Connect-U

Design Specification

Team 11

2017310605 김도현

2019311895 김연재

2016310401 이광호

2017314655 이인수

2017312222 최영우

Table of Contents

1. Preface....8
1.1. Readership....8
1.2. Scope....8
1.3. Objective....8
1.4. Document Structure....8
A. Preface....8
B. Introduction....8
C. System Architecture – Overall....9
D. System Architecture – Front-end....9
E. System Architecture – Back-end....9
F. Protocol Design....9
G. Database Design....9
H. Testing Plan....9
I. Development Plan....9
J. Supporting Information....9
2. Introduction....9
2.1. Objectives....9
2.2. Applied Diagrams....10
A. UML....10
B. Use case Diagram....10
C. Sequence Diagram....10
D. Class Diagram....11
E. Entity Relationship Diagram....11
2.3. Applied Tools....12
A. Microsoft PowerPoint....12

B. Draw.io....12
2.4. Project Scope....13
2.5. References....13
3. System Architecture – Overall....13
3.1. Objectives....13
3.2 System Organization....14
A. Frontend Application....14
B. Backend Application....15
4. System Architecture – Frontend....16
4.1. Objective....16
4.2. Components....16
A. Registration & Authentication System....16
1) RegisterPage....16
A. Attributes....16
B. Methods....17
2) AuthenticationPage....17
A. Attributes....17
B. Methods....17
3) LoginPage....18
A. Attributes....18
B. Methods....18
B. Profile System....19
1) ProfilePage....20
A. Attributes....20
B. Methods....20
C. Community System....21
1) CommunityMainPage....21

A. Attributes....21
B. Methods....21
2) Sorting....22
A. Attributes....22
3) Post....22
A. Attributes....22
D. Rating & Report System....24
1) RatingPage....24
A. Attributes....24
B. Methods....24
2) ReportPage....24
A. Attributes....24
B. Methods....25
5. System Architecture - Backend....25
5.1. Objective....25
5.2. Components....26
5.2.1. Controller....26
A. Method Description....26
5.2.2. Rating System....26
A. Method Description....27
5.2.3. User Authentication System....28
A. Method Description....28
5.2.4. Penalty System....29
A. Method Description....29
5.2.5. Post Search System....30
A. Method Description....31
6. Protocol Design....32

6.1 Objectives....32
6.2 REST API....32
1) REST API의 장점....32
a. client - server 구조....32
b. Uniform Interface....32
c. Stateless....32
d. Cacheable....32
e. Self-descriptiveness....33
2) REST API의 구성....33
a. 자원 (Resource) : URL....33
b. 행위 (Verb) : HTTP Method....33
c. 표현 (Representation) : JSON....33
6.3 Details....34
1) Authorization....34
a. Signup....34
b. Login....34
2) User....35
a. My Page....35
b. Edit My Page....36
3) Post....37
a. Search....37
b. Enroll post....37
c. Recommend post....38
d. matching....39
4) Apply history....39
a. Get....39
5) Search history....40

a. Get....40
6) Enroll history....40
a. Get....41
7) Report....41
a. Enroll report....41
8) Rating....42
a. Enroll rate....42
b. Get....42
7. Database Design....43
7.1. Objectives....43
7.2. ER Diagram....43
7.2.1. Entities....44
7.2.1.1. User....44
7.2.1.2 Search History....45
7.2.1.3 Enroll Histrory....46
7.2.1.4 Sign-Up History....46
7.2.1.5 Post....47
7.3 Relational Schema....48
7.4 SQL DDL....48
7.4.1 User....48
7.4.2 Search History....48
7.4.3 Enroll History....49
7.4.4 Sign-Up History....49
7.4.5 Post....49
7.4.6 recruit_condition....50
8. Testing Plan....50
8.1. Objectives....50

8.2. Testing Policy....50
8.2.1. Development Testing....51
A. Performance....51
B. Reliability....51
C. Security....51
8.2.2. Release Testing....51
8.2.3. User Testing....52
8.2.4. Testing Case....52
9. Development Plan....52
9.1. Objectives....52
9.2. Frontend Environment....52
A. Adobe Photoshop....52
B. Zeplin....53
C. Flutter with Android Studio....53
9.3. Backend Environment....54
A. Github....54
B. Firebase....54
9.4. Constraints....55
9.5. Assumptions and Dependencies....55
10. Supporting Information....56
10.1. Software Design Specification....56
10.2. Figures....56
10.3. Tables....56
10.4. Document History....59

1. Preface

1.1. Readership

본 문서의 예상 독자는 소프트웨어 개발자와 그 외에 관련된 stakeholders이다. 소프트웨어 개발자에는 소프트웨어 엔지니어, Architecture, Database Manager, 서버 운영자 및 추후 기술 지원을 위한 고객 서비스 팀 등이 이에 해당된다.

1.2. Scope

Design Specification은 소프트웨어 엔지니어링 과정에서 대학생 인력 중개 시스템인 Connect-U를 구현하는데 사용할 디자인의 정의로 사용된다.

