멘토-멘티 매칭 프로그램

Software Design Specification

2021.05.16

소프트웨어공학 TEAM2

Team Leader 2015310783 서기용

Team Member 2015318579 김정훈

Team Member 2017310474 김시인

Team Member 2018312827 김민지

<CONTENTS>

<List of Figures>

<List of Tables>

1. Preface

이 문서는 멘토-멘티 매칭 애플리케이션의 Design Specification을 기술하기 위해 작성된 디자인 명세서입니다.

1.1. Readership

이 문서는 크게 10개의 Chapter와 각 Chapter 내 세부적인 subsections으로 작성되었습니다. 이 문서의 주요 독자는 2021년 1학기 성균관대학교 소프트웨어공학 수업의 Team2입니다. 또한 소프트웨어공학 수업의 교수님, TA, 다른 Team의 학생들 역시 이 문서의 주요 독자가 될 수 있습니다. 그 외 개발자, 테스터 등 모든 이해관계자도 예상 독자층에 포함됩니다.

1.2. Scope

문서의 10개 Chapter는 애플리케이션의 소프트웨어 엔지니어링을 위해 필요한 frontend, backend 시스템 아키텍처, protocol, database 디자인 등 시스템 디자인 설계 및 구현 관련 전반적인 내용을 포함합니다.

1.3. Objective

이 문서의 주요 목적은 멘토-멘티 매칭 애플리케이션 개발과 실행에 필요한 소프트웨어 아키텍처 디자인 구조 및 아키텍처 설계 관련 정보를 제공하는 것입니다. 문서의 내용은 객체 지향 소프트웨어 디자인 및 개발을 전제로 합니다. 전체적인 시스템 아키텍처 구조와 더불어 frontend, backend 시스템 아키텍처로 구분하여 주요 서브 컴포넌트 구성요소들을 여러 UML 다이어그램 기반으로 설명합니다. 시스템 아키텍처와 더불어 protocol, database 디자인과 테스트, 개발 계획에 대한 정보를 전달하는 것이 문서의 목적입니다.

1.4. Document Structure

- 1. Preface: 문서의 예상 독자층, 범위 정의와 더불어 전체적 목표, 문서 구조를 다룹니다.
- 2. Introduction: 시스템 디자인에 활용할 다이어그램, 도구 및 시스템 목표를 설명합니다.
- 3. Overall System Architecture: 시스템의 전체적인 아키텍처 구조를 class, sequence, use case

다이어그램으로 표현합니다.

- 4. System Architecture – Frontend: 시스템의 frontend 아키텍처를 구성하는 세부 요소를 class, sequence 다이어그램으로 표현합니다.
- 5. System Architecture - Backend: 시스템의 backend 아키텍처를 구성하는 세부 요소를 class, sequence 다이어그램으로 표현합니다.
- 6. Protocol Design: 서버와 클라이언트 간 통신에 필요한 protocol 디자인을 다룹니다.
- 7. Database Design: 시스템의 데이터베이스를 ERD, SQL DDL으로 설명합니다.
- 8. Testing Plan: 시스템 테스트 계획을 설명합니다.
- 9. Development Plan: 시스템 개발에 활용할 개발 도구 및 개발 관련 이슈를 설명합니다.
- 10. Supporting Information: 문서 작성 기준 및 문서 작성 기록 정보입니다.

2. Introduction

이 프로젝트는 캠퍼스 내 소셜 미디어로 사용되는 모바일 애플리케이션을 개발하고 설계하는 것이다. 이 어플리케이션은 '멘토 매칭 프로그램'은 다른 커뮤니티 앱과 다르게 익명에서 벗어나 학생들 간에 1:1 소통을 할 수 있는 서비스를 제공합니다. 또, 일반 동아리 가입/개설이 부담스럽거나 원활한 학업을 위한 정보가 필요한 사용자에게 소모임 기능이 제공되기 때문에 사용자들은 본인의 상황에 맞게 타 사용자들과 교류할 수 있습니다. 시스템은 사용자의 요구에 맞춘 일치를 찾기 위해 특수 알고리즘을 사용하여 사용자에게 가장 필요한 정보를 보여주며 유저들이 읽고 싶은 것들을 눌러서 유저가 원하는 것을 알아 볼 수 있다. 이 설계 문서는 프로젝트 구현에 사용되거나 사용되도록 의도된 설계를 제시한다. 설명된 설계는 프로젝트를 위해 미리 준비한 소프트웨어 요구 사항 사양 문서에 지정된 요구 사항을 따릅니다.

2.1. Objectives

이 장에서는 설계 단계에서 이 프로젝트에 적용된 다양한 도구와 도표를 설명을 하고 있습니다.

2.2. Applied Diagram

2.2.1 UML

UML은 Unified Modeling Language의 약자이다. 간단히 말해서 UML은 소프트웨어의 모델링 및 문서화 방식에 대한 현대적인 접근 방식입니다. 사실, 이것은 가장 인기 있는 비즈니스 프로세스 모델링 기법 중 하나입니다. UML은 소프트웨어 시스템의 아티팩트의 기존 또는 새로운 비즈니스 프로세스, 구조 및 동작을 기술, 지정, 설계 및 문서화하는 데 사용되는 비즈니스 분석가, 소프트웨어 설계자 및 개발자를 위한 공통 언어입니다. UML은 다양한 애플리케이션 도메인(예: 은행, 금융, 인터넷, 항공우주, 의료 등)에 적용할 수 있습니다. 모든 주요 객체 및 구성요소 소프트웨어 개발 방법과 다양한 구현 플랫폼(예: J2EE, .NET)에 사용할 수 있다. 소프트웨어 구성 요소의 다이어그램 표현을 기반으로 합니다. 옛 속담에 있듯이: "그림은 천 마디의 가치가 있다." 시각적 표현을 사용함으로써 소프트웨어 또는 비즈니스 프로세스에서 발생할 수 있는 결함이나 오류를 더 잘 이해할 수 있습니다.

2.2.2. Use Case Diagram

시스템의 주춧돌 부분은 시스템이 충족시키는 기능적 요건이다. 사례 다이어그램을 사용하여 시스템의 높은 수준 요구 사항을 분석합니다. 이러한 요건은 다른 사용 사례를 통해 표현된다. 이 UML 다이어그램의 세 가지 주요 구성 요소가 있습니다. 기능적 요건 – 유스 케이스로 표현되며, 동작을 설명하는 동사. 행위자 – 행위자는 시스템과 상호작용하며, 행위자는 인간, 조직 또는 내부 또는 외부 응용자가 될 수 있습니다. 행위자와 사용 사례 간의 관계 - 직선 화살표를 사용하여 표현됩니다.

2.2.3 Sequence Diagram

시퀀스 다이어그램은 컴퓨터 과학 커뮤니티뿐만 아니라 비즈니스 애플리케이션 개발을 위한 설계 레벨 모델로서도 가장 중요한 UML 다이어그램일 것이다. 최근에는 시각적 자기 설명적 특성 때문에 비즈니스 프로세스를 묘사하는 데 인기를 끌고 있습니다. 이름에서 알 수 있듯이, 시퀀스 다이어그램은 행위자와 객체 간에 발생하는 메시지 및 상호 작용의 순서를 설명합니다. 행위자나 사물은 필요하거나 다른 사물이 그 행위자들과 의사소통하기를 원할 때만 활동할 수 있다. 모든

의사소통은 연대순으로 표현된다.

2.2.4. Class Diagram

통합 모델링 언어(Unified Modeling Language, UML)의 클래스 다이어그램(class diagram)은 시스템의 클래스, 속성, 연산(또는 메서드) 및 객체 간의 관계를 보여줌으로써 시스템의 구조를 설명하는 정적 구조 다이어그램의 일종이다. 클래스는 객체의 구성 블록이기 때문에 클래스 다이어그램은 UML의 구성 블록이다. 클래스 다이어그램의 다양한 구성 요소는 실제로 프로그래밍될 클래스, 주 객체 또는 클래스와 객체 사이의 상호 작용을 나타낼 수 있다. 클래스 셰이프 자체는 세 개의 행이 있는 직사각형으로 구성됩니다. 위쪽 행에는 클래스의 이름이 포함되고 가운데 행에는 클래스의 속성이 포함되며 아래쪽 섹션에는 클래스에서 사용할 수 있는 메서드 또는 작업이 표시됩니다. 클래스와 하위 클래스가 함께 그룹화되어 각 개체 간의 정적 관계를 표시합니다.

2.2.5. Context Diagram

시스템 컨텍스트 다이어그램(레벨 0 DFD라고도 함)은 데이터 흐름 다이어그램에서 가장 높은 레벨이며 모델링할 시스템의 컨텍스트와 경계를 설정하는 전체 시스템을 나타내는 하나의 프로세스만 포함합니다. 그것은 시스템과 외부 실체(즉, 행위자) 사이의 정보 흐름을 식별한다. 상황별 다이어그램은 일반적으로 요구사항 문서에 포함되어 있습니다. 모든 프로젝트 이해 관계자가 읽어야 하므로 이해 관계자가 항목을 이해할 수 있도록 쉬운 언어로 작성해야 합니다. 시스템 컨텍스트 다이어그램의 목적은 시스템 요건 및 제약 조건의 완전한 세트를 개발하는 데 고려해야 하는 외부 요인 및 사건에 주의를 집중하는 것이다. 시스템 컨텍스트 다이어그램은 프로젝트 초기에 조사 대상 범위를 결정하는 데 종종 사용됩니다. 따라서, 문서 내에서. 시스템 컨텍스트 다이어그램은 시스템과 상호 작용할 수 있는 모든 외부 엔터티를 나타냅니다. 전체 소프트웨어 시스템이 단일 프로세스로 표시됩니다. 이러한 다이어그램은 모든 외부 실체, 상호 작용하는 시스템 및 환경에 의해 둘러싸인 내부 구조에 대한 세부 정보가 없는 중앙의 시스템을 보여줍니다.

2.2.6 Entity Relationship Diagram

ERD(Entity Relationship diagram)는 데이터베이스에 저장된 엔티티 세트의 관계를 보여 줍니다. 이 컨텍스트에서 엔티티는 개체이며 데이터의 구성 요소입니다. 엔티티 집합은 유사한 엔티티의 컬

렉션입니다. 이러한 엔티티는 속성을 정의하는 속성을 가질 수 있습니다. 엔티티, 속성을 정의하고 이들 간의 관계를 표시함으로써, ER 다이어그램은 데이터베이스의 논리적 구조를 보여준다. 엔티티 관계 다이어그램은 데이터베이스 설계를 스케치하는 데 사용됩니다.

2.3. Applied Tools

2.3.1. Microsoft PowerPoint

이것은 도면 텍스트와 그림을 지원하는 도구입니다. 다양한 도형을 이용하여 도표를 그리는 것이 편리하다. 또한 전체 워드프로세서 서식(문서 작업 도구)과 도표 그리기용 텍스트가 부착된 그래픽 도형으로 작동하기 때문에 편집이 용이하다.

2.3.2. ERD Plus

이 도구는 ERD(실체 관계 다이어그램), 관계 스키마(관계 다이어그램) 및 스타 스키마(차원 모델)를 빠르고 쉽게 만들 수 있는 웹 기반 데이터베이스 모델링 도구입니다.

2.4. Project Scope

‘멘토 매칭 프로그램’은 코로나로 인해 학생간 비대면 친목, 교류가 힘든 학생들에게 1:1 멘토 매칭 서비스를 제공합니다. 사용자가 가입 시점에 기입 했던 정보를 이용하여 그에 적합한 멘토를 볼 수 있고, 멘토링을 신청할 수 있습니다. 사용자가 기입 한 정보는 데이터베이스에 저장되고 사용자는 실시간으로 멘토 성사 여부를 확인할 수 있습니다
동아리 가입이나 개설이 부담스러운 사용자에게 소모임 기능도 제공합니다. 검색어 입력을 통해 원하는 분야의 소모임 개설 여부를 확인하고 가입하거나 개설할 수 있습니다.

2.5. References

- IEEE Recommended Practice for Software Requirements Specifications
- Ian Sommerville, ‘Software Engineering 10th Edition’
- 2020 SKKUSE Team 1, ‘https://github.com/skkuse/2020spring_41class_team1’

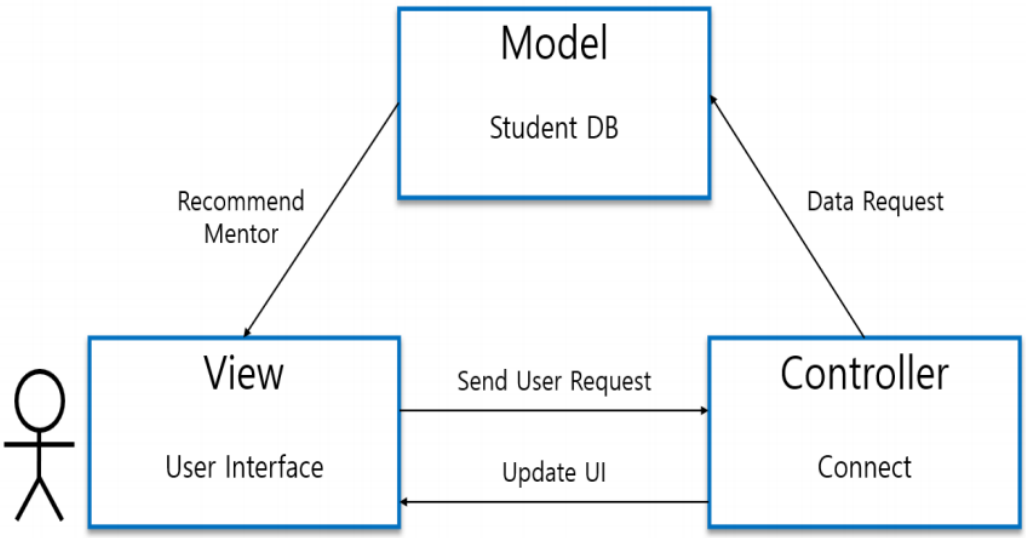
3. System Architecture – Overall

3.1 Objectives

3장에서는 Front-end와 Back-end에 걸쳐서 '멘토 추천 시스템'을 구성하는 subcomponent들을 기술하고 이들이 어떻게 구성되는지를 설명합니다.

3.2 System Organization

저희가 개발하는 시스템은 Client – Server 형식의 system organization을 따르고 있습니다. 저희가 제공하는 서비스는 사용자가 입력한 데이터가 DB로 전송되고 이것이 처리된 뒤 사용자가 원하는 Output을 사용자에게 출력합니다. 사용자에 의해 변경되는 데이터가 다른 사용자가 얻는 데이터에 실시간으로 반영이 되기 위해서 Front-end 와 Back-end application은 HTTP를 통해서 JSON 형식으로 전달됩니다. Back-end application은 사용자가 처음 가입할 때 입력한 정보를 프로필 처리시스템으로 전달합니다. 프로필 처리시스템에서 처리된 사용자의 정보는 Student DB에 저장됩니다. 사용자의 reuqest는 Controller에 의해 Student DB로 전달되고 DB에서 처리된 정보는 다시 Controller에 의해 사용자에게 출력됩니다.

