Finding Friends for the Same Purpose at Close Distance

Software Design Specification

2021.05.15.

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

LIST OF FIGURES

LIST OF TABLES

1. Preface

This chapter contains the readership information, readership, scope, objective of this document and the document structure of this Software Design Document for “유생찾기”.

1.1. Readership

This Software Design Document is divided into 10 sections with various subsections. Each section contains the overall structure of the system, the structure at the front end, the structure at the back end, protocols, database design and testing plan, and development plan. The detailed structure of the Software Design Document can be found as listed below, in the Document Structure subsection of this SDD. In this document, Team 7 is the main reader. Additionally, professors, TAs, and team members in the Introduction to Software Engineering class can be the main readers.

1.2. Scope

This Design Specification is to be used by Software Engineering and Software Quality Engineering as a definition of the design to be used to implement an application that users can find friends for the same purpose at close distance.

1.3. Objective

The primary purpose of this Software Design Document is to provide a description of the technical design aspects for our mobile phone recommendation application, "유생찾기". This document describes the software architecture and software design decisions for the implementation of "유생찾기". It also provides an architectural overview of the system to depict different aspects of the system. It further specifies the structure and design of some of the modules discussed in the SRS document and in addition, displays some of the use cases that have been transformed into sequential and activity diagrams, including the class

diagrams which show how the programming team would implement the specific module. The intended audience of this document is, but not limited to, the stakeholders, developers, designers, and software testers of the "유생찾기"mobile application.

1.4. Document Structure

- **1. Preface:** this chapter describes readership, scope of this document, object of this system, and structure of this document
- **2. Introduction:** this chapter describes several tools used for this document, several diagrams used in this document and the references, and object of this project.
- **3. Overall System Architecture:** this chapter describes the overall architecture of the system using context diagram, sequence diagram, and use case diagram.
- **4. System Architecture - Frontend:** this chapter describes architecture of the frontend system using class diagram and sequence diagram.
- **5. System Architecture - Backend:** this chapter describes architecture of the backend system using class diagram and sequence diagram.
- **6. Protocol Design:** this chapter describes design of several protocols which are used for communication between client and server.
- **7. Database Design:** this chapter describes database design using several ER diagrams and SQL DDL.
- **8. Testing Plan:** this chapter describes the testing plan for our system.
- **9. Development Plan:** this chapter describes which tools to use to develop the system, constraints, assumption, and dependencies for developing this system.
- **10. Supporting Information:** this chapter describes the baseline of this document and history of this document.

2. Introduction

The purpose of this project is to create a mobile application that is to provide map information around Sungkyunkwan University to Sungkyunkwan University students and is used for Sungkyunkwan University students to hold meetings according to their purpose.

The system provides the user with information about the venue and shows the meeting taking place at that venue. Users can decide whether to join the meeting, and after the meeting is over, evaluate where the meeting was held.
This document is a design document and contains information about the design used to implement the project. For design details, refer to the description of the requirements specification.

2.1 Objectives

This part explains the diagrams and tools applied to the project.

2.2 Applied Diagrams

2.2.1 UML

UML is a standardized modeling language to facilitate communication between developers in the process of system development such as requirements analysis, system design, and system implementation. UML has the advantage of being a language with a strong expressive power for modeling and a logical notation with relatively few contradictions. Therefore, communication between developers is easy, and it is easy to point out modeling structures that are omitted or inconsistent and can be applied to all systems regardless of the scale of the system to be developed. UML provides a wealth of analysis and design devices for developing object-oriented software based on diagrams such as use case diagrams and class diagrams, so it is expected to be used as an industry standard for a considerable period in the future.

2.2.2 Use case Diagram

The use case of UML describes the functional unit provided by the system. The main purpose of the Use Case Diagram is for development teams to visualize the functional requirements of the system, including relationships between different use cases as well as the relationships between the system and interacting actors for the main process. Use Case Diagram is used to describe the advanced functions of the system and the scope of the system.

2.2.3 Sequence Diagram

UML’s sequence diagram shows a detailed flow of a specific use case or even a part of a specific use case. Sequence Diagram shows the calling relationship between different objects in the sequence, and it can also show different calls to different objects in detail. The sequence diagram is expressed in two dimensions, the vertical shows the message/call sequence in the order of occurrence time, and the horizontal shows the object instance to which the message is transmitted. The sequence diagram is very simple, and class instances (objects) are classified by placing an instance of each class in a box at the top of the diagram.

2.2.4 Class Diagram

UML’s class diagram represents how different entities (people, products, data) relate to each other. In other words, it can be said to be the static structure of the system. Class diagrams are mainly used to show implementation classes handled by programmers, and implementation class diagrams show classes like logical class diagrams.

2.2.5 Context Diagram

Context diagram in UML is a diagram that defines the boundary between the system, or part of a system, and its environment, showing the entities that interact with it. Context diagram represents a central system without any details of the internal structure surrounded by all the systems, environments, and activities with which it interacts. The purpose of the context diagram is to focus on external factors and events that must be considered when developing overall system requirements and constraints. Best context diagrams are used to show how systems interoperate at a very high level, or how systems work and interact logically. Context diagrams are the tools you need to develop basic interactions between systems and actors.

2.2.6 Entity Relationship Diagram

When modeling the database structure of a system, the entity relationship diagram is a diagram showing the properties of entities that have unique characteristics that constitute them and the relationships between them in a network-type structure. In the entity relationship diagram, the entity set is represented by a rectangle, the attribute is represented by an ellipse, the entity set and its attributes are connected by a line, the relationship set is represented by a rhombus, and the mapping form of the relationship set is represented by an arrow.

2.3 Applied Tools

2.3.1 ERD Plus

ERD Plus is a database modeling tool for creating Entity Relationship Diagrams, Relational Schemas, Star Schemas, and SQL DDL statements. ERD Plus is a web-based database modeling tool that lets you quickly and easily create Entity Relationship Diagrams (ERDs), Relational Schemas (Relational Diagrams), Star Schemas (Dimensional Models). ERD Plus enables drawing standard ERD components (Entities, Attributes, Relationships).

2.3.2 Microsoft PowerPoint

The PowerPoint gives you access to the familiar slideshow maker tool you already know. Create, edit, view, present, or share presentations quickly and easily. PowerPoint provides a quick view of your recent slides and presentations for easy access on any device. PowerPoint lets you make a lasting impression with powerful and customizable slides and presentations that make you stand out. PowerPoint gives you templates to work from and automatic design ideas for your presentations.

PowerPoint is widely used as presentation software by a variety of users such as businessmen, teachers, and students, and is one of the most appropriate ways to use it in the form of persuasion skills. Microsoft Office PowerPoint, part of Microsoft Office, has become the most widely used presentation program in the world.

2.4 Project Scope

Through this project, users will have an opportunity to easily grasp the location information around Sungkyunkwan University, and to communicate with students who have no other relationship. With continuous updates, we will have a database server that supports location data for hundreds or thousands of places around Sungkyunkwan University. Users will be able to build a new relationship and develop it in a good way through our application.

2.5 References

The user of this SDD may need the following documents for reference:

- Team 1, 2020 Spring, Software Design Document, SKKU.
- Terrain and Geospatial Information System Glossary, 2016. 1. 3., Kangwon Lee, Howoong Son.

3. System Architecture - Overall

3.1. Objectives

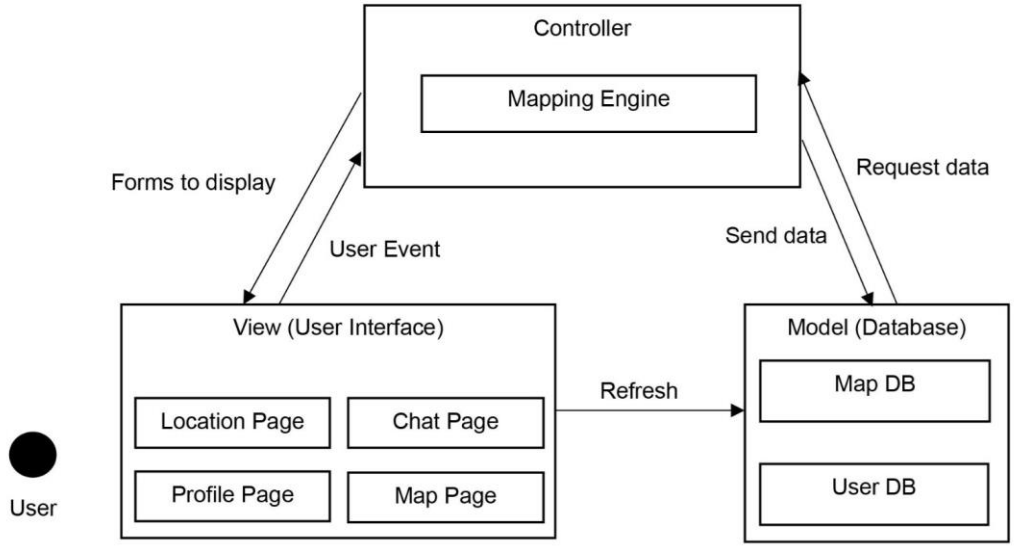
This chapter describes the overall system architecture with organization of the frontend and backend design of the system.

3.2. System Organization

This service is designed by applying the Model-View-Controller pattern. The frontend handles parts of interacting with the user in the view, such as signing in, signing up, searching for a place on the map, managing a list of friends, opening and participating in a chat room suitable for the purpose, chatting, and writing a review. In the backend, the user's interaction information is delivered to the controller, and requested data such as the user information, filtering conditions, user's friend status, rating and review information are delivered from the database, and the required result is delivered to the user with the view. When the frontend receives the location searching information from the user, the backend

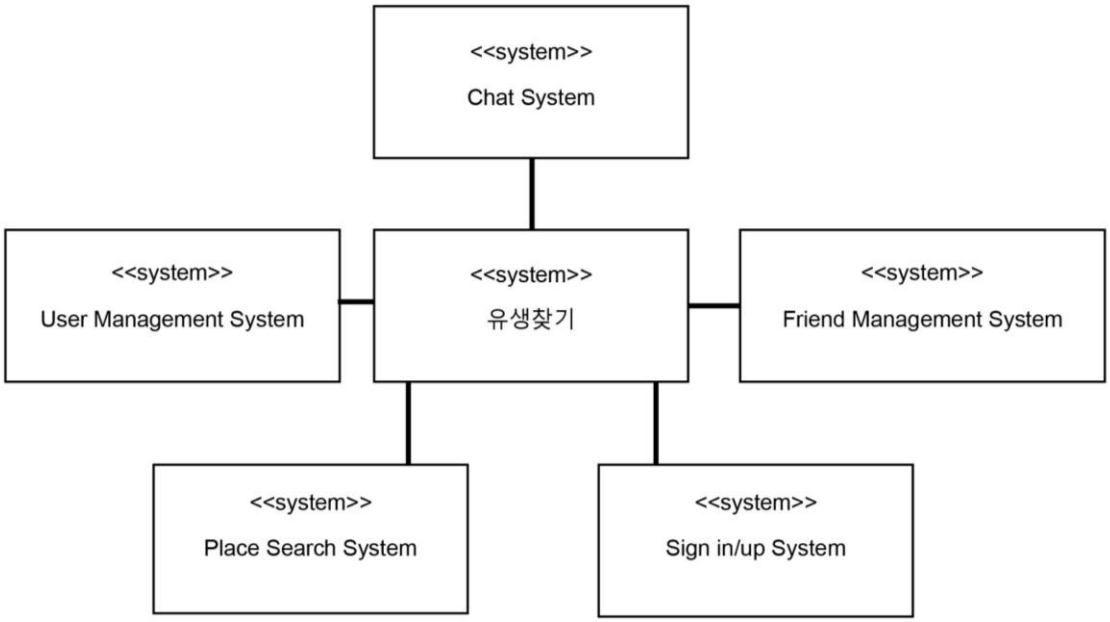
application gets information using the Naver Map API, and the controller receives the chat room list information that meets the filtering conditions of the user near the location from the database through the filtering algorithm. This is passed to the frontend and displayed on the user's screen, allowing the user to join or open a chat room. When the chat room is closed, users can write ratings and reviews for the place, and this information is stored in a database on the backend, which is then used when the other users search for the location. For the communication between frontend and backend, we use HTTP communication with JSON format.

