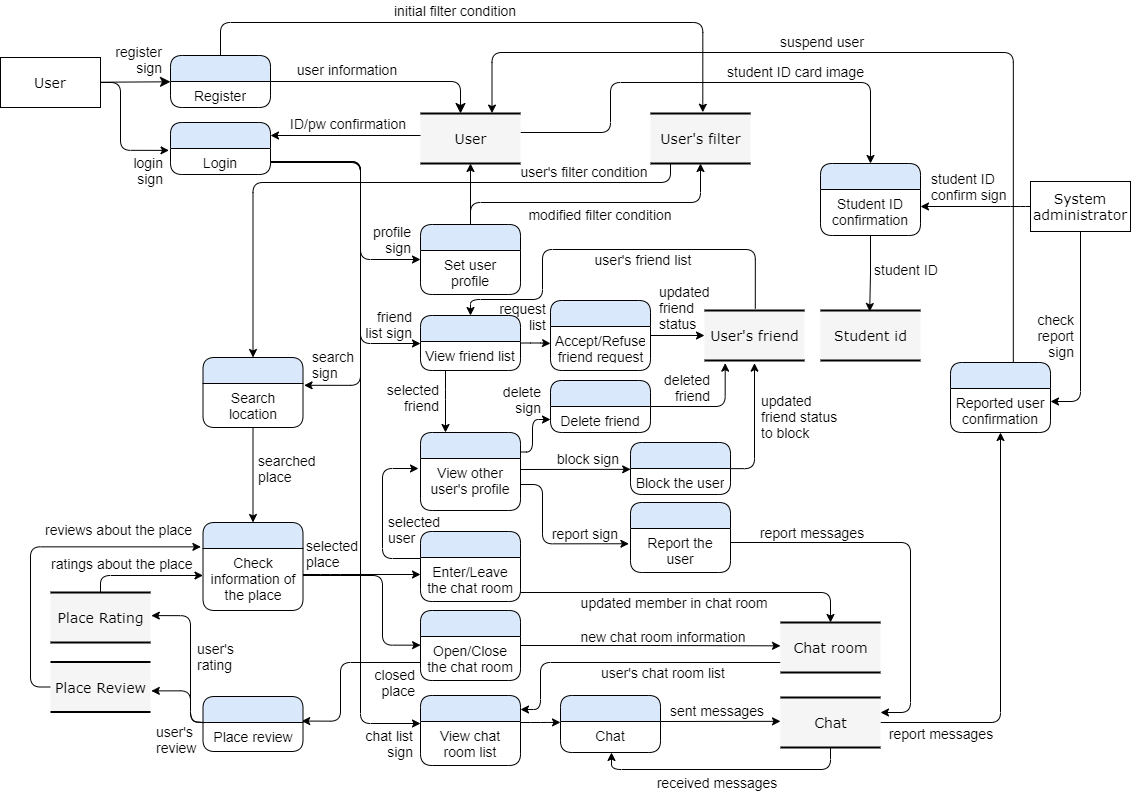
[Table ] Student id

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Key | Constraint | Description |
| user\_id | PK, FK | Not Null | User’s id |
| student\_id | PK | Not Null | User’s student id(encrypted) |



[Figure ] Entity Relationship Diagram

**3.2.4. Data Flow Diagram**



[Figure ] Data flow diagram

## **3.3. Performance Requirements**

The following requirements are based on estimates and are subject to change as the application is complete.

### **3.3.1. Static numerical requirement**

- The system only supports one concurrent user and after disconnecting, users can switch accounts and access.

- The system is recommended to run on a mobile device with 4 GB of RAM and a 2.0 GHz single processor. And the system supports the latest version of Android 8.0 and above, IOS 13 and above.

### **3.3.2. Dynamic numerical requirement**

- The system works well in environments with at least 300 concurrent users. And the system is built to manage at least 10000 members.

- The mapping within the application must be performed within 5 seconds.

- The basic function and connection of the application must be performed within 3 seconds.

## **3.4. Logical Database Requirements**

The system manages user information through a database called Firebase. The system stores user information and location information in a database. The database is managed so that it can have basic performance for processing information through the database.

## **3.5. Design Constraints**

The system must not contain components that are not covered under the license. The system must be accessible from various mobile devices using the Android and IOS operating system. Administrators should be able to access and manage the system through a web browser and administrator application. The system should be designed to use Firebase’s database.

## **3.6. Standards compliance**

All programs in the system are written according to JAVA standards. Function and variable names in the program use camel notation and underscore notation apply to the database. System management tools follow the HTML5 standard.

## **3.7. Software System Characteristics**

This section describes non-functional requirements that are categorized as product requirements, organizational requirements, and external requirements. Software system characteristics are revealed through non-functional requirements.

### **3.7.1. Product Requirements**

The system must meet the following product requirements.

**3.7.1.1. Usability Requirement**

The most important of the non-functional requirements, the system should be designed so that it can be easily used by non-experts. Users should be able to use the functions of the system without having to go through a separate manual. If a function is required to be described, it should be described as easily as possible. To satisfy the above, the system needs to have a simple and easy user interface.

**3.7.1.2. Performance Requirement**

The algorithm for mapping aims to provide results to users within 5 seconds. This is the most time-consuming operation on the system, which can reduce usability.

**3.7.1.3. Security Requirement**

The user must be certified as a student at Sungkyunkwan University before using the application. User can use the system after receiving the certification, and for system management, user must obtain a separate certification for the administrator in order to have the authority for system management.

### **3.7.2. Organizational Requirements**

This section covers a wide range of requirements regarding the policy.

**3.7.2.1. Environmental Requirement**

Location information and location details are taken from ‘Naver Map’ and used it in the system. Information is provided by the application based on the rich API provided by ‘Naver Map’.