1.3. Objective

Design Specification의 주요 설계 목적은 대학생 인력 중개 시스템인 Connect-U의 기술적 디자인에 대한 설명을 제공하는 것이다. 이 문서는 Connect-U 구현을 위한 소프트웨어 architecture과 소프트웨어 design 결정에 대해 설명한다. 또한 시스템의 전반적인 구조를 설명하고, 다양한 다이어그램을 통하여 시스템 모듈의 구조와 설계를 프로그래밍 팀이 알아보기 쉽게 한다.

1.4. Document Structure

본 문서는 Preface, Introduction, System architecture - Overall, System architecture – Front-end, System architecture – Back-end, Protocol design, Database design, Testing plan, Development plan, Supporting Information의 10개의 단원으로 이루어졌다. 각 단원에 대한 설명은 다음과 같다.

A. Preface

이 장에서는 문서의 예상 독자층을 선정하고, 문서의 scope, objective와 Connect-U의 Design Specification의 구조를 포함한다.

B. Introduction

이 장에서는 문서에서 사용된 여러 도구, 다이어그램 및 프로젝트의 개발 범위와 레퍼런스를 기술한다.

C. System Architecture – Overall

이 장에서는 본 시스템의 전체 구조에 대해 설명하고 각 시스템 간 연관 상태를 나타낸다. 또한 각 시스템을 간략하게 설명한다.

D. System Architecture – Front-end

이 장에서는 Front-end 시스템의 구조와 하위 구성 요소 간의 구성도 및 각 요소의 역할 등을 여러 종류의 다이어그램을 통해 설명한다.

E. System Architecture – Back-end

이 장에서는 Back-end 시스템의 구조와 하위 구성 요소 간의 구성도 및 각 요소의 역할 등을 여러 종류의 다이어그램을 통해 설명한다.

F. Protocol Design

이 장에서는 Front-end와 Back-end간의 상호작용을 규정하는 인터페이스와 프로토콜을 어떻게 구성하는지에 대해 기술하고, 해당 인터페이스가 어떤 기술에 기반해 있는지 설명한다.

G. Database Design

이 장에서는 시스템에서 활용되는 데이터를 저장하고 관리하고 활용하기 위한 Database의 구조를 설명한다. 데이터의 속성과 데이터베이스의 스키마, 데이터베이스 간의 관계에 대해서 ER diagram과 Relational schema를 이용해 설명한다.

H. Testing Plan

이 장에서는 시스템이 요구사항을 만족시키는지를 확인하고, 시스템의 내부적 결함을 찾기 위한 testing plan을 설명한다.

I. Development Plan

이 장에서는 시스템을 구현하는 데 사용되는 개발 도구와 프로그래밍 언어, 라이브러리, 플랫폼 등의 개발 환경에 대해 설명한다.

J. Supporting Information

이 장에서는 문서의 기저와 작성 일정에 대해 기술한다.

2. Introduction

2.1. Objectives

이 장에서는 시스템의 design process에 사용된 다양한 다이어그램 및 tool들을 소개하고, 시스템의 개발 범위와 레퍼런스를 기술한다.

2.2. Applied Diagrams

A. UML

UML(Unified Modeling Language)은 통합 모델링 언어라는 뜻으로 객체 지향 소프트웨어 엔지니어링 분야의 표준화된 범용 모델링 언어를 말한다. UML을 이용하여 시스템의 모든 것을 문서화, 지정, 구축할 수 있으며 객체 지향 문제 해결에 적용할 수 있다. 이에 더해 복잡한 소프트웨어 시스템 개발 모델링에 필요한 구성요소를 제시하고 이를 이용한 추상화 방법과 산출물들을 프로젝트 참여자들이 쉽게 이해할 수 있도록 한다.
따라서 UML을 통해 사용자, 분석가, 설계자, 개발자 등의 개발을 위한 의사소통을 원활하게 한다. UML에는 대표적으로 class diagram, use case diagram, sequence diagram, component diagram 등이 있다.

B. Use case Diagram

Use case diagram은 개발하고자 하는 시스템을 목적에 맞게 사용자 입장에서 시나리오를 그린 다이어그램이다. 시스템의 기능적 요구사항에 대한 기저이고 사용자의 시각에서 시스템의 범위와 기능을 정의한 모델이다. 이렇게 actor가 어떤 기능을 사용할 수 있는지를 보여준다.

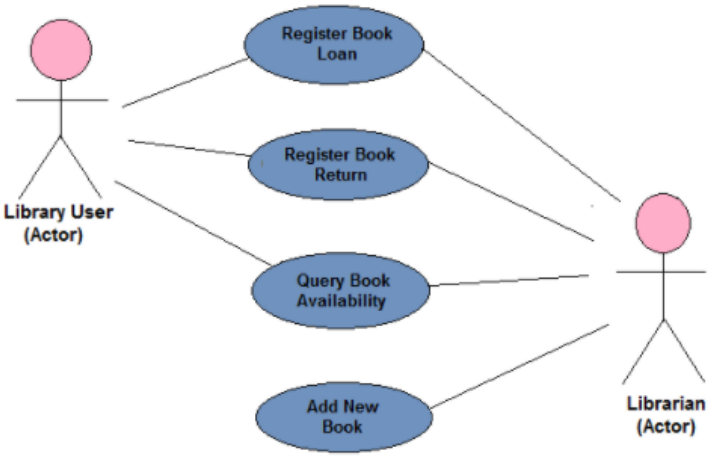

Figure 1. Use case diagram의 예

C. Sequence Diagram

Sequence Diagram은 시간 순서로 정렬된 객체의 상호작용을 보여준다. 시나리오 기능을 수행하는데 필수적인 객체들 간에 교환되는 일련의 메세지들과 시나리오에 수반되는 객체와 클래스를 표현한다.