[Figure 1] Overall system architecture



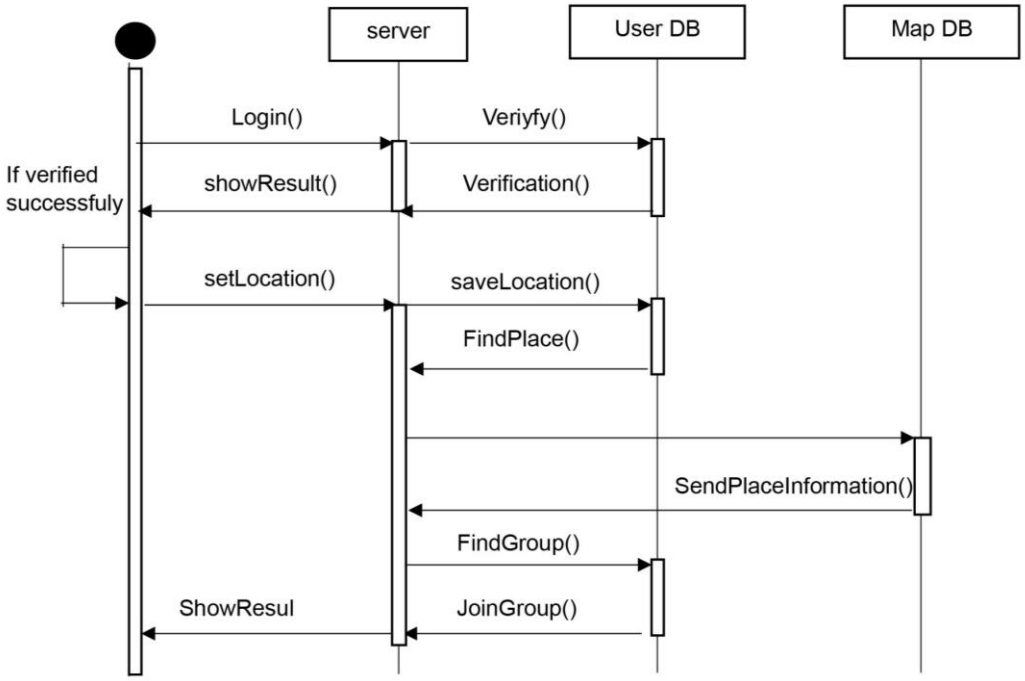
3.2.1 Context Diagram

[Figure 2] Overall context diagram



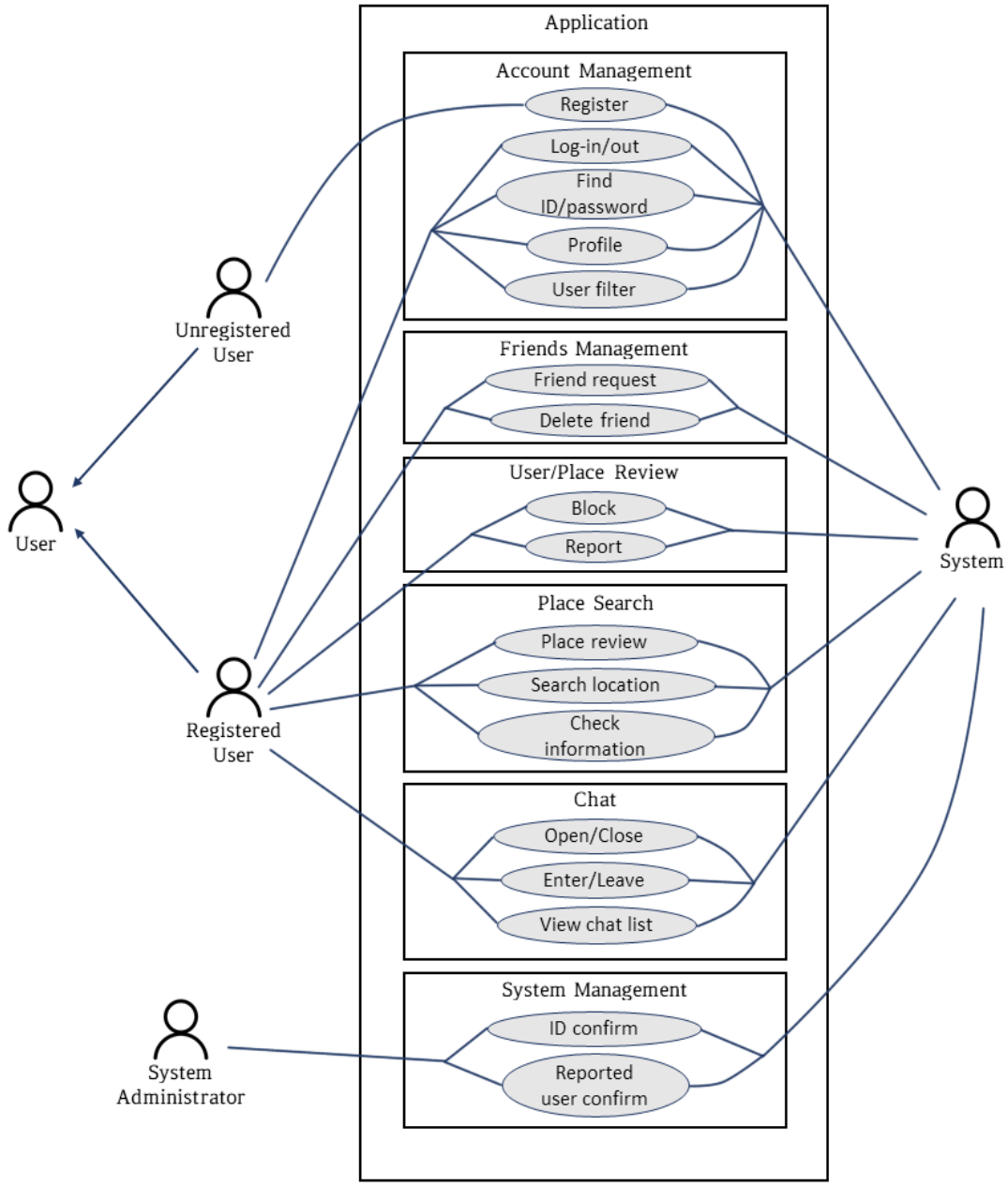
3.2.2 Sequence Diagram

[Figure 3] Overall sequence diagram



3.2.3. Use Case Diagram

[Figure 4] Use case diagram



4. System Architecture - Frontend

4.1. Objectives

This chapter describes the architecture of the frontend system, including the attributes and methods of each subcomponent and the relationship between components.

4.2. Subcomponents

4.2.1. Profile

The profile class is a class that handles user’s profile information. When a user registers, the initial profile information is set, and then the profile page can be accessed, or the information can be modified.

4.2.1.1. Attributes

These are the attributes that the profile class has.

- **user_id** : id of the user. Each user signs up with a unique id.
- **profile** : profile information of the user.

These are the attributes that the profile object has.

- **user_id** : id of the user. Each user signs up with a unique id.
- **email_address** : email address of the user.
- **nickname** : nickname of the user.
- **age** : age of the user.
- **gender** : gender of the user.
- **major** : major of the user.
- **entrance_year** : entrance year of the user.
- **profile_image** : profile image of the user.

- **profile_message** : profile message of the user.

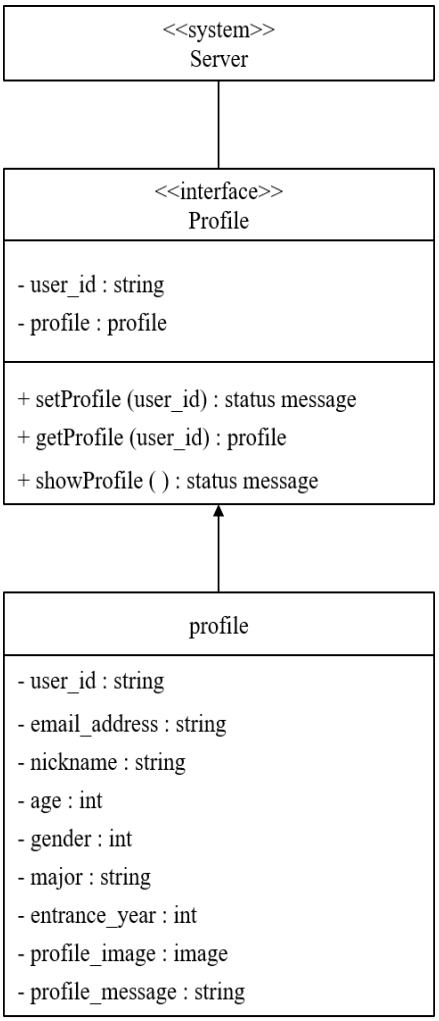
4.2.1.2 Methods

These are the methods that the profile class has.

- setProfile()
- getProfile()
- showProfile()

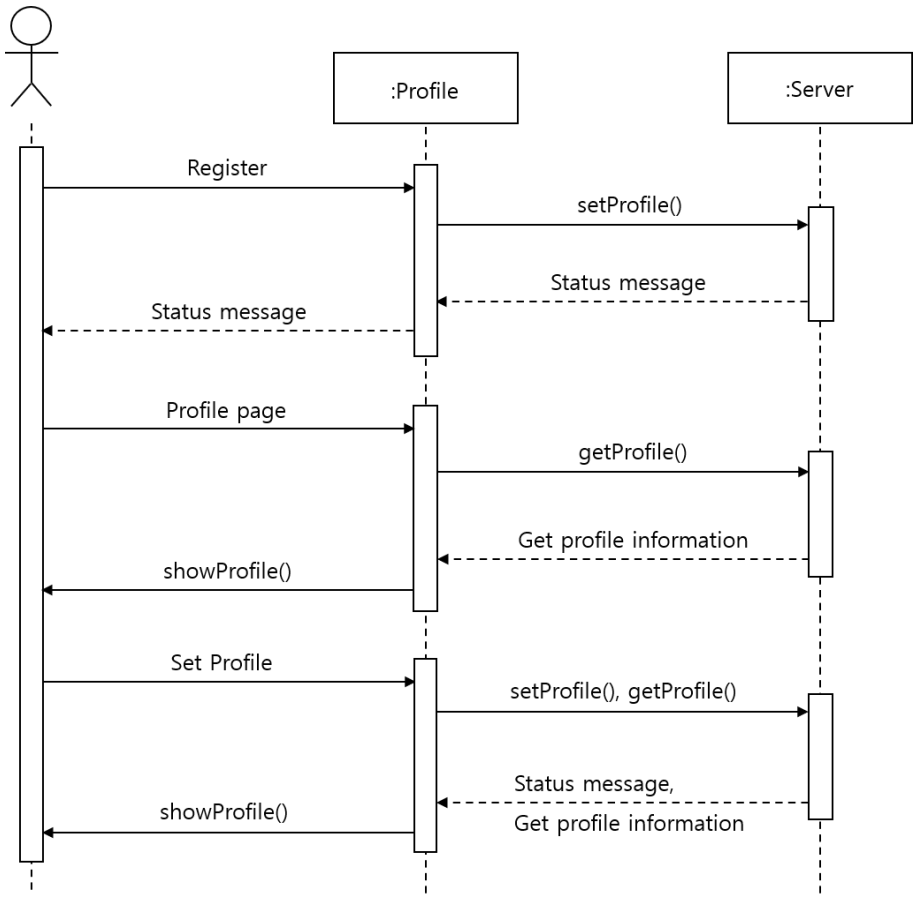
4.2.1.3 Class Diagram

[Figure 5] Class diagram - Profile



4.2.1.4 Sequence Diagram

[Figure 6] Sequence diagram - Profile



4.2.2. Place Search

The place search class receives the name of the place that the user wants to search and the information of the filter tag and requests a search.

4.2.2.1. Attributes

These are the attributes that the place search class has.

- **user_id** : id of the user.
- **place_name** : the name of the location that user wants to search.
- **filter** : filtering condition of the user.

These are the attributes that the search_info object has.

- **place_name** : the name of the location that user wants to search.
- **filter_tag** : filter condition that is set by tag when the user searches the location.

These are the attributes that the user_filter object has.

- **user_id** : id of the user.
- **filter_condition** : filter condition that is set in the user's profile.

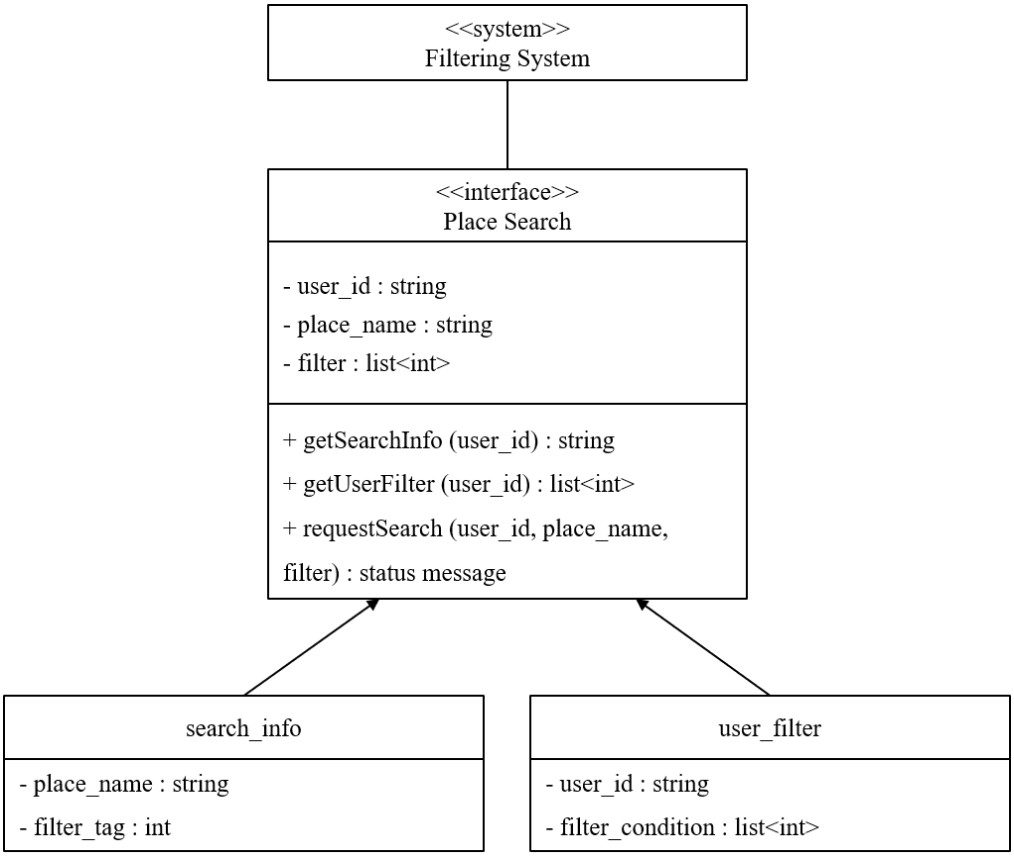
4.2.2.2 Methods

These are the methods that the place search class has.

- getSearchInfo()
- getUserFilter()
- requestSearch()

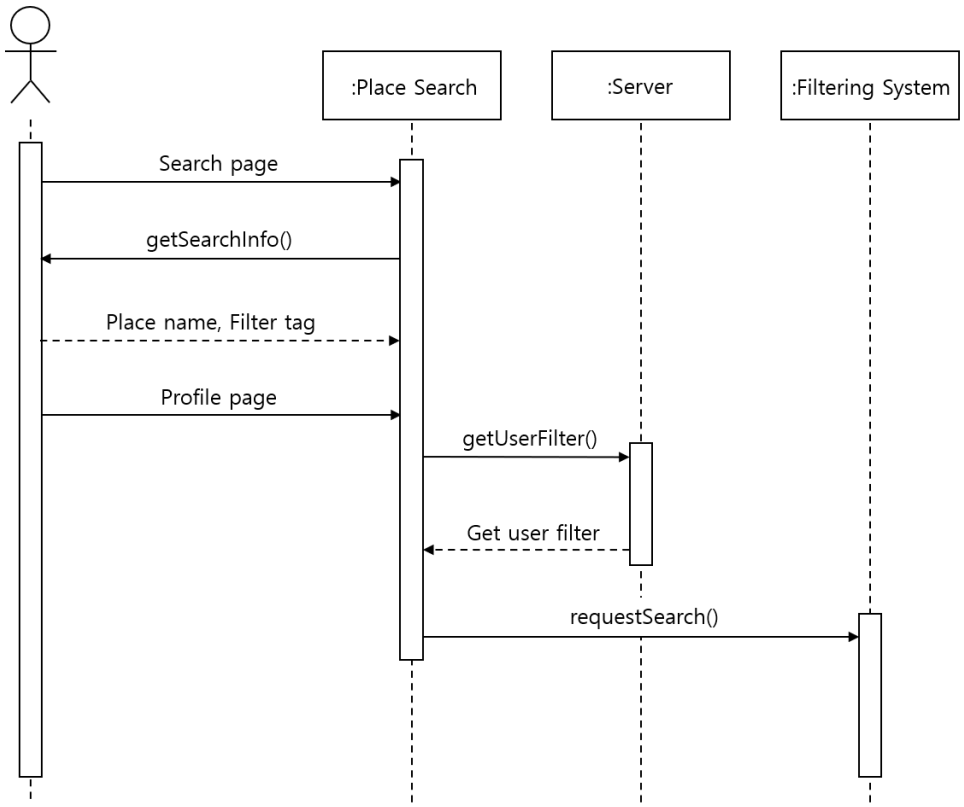
4.2.2.3 Class Diagram

[Figure 7] Class diagram - Place Search



4.2.2.4 Sequence Diagram

[Figure 8] Sequence diagram - Place Search



4.2.3. Search Result

The search result class receives information of the search result and chat room list created at the location and displays it to the user.