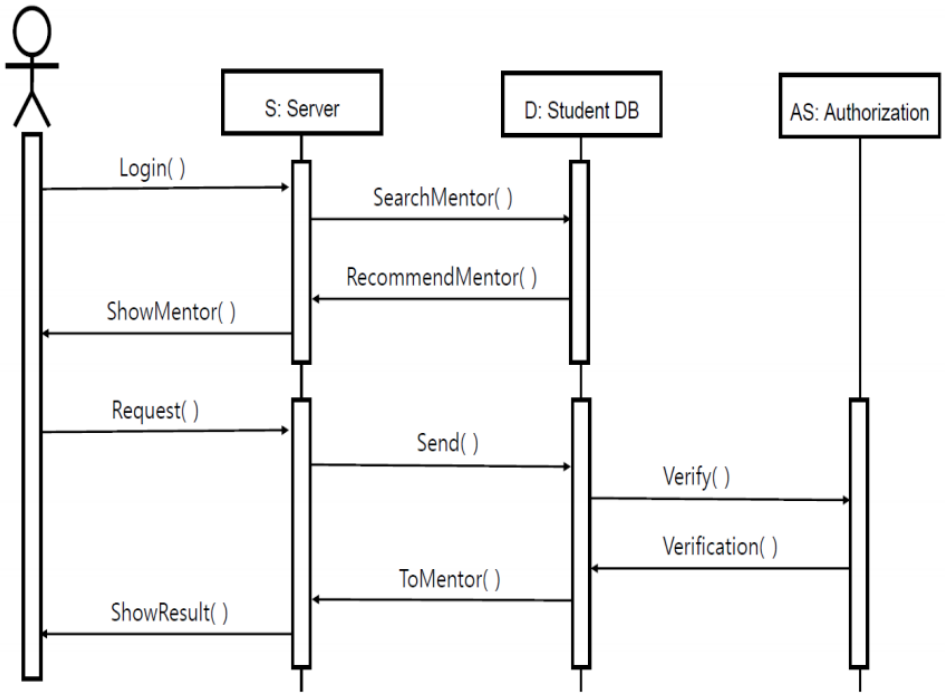


[Figure 1] Overall system architecture

3.2.1 Context Diagram

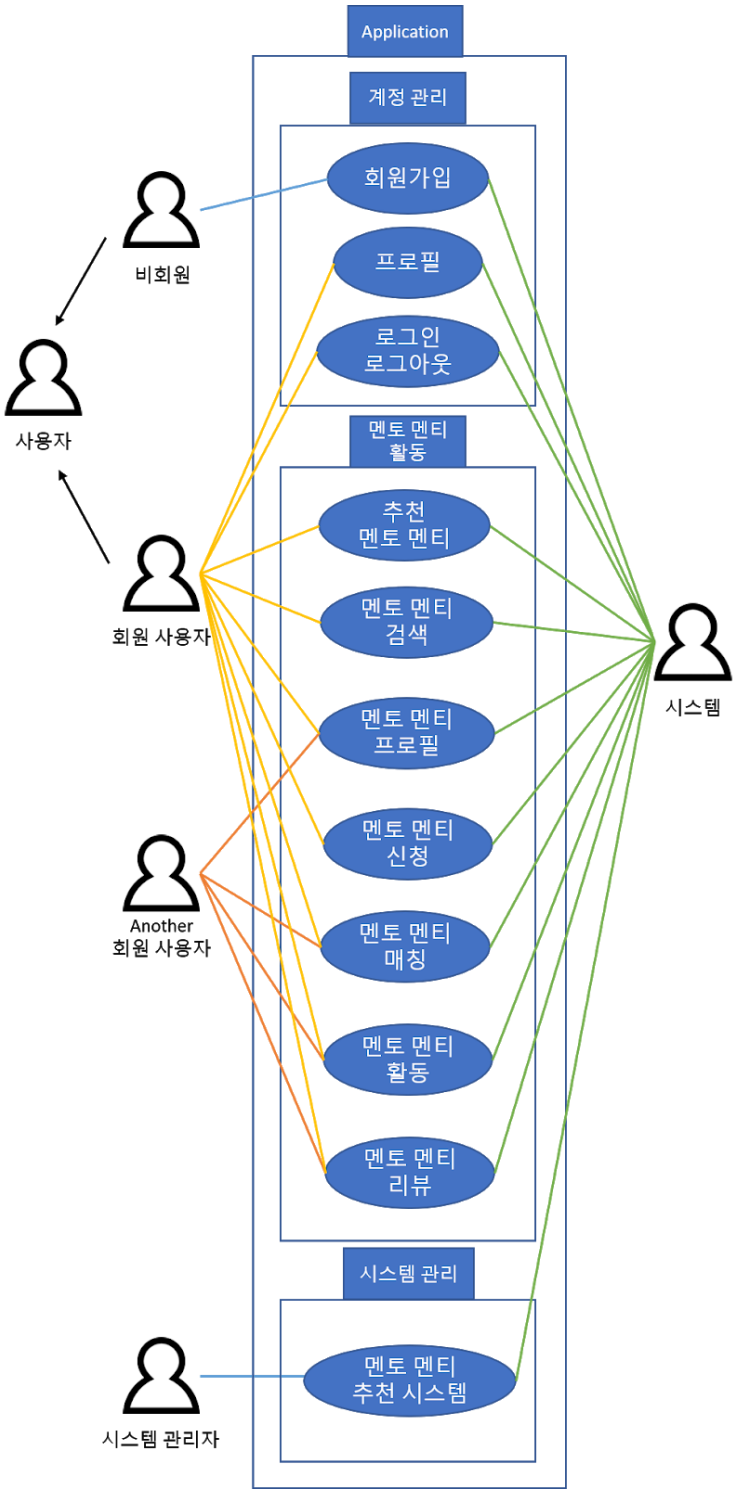
[Figure 2] Overall context diagram

3.2.2 Sequence Diagram



[Figure 3] Overall sequence diagram

3.2.2 Use Case Diagram



[Figure 4] Use case diagram

4. System Architecture – Frontend

4.1. Objectives

Chapter4는 애플리케이션 frontend 시스템의 아키텍처 구조와 시스템을 구성하는 여러 컴포넌트의 오브젝트, 속성, 기능과 더불어 컴포넌트 간 관계에 대한 기술입니다.

4.2. Subcomponents

4.2.1. 프로필

성균관대학교 ID Email, 비밀번호, 닉네임 등 사용자의 기본정보를 다룹니다. 또한 멘토/멘티 역할, 매칭 희망 분야, 소모임 관심 분야 등 선호정보를 함께 다룹니다. 사용자는 프로필을 애플리케이션 회원 등록 시 입력해야 하고, 필요시 수정 가능합니다.

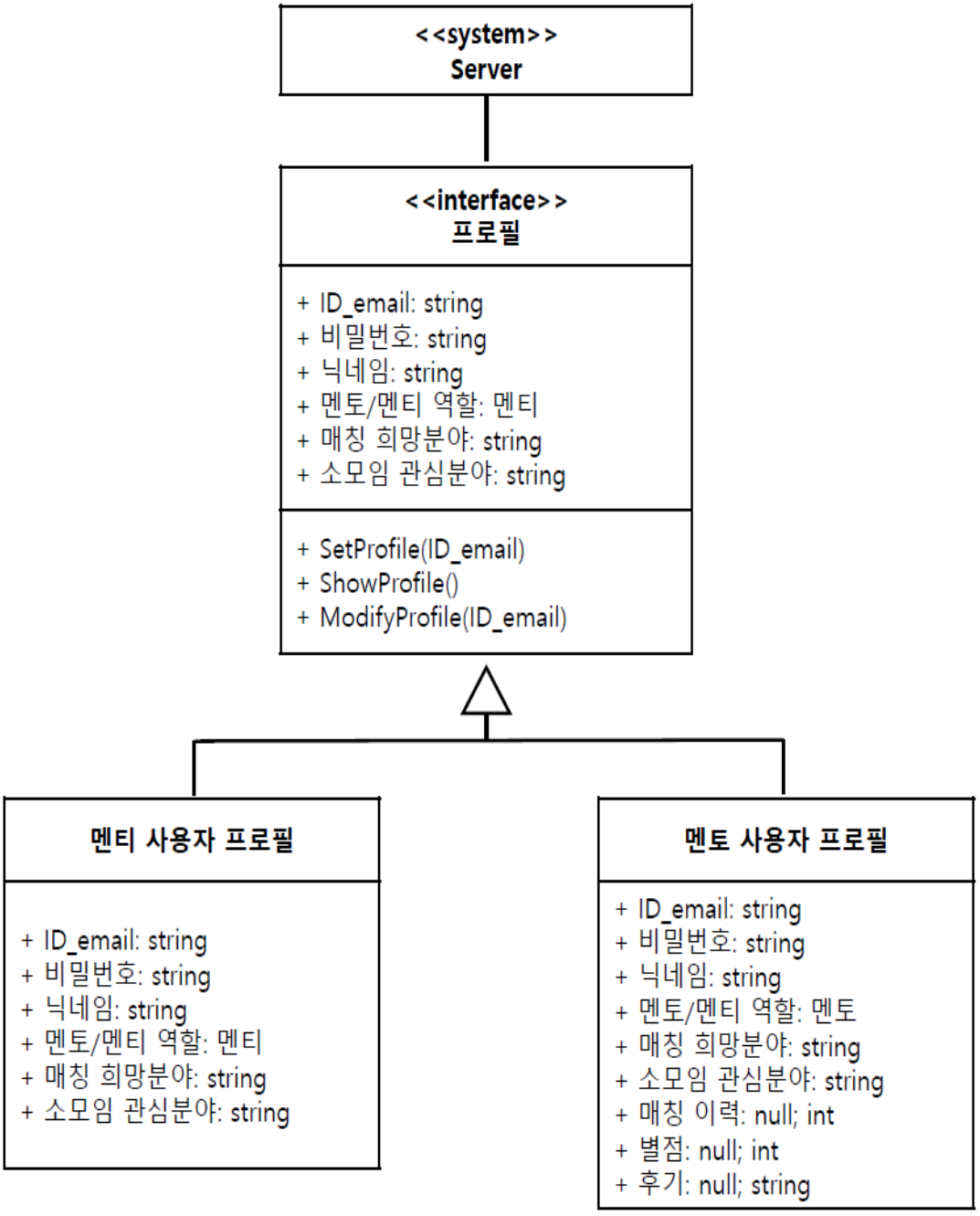
4.2.1.1. Attributes

- ID Email: 성균관대학교 웹메일 주소
- 비밀번호
- 닉네임
- 멘토/멘티 역할
- 매칭 희망 분야
- 소모임 관심 분야

4.2.1.2. Methods

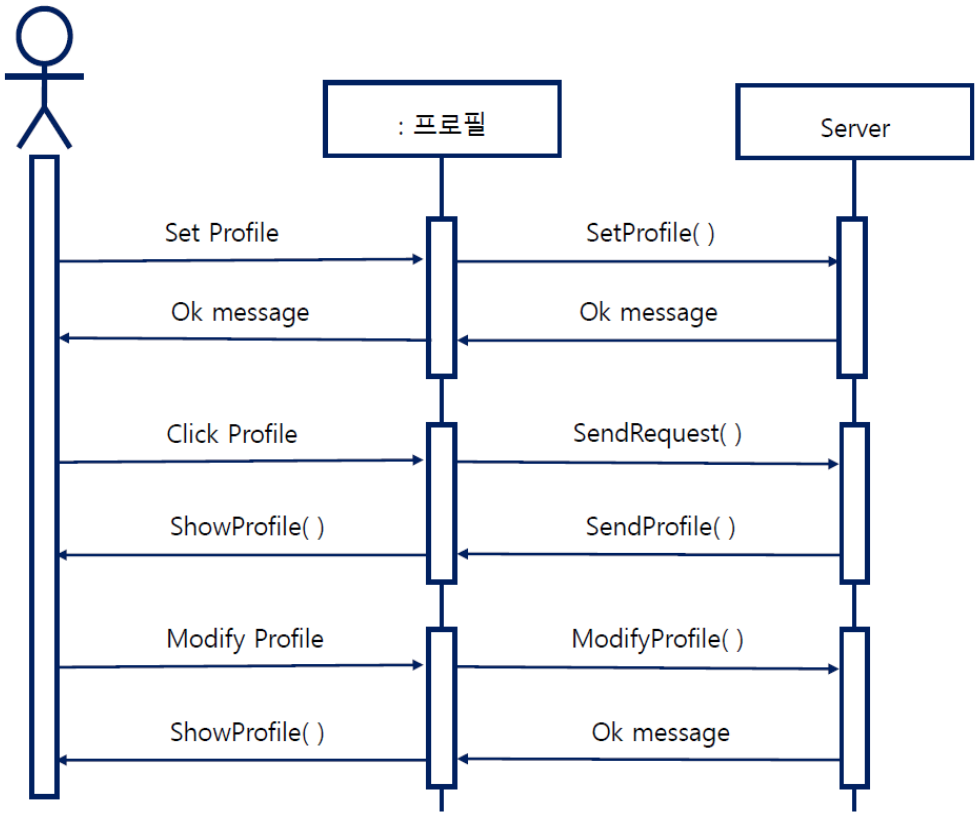
- SetProfile()
- ShowProfile()
- ModifyProfile()

4.2.1.3. Class Diagram



[Figure 5] Class diagram – 프로필

4.2.1.4. Sequence Diagram



[Figure 6] Sequence diagram – Profile

4.2.2. 멘토 검색

멘토와 매칭을 원하는 멘티 사용자는 검색 키워드를 입력하여 희망 분야에 멘토가 있는지 검색할 수 있습니다. 검색 키워드는 특정 멘토의 닉네임 또는 경제학, 파이썬 등 희망 분야의 명칭 모두 가능합니다.

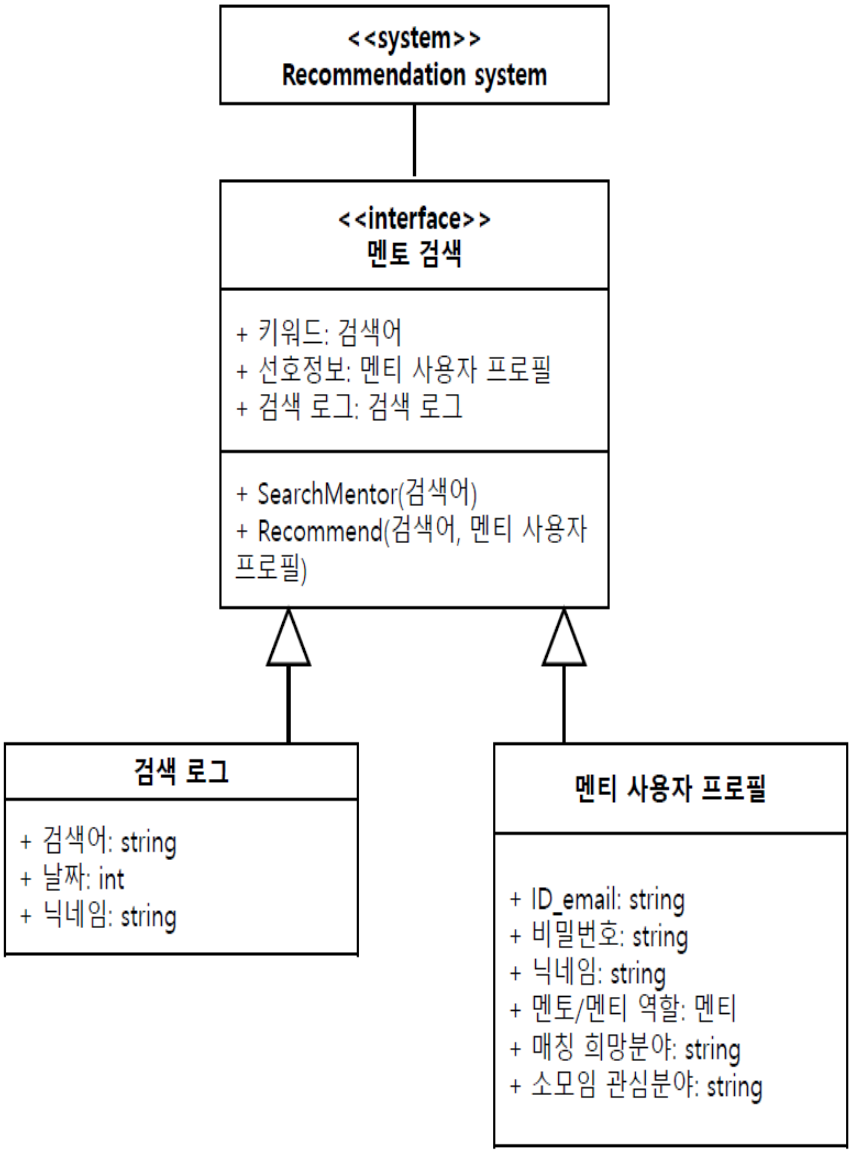
4.2.2.1. Attributes

- 검색어
- 멘토/멘티 희망 분야

4.2.2.2. Methods

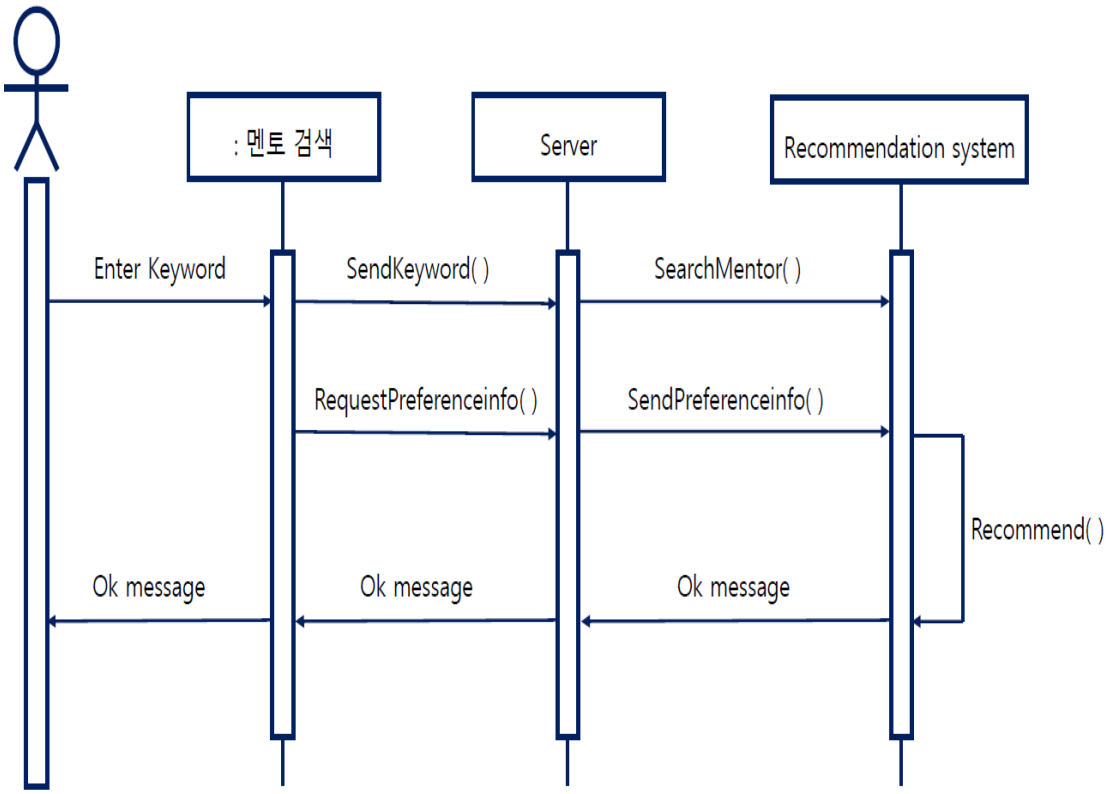
- SearchMentor()
- Recommend()