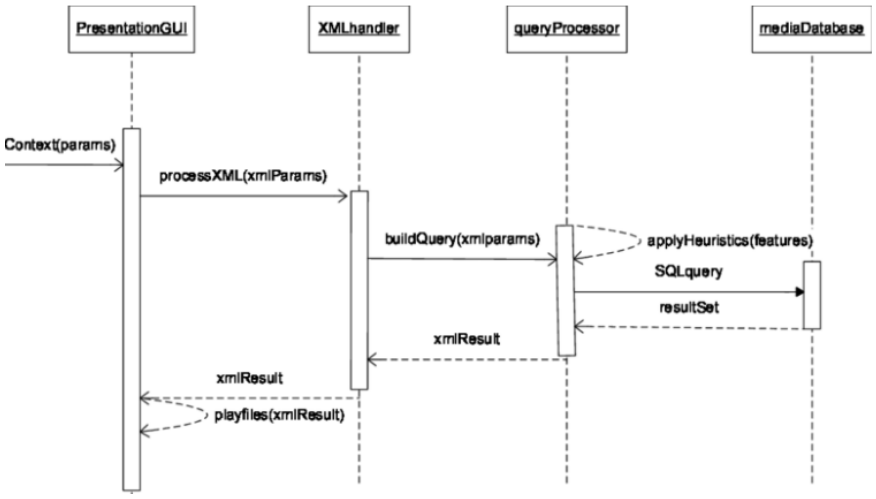

Figure 2. Sequence diagram의 예

D. Class Diagram

Class diagram은 시스템에서 사용되는 class를 정의하고 그들 간에 존재하는 정적인 관계를 다양한 방식으로 표현한 다이어그램이다.

객체지향 SW 시스템은 클래스와 그 관계로 뼈대가 구성되기 때문에 이를 정의한 클래스 다이어그램은 곧 시스템의 구현될 모습을 정의한 것이다. 클래스 다이어그램은 분석되거나 설계되는 모든 클래스를 한 장의 다이어그램으로 정의한 것이다.

클래스 다이어그램을 통해 문제 도메인의 구조를 나타낼 수 있고, 실제 소프트웨어의 설계 혹은 구현을 위한 용도로 사용되기도 한다

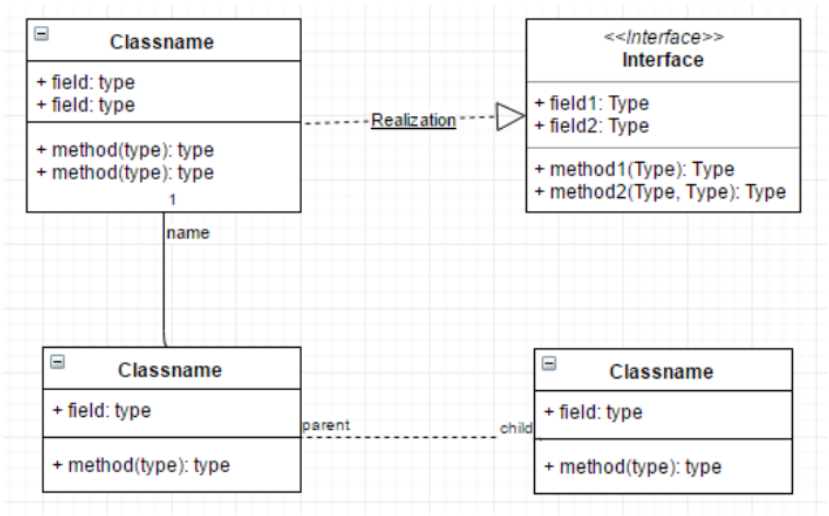

Figure 3. Class diagram의 예

E. Entity Relationship Diagram

ER diagram은 Entity-relationship model을 표현하며 데이터베이스를 설계할 때 사용된다. ER diagram은 개체, 속성, 관계성을 표현하여 데이터베이스를 모델링하는 방식이다.