4.2.3.1. Attributes

These are the attributes that the search result class has.

- **user_id** : id of the user.
- **search_result** : the result information of the search.
- **chat_room_list** : list of chat rooms of the search location.

These are the attributes that the search_result object has.

- **place_name** : name of the search place.
- **place_info** : the information of the search place.

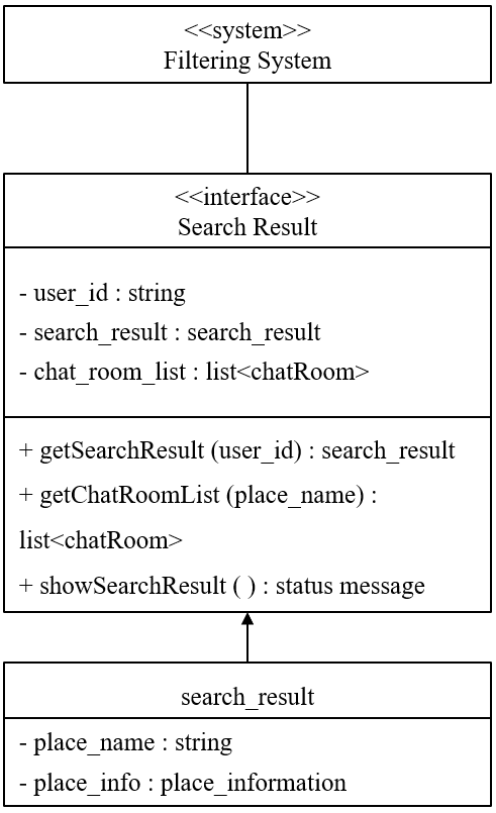
4.2.3.2 Methods

These are the methods that the search result class has.

- getSearchResult()
- getChatRoomList()
- showSearchResult()

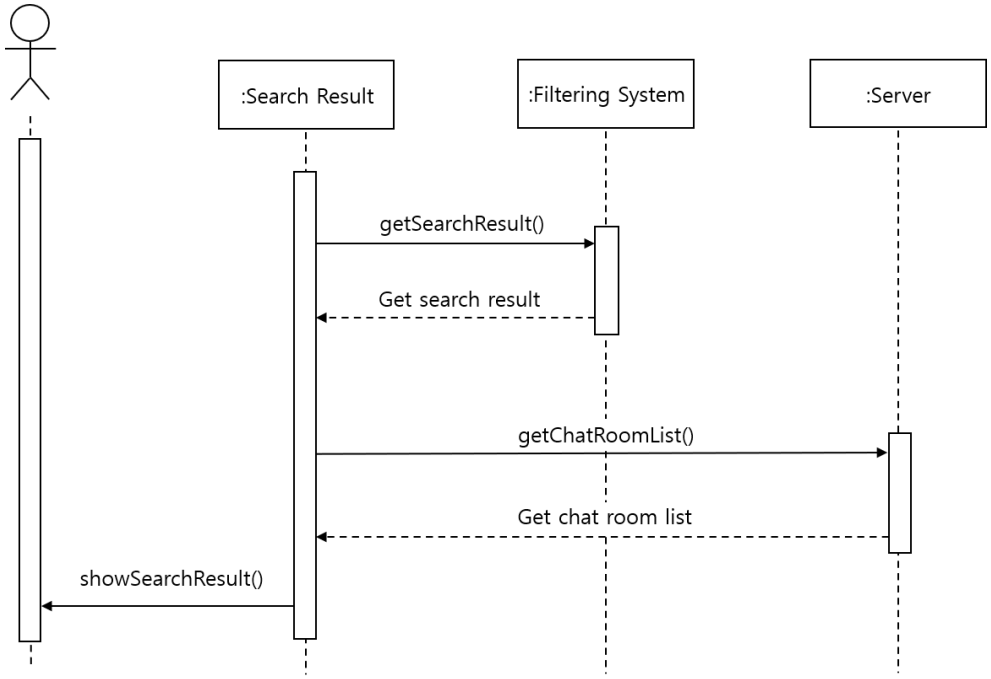
4.2.3.3 Class Diagram

[Figure 9] Class diagram - Search Result



4.2.3.4 Sequence Diagram

[Figure 10] Sequence diagram - Search Result



4.2.4. Friend

4.2.4.1. Attributes

These are the attributes that the friend object has.

- **user_id** : id of the user.
- **freind_name** : name of the friend.
- **number** : number for the friend.

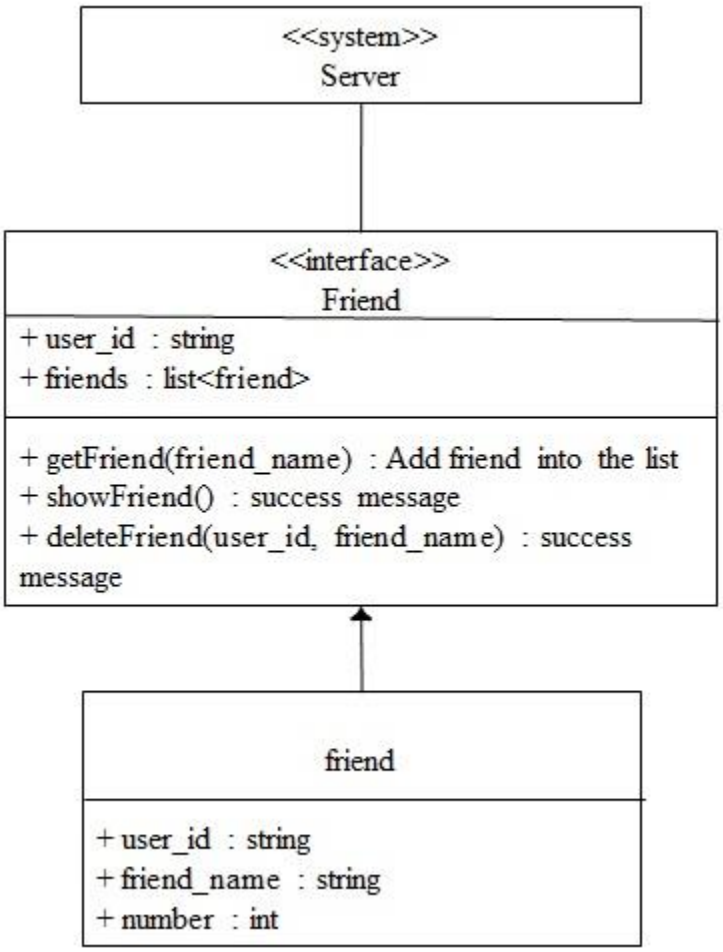
4.2.4.2 Methods

These are the methods that the friend class has.

- getFriend()
- showFriend()
- deleteFriend()

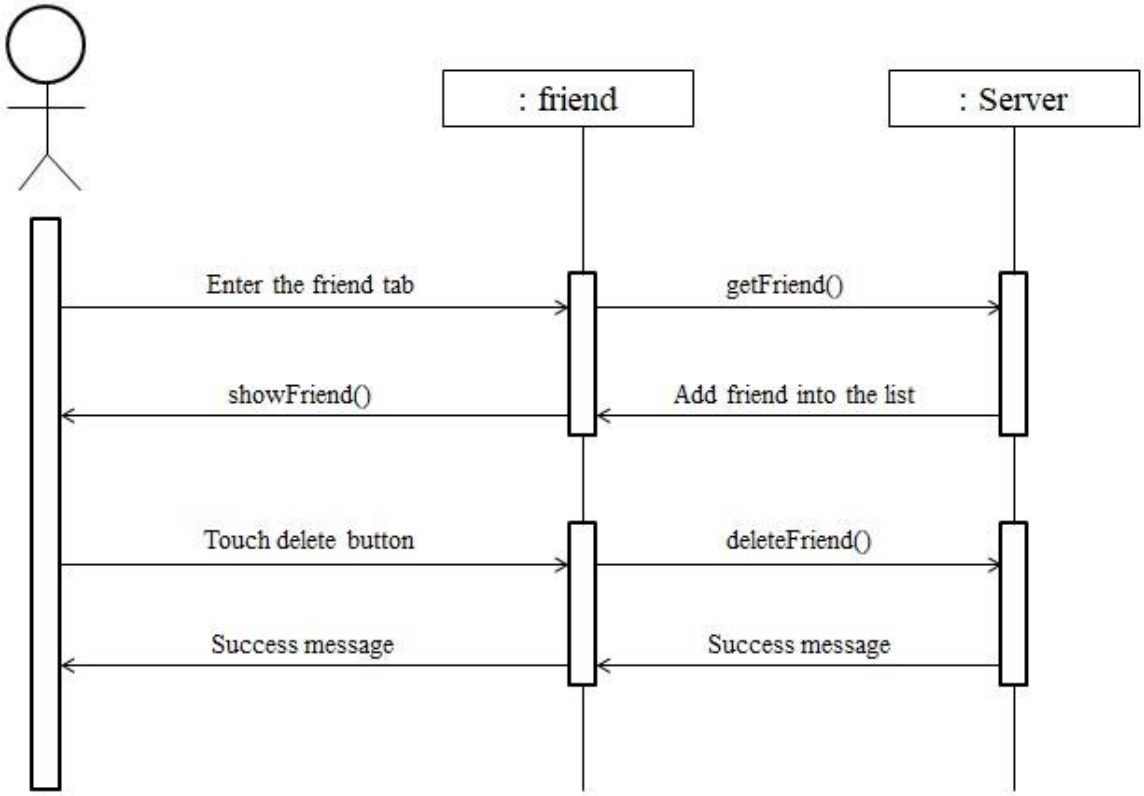
4.2.4.3 Class Diagram

[Figure 11] Class diagram - Friend



4.2.4.4 Sequence Diagram

[Figure 12] Sequence diagram - Friend



4.2.5. Chat Room

4.2.5.1. Attributes

These are the attributes that the chat room object has.

- **user_id** : id of the user.
- **chat_room_name** : name of the chat room.
- **number** : number for the chat room.
- **place_name** : name of the place of the chat room.

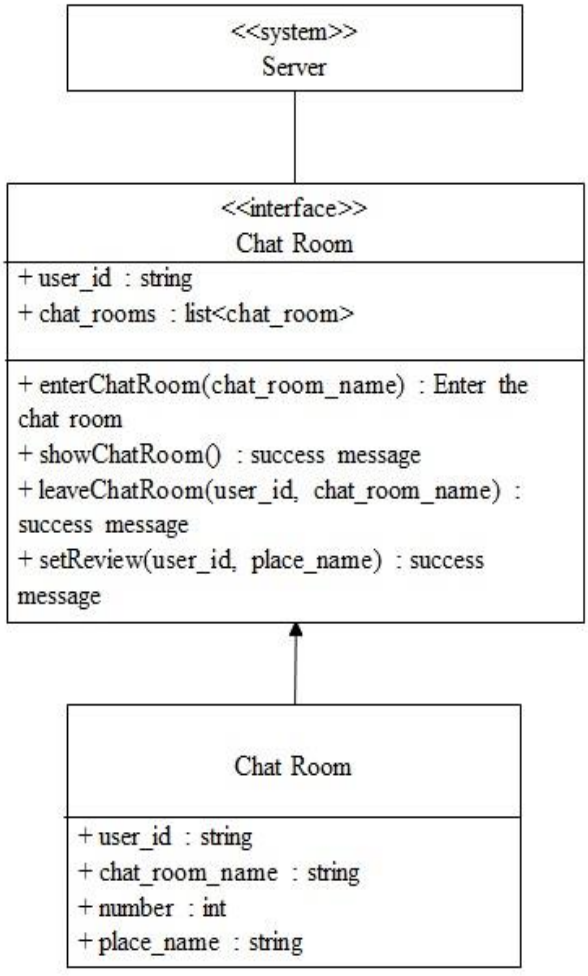
4.2.5.2 Methods

These are the methods that the chat room class has.

- enterChatRoom()
- showChatRoom()
- leaveChatRoom()
- setReview()

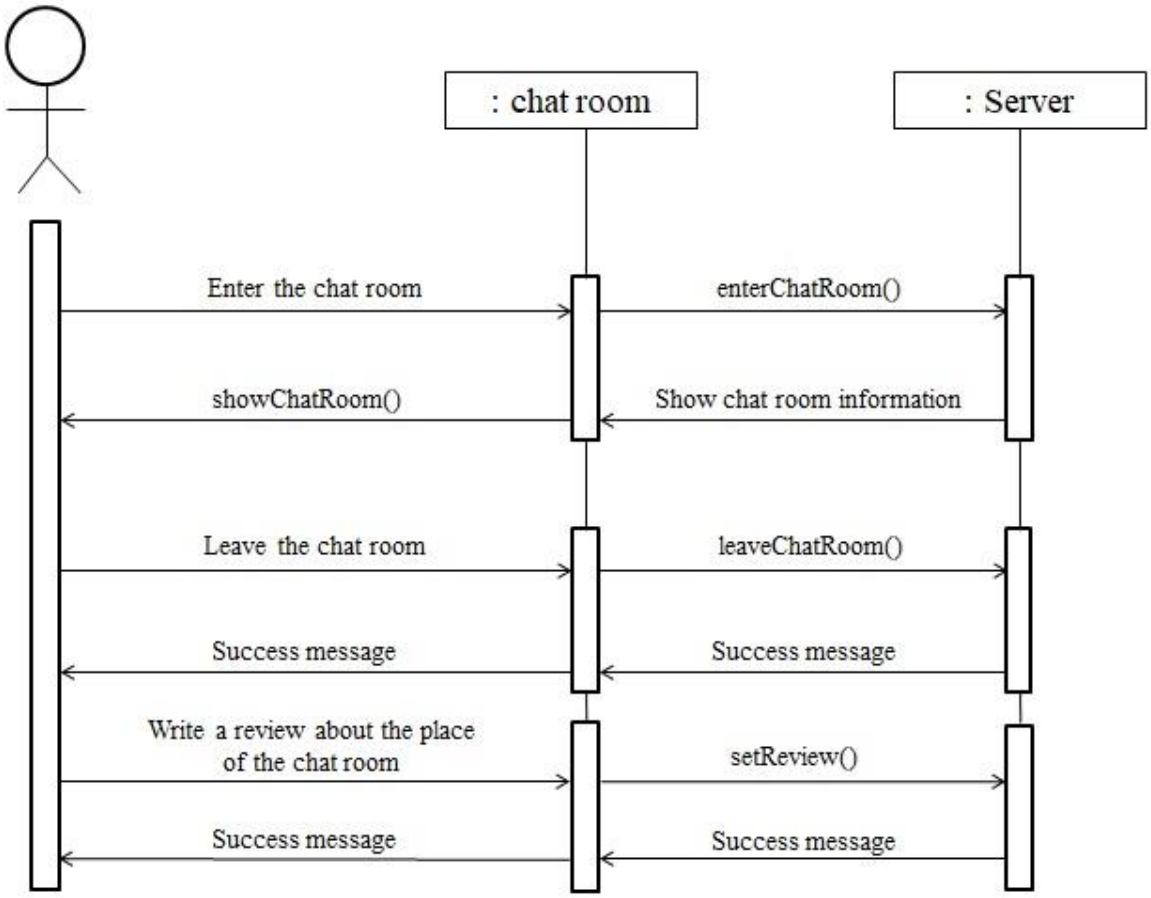
4.2.5.3 Class Diagram

[Figure 13] Class diagram - Chat Room



4.2.5.4 Sequence Diagram

[Figure 14] Sequence diagram - Chat Room



4.2.6. Place Detail

4.2.6.1. Attributes

These are the attributes that the place detail object has.