4.2.2.3. Class Diagram



[Figure 7] Class diagram – 멘토 검색

4.2.2.4. Sequence Diagram



[Figure 8] Sequence diagram – 멘토 검색

4.2.3. 소모임 검색

소모임 가입을 원하는 사용자는 검색 키워드를 입력하여 관심 분야 내 소모임이 있는지 검색할 수 있습니다. 검색 키워드는 특정 소모임 이름 또는 사진, 기타, 밴드 등 희망 분야의 명칭 모두 가능합니다.

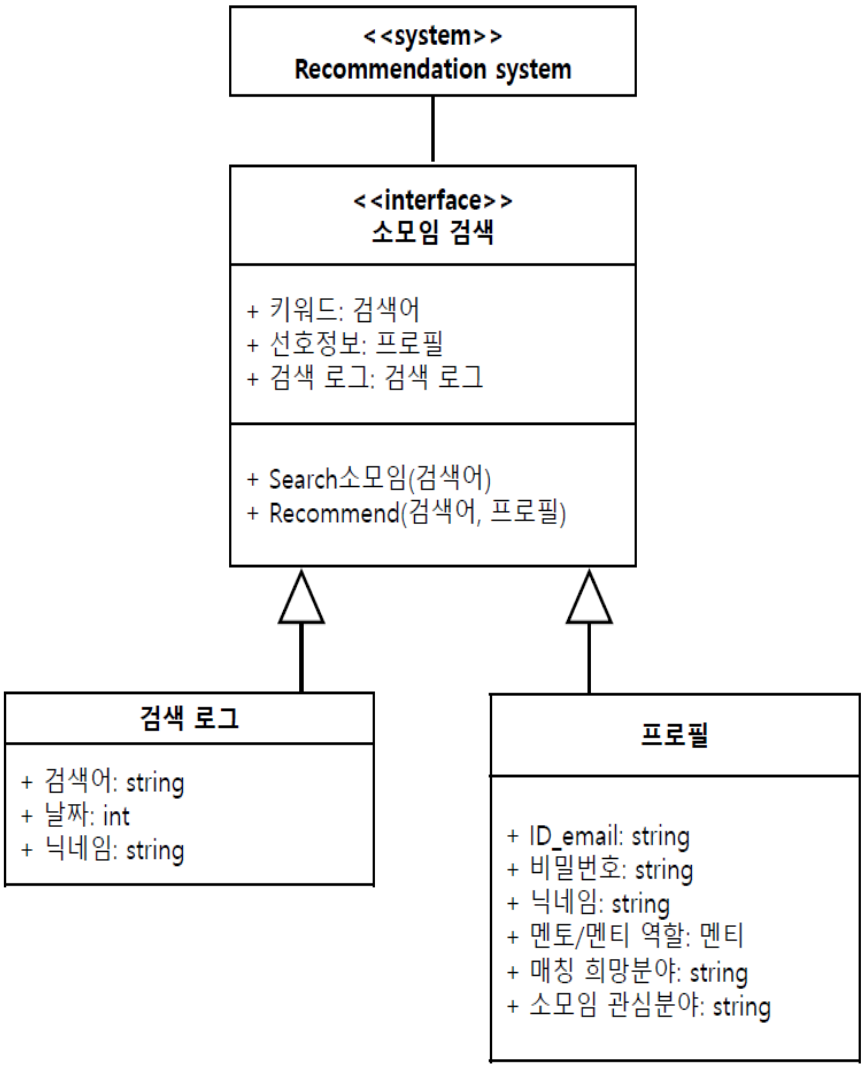
4.2.3.1. Attributes

- 검색어
- 소모임 관심 분야

4.2.3.2. Methods

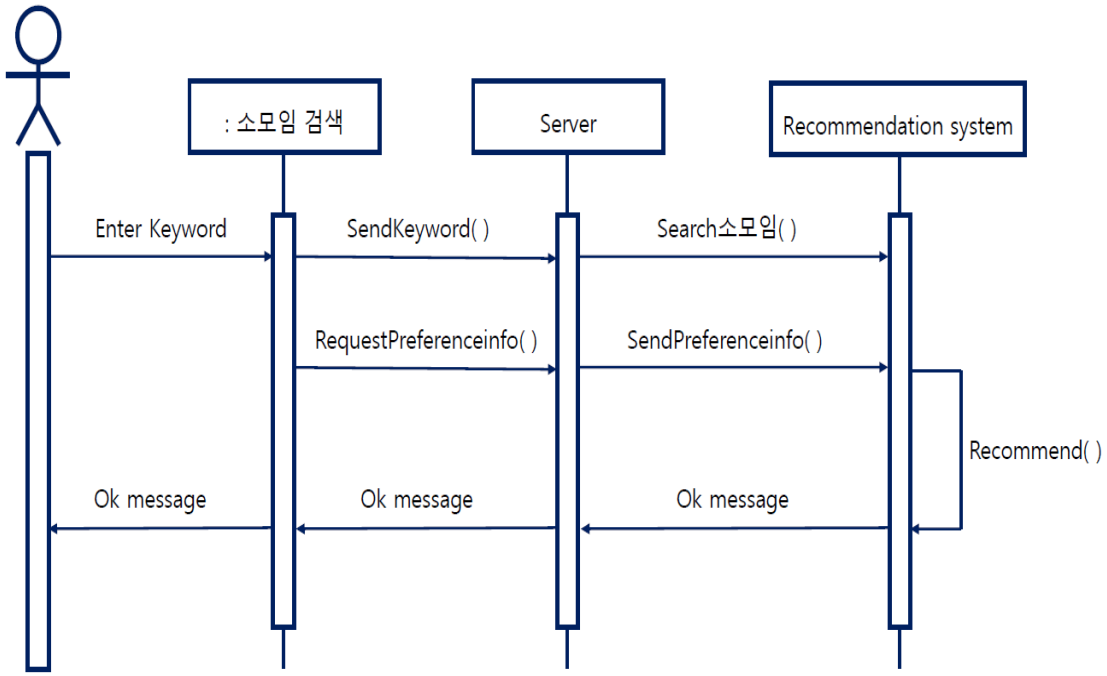
- Search소모임()
- Recommend()

4.2.3.3. Class Diagram



[Figure 9] Class diagram – 소모임 검색

4.2.3.4. Sequence Diagram



[Figure 10] Sequence diagram – 소모임 검색

4.2.4. 멘토 검색결과

사용자가 입력한 멘토 검색 키워드와 회원등록시 입력한 사용자의 멘토-멘티 매칭 희망 분야를 추천 시스템에 전달하여 추천 시스템이 제시하는 멘토 추천 결과 리스트를 확인할 수 있습니다.

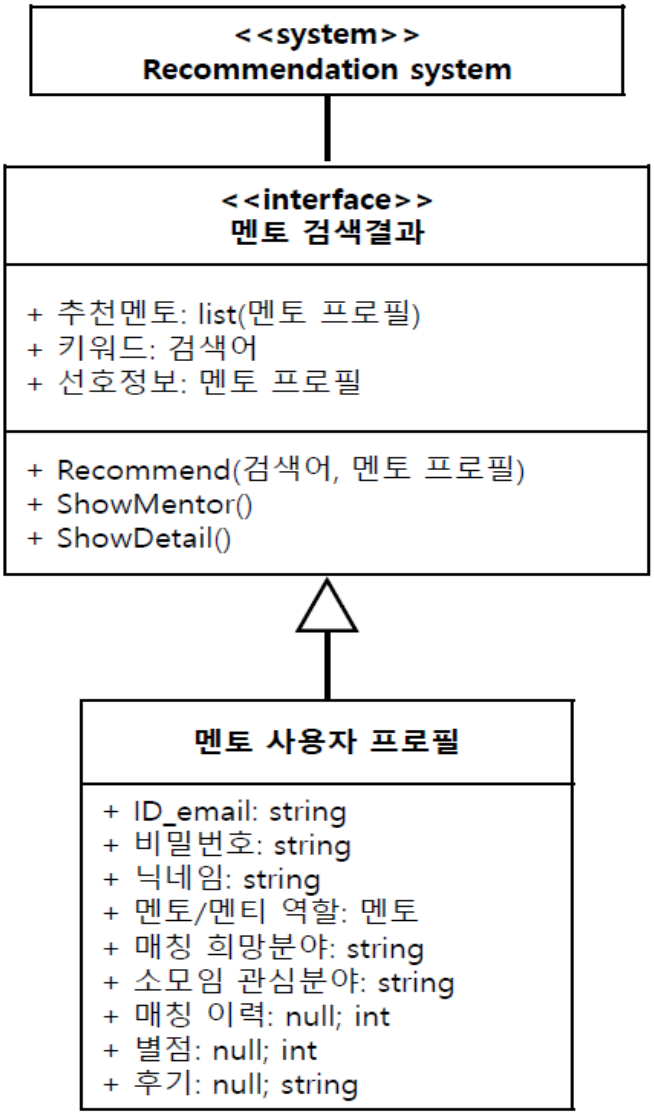
4.2.4.1. Attributes

- 추천멘토
- 키워드
- 선호정보

4.2.4.2. Methods

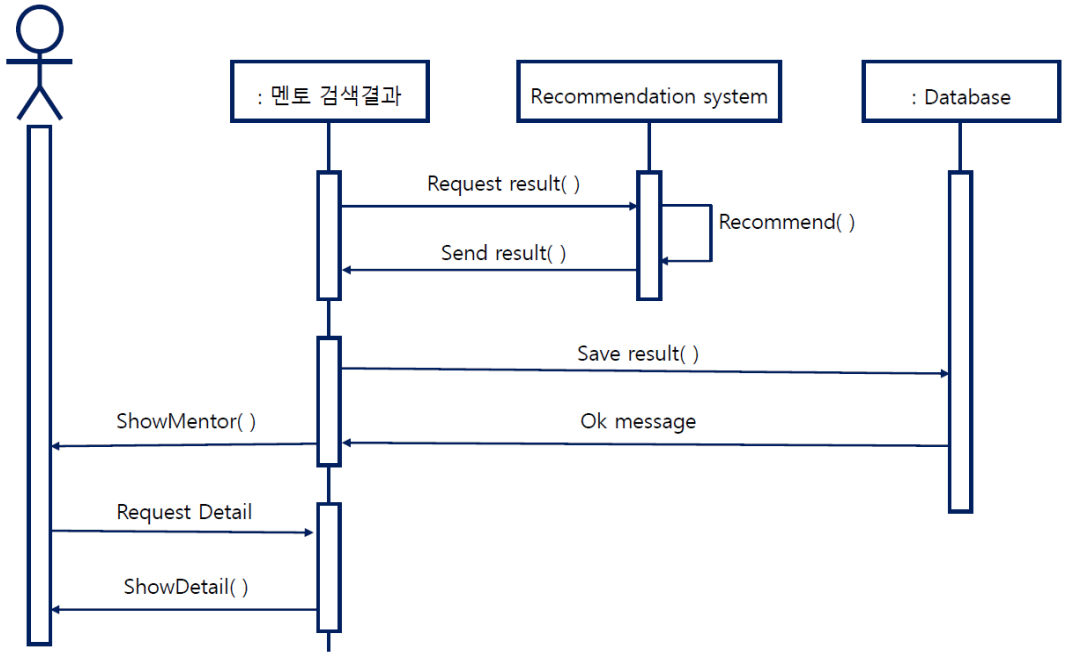
- Recommend()
- ShowMentor()
- ShowDetail()

4.2.4.3. Class Diagram



[Figure 11] Class diagram – 멘토 검색결과

4.2.4.4. Sequence Diagram



[Figure 12] Sequence diagram – 멘토 검색결과

4.2.5. 소모임 검색 결과

사용자가 입력한 소모임 검색 키워드와 회원등록시 입력한 사용자의 소모임 관심 분야를 추천 시스템에 전달하여 추천 시스템이 제시하는 소모임 추천 결과 리스트를 확인할 수 있습니다.