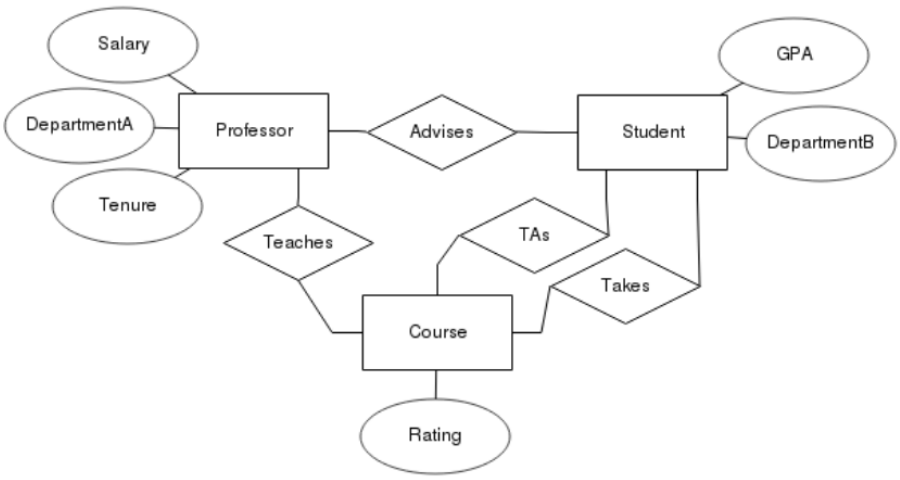

Figure 4. ER diagram의 예

2.3. Applied Tools

A. Microsoft PowerPoint

Power Point는 Ms사가 제공하는 Tool으로 다양한 도형, 아이콘 등을 그릴 수 있으며 이를 이용해 다양한 diagram을 그리는데 유용하게 사용된다. 또한 텍스트를 삽입하고 수정하기에도 용이하다.

Figure 5. Power Point

B. Draw.io

Draw.io는 많은 기본 템플릿과 도형을 제공해 사용자는 drag and drop으로 손쉽게 다이어그램을 그릴 수 있다. 또한 도형 간 연결선을 간단하게 만들 수 있고, 격자에 위치를 맞출 수 있는 등 다이어그램 작성에 필요한 다양한 기능을 제공해주기 때문에 많은 다이어그램이 draw.io를 통해

작성되었다.

Figure 6. Draw.io

2.4. Project Scope

Connect-U는 기존에 대학생들을 위한 인력중개 플랫폼의 부족한 점을 개선하기 위하여 모집자가 모집 카테고리를 설정하고 모집을 원하는 사람들의 조건을 설정해서 시스템이 제공하는 양식에 맞게 인원 모집 공고를 작성하고 올리면 참여를 원하는 지원자가 공고를 보고 신청해서 연결해주는 인력 중개 시스템이다. 이를 위하여 Connect-U는 사용자에게 다양한 기능들을 제공한다. 처음으로, 사용자가 시스템에 가입하고 사용자를 인증 받기 위한 Authentication기능, 가입한 정보를 바탕으로 한 log in, out 기능, 개인 정보를 변경하기 위한 Profile change 기능, 시스템에 프로젝트를 등록하는 기능, 사용자가 관심있는 프로젝트를 찾기 위한 search 기능, 사용자에게 프로젝트를 추천해주는 추천기능, 프로젝트에 참여한 이후 상호 평가를 하기 위한 평가 기능, 신고가 누적되는 경우 특정 기간 동안 프로젝트 신청을 할 수 없도록 하는 블랙리스트 기능 등이 있다.

2.5. References

이 문서를 작성하기 위해 사용한 references는 다음과 같다.

A. Team 1, 2020 Spring. Software Design Document, SKKU.

B. Appleton, Brad. A Software Design Specification Template. N.d.

3. System Architecture – Overall

3.1. Objectives

본 챕터는 본 시스템의 전체적인 구조와 하위 시스템의 개괄적 설계도에 대한 서술이다. 프론트

엔드와 백엔드 구조로 나누어 서술한다.

3.2 System Organization

본 시스템은 클라이언트-서버 모델을 적용해 설계했다. 프론트엔드 어플리케이션은 사용자와 사용자 인터페이스를 통해 상호작용한다. 프론트엔드 어플리케이션과 백엔드 어플리케이션은 Rest Api를 사용하여 JSON 형식의 데이터를 송수신한다. 백엔드 어플리케이션은 프론트엔드로부터 들어오는 요청을 로직에 맞게 처리하여 적절한 응답 객체를 해당하는 컨트롤러로 반환한다.