- **place_name** : name of the place.
- **place_address** : address of the place.
- **phone_number** : phone number of the place.

- **opening_hours** : opening hours of the place.
- **place_menu** : menu of the place.

These are the attributes that the place review object has.

- **place_name** : name of the place.
- **place_review** : review of the place. Saved in list of integers.

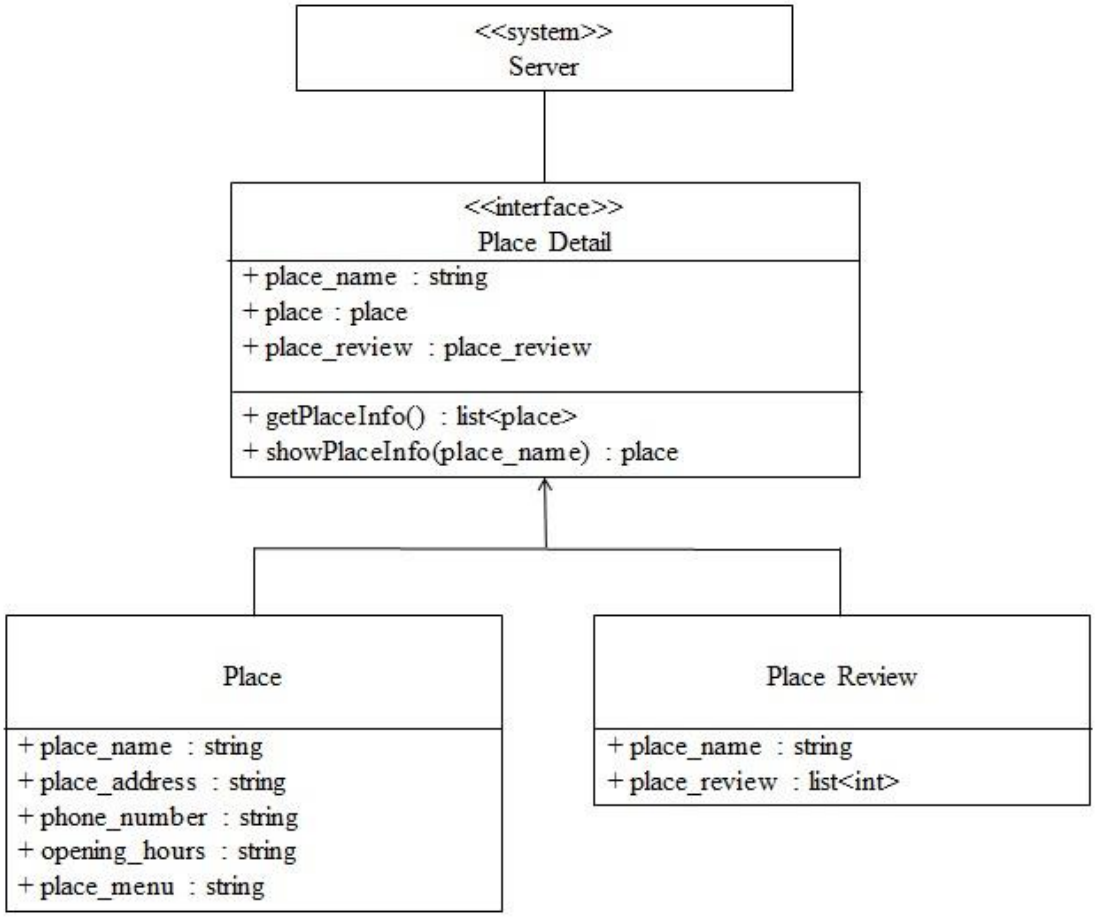
4.2.6.2 Methods

These are the methods that the place detail class has.

- getPlaceInfo()
- showPlaceInfo()

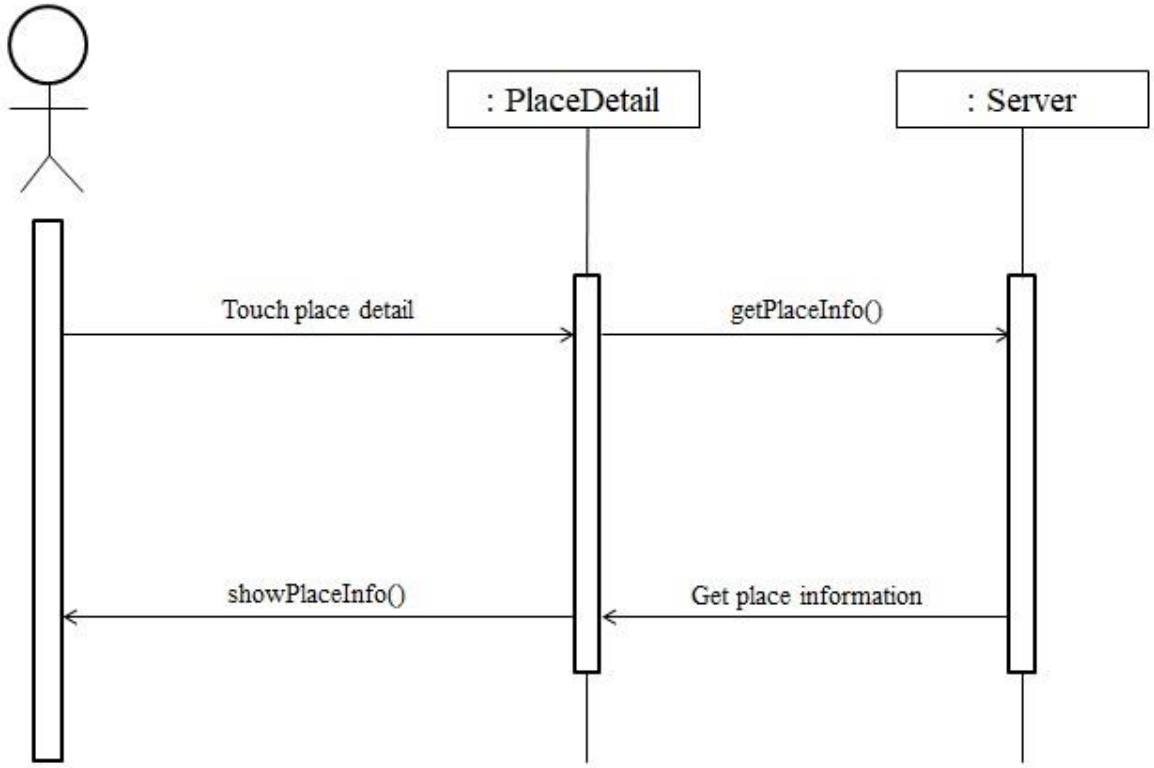
4.2.6.3 Class Diagram

[Figure 15] Class diagram - Place Detail



4.2.6.4 Sequence Diagram

[Figure 16] Sequence diagram - Place Detail



5. System Architecture - Backend

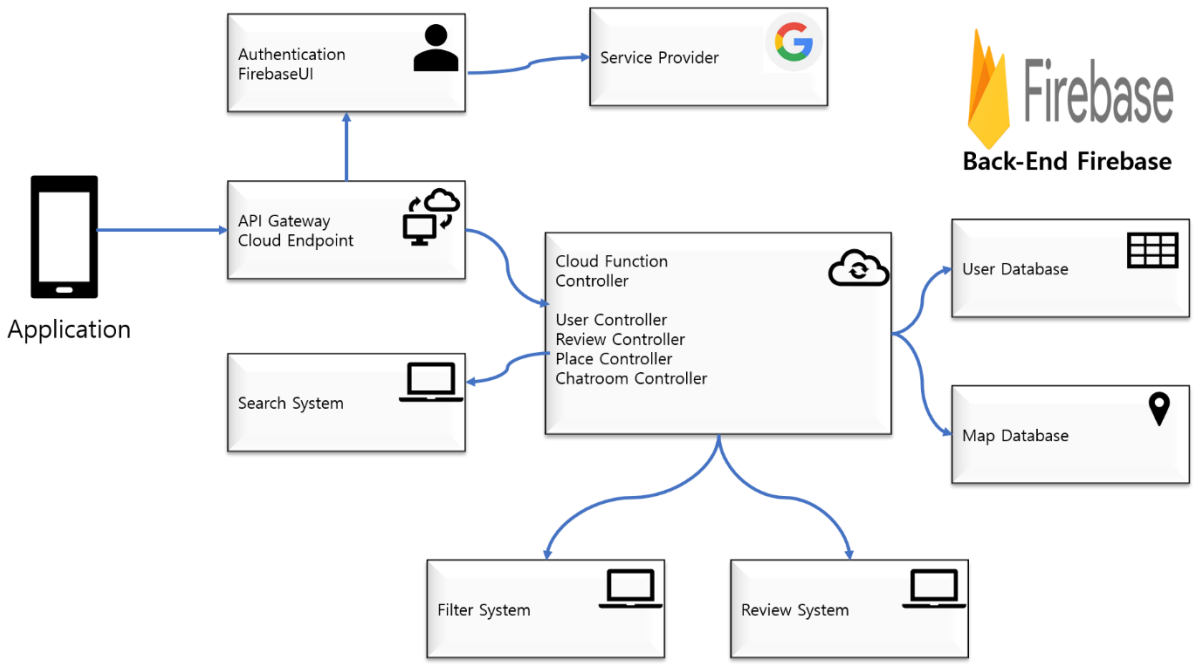
5.1. Objectives

This chapter describes the structure of the back-end system include DB and API Cloud.

5.2. Overall Architecture

The overall architecture of the system is as above. The API gateway (Request Handler) receives the request from the front-end and distributes it to appropriate cloud function (Controller/Manager). In this process, function that use external API such as authentication are handled. Other requests (processed internally) sent to the corresponding controller. The controller interacts with the cloud (Map/User database) and process request.

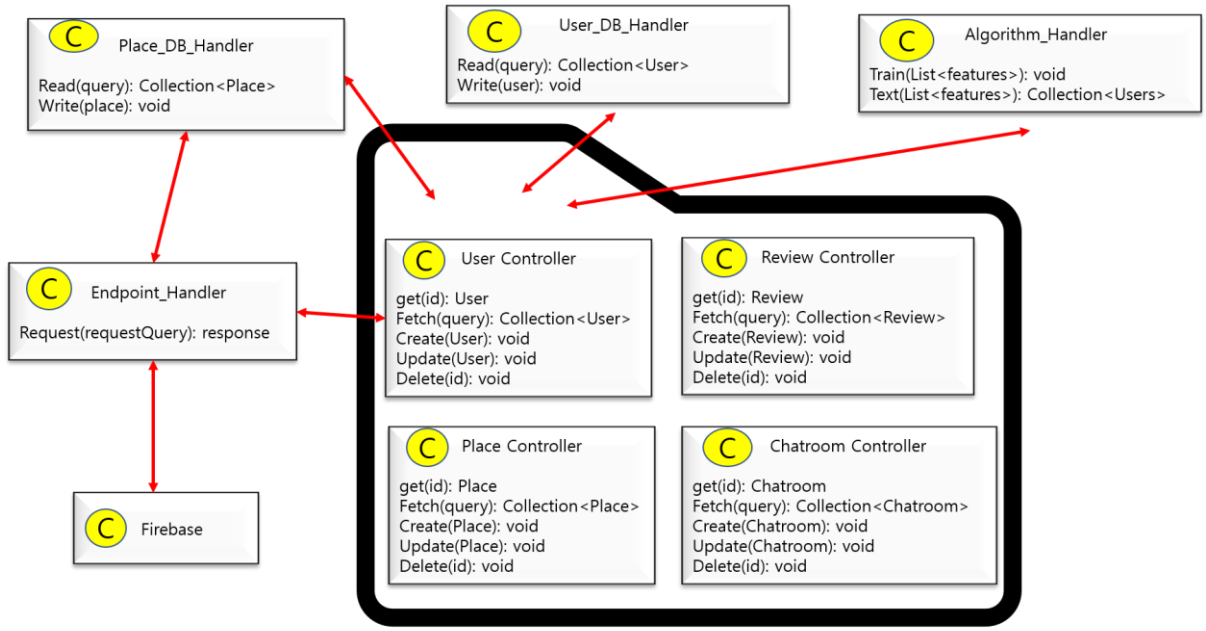
[Figure 17] Overall architecture



5.3. Subcomponents

5.3.1. Cloud Function

[Figure 18] Cloud Function



5.3.1.1. Endpoint Handler Class

API gateway. Distribute requests from the frontend to the appropriate controller or API.

5.3.1.2. FirebaseUI

Class to implement authentication through FirebaseUI

5.3.1.3. Place DB Handler Class

Interface to communicate with DB. The specific place which user searched is fetched from the DB, and the review, chatroom related with place is added to each collection of the place and stored.

5.3.1.4. User DB Handler Class

Interface to communicate with DB. The user information is fetched from the DB such as friends, profile, and reviews about visited place.