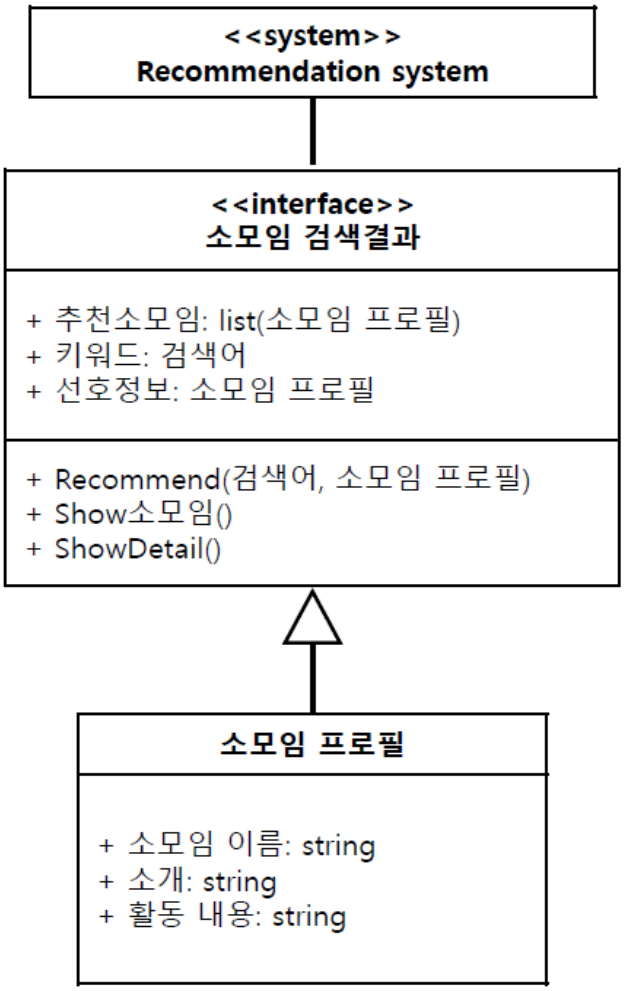
4.2.5.1. Attributes

- 소모임 이름
- 소개
- 활동 내용

4.2.5.2. Methods

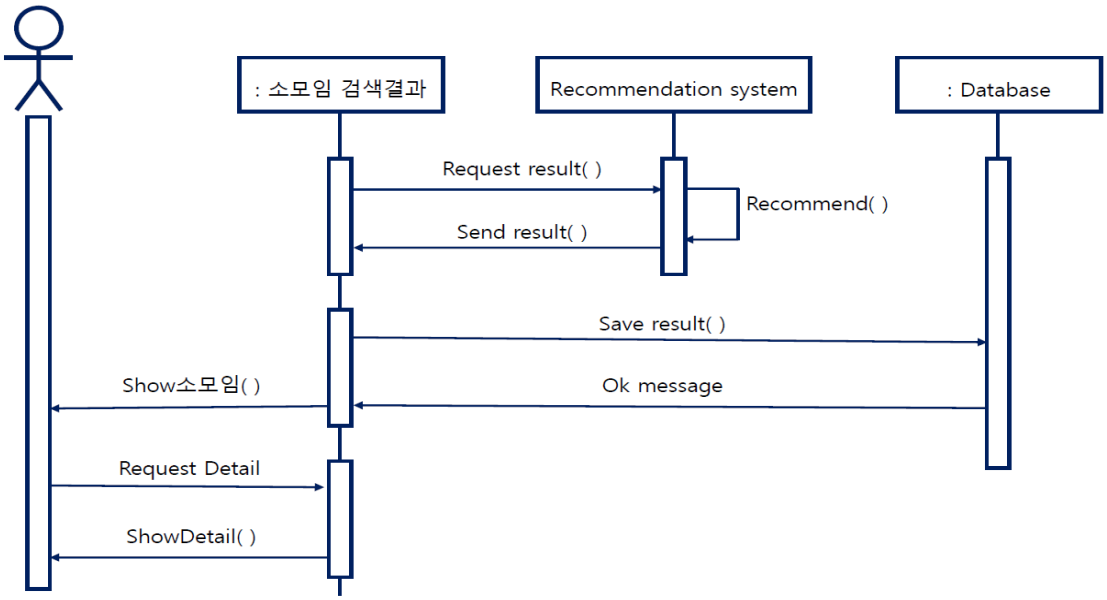
- Recommend()
- Show소모임()
- ShowDetail()

4.2.5.3. Class Diagram



[Figure 13] Class diagram – 소모임 검색결과

4.2.5.4. Sequence Diagram



[Figure 14] Sequence diagram – 소모임 검색결과

4.2.6. 매칭정보

매칭 요청/조회 페이지에서 멘티 사용자는 원하는 멘토 사용자에게 매칭 요청을 할 수 있습니다. 요청을 받은 멘토 사용자는 매칭 요청/조회에서 신청을 수락 또는 거절 모두 가능합니다. 요청을 받은 멘토가 수락을 클릭하면 매칭이 성사되고, 거절을 클릭하면 매칭이 결렬됩니다. 매칭 요청/조회에서 사용자는 매칭 현황을 확인할 수 있습니다.

4.2.6.1. Attributes

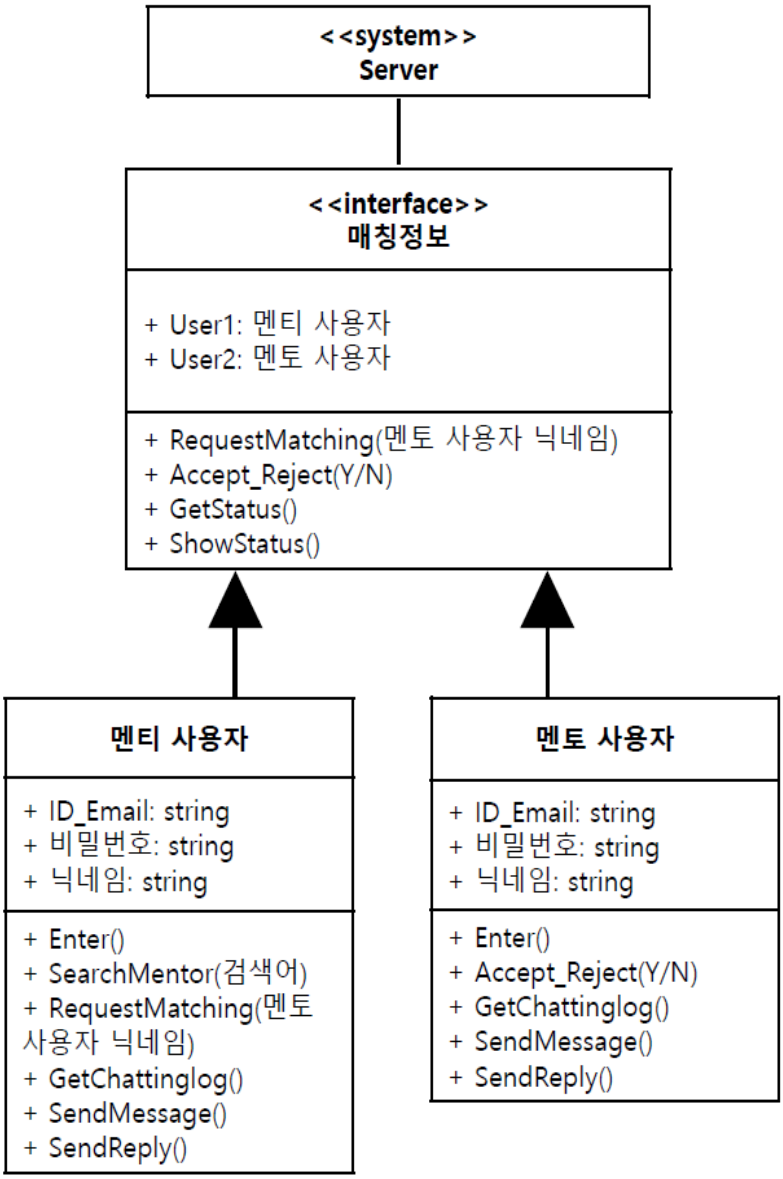
- User1: 멘티 사용자
- User2: 멘토 사용자

4.2.6.2. Methods

- RequestMatching()
- Accept_Reject()
- GetStatus()

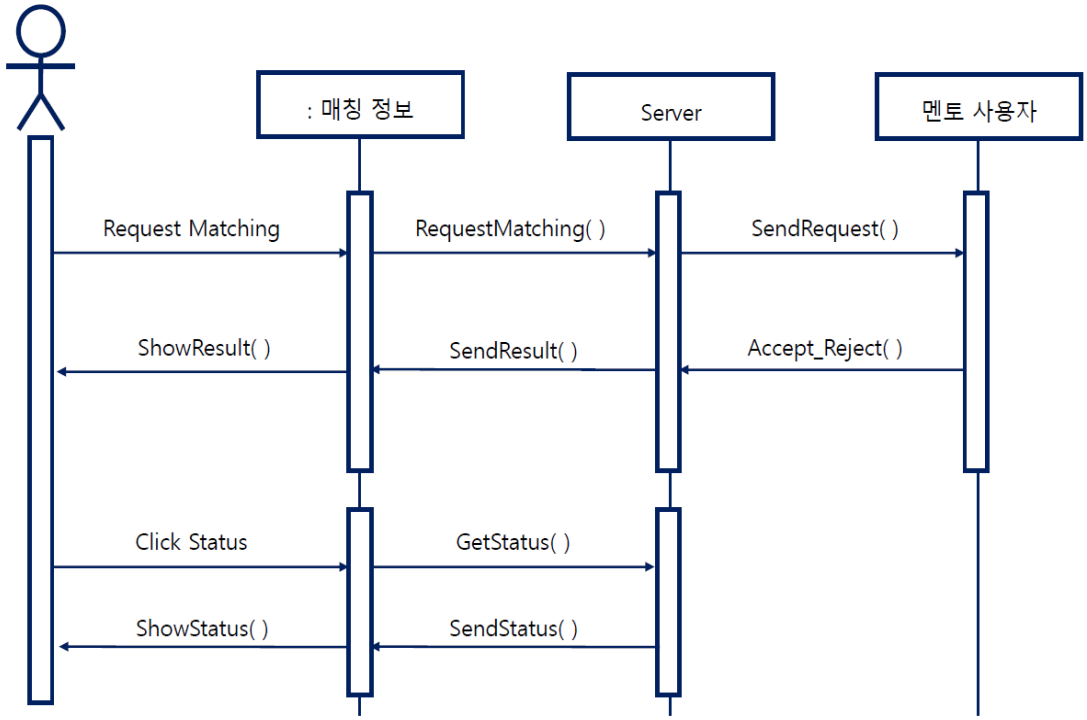
- ShowStatus()

4.2.6.3. Class Diagram



[Figure 15] Class diagram – 매칭정보

4.2.6.4. Sequence Diagram



[Figure 16] Sequence diagram – 매칭정보

4.2.7. 채팅

멘토-멘티 매칭이 성사된 사용자는 채팅방 페이지에서 대화할 수 있습니다. 사용자의 모든 채팅 로그는 일정 주기로 데이터베이스에 임시 저장됩니다.

4.2.7.1. Attributes

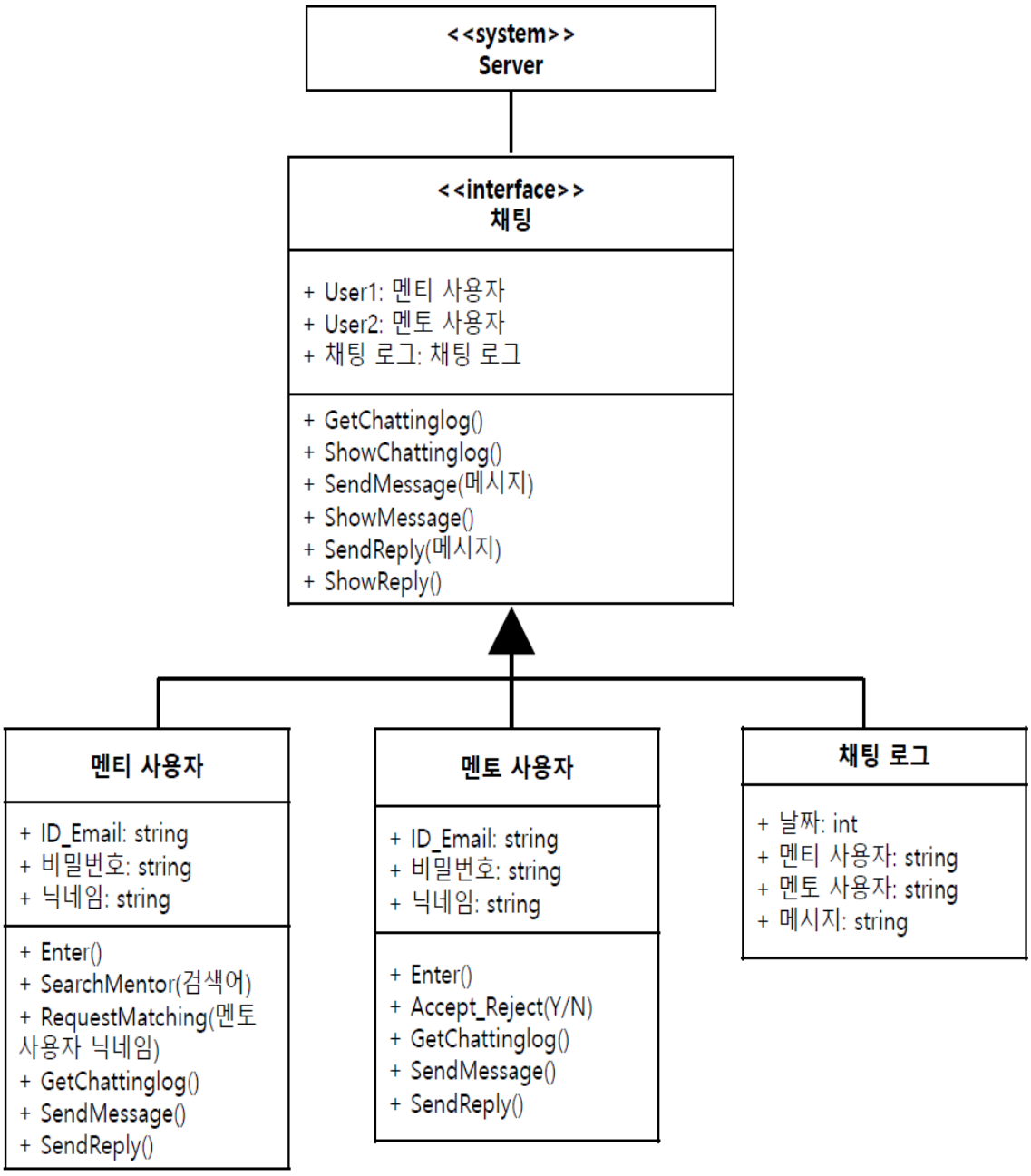
- User1: 멘티 사용자
- User2: 멘토 사용자
- 채팅 로그

4.2.7.2. Methods

- GetChattinglog()
- ShowChattinglog()

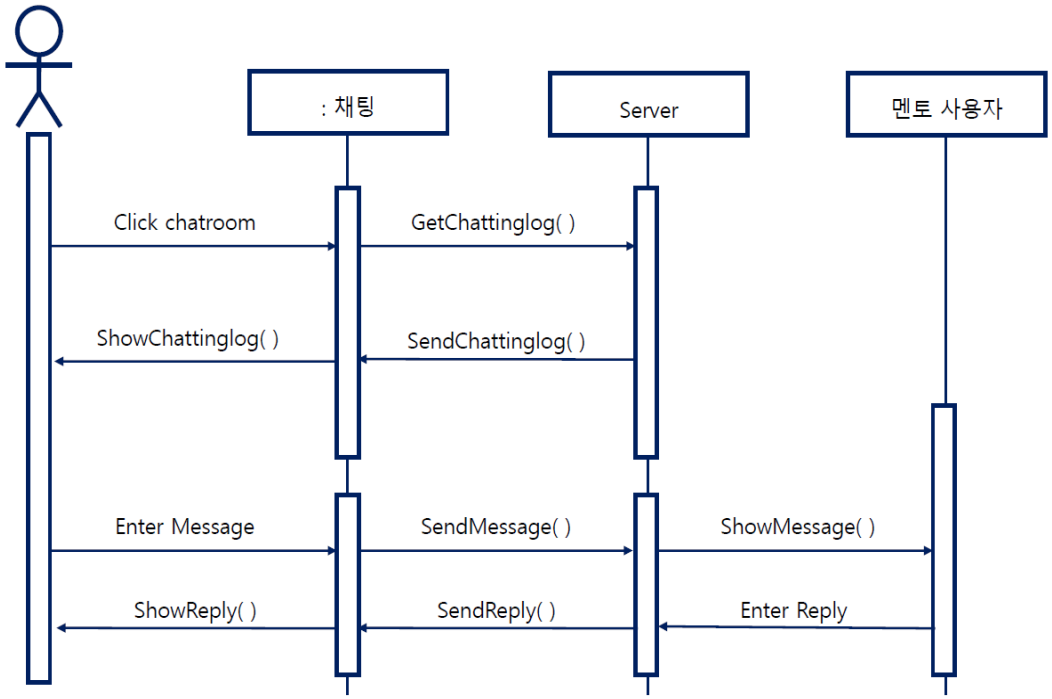
- SendMessage()
- ShowMessage()
- SendReply()
- ShowReply()

4.2.7.3. Class Diagram



[Figure 17] Sequence diagram – 채팅

4.2.7.4. Sequence Diagram



[Figure 18] Sequence diagram – 채팅

4.2.8. 리뷰

매칭 시작일 기준 30일 이후부터 멘티 사용자는 멘토 사용자에 대한 리뷰를 작성할 수 있습니다. 멘티 사용자가 작성한 리뷰 후기 내용은 멘토 사용자 프로필에 저장되고 리뷰 별점은 멘토 사용자 프로필에 실시간으로 반영됩니다.