A. Frontend Application

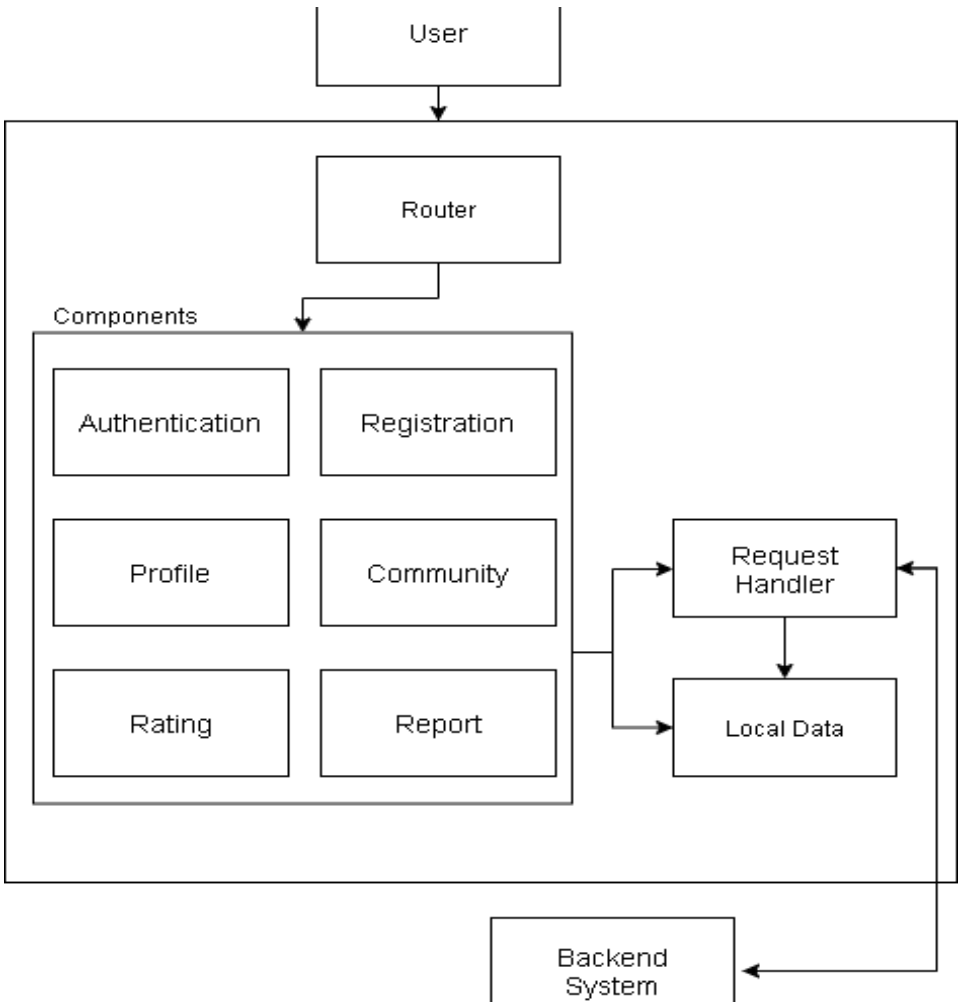

Figure 7. Frontend context diagram

프론트엔드는 사용자와 상호작용하는 UI를 제공한다. Flutter 프레임워크를 사용하여 자체 OS 시스템에 맞게 컴포넌트를 관리한다. 컴포넌트에서 필요한 자원 중 빈번히 사용되는 데이터들은 로컬 데이터 저장소에서 가져올 수 있도록 설계한다. 프론트엔드에서 백엔드로 요청을 보내기 위해

서는 Request Handler를 호출하여 적절한 응답을 받아온다.

B. Backend Application

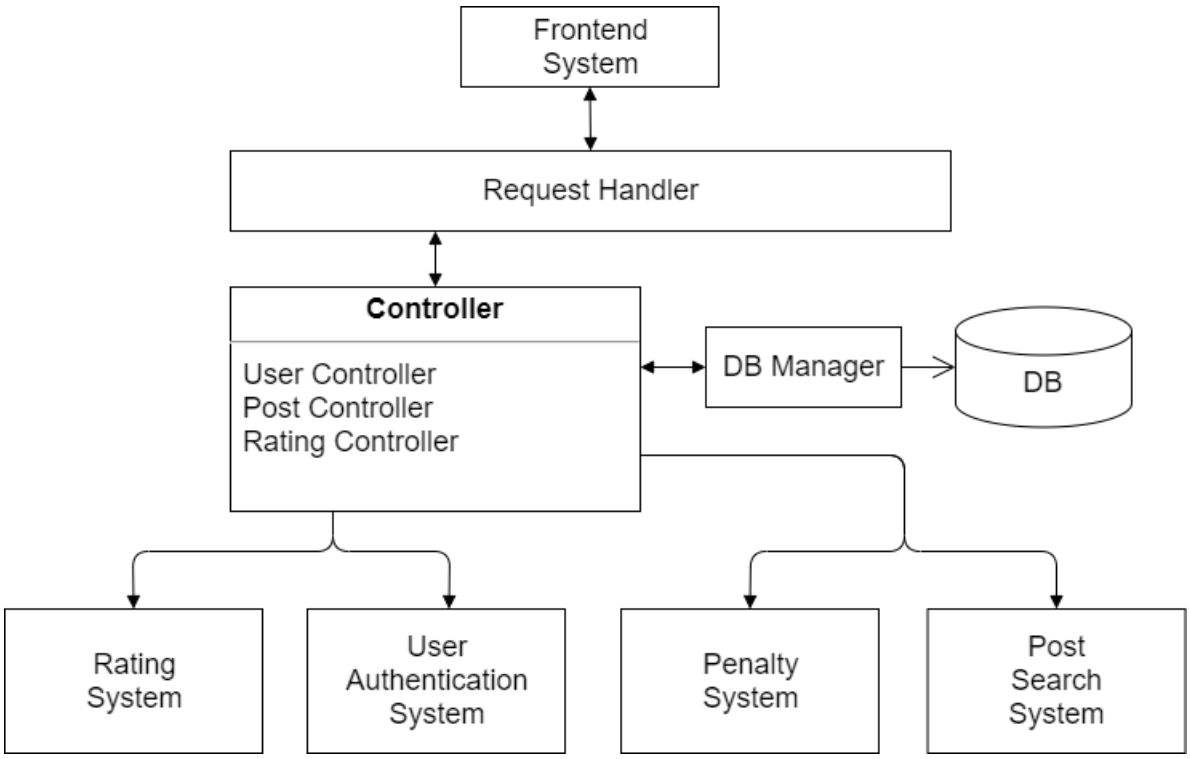

Figure 8. Architecture Diagram - Backend

벡엔드는 프론트엔드에서 Request Handler 를 통해 요청을 받아 Controller 를 통해 요구사항을 분석하고 DB Manager를 통해 DB에서 필요한 정보를 받아온다. 이 때 단순한 CRUD 작업은 서브 시스템인 각 개체의 Controller 를 통해서 진행하고 외부 API 와의 통신 등이 필요한 작업은 별개로 이뤄진 컴포넌트를 통해서 이뤄진다.