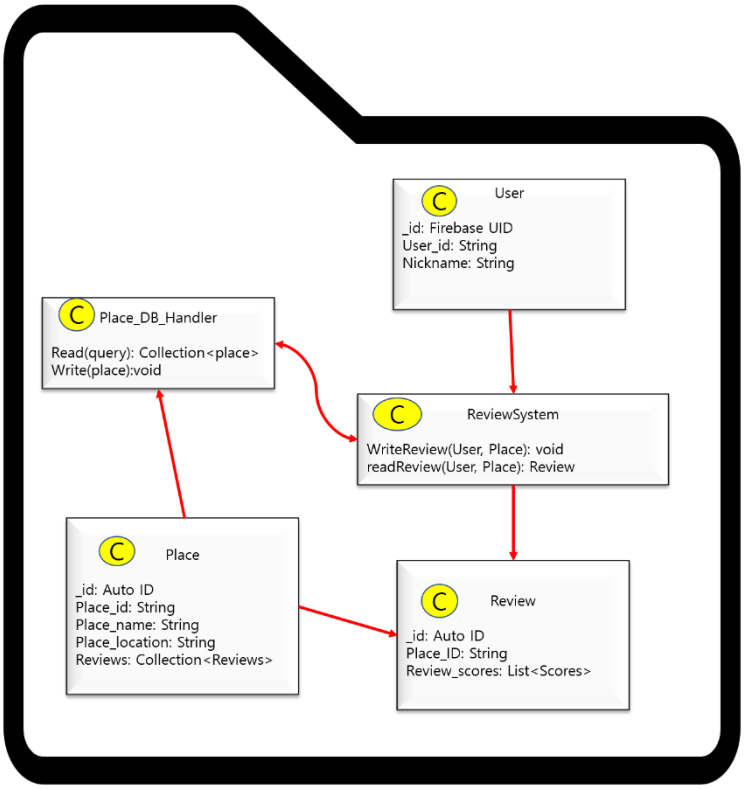
5.3.1.5. Algorithm_Handler

Interface for machine learning. It is used when match users with similar preferences and recommend users to add friend to other user who has same common things (friends, visited place etc.)

5.3.2. Review System

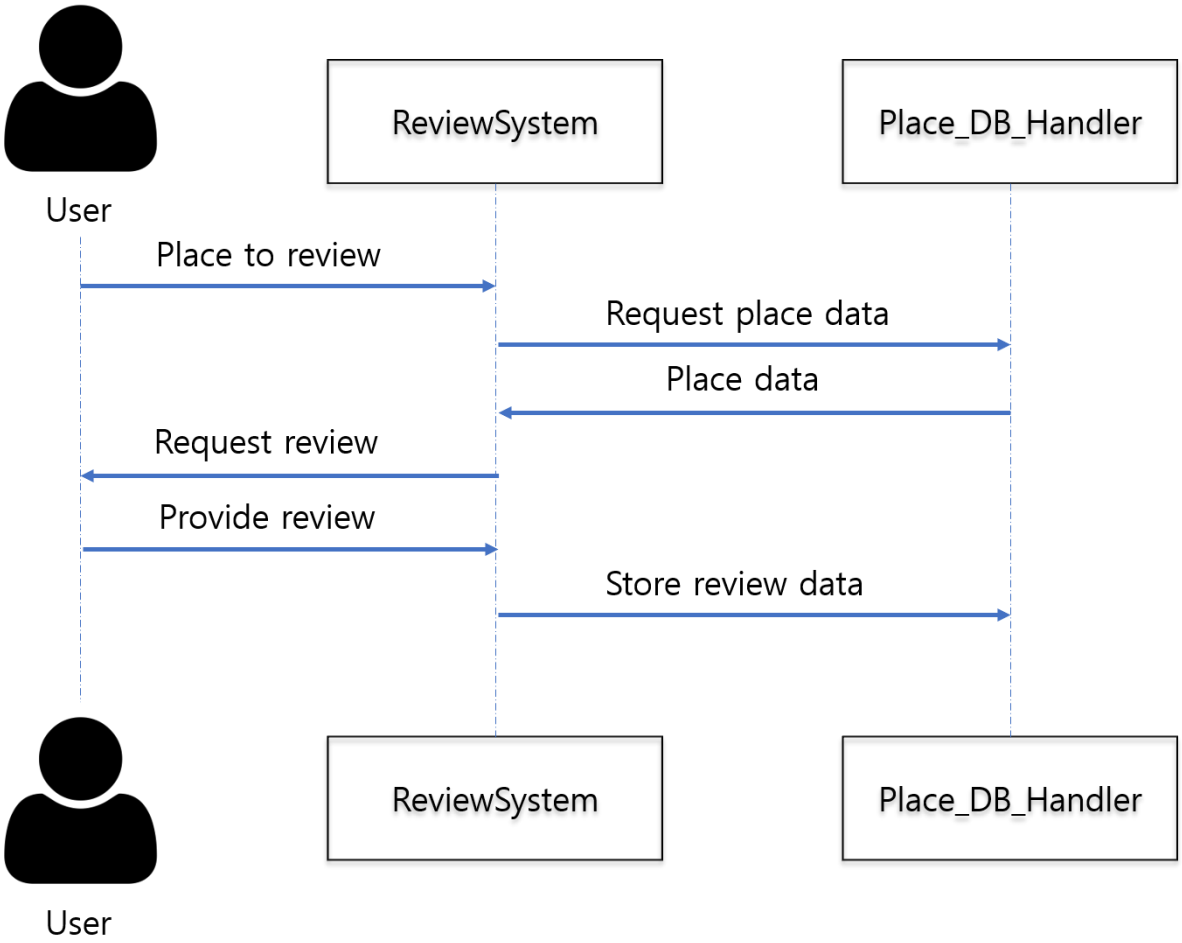
5.3.2.1. Class Diagram

[Figure 19] Class diagram - Review System



5.3.2.2. Sequence Diagram

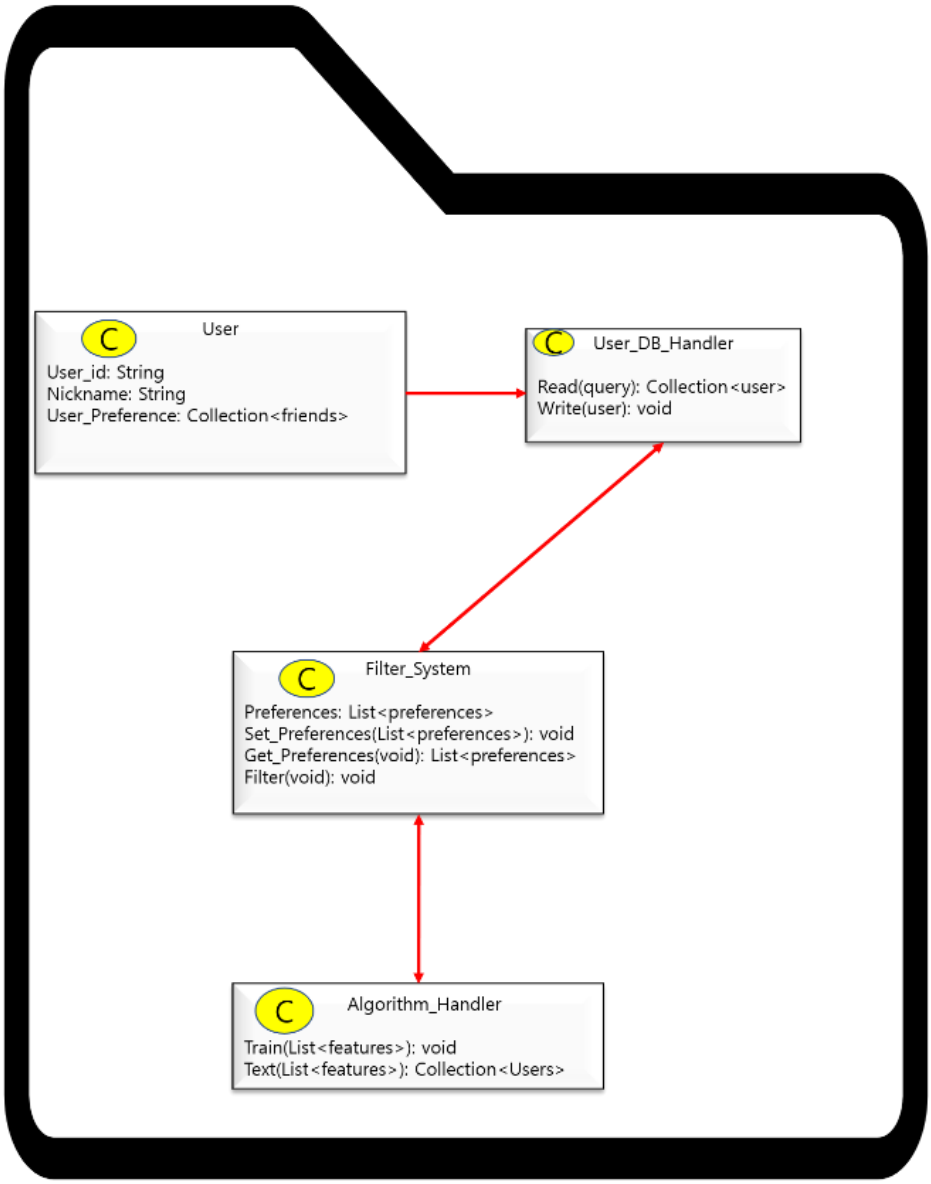
[Figure 20] Sequence diagram- Review System



5.3.3. Filter System

5.3.3.1. Class Diagram

[Figure 21] Class diagram - Filter System

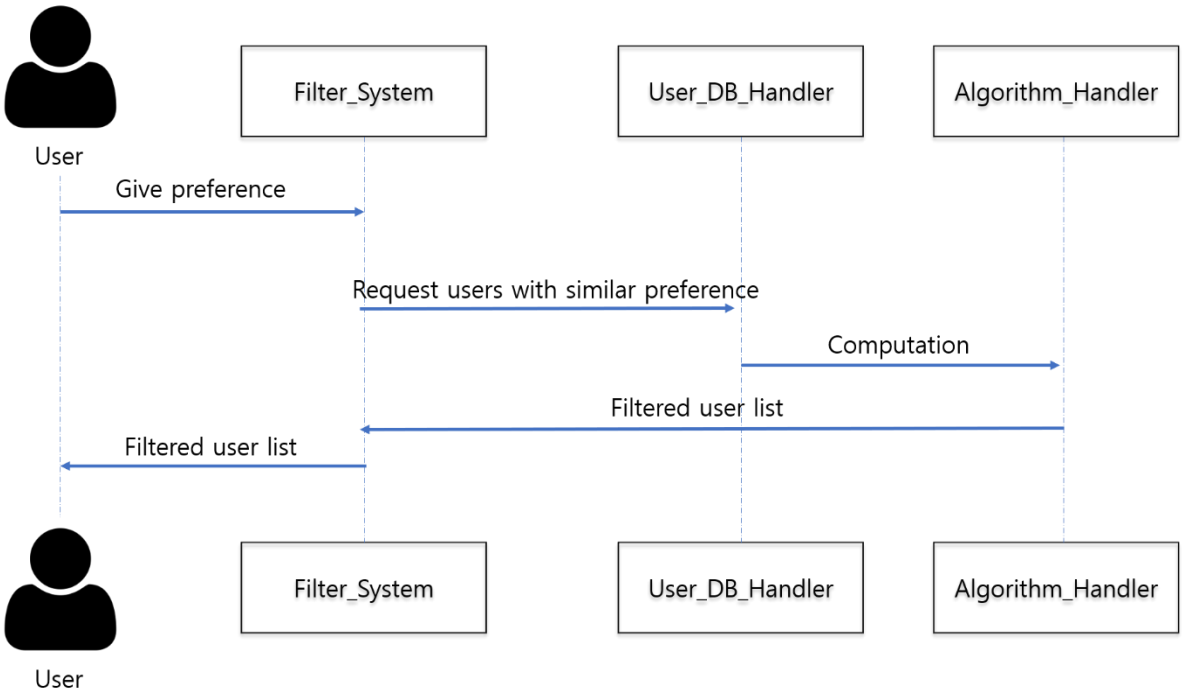


- Class description

  ✓ **Filter Class:** It is an interface for filter friends list to recommend users based on their preferences and other things related with user information or user's friends. Called when users give their new or updated preference, give filtered friends list to user based on database through algorithm handler.

5.3.3.2. Sequence Diagram

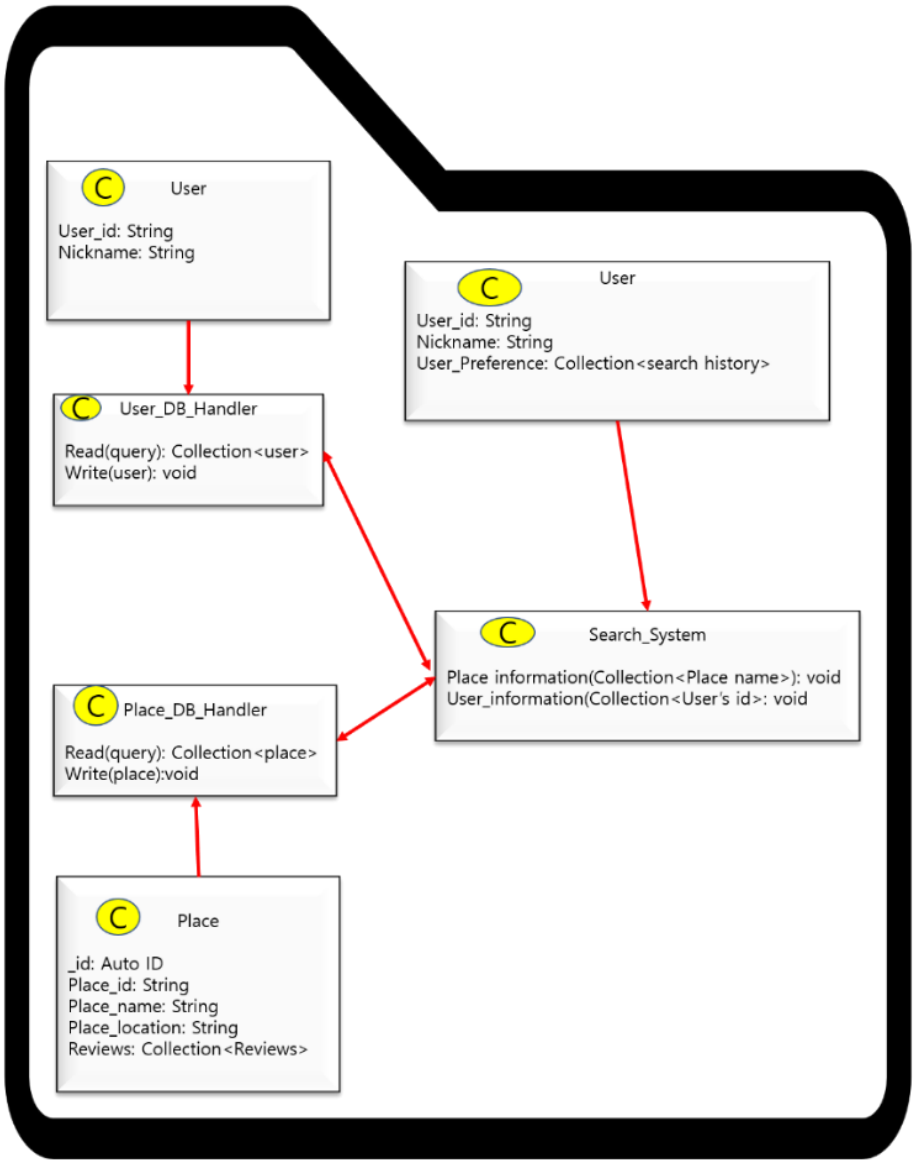
[Figure 22] Sequence diagram - filter system