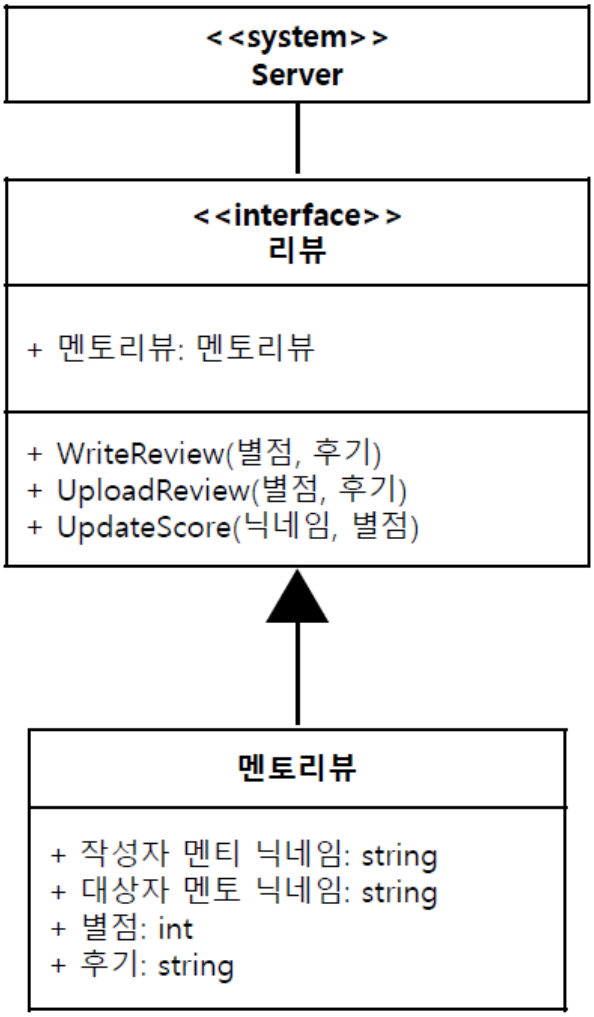
4.2.8.1. Attributes

- 작성자 멘티 닉네임
- 대상자 멘토 닉네임
- 별점
- 후기

4.2.8.2. Methods

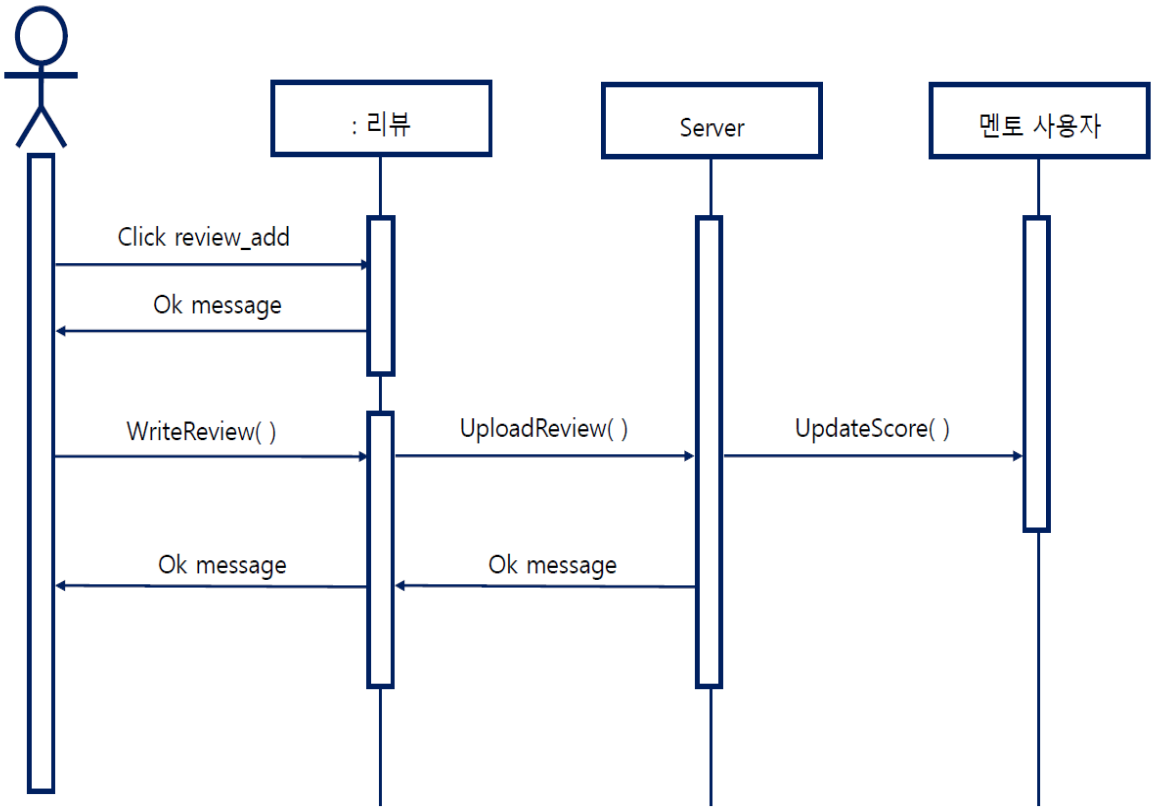
- WriteReview()
- UploadReview()
- UpdateScore()

4.2.8.3. Class Diagram



[Figure 19] Sequence diagram – 리뷰

4.2.8.4. Sequence Diagram



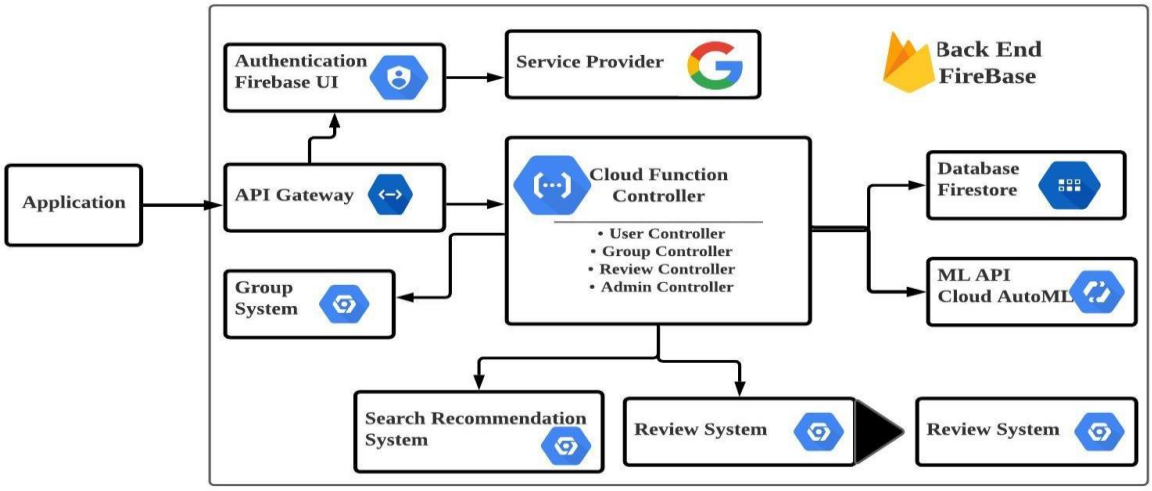
[Figure 20] Sequence diagram – 리뷰

5. System Architecture – Backend

5.1. Objective

이 장에서는 DB 및 API Cloud 포함 백엔드 시스템의 구조를 설명을 합니다.

5.2. Overall Architecture

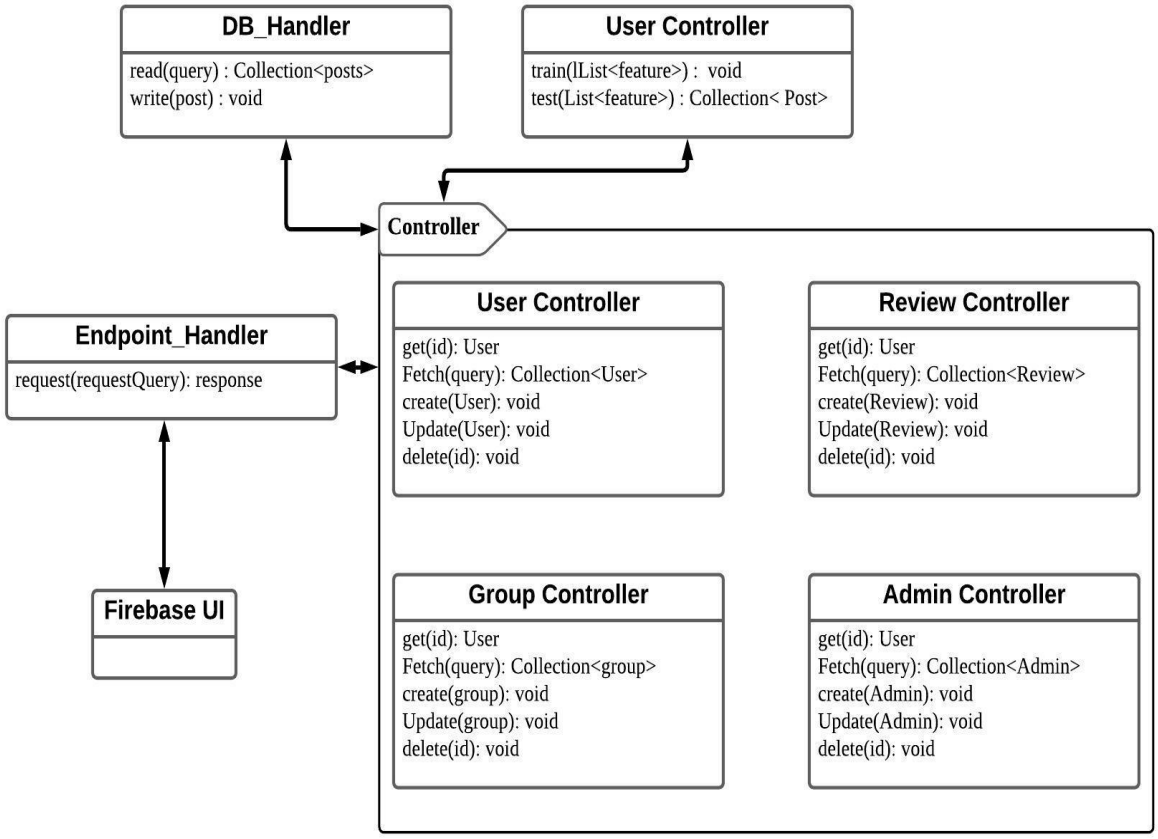


[Figure 21] Overall architecture

시스템의 전체적인 구조는 위와 같다. API 게이트웨이(Request Handler)는 프런트엔드로부터 요청을 수신하여 적절한 클라우드 기능(Controller / Manager)으로 배포합니다. 이 과정에서 인증 등 외부 API를 사용하는 기능이 처리된다. 해당 컨트롤러로 전송된 기타 요청(내부 처리)입니다. 컨트롤러는 클라우드(Database / Cloud AutoML)와 상호 작용하여 요청을 처리합니다.

5.3. Subcomponents

5.3.1. Cloud Function



[Figure 22] Class diagram – Cloud functions

5.3.1.1. Endpoint Handler

API 게이트웨이 - 프런트 엔드에서 적절한 컨트롤러 또는 API로 요청을 배포합니다.

5.3.1.2. FirebaseUI

FirebaseUI는 클래스를 통한 인증 구현 입니다.

5.3.1.3. DB_Handler Class

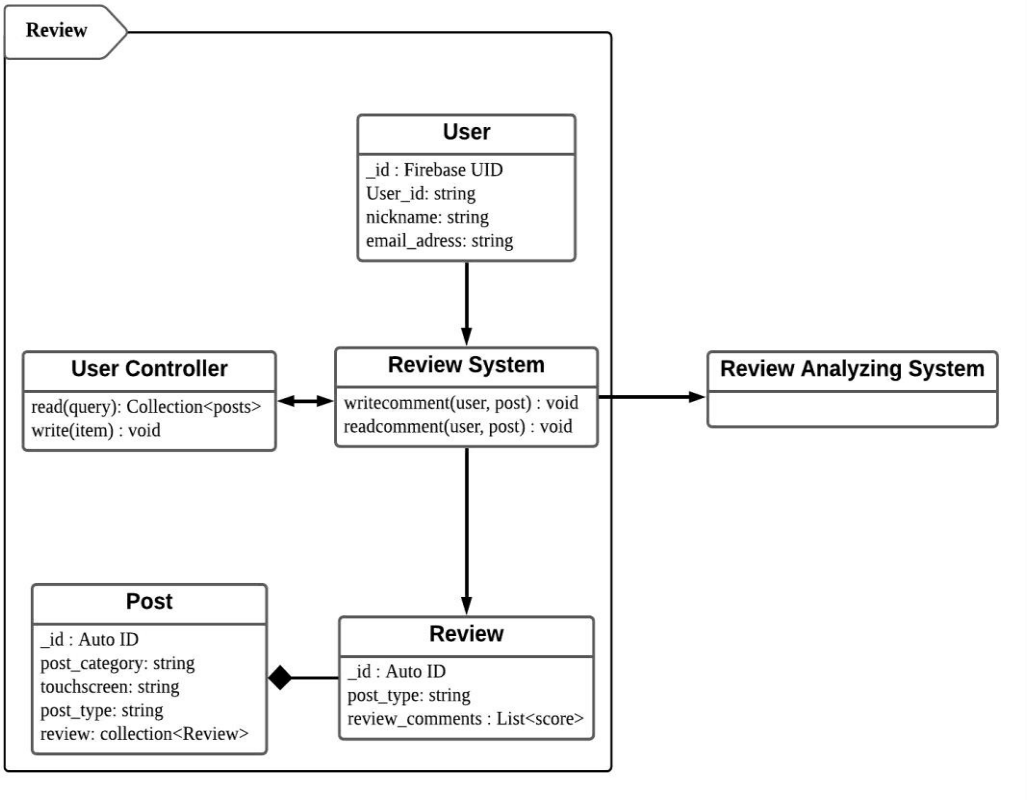
DB와 통신하는 인터페이스입니다. 특정 게시물 또는 그룹을 DB에서 가져오고 검토 개체가 게시물 또는 그룹의 검토 컬렉션에 추가되어 저장됩니다.

5.3.1.4. Model_Handler

그것은 기계 학습을 위한 인터페이스이다. 텐서 플로우 모델은 Firebase ML 키트의 사용자 정의 모델을 통해 사용이 가능 합니다.

5.3.2 Review System

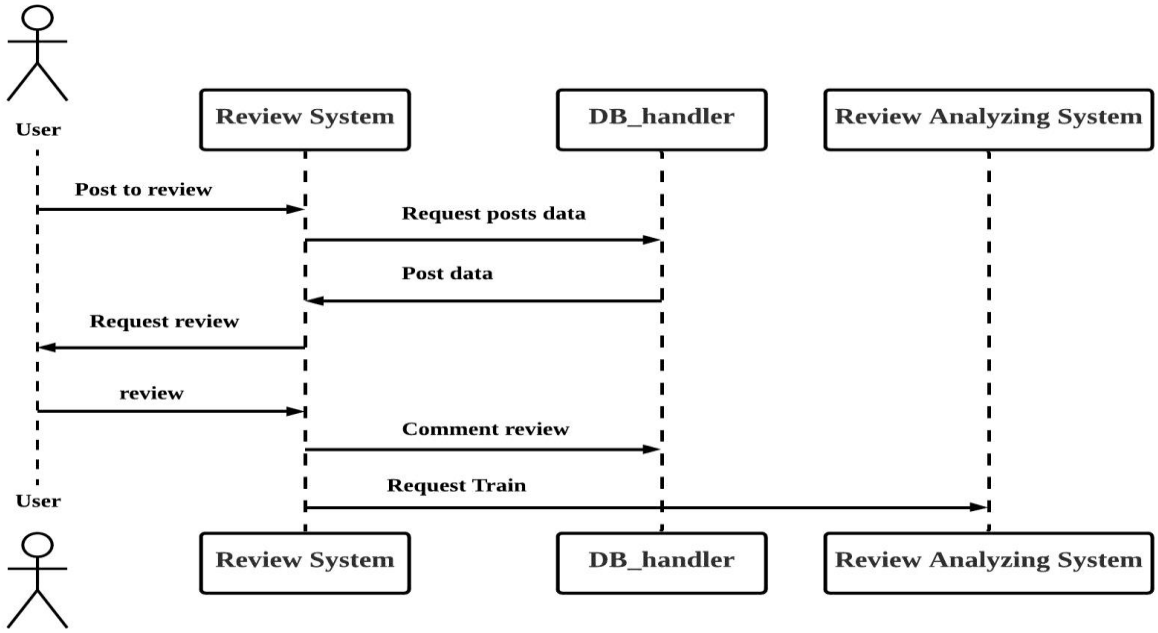
5.3.2.1. Class Diagram



[Figure 23] Class diagram – Review system

- Class Description
  - ✓ **Review System Class:** 검토 개체를 제어하기 위한 인터페이스입니다. 사용자가 시스템에 리뷰 쓰기를 요청할 때, 시스템은 사용자의 리뷰를 포스트 또는 그룹 리뷰 모음에 추가합니다. 사용자가 리뷰를 보고자 할 때 포스트 또는 그룹 리뷰 컬렉션에서 가져온다.