4. System Architecture – Frontend

4.1. Objective

전체 시스템 아키텍처 중 사용자와의 상호작용을 담당하는 프론트엔드 시스템의 구조와 각 컴포넌트의 구성, 컴포넌트 사이의 관계를 기술한다.

4.2. Components

A. Registration & Authentication System

Class Diagram

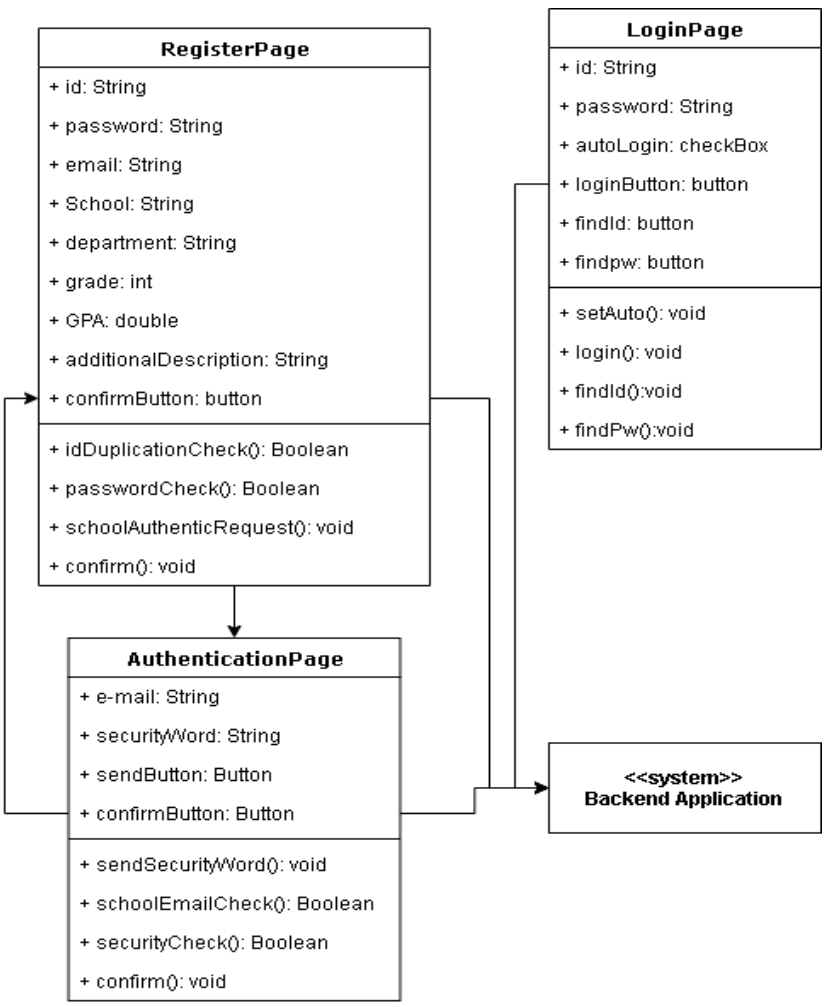

Figure 9. Registration & Authentication System Class Diagram

1) RegisterPage

A. Attributes

+ id: 사용자의 id

+ password: 사용자의 password

+ email: 사용자가 소속된 학교의 email

+ school: 사용자가 소속된 학교

+ department: 사용자가 소속된 학과

+ grade: 사용자의 학년

+ GPA: 사용자의 총 GPA

+ additionalDescription: 사용자가 자신의 프로필에 적고 싶은 말

+ confirmButton: 회원가입 완료 버튼

B. Methods

+ idDuplicationCheck(): id 중복 여부를 체크하는 요청을 백엔드에 보낸다.

+ passwordCheck(): password가 양식에 맞는지 체크한다.

+ schoolAuthenticRequest(): 학교 인증 화면으로 이동한다.

+ confirm(): 회원 정보를 백엔드에 보내 회원가입을 완료한다.

2) AuthenticationPage

A. Attributes

+ e-mail: 사용자가 소속된 학교의 email

+ securityWord: 사용자의 email로 보내진 보안문자

+ sendButton: 사용자의 email로 보안문자를 보내는 버튼

+ confirmButton: 학교 인증 완료 버튼

B. Methods

+ sendSecurityWord(): 백엔드에 이메일로의 보안문자 전송을 요청한다.

+ schoolEmailCheck(): 해당 학교에서 등록한 학교 이메일인지 확인한다.

+ securityCheck(): 보안문자를 확인한다.

+ confirm(): 학교 인증이 완료를 백엔드에 알린다.

3) LoginPage

A. Attributes

+ id: 사용자의 id

+ password: 사용자의 password

+ autologin: 자동로그인 여부

+ loginButton: 로그인 버튼

+ findId: 아이디 찾기 버튼

+ findPw: 비밀번호 찾기 버튼

B. Methods

+ setAuto(): 백엔드에 자동로그인 활성화를 알린다.