5.3.4. Search System

5.3.4.1. Class Diagram

[Figure 23] Class diagram - Search System



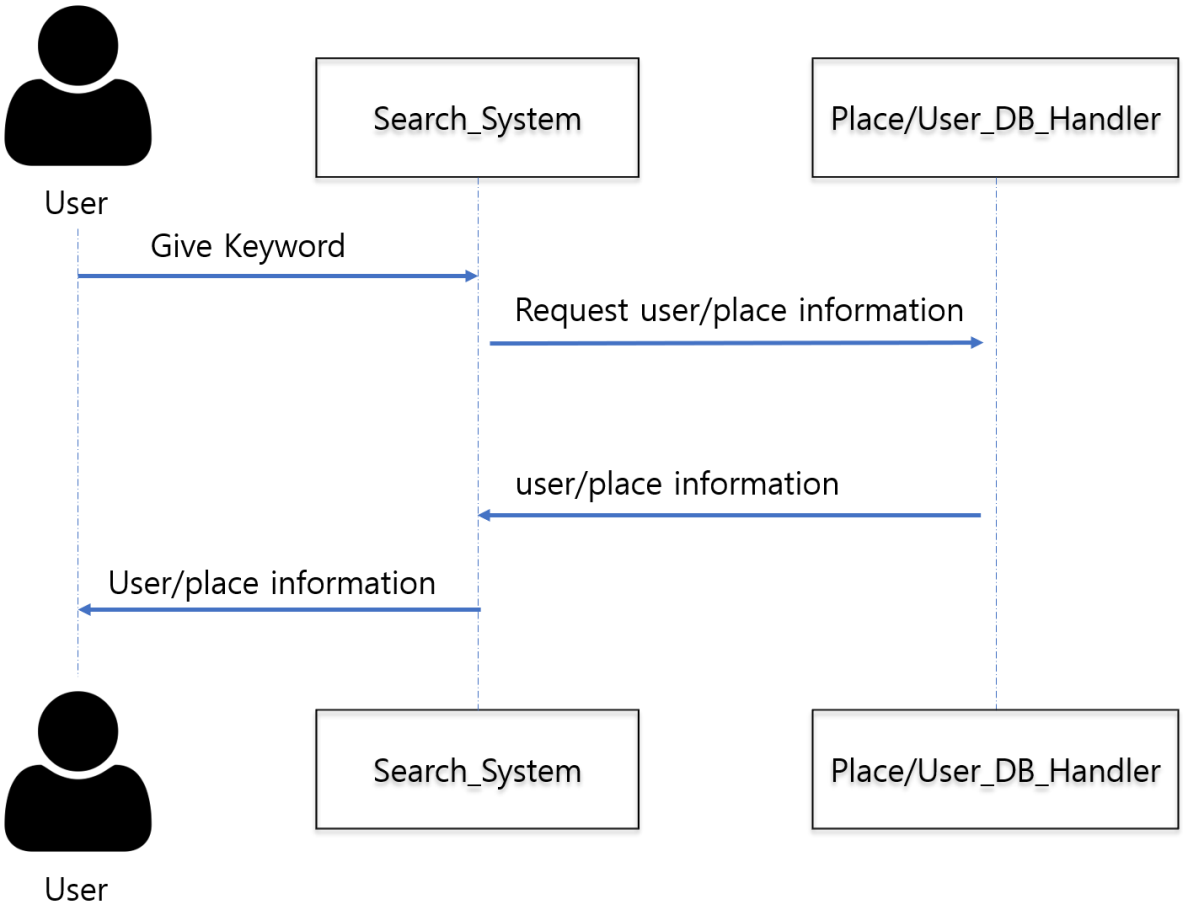
- Class description

  ✓ Search System: This is the interface for search users or places information. When

user request keywords that he searches, the system request user DB or place DB to get matched information and give this information to user.

5.3.4.2. Sequence Diagram

[Figure 24] Sequence diagram - Search System



6. Protocol Design

6.1. Objectives

This chapter explains the protocols that are used for interaction between each subsystem and describe the whole structure of that protocols. It will focus on the application and its server and also explain the definition method of each interface in application.

6.2. JSON

JSON is abbreviation for JavaScript Object Notation, a lightweight data exchange format that is widely used when storing or storing data. And it refers to an expression used when creating an object in Javascript. JSON expressions are easy to understand for both humans and machines and its capacity is small, so recently, JSON has replaced XML and is widely used for data transmission. JSON is just a data format, not any communication method or programming grammar, it is simply a way of expressing data.

6.3. OAuth

OAuth is an open standard for delegation of access, used as a common means by which Internet users can grant website or application access to their information on other websites without providing a password. This mechanism is used by several companies, such as Amazon, Google, Facebook, Microsoft, and Twitter, and allows users to share information about accounts on third-party applications or websites. Before OAuth was used, there was no standard for authentication method, so the existing basic authentication ID and password were used, which is a weak structure in terms of security. In the case of non-basic authentication, each application confirmed the user according to the method of their own developed company. OAuth is a standardized authentication method. With OAuth, applications that share this authentication do not need separate authentication. Therefore, it becomes possible to integrate and use multiple applications.

6.4. Authentication

6.4.1. Register

- Request

[Table 1] register request

Attribute	Detail	
Protocol	OAuth	
Request body	Email	User's school email
	Request token	Token for OAuth
	User	User information

- Response

[Table 2] register response

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 (Bad request, overlap)	
	HTTP 401 (unauthorized)	
	HTTP 404 (Not found)	
	HTTP 500 (Not found)	
Success response body	Access Token	Token for access
	Message	Message: "Access success"
Failure response body	Message	Message: "Access fail"

6.4.2. Log-In

- Request

[Table 3] Log-in request

Attribute	Detail	
Protocol	OAuth	
Request body	Email	User's school email
	Request token	Token for OAuth
	User	User information

- Response

[Table 4] Log-in response

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 (Bad request, overlap)	
	HTTP 401 (unauthorized)	
	HTTP 404 (Not found)	
	HTTP 500 (Not found)	
Success response body	Access Token	Token for access
	Message	Message: "Access success"
Failure response body	Message	Message: "Access fail"

6.5. User profile

6.5.1. Set User Profile

- Request

[Table 5] Set user profile request

Attribute	Detail	
URI	/user/:id/profile	
Method	POST	
Parameter	User	Basic User Information
	Interest	User interest / purpose
Header	Authorization	User authentication

- Response

[Table 6] Set user profile response

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 (Bad request, overlap)	
	HTTP 403 (Forbidden)	
	HTTP 404 (Not found)	
Success response body	Access Token	Token for access
	Message	Message: “Access success”
Failure response body	Message	Message: “Access fail”

6.5.2. Get User Profile

- Request

[Table 7] Get user profile request

Attribute	Detail	
URI	/user/:id/profile	
Method	GET	
Header	Authorization	User authentication

- Response

[Table 8] Get user profile response

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 (Bad request, overlap)	
	HTTP 404 (Not found)	
Success response body	Access Token	Token for access
	Message	Message: “Access success”
	User	User objects (profile)
Failure response body	Message	Message: “Access fail”

6.6. View the Map

- Request

[Table 9] View the map request

Attribute	Detail	
URI	/map	
Method	Get	
Parameter	User	Basic User Information
	location	User's location
Header	Authorization	User authentication

- Response

[Table 10] View the map response

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 (Bad request, overlap)	
	HTTP 404 (Not found)	
Success response body	Access Token	Token for access
	Map view	View the close range of map
	Message	Message: "Access success"
Failure response body	Message	Message: "Access fail"

6.7. Search Friends

- Request

[Table 11] Search Friends request

Attribute	Detail	
URI	User/:id/map	
Method	Get	
Parameter	User	Basic User Information
	interest	User's interest, purpose
	location	User's location
Header	Authorization	User authentication

- Response

[Table 12] Search Friends response

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 (Bad request, overlap)	
	HTTP 404 (Not found)	
Success response body	Access Token	Token for access
	Map view	View the close range of map
	User	Basic user information
	Message	Message: "Access success"
Failure response body	Message	Message: "Access fail"

6.8. Chat

6.8.1. Start chat

- Request

[Table 13] Start chat request

Attribute	Detail	
URI	/user/:id/profile	
Method	GET	
Header	Authorization	User authentication

- Response

[Table 14] Start chat response

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 (Bad request, overlap)	
	HTTP 404 (Not found)	
Success response body	Access Token	Token for access
	Message	Message: “Access success”
	Chatting room	View the new chatting room
Failure response body	Message	Message: “Access fail”

6.8.2. Finish chat

- Request

[Table 15] Finish chat request

Attribute	Detail	
URI	/user/:id/profile	
Method	GET	
Header	Authorization	User authentication

- Response

[Table 16] Finish chat response

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 (Bad request, overlap)	
	HTTP 404 (Not found)	
Success response body	Access Token	Token for access
	Message	Message: “Access success”
	Review page	View the user review page
Failure response body	Message	Message: “Access fail”

6.9. Review

6.9.1. Write review

- Request

[Table 17] Write review request

Attribute	Detail	
URI	User/:id/map	
Method	Get	
Parameter	User	Basic User Information
	Contents	User's review contents
Header	Authorization	User authentication

- Response

[Table 18] Write review response

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 (Bad request, overlap)	
	HTTP 404 (Not found)	
Success response body	Access Token	Token for access
	Message	Message: "Access success"
Failure response body	Message	Message: "Access fail"

6.9.2. Modify review

- Request

[Table 19] Modify review request

Attribute	Detail	
URI	User/:id/review	
Method	Get	
Parameter	User	Basic User Information
	Contents	User's review contents
Header	Authorization	User authentication

- Response

[Table 20] Modify review response

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 (Bad request, overlap)	
	HTTP 404 (Not found)	
Success response body	Access Token	Token for access
	Message	Message: "Access success"
Failure response body	Message	Message: "Access fail"

6.9.3. Delete review

- Request

[Table 21] Delete review request

Attribute	Detail	
URI	User/:id/review	
Method	Get	
Parameter	User	Basic User Information
Header	Authorization	User authentication

- Response

[Table 22] Delete review response

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 (Bad request, overlap)	
	HTTP 404 (Not found)	
Success response body	Access Token	Token for access
	Message	Message: “Access success”
Failure response body	Message	Message: “Access fail”

6.9.4. View the review

- Request

[Table 23] View the review request

Attribute	Detail	
URI	User/:id/review	
Method	Get	
Parameter	User	Basic User Information
Header	Authorization	User authentication

[Table 23

- Response

[Table 24] View the review response

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 (Bad request, overlap)	
	HTTP 404 (Not found)	
Success response body	Access Token	Token for access
	Message	Message: “Access success”
Failure response body	Message	Message: “Access fail”

7. Database Design

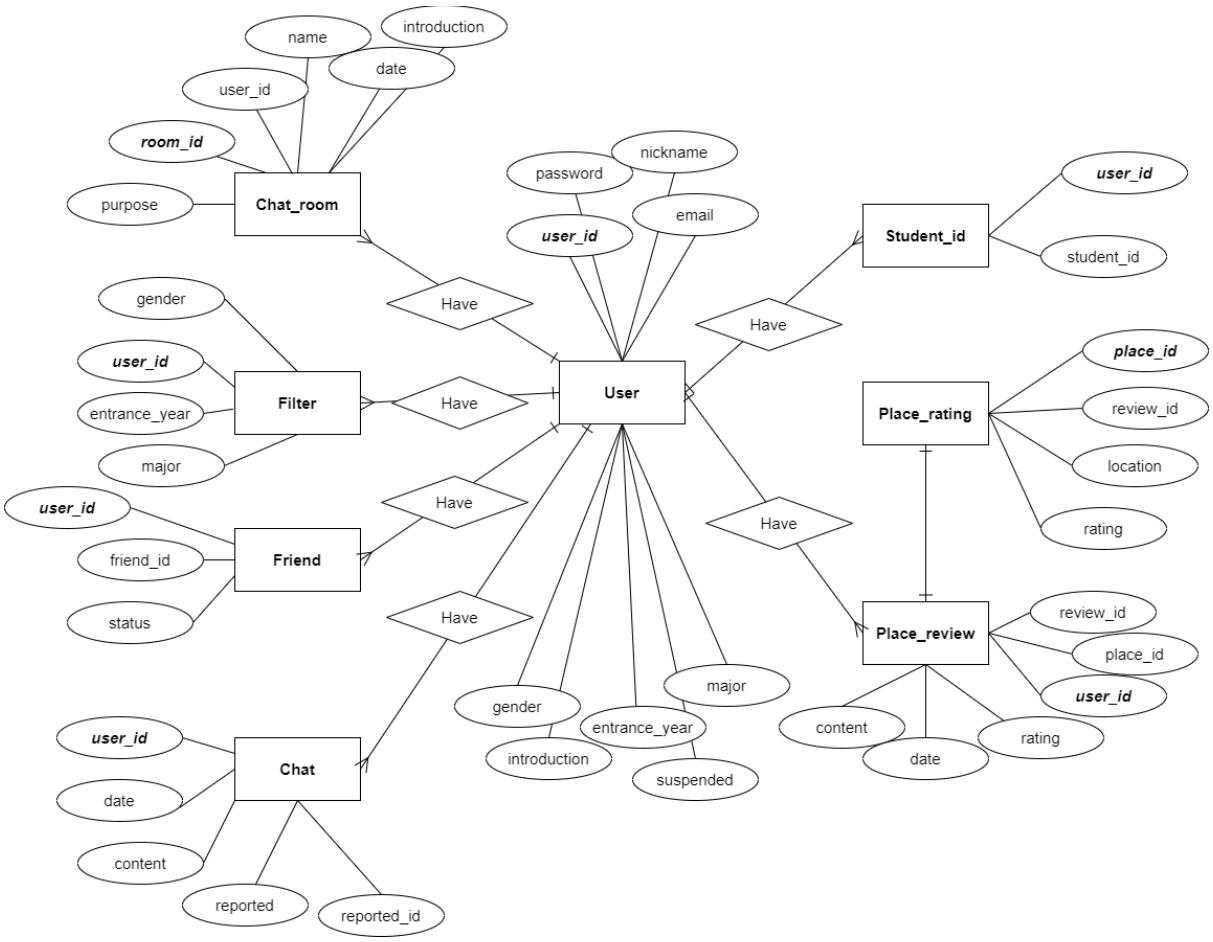
7.1. Objectives

This section describes the system data structures and how these are to be represented in a database. It first identifies entities and their relationship through ER-diagram (Entity Relationship diagram). Then, it generates Relational Schema and SQL DDL (Data Description
Language) specification.