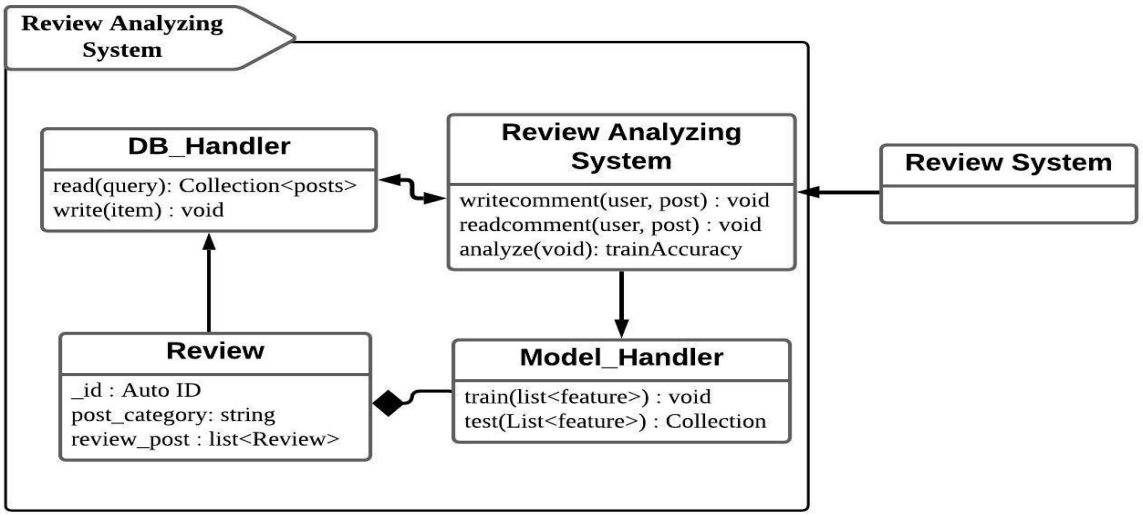
5.3.2.2. Sequence Diagram



[Figure 24] Sequence diagram – Review system

5.3.3. Review Analyzing System

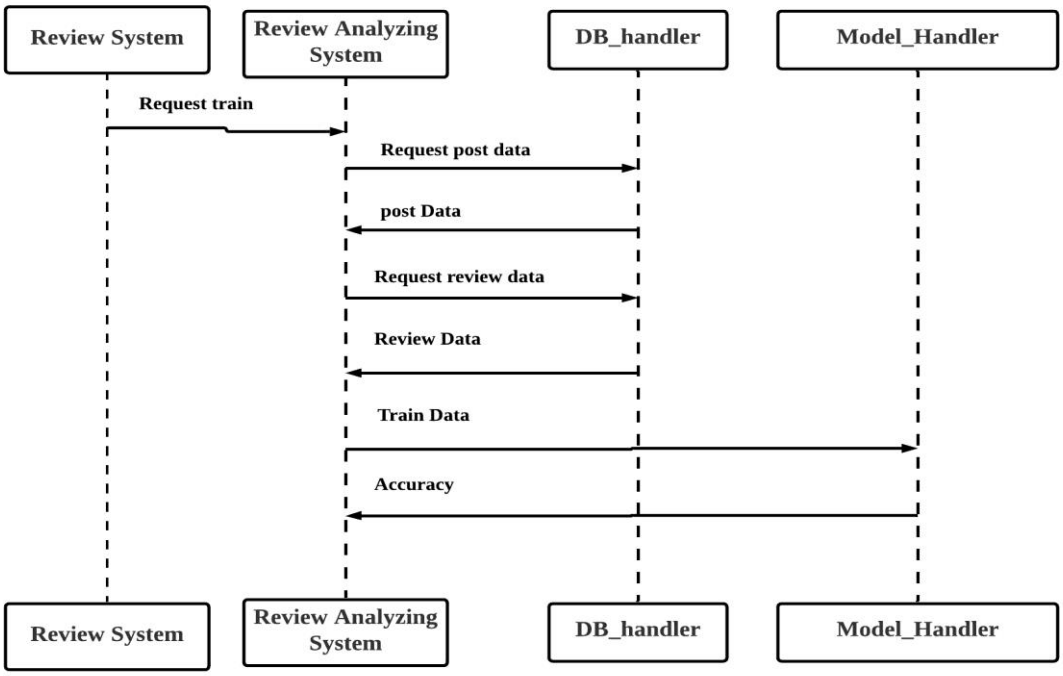
5.3.3.1. Class Diagram



[Figure 25] Class diagram – Review analyzing system

- Class Description
  - ✓ **Review Analyzing Class:** 리뷰를 분석하는 인터페이스입니다. 새 검토가 추가되면 호출되며 model_handler 에 교육을 요청합니다.

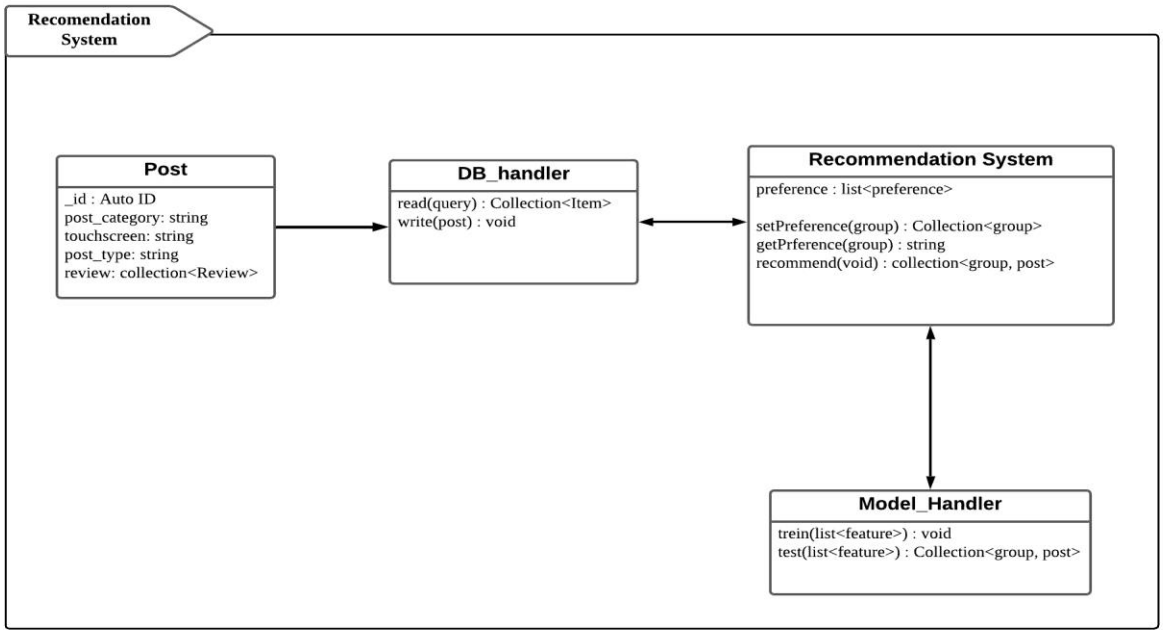
5.3.3.2. Sequence Diagram



[Figure 26] Sequence diagram – Review analyzing system

5.3.4. Recommendation System

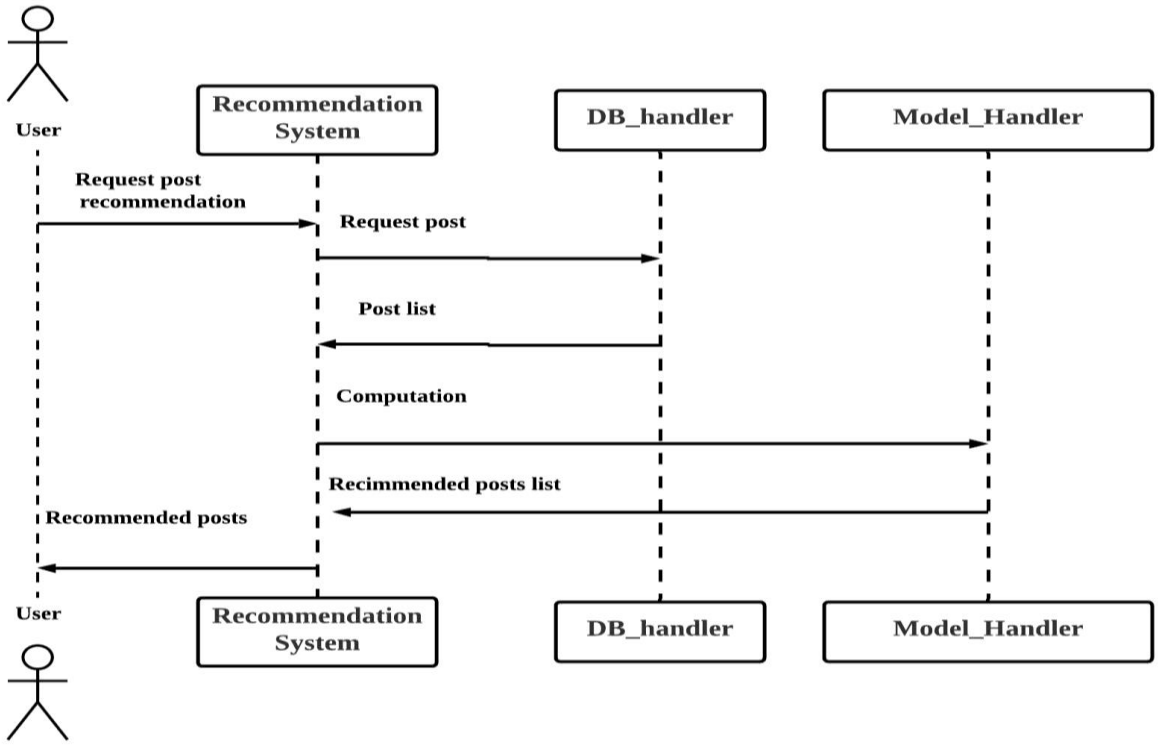
5.3.4.1. Class Diagram



[Figure 27] Class diagram - Recommendation system

- Class Description
  - ✓ **Recommendation System Class:** 그룹 또는 포스트 권장 사항에 대한 인터페이스입니다. 사용자가 조건을 입력하면 모델 처리기를 통해 상관 관계가 높은 그룹 또는 게시물이 권장됩니다.

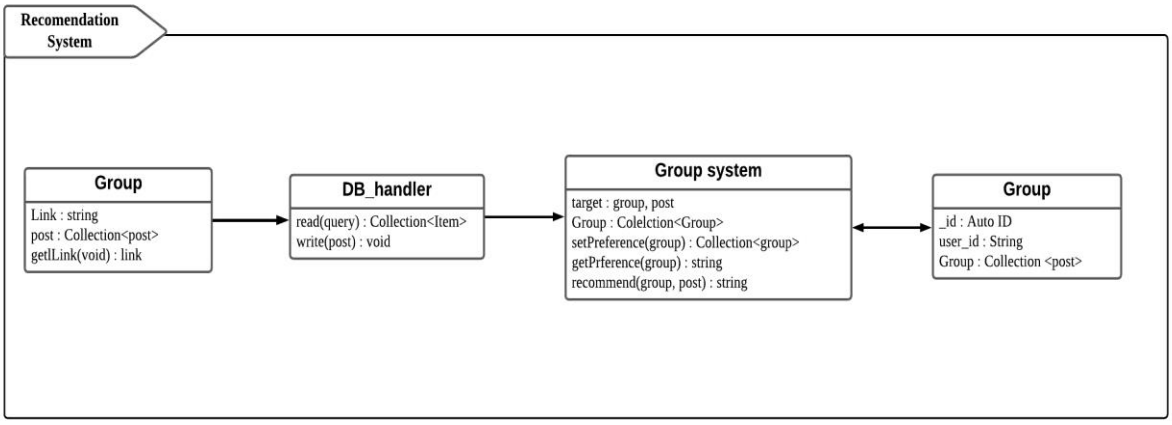
5.3.4.2. Sequence Diagram



[Figure 28] Sequence diagram – Recommendation system

5.3.5. Group System

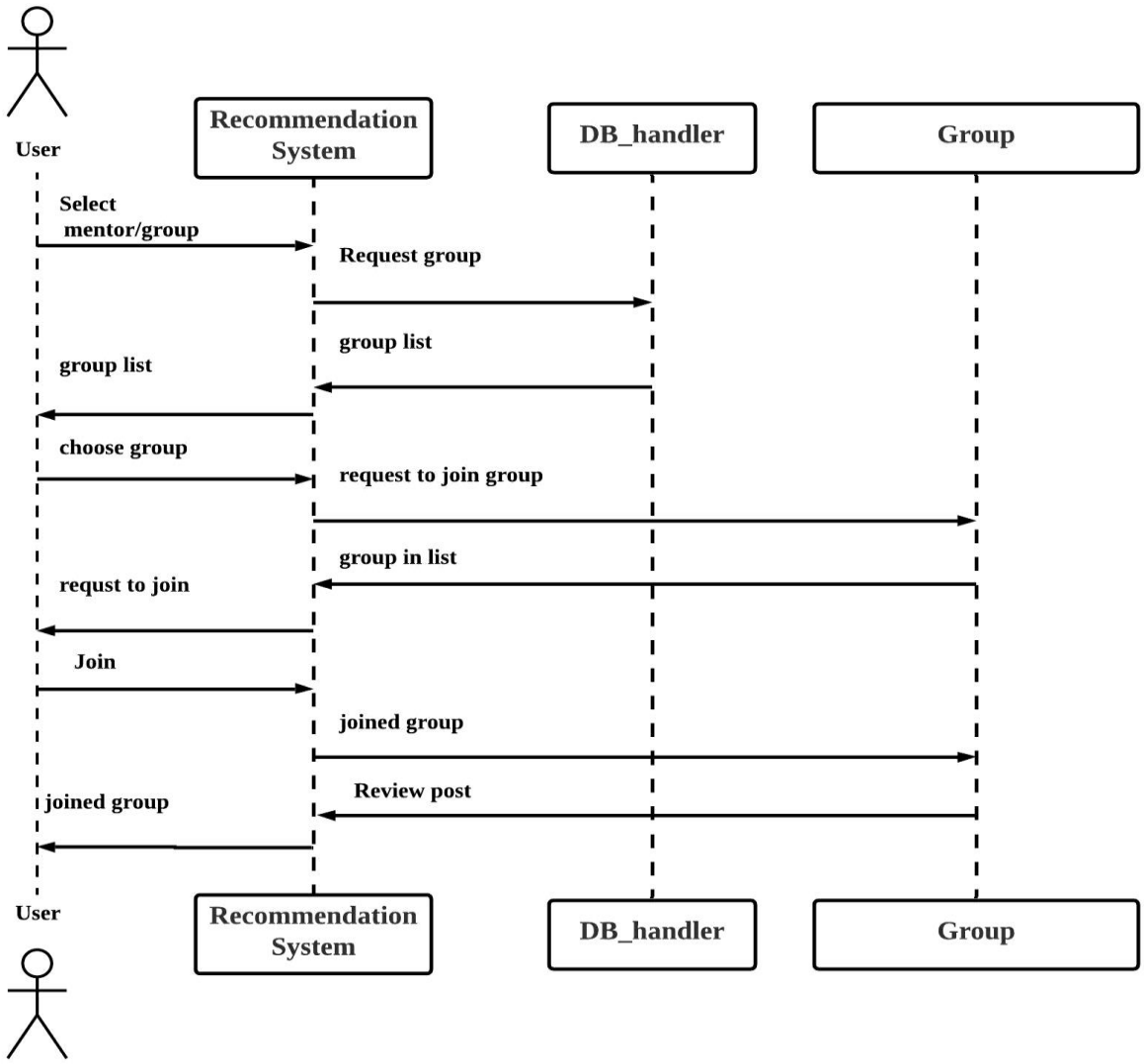
5.3.5.1. Class Diagram



[Figure 29] Class diagram – Group system

- Class Description
  - ✓ **Group System Class:** 그룹에 가입하기 위한 인터페이스입니다. 사용자가 원하는 그룹을 입력하면 그룹에 연결되고 사용자가 그룹을 검토할 수 있습니다.
  - ✓ **Group Class:** 학생들을 위한 그룹 활동.