+ login(): 백엔드에 로그인 요청을 보낸다.

+ findId(): 아이디 찾기 페이지로 이동한다.

+ findPw(): 비밀번호 찾기 페이지로 이동한다.

Sequence Diagram

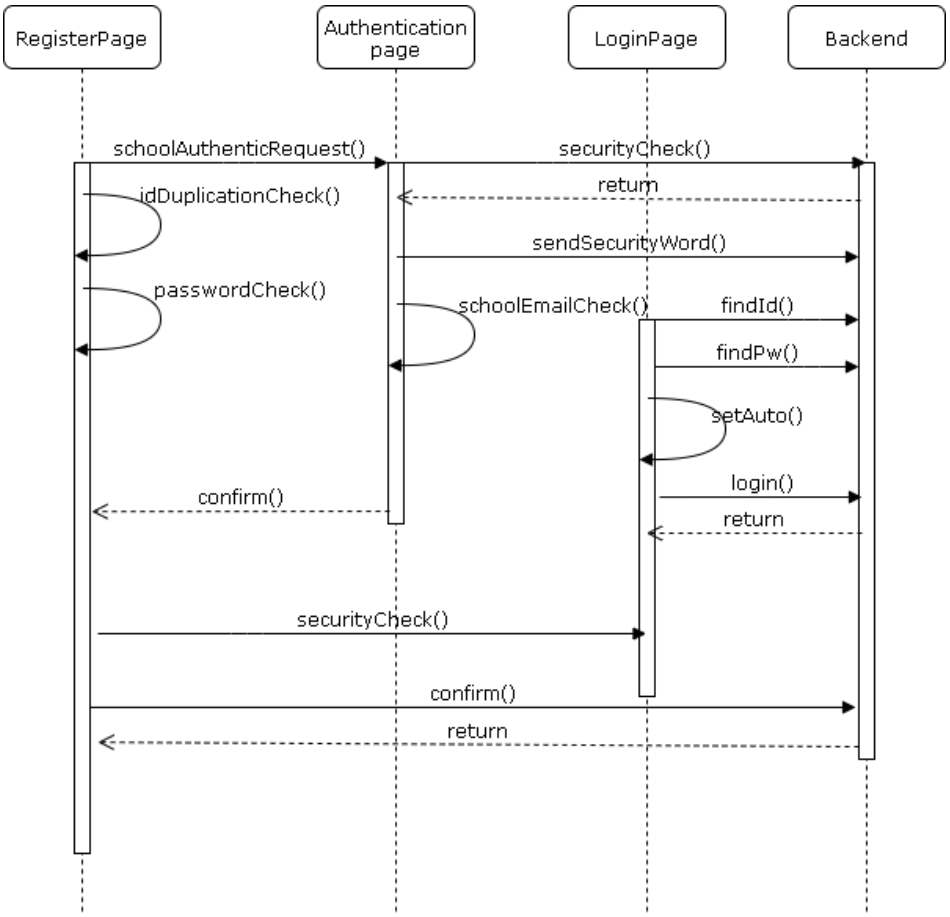

Figure 10. Registration & Authentication System Sequence Diagram

B. Profile System

Class Diagram

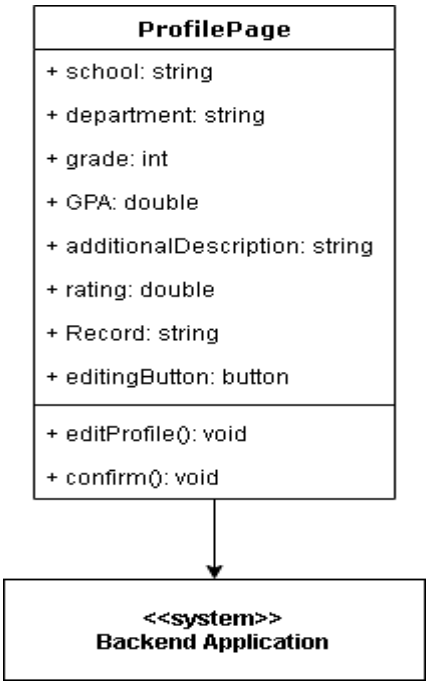

Figure 11. Profile System Class Diagram

1) ProfilePage

A. Attribute

+ school: 사용자가 소속된 학교

+ department: 사용자가 소속된 학과

+ grade: 사용자의 학년

+ GPA: 사용자의 GPA

+ additionalDescription: 사용자의 프로필에 남는 추가적인 설명

+ rating: 사용자의 평점

+ Record: 사용자가 참여한 프로젝트 이력

+ editingButton: 프로필 수정 버튼

B. Method

+ editProfile(): 프로필에서 변경 가능한 부분을 inputText로 변경

+ confirm(): 변경사항을 백엔드에 보내 profile수정 요청

Sequence Diagram

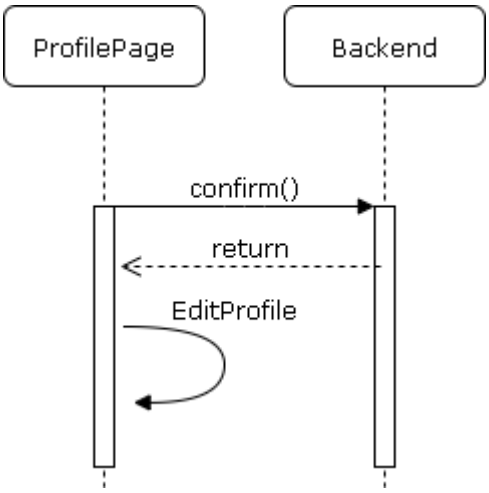

Figure 12. Profile System Sequence Diagram

C. Community System

Class Diagram