7.2. ER Diagram

The system consists of eight entities: User, Chat_room, Filter, Friend, Chat, Student_id, Place_rating, Place_review, Cart. ER-diagram expresses each entity as rectangular and their relationship as rhombus. When an entity has multiple relationships with another entity, trident (three line) is used to indicate it. When an entity has just one relationship with another entity, the cross (two line) is used to indicate it. The attribute of an entity is expressed as an ellipse. The unique attribute which uniquely identifies an entity is bolded and tilted.

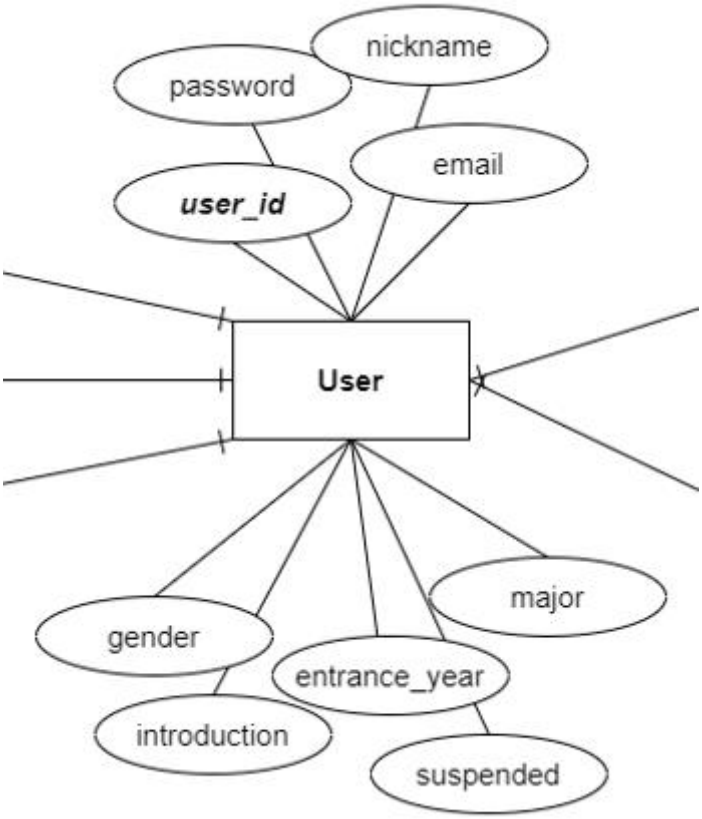
[Figure 25] ER-Diagram



7.2.1 Entities

7.2.1.1. User

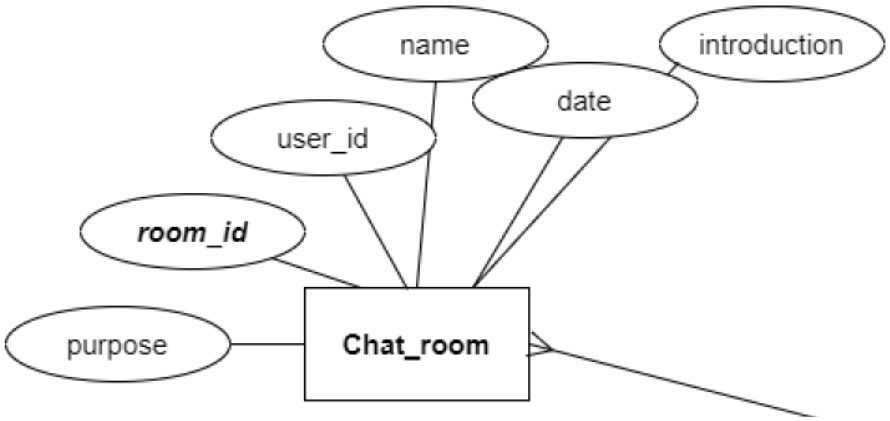
[Figure 26] ER diagram, Entity, User



User entity represents user of "유생찾기". This entity consists of user_id, password, nickname, email, gender, introduction, entrance_year, suspended, major and user_id attribute is the primary key. Users can have a friend list DB and this list contains friends of the user.

7.2.1.2. Chat room

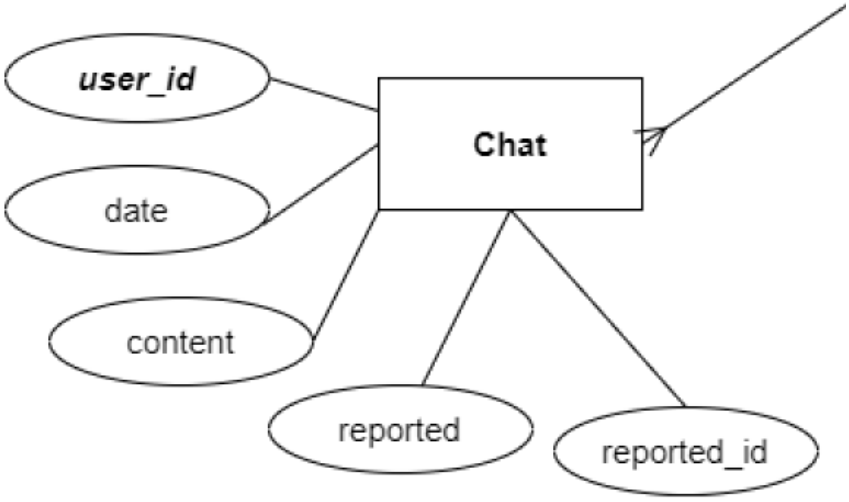
[Figure 27] ER diagram, Entity, Chat room



Chat room represents the chat room made by a user. It consists of user_id, room_id, purpose, name, date, and introduction. And room_id is used as the primary key. In the chat room, user_id means the id of the user who made that room.

7.2.1.3. Chat

[Figure 28] ER diagram, Entity, Chat

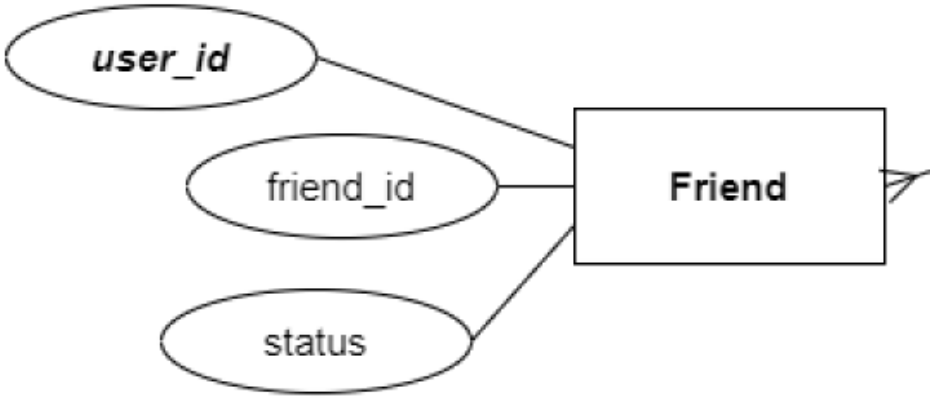


Chat represents the chatting message that the user can send to other users in the chat room. This chat message has the attribute, user_id, date, content, reported and reported_id. Like the chat room entity, the chat entity's user_id also means the id of the user who sends the

message. Reported means that some user reports the message to the system for some reasons and reported_id means the id of the user who reported that message to the system.

7.2.1.4. Friend

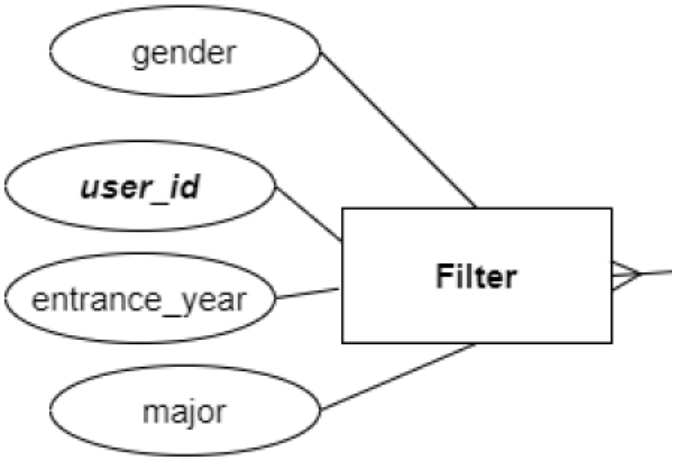
[Figure 29] ER diagram, Entity, Friend



All users can have friends, and a friend entity represents the friend of the user. The user can have a friend list and the friend list contains a friend that consists of user_id, friend_id, and status. Status refers to the status between the user and the friend user, and indicates the block status, the status of being a friend, etc. as a number.

7.2.1.5. Filter

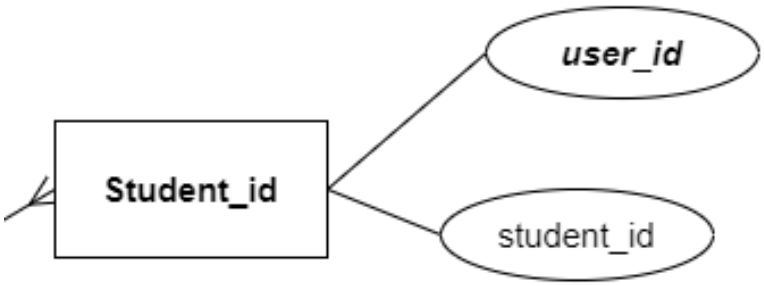
[Figure 30] ER diagram, Entity, Filter



In our system, every user has the filter information. The filter entity represents information on how to filter out the people the user wants to see. It consists of user_id, gender, entrance_year, major, and user_id is worked as the primary key in this entity. The user can filter other users by gender, entrance year, and major, so they can select only the type of person they want to see.

7.2.1.6. Student id

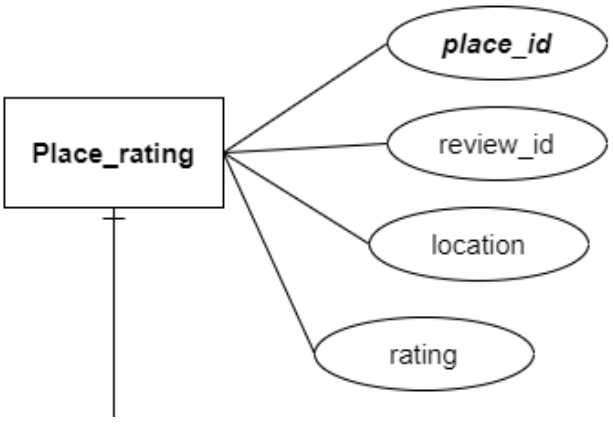
[Figure 31] ER diagram, Entity, Student id



Our system can only be joined by members of the school. Therefore, it should be blocked when a person with an existing user's student ID attempts to sign up. For this, the student_id entity exists.

7.2.1.7. Place rating

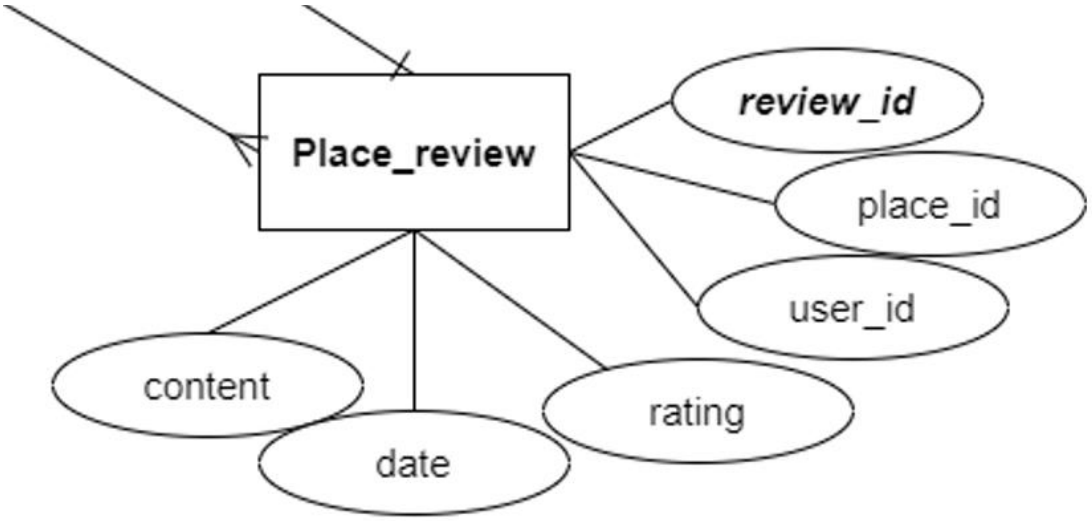
[Figure 32] ER diagram, Entity, Place rating



The user can review and rate the place in the system. Place_rating entity represents the place, review, and ratings. It consists of place_id, review_id, location, rating and place_id is the primary key. The place also has a review list, and the review list consists of reviews about that place.