5.3.5.2 Sequence Diagram



[Figure 30] Sequence Diagram – Group system

6. Protocol Design

6.1 Objectives

6장에서는 각 subsystem 사이에서의 상호작용을 위해서 어떤 protocol structure가 사용되었는지를 기술합니다.

6.2 JSON

JSON 형식은 key-value의 쌍으로 이루어진 데이터 오브젝트를 전달하기 위해 사용하는 개방형 표준 파일 포멧으로 컴퓨터 뿐만이 아니라 사람이 읽을 수도 있는 형식입니다. JSON은 자바스크립트의 구문 형식을 따르지만 데이터를 생성/분석을 위해서 다른 언어를 사용할 수도 있습니다. Firebase의 실시간 데이터베이스 데이터는 JSON 객체로 저장됩니다. SQL 데이터베이스와 달리 테이블이나 레코드가 없으며, JSON 트리에 추가된 데이터는 연결된 키를 갖는 기존 JSON 구조의 노드가 됩니다.

6.3 HTTP Method

클라이언트가 서버로 요청을 보내는 방법에는 크게 GET과 POST 두 가지 방식이 있습니다.

6.3.1 GET

GET method는 어떤 정보를 가져와서 조회하기 위해서 사용하는 방식입니다. GET method를 실행하기 위해서는 URL에 변수를 포함시켜 요청하거나 데이터를 Headerd에 포함시켜 요청합니다.

6.3.2 POST

POST method는 데이터를 서버로 제출하여 추가 또는 수정하기 위해 사용합니다. POST방식은 BODY에 데이터를 포함시켜서 전송합니다.

6.4 Authentication

6.4.1 Register

● Request

[Table1] Table of register request

Attribute	Detail	
Protocol	HTTP	
Request Body	Email	사용자의 학교 이메일
	Request Register Link	인증을 위한 Hyper Link

● Response

[Table2] Table of register response

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 400 (Bad Request, overlap)	
Success Response Body	Access Token	접근을 위한 token
Failure Response Body	Message	Fail message

6.5 Profile

6.5.1 Set Profile

● Request

[Table3] Table of set profile request

Attribute	Detail	
Method	POST	
URI	/user/:id/profile	
Parameter	User	사용자가 기입한 정보
Header	Authorization	User authentication

● Response

[Table4] Table of set profile response

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 403 (Forbidden)	
Success Response Body	Message	Success message
Failure Response Body	Message	Fail message

6.5.2 Show Profile

● Request

[Table5] Table of show profile request

Attribute	Detail	
Method	GET	
URL	/user/:id/profile	
Parameter	User	N/A
Header	Authorization	User authentication

● Response

[Table6] Table of show profile response

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 404 (Not Found)	
Success Response Body	User	User objects
Failure Response Body	Message	Empty

6.6 Search Mentor

6.6.1 Recommend Mentor

● Request

[Table7] Table of mentor recommendation request

Attribute	Detail	
Method	GET	
URI	/user/recommendation	
Parameter	User	프로필에 기입된 사용자 정보
Header	Authorization	User authentication

● Response

[Table8] Table of mentor recommendation response

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 404 (Not Found)	
Success Response Body	Recommend Result	사용자에게 맞는 멘토들의 list
Failure Response Body	Message	Fail Message

6.6.2 Search Mentor

● Request

[Table9] Table of search Mentor request

Attribute	Detail	
Method	GET	
URI	/cloud/mentor	
Parameter	User	검색 키워드
Header	Authorization	User authentication

● Response

[Table10] Table of search Mentor response

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 404 (Not Found)	
Success Response Body	Search Result	검색 키워드에 맞는 멘토 검색 결과
Failure Response Body	Message	Fail Message

6.7 Search 소모임

● Request

[Table11] Table of search 소모임 request

Attribute	Detail	
Method	GET	
URI	/cloud/club	
Parameter	User	검색 키워드
Header	Authorization	User authentication

● Response

[Table12] Table of search Mentor response

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 404 (Not Found)	
Success Response Body	Search Result	검색 키워드에 맞는 소모임 검색 결과
Failure Response Body	Message	Fail Message

7. Database Design

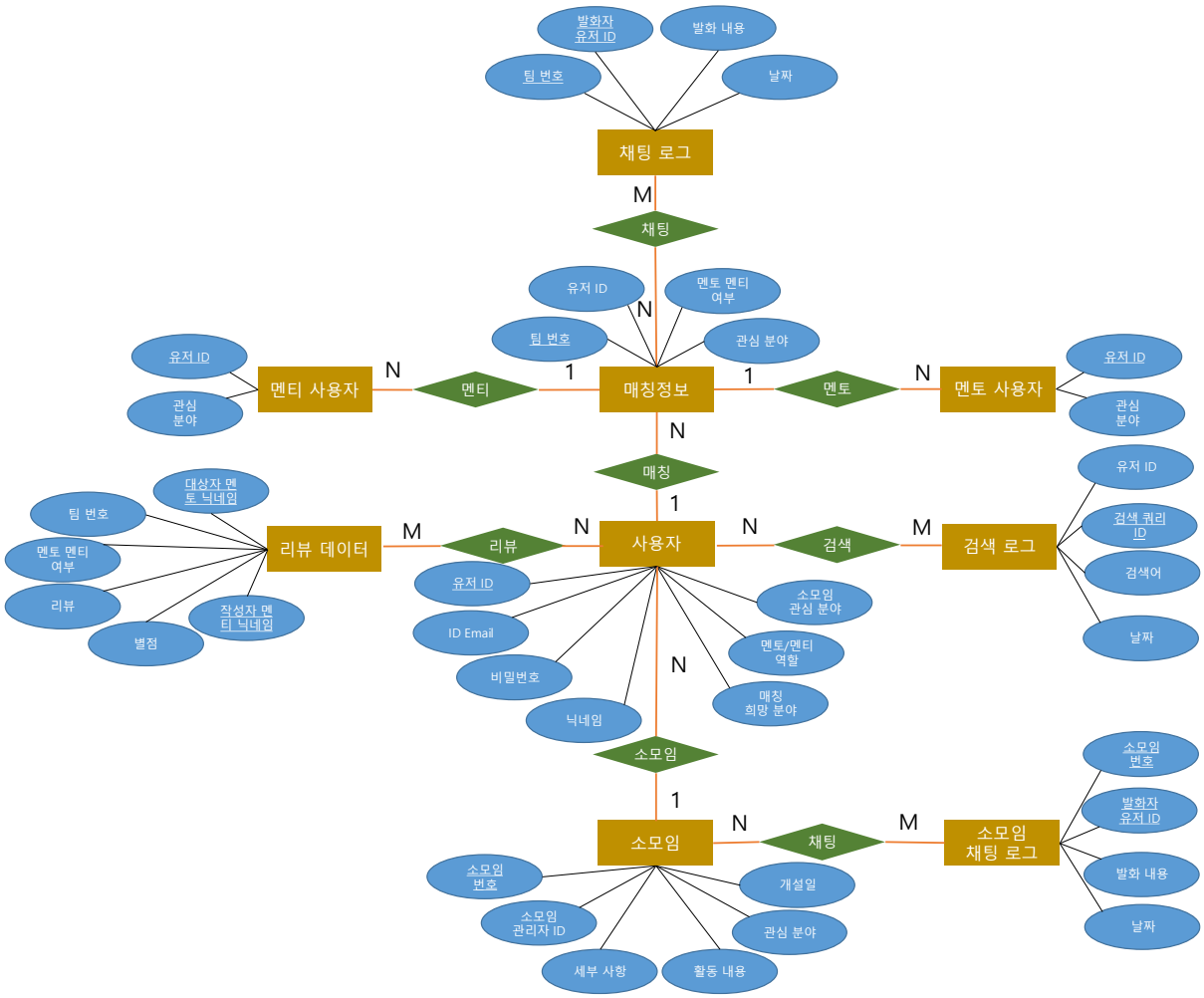
7.1 Objectives

이 섹션에서는 시스템 데이터 구조와 해당 구조가 데이터베이스에 표시되는 방법에 대해 서술합니다. 먼저 개체-관계 모델(ER-diagram)을 통해 개체와 그 관계를 기술합니다. 그 후 관계형 스키마 및 SQL DDL(Data Description Language) 세부사항을 설명합니다.

7.2 ER Diagram

시스템은 사용자, 매칭 정보, 멘티 사용자, 멘토 사용자, 채팅 로그, 리뷰 데이터, 검색 로그, 소모임과 소모임 채팅로그, 총 9개의 개체로 구성됩니다. 개체-관계 모델은 각 개체를 직사각형으로,

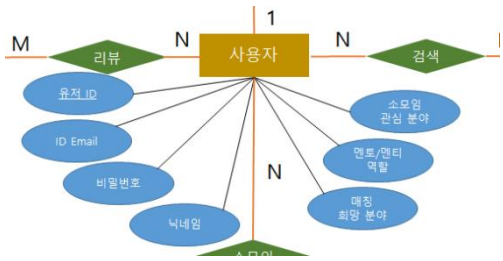
관계를 마름모로 표현합니다. 개체가 다른 개체와 관계가 있는 경우 주황색을 사용해 이를 나타냅니다. 개체의 속성은 타원으로 표현됩니다. 개체를 고유하게 식별하는 고유 속성은 밑줄로 표현됩니다. 개체에 동일한 속성이 여러가지 있는 경우 해당 속성은 이중 경계선이 있는 타원으로 표시됩니다.



[Figure 31] ER-diagram

7.2.1 Entities

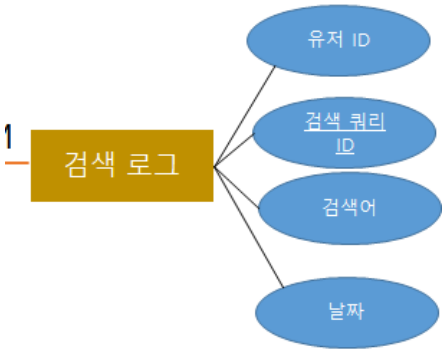
7.2.1.1 사용자



[Figure 32] 개체-관계 모델, 개체, 사용자

사용자 개체는 멘토멘티 매칭 프로그램의 사용자를 나타냅니다. 사용자 개체는 유저 ID, ID Email, 비밀번호, 닉네임, 매칭 희망 분야, 멘토/멘티 역할, 소모임 관심 분야로 이루어져 있으며 유저 ID 가 기본키의 역할을 합니다. 또한 여러 개의 리뷰 데이터와 소모임, 검색로그, 매칭 정보 개체와 연결될 수 있습니다.

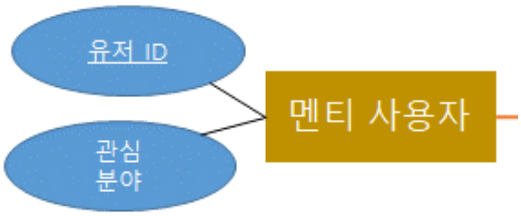
7.2.1.2 검색 로그



[Figure 33] 개체-관계 모델, 개체, 검색 로그

검색 로그 개체는 사용자의 멘토, 멘티 혹은 소모임 검색 기록을 표시합니다. 이 개체는 사용자 개체와 관계가 있으며, 유저 ID, 검색 쿼리 ID, 검색어, 날짜를 속성으로 가지며 기본 키는 검색 쿼리 ID 속성입니다.

7.2.1.3 멘티 사용자



[Figure 34] 개체-관계 모델, 개체, 멘티 사용자

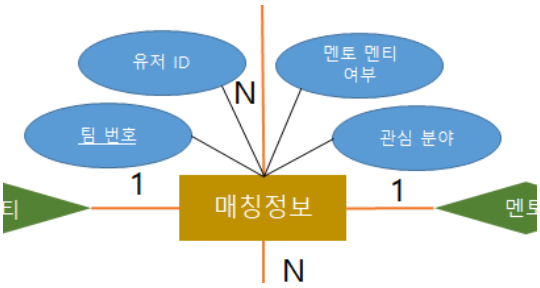
멘티 사용자 개체는 멘티로 매칭된 유저들을 표시하는 개체입니다. 이 개체는 매칭 정보 개체와 관계가 있으며, 유저 ID와 관심분야를 속성으로 가지며, 기본 키는 유저 ID 속성입니다.

7.2.1.4 멘토 사용자

[Figure 35] 개체-관계 모델, 개체, 멘토 사용자

멘토 사용자 개체는 멘토로 매칭된 유저들을 표시하는 개체입니다. 세부사항은 멘티 사용자 개체와 동일합니다.

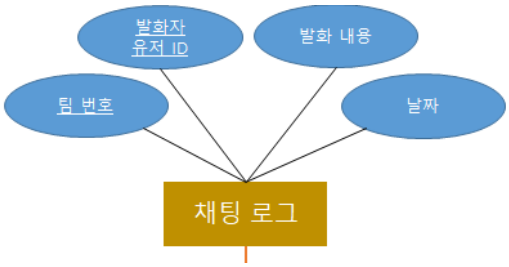
7.2.1.5 매칭 정보



[Figure 36] 개체-관계 모델, 개체, 매칭 정보

매칭 정보 개체는 사용자의 멘토-멘티 매칭 정보를 표시하는 개체입니다. 이 개체는 사용자 개체와 채팅 로그 개체와 관계가 있으며, 팀 번호, 유저 ID, 멘토 멘티 여부, 관심 분야를 속성으로 가지며 팀 번호가 기본키의 역할을 합니다. 또한 여러 개의 사용자와 멘티 사용자, 멘토 사용자, 채팅 로그 개체와 연결될 수 있습니다.

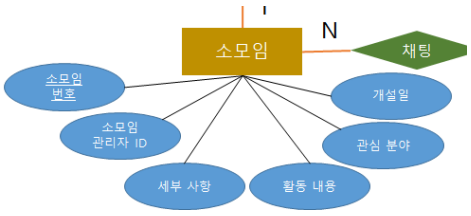
7.2.1.6 채팅 로그



[Figure 37] 개체-관계 모델, 개체, 채팅 로그

채팅 로그 개체는 멘토와 멘티로 매칭된 팀의 채팅 기록을 표시합니다. 이 개체는 매칭 정보 개체와 관계가 있으며, 팀 번호, 발화자 유저 ID, 발화 내용, 날짜를 속성으로 가지며 팀 번호와 발화자 유저 ID 속성이 기본 키 역할을 합니다.

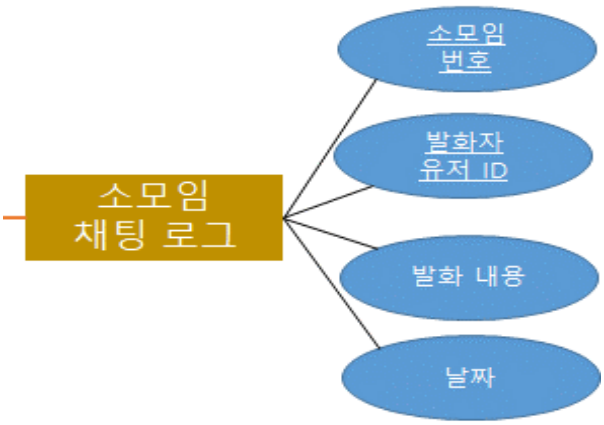
7.2.1.7 소모임



[Figure 38] 개체-관계 모델, 개체, 소모임

소모임 개체는 각 소모임의 정보를 기록합니다. 이 개체는 사용자와 소모임 채팅로그 개체와 관계가 있으며, 소모임 번호, 소모임 관리자 ID, 세부사항, 관심 분야, 활동 내용과 개설일을 속성으로 가지며 소모임 번호가 기본키의 역할을 합니다. 또한 여러 개의 사용자와 소모임 채팅 로그 개체와 연결될 수 있습니다.

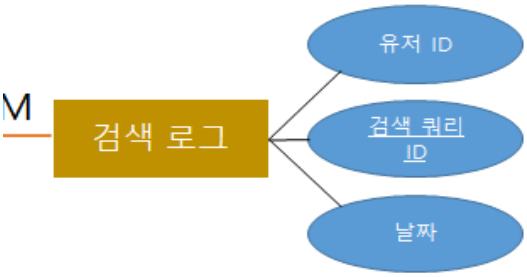
7.2.1.8 소모임 채팅 로그



[Figure39] 개체-관계 모델, 개체, 소모임 채팅 로그

소모임 개체는 각 소모임의 정보를 기록합니다. 이 개체는 사용자와 소모임 채팅로그 개체와 관계가 있으며, 소모임 번호, 소모임 관리자 ID, 세부사항, 관심 분야와 개설일을 속성으로 가지며 소모임 번호가 기본키의 역할을 합니다. 또한 여러 개의 사용자와 소모임 채팅 로그 개체와 연결될 수 있습니다.