**3.7.2.2. Operational Requirement**

Users are identified through ID, and users are also identified through ID, but detailed ID, but detailed information of each other can be provided by agreement between users. User information is managed in a database, and user information can be provided within 3 seconds, and location information can be provided within 5 seconds.

### **3.7.3. External Requirements**

This section covers requirements for external factors.

**3.7.3.1. Safety Requirement**

The system must be designed so that users of external systems cannot access the system. In addition, safety must be provided so that data is not damaged by external shocks.

**3.7.3.2. Regulatory Requirement**

Permission should be given from Naver to get location information through ‘Naver Map’. Users’ personal information must be legally protected and not infringed. The system should be developed in accordance with national privacy standards.

## **3.8. Organizing the Specific Requirements**

This section allows finding out specific system model for requirements. The system model uses a graphical notation based on UML (Unified Modeling Language) and tabular format.

### **3.8.1. Context Model**

<<system>>

Place Search System

<<system>>

User Management System

<<system>>

Friend Management System

Syste

<<system>>

Sign in/up System

Syste

<<system>>

Chat System

Syste

<<system>>

유생찾기

System

[Figure ] Context Model

### **3.8.2. Process Model**

[be Registered]

[not be Registered]

Login

Register

Set Location Information

Place Search

View/Edit Profile

Manage Friend

Manage Chat or Group

Chat or Join the group

[Figure ] Overall Process Model

### **3.8.3. Interaction Model**

See 3.2.2 Use Case Diagram

### **3.8.4. Behavior Model**

**3.8.4.1. Data Flow Diagram**

See 3.2.4 Data Flow Diagram

**3.8.4.2 Sequence Diagram**

It mainly describes our main system, the locating sequence.

FindGroup()

SendPlaceInformation()

JoinGroup()

Login()

Veriyfy()

Verification()

showResult()

setLocation()

saveLocation()

FindPlace()

ShowResult()

If verified

successfuly

server

Map DB

User DB

[Figure ] Sequence Diagram

## **3.9. System Architecture**

This section describes the relationship and functionality between system modules using the MVC pattern.

Refresh Request

Request data

Send data

User Event

Forms to display

Controller

Mapping Engine

Model (Database)

Z

View (User Interface)

Location Page

Chat Page

User

Map DB

Map Page

Profile Page

User DB

[Figure ] System architecture of the system

## **3.10. System Evolution**

This section covers system changes that will benefit system designers. Also, this section describes changes to the system caused by advances in hardware and changes in requirements.

### **3.10.1. Limitation and Assumption**

First of all, since the goal of the application is to find friends and search for places around Sungkyunkwan University, we are servicing location information near Sungkyunkwan University. In the future, if more users use the application or can service a wider range of location information through the development of technology, we will update the application accordingly. In addition, we plan to continuously update the functions of the application if there are functions that users need, including various functions currently in service.

### **3.10.2. Evolutions of Map Application and Change of User Requirements**

With the development of the Map application, the purpose of using our application may be lacking. For example, if a newly developed Map application provides location information of all regions including Sungkyunkwan University and also provides functions of our application, our application will not be used because it is backward compatible with the application.

In preparation for this situation, there will be a way to develop a functionally better application. However, it may be better to provide information about the school as well as location information in connection with the school, taking advantage of the merit that only Sungkyunkwan University students can use it.

In the future, there will be a need to develop an application that meets changing requirements by exploring the direction of application development.

# **4. Supporting Information**

## **4.1. Software Requirement Specification**

This software requirements specification was written in accordance with the IEEE Recommendation (IEEE Recommended Practice for Software Requirements Specifications, IEEE-Std-830).

## **4.2. Document History**

[Table ] Document History

| **Date** | **Version** | **Description** | **Writer** |
| --- | --- | --- | --- |
| 2021/04/08 | 0.1 | Style and overview | Eunju Seok |
| 2021/04/09 | 1.0 | Addition of 3.2.1., 3.2.2. | Hyejoon Jang |
| 2021/04/09 | 1.1 | Addition of 3.1.1, 3.1.2 | Jiwon Seo |
| 2021/04/10 | 1.2 | Addition of 1.1, 1.2, 1.3 | Eunji Gil |
| 2021/04/10 | 1.3 | Addition of 2.1, 2.2, 2.3 | Hyeyeong Kim |
| 2021/04/11 | 1.4 | Addition of 3.3, 3.4 | Georyang Park |
| 2021/04/17 | 1.5 | Addition of 3.2.3., 3.2.4 | Eunju Seok |
| 2021/04/18 | 1.6 | Revision of 3.2.1 | Hyejoon Jang |
| 2021/04/19 | 1.7 | Addition of 3.1.3 | Jiwon Seo |
| 2021/04/19 | 1.8 | Addition of 1.4, 4.1, 4.2 | Eunji Gil |
| 2021/04/20 | 1.9 | Addition of 2.4., 2.5. | Hyeoyeong Kim |
| 2021/04/20 | 1.10 | Addition of 3.5, 3.6 | Georyang Park |
| 2021/04/21 | 1.11 | Addition of 3.1.1, 3.1.2 | Eunju Seok |
| 2021/04/21 | 1.12 | Revision of 3.1.3., 3.1.4. | Hyejoon Jang |
| 2021/04/22 | 1.13 | Addition of 3.1.4 | Jiwon Seo |
| 2021/04/22 | 1.14 | Revision of 1,4 and style | Eunji Gil |
| 2021/04/23 | 1.15 | Addition of 2.6., 2.7. | Hyeoyeong Kim |
| 2021/04/23 | 1.16 | Addition of 3.7 | Georyang Park |
|  |  |  |  |
|  |  |  |  |