7.2.1.7. Place review

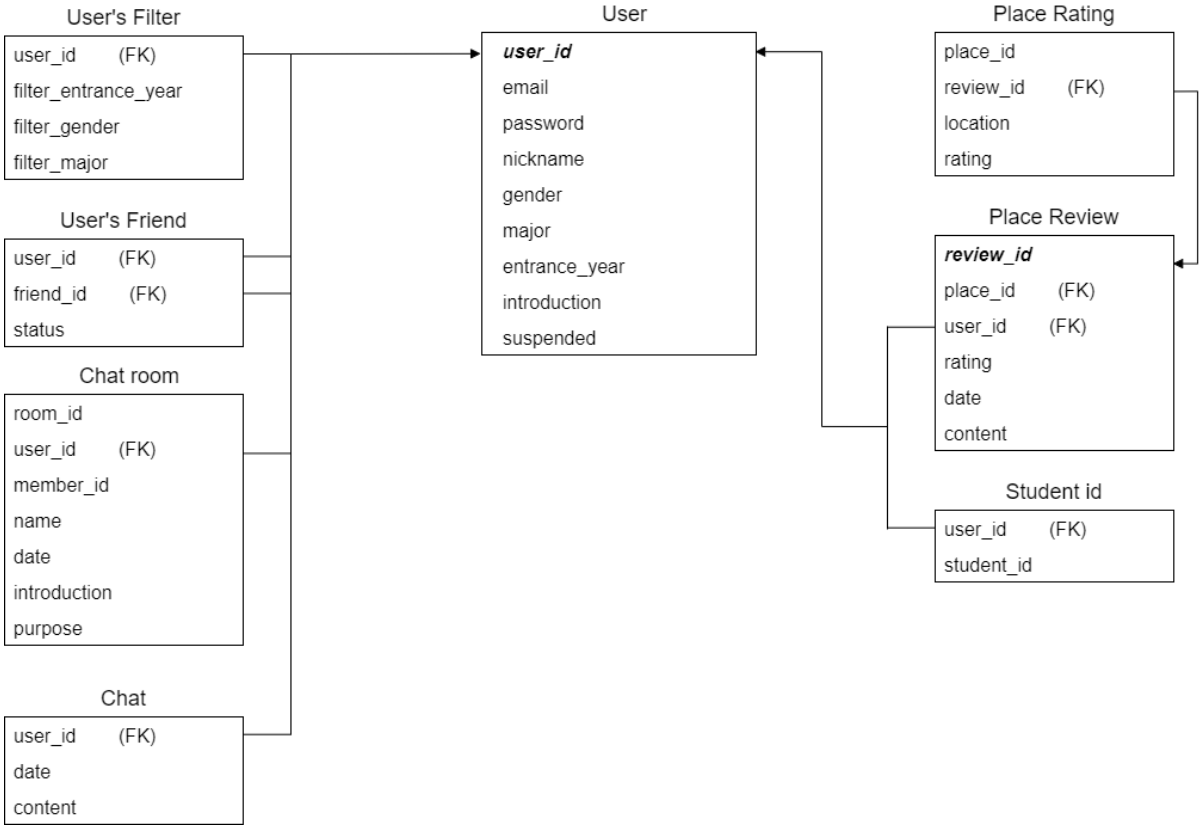
[Figure 33] ER diagram, Entity, Place review



The place review represents a review on the place. It consists of review_id, place_id, user_id, rating, date, content, and review_id is the primary key in this entity. Each review has the author's user_id and place_id.

7.3. Relational Schema

[Figure 34] Relational Schema



7.4. SQL DDL

7.4.1 User

```
CREATE TABLE User
(
   user_id INT NOT NULL
   email VARCHAR NOT NULL
   password VARCHAR NOT NULL
   nickname VARCHAR NOT NULL
   gender VARCHAR NOT NULL
   major VARCHAR NOT NULL
   entrance_year INT NOT NULL
   introduction VARCHAR NOT NULL
   suspended INT NOT NULL
   PRIMARY KEY(user_id)

);
```

7.4.2 User_filter

```
CREATE TABLE User_filter
(
   user_id INT NOT NULL
   filter_entrance_year INT NOT NULL
   filter_gender VARCHAR NOT NULL
   filter_major VARCHAR NOT NULL
   FOREIGN KEY (user_id) REFRENCES
   User(user_id)

);
```

7.4.3 User_friend

```
CREATE TABLE User_friend
(
   user_id INT NOT NULL
   friend_id INT NOT NULL
   status INT NOT NULL
   FOREIGN KEY (user_id) REFRENCES
   User(user_id)
   FOREIGN KEY (friend _id) REFRENCES
   User(user_id)

);
```

7.4.4 Chat_room

```
CREATE TABLE Chat_room
(
   room_id INT NOT NULL
   user_id INT NOT NULL
   member_id INT NOT NULL
   name VARCHAR NOT NULL
   date DATE NOT NULL
   introduction VARCHAR NOT NULL
   purpose VARCHAR NOT NULL
   PRIMARY KEY(room_id)
   FOREIGN KEY (user_id) REFRENCES
   User(user_id)

);
```

7.4.5 Chat

```
CREATE TABLE Chat(
   user_id INT NOT NULL
   date DATE NOT NULL
   content NVARCHAR
   FOREIGN KEY (user_id) REFRENCES
   User(user_id)
);
```

7.4.6 Place_rating

```
CREATE TABLE Place_rating
(
   place_id INT NOT NULL
   review_id INT NOT NULL
   location VARCHAR NOT NULL
   rating VARCHAR NOT NULL
   PRIMARY KEY(place_id)
   FOREIGN KEY (review_id) REFRENCES
   Place_review(review_id)

);
```

7.4.7 Place_review

```
CREATE TABLE Place_review
(
   review_id INT NOT NULL
   place_id INT NOT NULL
   user_id INT NOT NULL
   rating INT NOT NULL
   date DATE NOT NULL
   content NVARCHAR NOT NULL
   PRIMARY KEY(review_id)
   FOREIGN KEY (place_id) REFRENCES
   Place_rating(place_id)
   FOREIGN KEY (user_id) REFRENCES
   User(user_id)

);
```

7.4.8 Student_id

```
CREATE TABLE Student_id
(
   user_id INT NOT NULL
   student_id INT NOT NULL
   PRIMARY KEY(user _id)
   FOREIGN KEY (user _id) REFRENCES
   User(user _id)

);
```

8. Testing Plan

8.1. Objectives

This chapter describes plans for tests that include three main subgroups: development testing, release testing, and user testing. These tests are critical in that they detect potential errors and defects in the product and ensure flawless operation and reliable product market and customer release.

8.2. Testing Policy

8.2.1. Development Testing

Development testing is mainly performed for synchronized application of a wide range of fault prevention and detection strategies to reduce the potential risk of software development and save time and costs.

At this stage, the software may be unstable because it has not been sufficiently tested, and components may crash. Therefore, static code analysis, data flow analysis, peer code review, and unit testing should be performed at this stage. Through these processes, we focus primarily on achieving 1) performance, 2) reliability, ensuring safe and fault-free operations, and 3) security, which define the identity of the software.

8.2.1.1. Performance

Mapping within applications is the most time-consuming task in the system, and it is important for developers to reduce mapping time for many concurrent users. As specified in the recommended specification, the system should provide the user with results within 5 seconds. We will prepare test cases for different preferences, evaluate the speed of mapping functions, and improve the flow of code with respect to mapping algorithms and communication with servers.

8.2.1.2. Reliability

For the system to operate safely without failure, the subcomponents and units that make up the system must first operate and connect correctly. Therefore, we have to go through development tests from the unit development stage and repeatedly check the failure while each unit is integrated into the system.

8.2.1.3. Security

Securing information of users and the system is a crucial matter to be handled by developers. Regardless of the value of the information, it should be protected from unwanted visitors to the system.

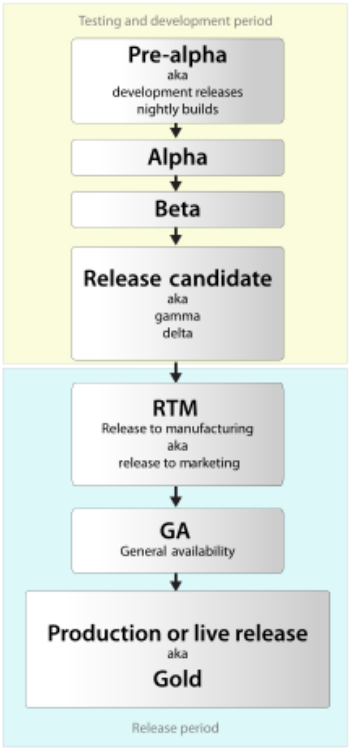
For the security of the system, we can access a near-finished version of the app to identify security issues and create reports through manual code review. In addition, other mobile app security testing services provided by Ostorlab, Appvigil, and others are available, indicating that developers have overlooked application vulnerabilities.

8.2.2. Release Testing

One of the most critical parts of any software development project is to release the product to the market, and customers. A technically good software can go wrong due to a wrong way of release. Therefore, release testing is inevitable for better connection between the product and the market. Release testing is testing a new version of a software/application to verify that that the software can be released in a flawless, so it does not have any defects on its working. It should be carried out before release.

Depending on the software release life cycle, tests typically begin with the 'alpha' version of the software that has completed its basic implementation. We will start development testing in Alpha version and release beta for further testing including user and release testing.

[Figure 35] Software Release Life Cycle



After Beta version is released, we would get feedback from actual users as well as developers.

8.2.3. User Testing

We need to set up a possible scenario and a realistic situation to proceed with the necessary user test. We assume that the 유생찾기 application will have 30 users. After setting up this situation, we conducted our own use case testing with an Android emulator, distributing the 유생찾기 application beta version to them and collecting user reviews.

8.2.4. Testing Case

Test cases are set up based on three basic aspects: functionality (interaction), performance, and security. Set up 5 test cases (15 cases in total) on each side, test the application, and complete

the assessment sheet.

9. Development Plan

9.1. Objectives

This chapter illustrates the technologies and environment for the development of the application.

9.2. Frontend Environment

9.2.1. Adobe Photoshop (UI/UX Design)

[Figure 36] Adobe Photoshop logo

It is a raster graphics editor developed and published by Adobe Inc. for Windows and macOS. This program would provide aesthetical layout and icons for improved user experience during our project.

9.2.2. Adobe Xd (UI/UX Design)

[Figure 37] Adobe Xd logo

It is an all-in-one UX/UI solution program for designing web and mobile app. It provides interactive prototypes in various platforms, which facilitates communication among teammates and enables prompt reflection of feedbacks.

9.2.3. Kakao Oven (UI/UX Design)

[Figure 38] Adobe Xd logo

It is a prototyping tool provided by Kakao. Prototyping refers to designing UI (interface) and UX (user experience) in advance before software development. We can use it for the 'planning' stage of the app. Unlike other tools such as 'Sketch' and 'Protofi', it is basically free. Although Adobe XD is also a free tool, Kakao Oven has the advantage of not having to install a separate program thanks to its HTML5 base. Even if it’s a free tool, but most of the planning we imagine is visualizable.

9.2.4. Android Studio (Application)

[Figure 39] Android Studio logo

It is the official integrated development environment for Google's Android operating system, and it is the most fundamental environment for developing 유생찾기 application. Supporting various programming languages and featuring an intelligent code editor equipped with IntelliJ IDEA interface, Android Studio enables developers to make code more easily and accurately.

For frontend development, we can build a structure of the application, by setting visible actions to be followed according to user interaction in the application using XML layouts.

9.3. Backend Environment

9.3.1. Github (Open source)

[Figure 40] Github logo

It provides hosting for software development version control using Git. It enables teammates to develop a single project together, resulting in easy integration of components. We are now using GitHub for developing iDecide application and controlling version of it.

9.3.2. Firebase (DBMS)

[Figure 41] Firebase logo

It supports development of mobile and web applications by providing various features such as cloud storage, real-time database, machine learning kit, etc.. Among them, we would use real-time database feature for managing data of users, laptops, and etc. Thanks to real-time database, data is synchronized across all the clients connected to this database. It means all clients can share a single real-time database instance and receive updated data with automated update.

9.3.3. SQLite Database (DBMS)

[Figure 42] SQLite Database logo

It is a database management system, such as MySQL and PostgreSQL, but is a relatively lightweight database used by applications, not servers. Although it is not suitable for large-scale tasks compared to typical RDBMS, it is good for small and medium-sized tasks. The API is also characterized by simply calling the library and using only one file to store data. It is also a database built into the Google Android operating system.

9.3.4. Android Studio (Application)

[Figure 43] Android Studio logo

We can add additional features by connecting the application to external APIs, and connect XML files to activities of the application. In addition, we connect the application to DB server for the control of data.

9.3.5. Node.js (Server)

[Figure 44] Nodejs logo

It is a software platform used to develop scalable network applications (especially server-side). It utilizes JavaScript as a written language and has high processing power over non-blocking I/O and single-thread event loops. It includes a built-in HTTP server library, which allows web servers to operate without separate software, such as Apache, allowing more control over the behavior of web servers.

9.3.6. AWS EC2 (Server)

[Figure 45] AWS logo

It is the center of Amazon's cloud computing platform Amazon Web services, allowing users to rent virtual machines and run their own computer applications on them. EC2 encourages scalable application deployment by providing a web service that allows users to boot into the Amazon Machine Image (AMI) and configure virtual machines that Amazon calls "instances" with the desired software. We can create, start, and shut down server instances if necessary, and they use the term "elastic" (elastic) because they pay per hour for running servers. EC2 provides us with control over geographic instance locations that allow for latency optimization and high levels of multiplexing.

9.4. Constraints

The system will be designed and implemented based on the contents mentioned in this document. Other details are designed and implemented by selecting the direction preferred by the developer, but the following items are observed.

- Use the technology that has already been widely proven.
- Mapping speed should not exceed 5 seconds.

- Avoid using technology or software that requires a separate license or pays for royalty. (Exclude this provision if this is the only technology or software that the system must require.)
- Decide in the direction of seeking improvement of overall system performance.
- Decide in a more user-friendly and convenient direction
- Use open source software whenever possible
- Consider the system cost and maintenance cost
- Consider future scalability and availability of the system
- Optimize the source code to prevent waste of system resources
- Consider future maintenance and add sufficient comments when writing the source code
- Develop with Windows 10 environment and Android Studio whose build tools version is 29.0.3
- Develop with minimum Android version 6.0 (API 23) and target Android version 29
- Emulate the system using Android version 10 (API 29)

9.5. Assumptions and Dependencies

All systems in this document are designed and implemented based on Android devices and open sources. Therefore, all content is based on the Android operating system with a minimum API version 23 and may not be applicable to other operating systems or versions.

10. Supporting Information

10.1. Software Design Specification

This software design specification was written in accordance with the IEEE Recommendation (IEEE Recommended Practice for Software Design Description, IEEE-Std-1016).

10.2. Document History

[Figure 46] Document History

Date	Version	Description	Writer
2021/05/08	0.1	Style and overview	Eunju Seok
2021/05/09	1.0	Addition of 8, 9	Eunji Gil
2021/05/10	1.1	Addition of 1,7	Eunju Seok
2021/05/10	1.2	Addition of 6	Hyeyeong Kim
2021/05/11	1.3	Addition of 4	Hyejun Jang
2021/05/12	1.4	Addition of 2, 3	Georyang Park
2021/05/12	1.5	Addition of 5	Jiwon Seo
2021/05/13	1.6	Addition of 4	Hyejun Jang
2021/05/13	1.7	Addition of 4	Georyang Park
2021/05/14	1.8	Revision of 5	Jiwon Seo
2021/05/14	1.9	Addition of 10	Eunji Gil
2021/05/14	2.0	Revision of 6	Hyeyeong Kim
2021/05/15	2.1	Revision of style	Eunju Seok