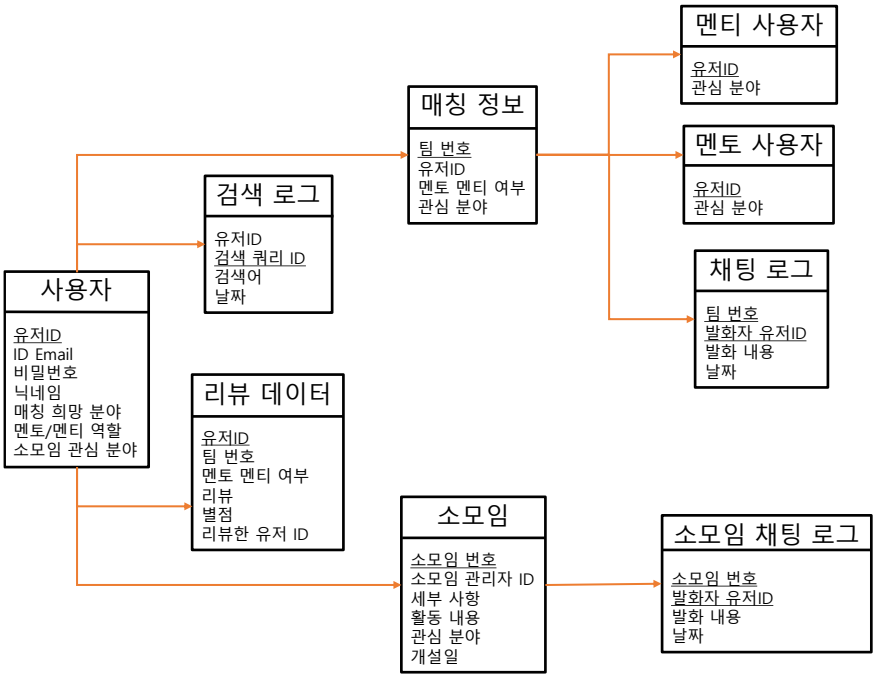
7.2.1.9 리뷰 데이터



[Figure40] 개체-관계 모델, 개체, 리뷰 데이터

검색 로그 개체는 사용자의 멘토, 멘티 혹은 소모임 검색 기록을 표시합니다. 이 개체는 사용자 개체와 관계가 있으며, 유저 ID, 검색 쿼리 ID, 날짜를 속성으로 갖습니다.

7.3 Relational Schema



[Figure41] 관계 스키마

7.4 SQL DDL

7.4.1 사용자

```
CREATE TABLE User
(
   user_id INT NOT NULL,
   web_mail VARCHAR NOT NULL,
   pass_word VARCHAR NOT NULL,
   nickname VARCHAR NOT NULL,
   mento_menti INT,
   match_interest VARCHAR,
   clud_interest VARCHAR,
   PRIMARY KEY (user_id)
);
```

7.4.2 검색 로그

```
CREATE TABLE Search_log
(
   search_query_id INT NOT NULL,
   user_id INT NOT NULL,
   search_key VARCHAR NOT NULL,
   date DATE NOT NULL,
   PRIMARY KEY (search_query_id)
);
```

7.4.3 멘티 사용자

```
CREATE TABLE Menti_user
(
   user_id INT NOT NULL,
   interest VARCHAR,

   PRIMARY KEY (user_id)

)
```

7.4.4 멘토 사용자

```
CREATE TABLE Mento_user
(
   user_id INT NOT NULL,
   interest VARCHAR,

   PRIMARY KEY (user_id)

)
```

7.4.5 매칭 정보

```
CREATE TABLE Matching_info
(
   team_number INT NOT NULL,
   user_id INT NOT NULL,
   mento_menti INT
   interest VARCHAR,

   PRIMARY KEY (team_number)

)
```

7.4.6 채팅 로그

```
CREATE TABLE Chat_log
(
   team_number INT NOT NULL,
   speaker_id INT NOT NULL,
   context VARCHAR,
   date DATE,
   PRIMARY KEY (team_number, speaker_id)
)
```

7.4.7 소모임

```
CREATE TABLE Club
(
   club_number INT NOT NULL,
   club_manager_id INT NOT NULL,
   generate_date DATE NOT NULL
   interest VARCHAR,
   additional_info VARCHAR,
   activity_content VARCHAR,
   PRIMARY KEY (clud_number)
)
```

7.4.8 소모임 채팅 로그

```
CREATE TABLE Club_chat_log
(
   club_number INT NOT NULL,
   speaker_id INT NOT NULL,
   context VARCHAR,
   date DATE,
   PRIMARY KEY (club_number, speaker_id)
)
```

7.4.9 리뷰 데이터

```
CREATE TABLE Review_data
(
   user_id INT NOT NULL,
   team_number INT NOT NULL,
   mento_menti INT,
   review VARCHAR NOT NULL,
   star_point INT NOT NULL,
   reviewer_id INT NOT NULL,
   PRIMARY KEY (user_id, reviewer_id),
);
```

8. Testing Plan

8.1 Objectives

이 섹션에서는 개발 테스트, 배포 테스트 및 사용자 테스트의 세가지 주요 테스트 계획을 설명합니다. 해당 테스트는 제품의 잠재적인 오류와 결함을 감지하고 안정적인 제품을 출시할 수 있도록 보장한다는 점에서 중요합니다.

8.2 Testing Policy

8.2.1 개발 테스트 (Development Testing)

개발 테스트는 주로 소프트웨어 개발의 잠재적인 오류를 줄이고 시간과 비용을 절약해 광범위한 오류를 방지하기 위해 수행됩니다. 이 단계에서는 소프트웨어가 충분한 테스트를 아직 거치지 않았기 때문에 불안정할 수 있으며 구성요소가 서로 충돌할 수 있습니다. 따라서 이 단계에서는 정적 프로그램 분석, 데이터 흐름 분석, 피어 프로그램 검토, 유닛 테스트를 수행해야 합니다. 이 작업을 통해 우리는 주로 우리 소프트웨어의 정체성을 정의하는 성능과 안전하고 오류 없는 작동을 보장하기 위한 신뢰성, 그리고 보안을 달성하는데 중점을 두고 있습니다.

8.2.1.1 Performance

추천 알고리즘은 시스템에서 가장 많은 시간이 소요되는 작업을 수행하기 때문에 개발자가 추천 결과를 필터링하고 사용자에게 제시하는 시간을 단축하는 것이 중요합니다. 권장 사양에 명시된 대로 시스템은 사용자에게 5초 이내에 결과를 제공해야 합니다. 다양한 선호도에 대한 테스트 케이스를 준비하고 추천 알고리즘의 속도를 평가하고 추천 알고리즘 및 서버와의 통신에 관한 코드 흐름을 개선합니다.

8.2.1.2 Reliability

시스템이 고장 없이 안전하게 작동하기 위해서는 먼저 시스템을 구성하는 하위 구성요소 및 장치가 작동하고 올바르게 연결되어야 합니다. 따라서 유닛 프로그램 개발 단계부터 개발 테스트를 거쳐 각 유닛 시스템을 통합하는 동안 반복적으로 오류를 확인하고 개선해야 합니다.

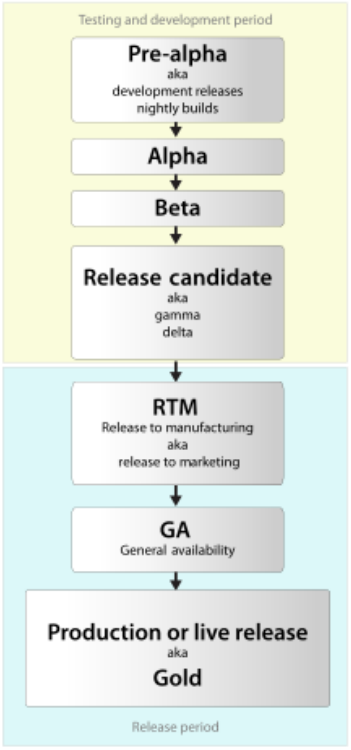
8.2.1.3 Security

사용자 및 시스템의 정보 보안은 개발자가 처리해야 할 중요한 문제입니다. 정보의 가치에 관계없이 원치 않는 시스템 방문자로부터 정보를 보호해야 합니다. 시스템 보안을 달성하기 위해 완성 단계에 있는 버전 어플리케이션에 접속하고 보안 문제를 식별합니다. 수동적으로 프로그래밍 코드를 컴토하고 관련 보고서를 작성합니다. 또한 Ostorlab, Appvigil 등에서 제공하는 다양한 모바일 어플리케이션 보안 테스트 서비스를 활용해 어플리케이션의 취약점을 파악하고 수정할 수 있습니다.

8.2.2 Release Testing

소프트웨어 개발에서 가장 중요한 부분 중 하나는 제품을 시장에 내놓고 고객에게 출시하는 것입니다. 기술적으로 훌륭한 소프트웨어도 잘못된 출시 방법으로 인해 괄목할 만한 성적을 얻을 수 없을 수 있습니다. 따라서 제품과 시장이 더욱 긴밀히 연결하기 위해 출시 테스트가 불가피 합니다. 출시 테스트는 소프트웨어나 어플리케이션의 새 버전을 테스트 해 소프트웨어가 완벽하게 출시될 수 있는지, 작동에 결함이 없는지 확인하는 작업입니다. 프로그램 출시 전에 우선적으로 수행되어야 합니다.

소프트웨어 출시 수명 주기에 따라 테스트는 일반적으로 프로그램의 기본 구현이 완료되는 '알파' 버전에서 시작됩니다. 알파 버전에서 개발 테스트를 시작하고 사용자 및 출시 테스트를 포함한 추가 테스트를 위해 베타 버전을 출시합니다. 베타 버전이 출시되면 소프트웨어는 개발자는 물론 실제 사용자에게로부터 피드백을 받게 됩니다.



[Figure42] 소프트웨어 배포 생명 주기

8.2.3 User Testing

유저 테스트에서는 필요한 사용자 테스트를 진행할 수 있는 가능한 시나리오와 현실적인 상황을 설정해야 합니다. 멘토-멘티 매칭 프로그램 어플리케이션에는 30명의 사용자가 있다고 가정합니다. 이 상황을 설정한 후 멘토-멘티 매칭 프로그램의 베타 버전을 배포하고 Android 에뮬레이터를 사용해 개인의 유스 케이스 테스트를 실행하면서 실질적인 사용자의 리뷰 데이터를 수집합니다.

8.2.4 Testing Case

테스트 케이스는 어플리케이션의 3가지 기본 측면인 기능과 성능, 보안에 따라 설정됩니다. 각 측면에서 5가지, 총 15가지의 테스트 케이를 설정하고 멘토-멘티 매칭 프로그램 어플리케이션에 대한 테스트를 진행할 평가지를 제작합니다.

9. Development Plan

9.1. Objectives

Chapter9는 애플리케이션 개발 환경 및 도구에 대해 기술합니다. frontend와 backend 개발 환경 및 도구와 개발 관련 constraints, assumptions, dependencies를 기술합니다.

9.2. Frontend Environment

9.2.1. Android Studio

[Figure 43] Android Studio logo

Android 스튜디오는 Android 애플리케이션 개발을 위한 IntelliJ IDEA를 기반 공식 통합 개발 환경(IDE)입니다. IntelliJ의 강력한 코드 편집기와 개발자 도구 외에도 Android 애플리케이션 개발할 때 생산성을 높여주는 여러 유용한 기능을 제공하는 통합 개발 환경입니다. 따라서 본 애플리케이션의 frontend 개발에서 Android 스튜디오를 주요 개발 도구로 사용할 예정입니다.

(https://developer.android.com/studio/intro?hl=ko 참조하십시오)

9.2.2. Kotlin

[Figure 44] Kotlin logo

Frontend 개발 언어로 Kotlin을 사용할 계획입니다. Kotlin은 정적 입력 방식의 최신 프로그래밍 언어로 전문 Android 개발자의 60% 이상이 사용하고 있으며 생산성, 개발자 만족도 및 코드 안전성을 높이는 데 도움이 되는 언어입니다. 함께 사용할 계획인 Android 스튜디오는 Kotlin 사용을 위한 최고의 지원을 제공하기 때문에 애플리케이션 개발을 위해 Kotlin언어가 적합할 것으로 예상

합니다. (https://developer.android.com/kotlin?hl=ko&gclid=CjwKCAjwnPOEBhA0EiwA609ReQPXcI4p2-CYvA9PuGDEm0VfB4V91icHheiTqlclKprXsNJ1_kUS0hoCdHIQAvD_BwE&gclsrc=aw.ds 참조하십시오)

9.3. Backend Environment

9.3.1. Github

[Figure 45] Github logo

Git를 이용한 소프트웨어 개발 버전 제어를 위한 호스팅을 제공한다. 이를 통해 팀원들이 단일 프로젝트를 함께 개발할 수 있어 구성 요소를 쉽게 통합할 수 있습니다. 우리는 현재 GitHub를 응용 프로그램 개발과 제어 버전으로 사용하고 있습니다.

9.3.2. Firebase

[Figure 46] Firebase logo

클라우드 스토리지, 실시간 데이터베이스, 머신러닝 키트 등 다양한 기능을 제공하여 모바일 및 웹 애플리케이션 개발을 지원합니다. 그중에서도 사용자 데이터, 기능 등을 관리하기 위한 실시간 데이터베이스 기능을 사용할 것입니다. 실시간 데이터베이스 덕분에 이 데이터베이스에 연결된 모든 클라이언트에서 데이터가 동기화됩니다. 이는 모든 클라이언트가 단일 실시간 데이터베이스 인스턴스를 공유하고 자동 업데이트를 통해 업데이트된 데이터를 수신할 수 있음을 의미합니다.

9.3.3 Android Studio

[Figure 47] Android Studio logo

애플리케이션을 외부 API에 연결하여 추가 기능을 추가하고 XML 파일을 애플리케이션의 작업에 연결할 수 있습니다. 또한, 우리는 데이터 제어를 위해 애플리케이션을 DB 서버에 연결합니다.

9.4 Constraints

앞으로 진행될 시스템 설계는 이 문서에 있는 내용에 맞춰 진행될 것입니다. 그 밖의 제약 사항은 다음과 같습니다.

- 가능한 한 범용되는 오픈 소스를 활용할 것입니다.

- 검색에 소요되는 시간은 5 초를 넘기지 않도록 합니다.

- 대안이 없는 경우를 제외하고 사용료를 지불해야 하는 제품은 사용하지 않습니다.

- 프론트엔드는 안드로이드 스튜디오와 Kotlin을 사용하여 개발합니다.

- 백엔드는 Python 과 Firebase를 이용하여 개발합니다.

- 사용자의 피드백을 기반으로 제품을 향상시킵니다.

- 향후에 발생하는 User requirement change에 대비해 코드를 간결하게 작성하고 code에 대해 가능한 한 comment를 달도록 합니다.

9.5 Assumptions and Dependencies

이 문서는 안드로이드 개발 환경임을 설정하고 작성되었습니다. 사용자는 안드로이드 5.1 버전 이상에서 서비스를 제공받을 수 있습니다.

10. Supporting Information

10.1. Software Design Specification

이 디자인 명세서 문서는 IEEE 표준안 (IEEE Std. 1012-2012, IEEE Standard for Software Verification & Validation)에 따라 작성되었습니다.

10.2. Document History

Date	Version	Description	Writer
2021/05/08	0.1	양식 작성	김시인
2021/05/10	1.0	1 작성, 4 작성	김시인
2021/05/11	1.1	2 작성, 5 작성	서기용
2021/05/11	1.2	3 작성, 6 작성	김정훈
2021/05/12	1.3	7 작성	김민지
2021/05/13	1.4	4 수정, 9.1, 9.2 작성	김시인
2021/05/14	1.5	9.3 작성	서기용
2021/05/14	1.6	9.4, 9.5 작성	김정훈
2021/05/15	1.7	7 수정, 8 작성	김민지
2021/05/16	1.8	최종본 양식 수합	김시인