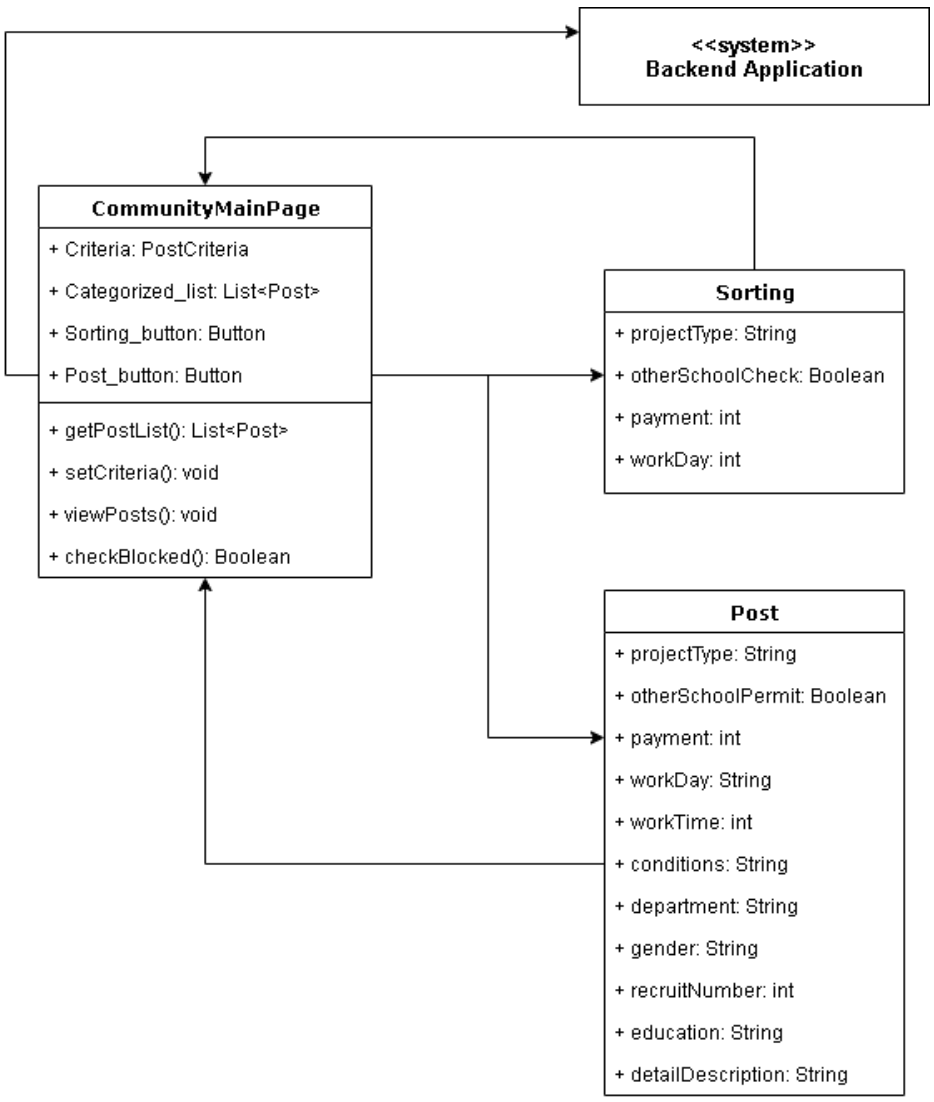

Figure 13. Community System Class Diagram

1) CommunityMainPage

A. Attributes

+ Categorized_list: 현재 게시판에 해당되는 카테고리의 리스트

+ Criteria: 리스트를 정렬하는 기준

+ Sorting_button: setCriteria()를 통해 Criteria를 정하는 버튼

+ Post_button: 새로운 Post를 작성하는 버튼

B. Methods

+ getPostList(): 지정된 Criteria에 맞는 Post의 list를 백엔드에 요청해 가져온다.

+ setCriteria(): Criteria를 지정한다.

+ viewPost(): 해당 Post 내부의 page를 백엔드에 요청해 가져온다.

+ checkBlocked(): 유저가 blacklist에 등록되어 있는지 확인한다.

2) Sorting

A. Attributes

+ projectType: 원하는 프로젝트의 직종(서비스, 전문직, 연구/개발, 인사, 마케팅...)

+ otherSchoolCheck: 다른 학교의 post도 확인할지 선택한다.

+ payment: 원하는 수당의 범위를 선택한다.

+ workday: 원하는 일하는 요일을 선택한다.

3) Post

A. Attributes

+ projectType: 개설하는 프로젝트의 직종

+ otherSchoolPermit: 다른 학교에서의 검색 허용

+ payment: 수당

+ workDay: 일하는 기간

+ workTime: 일하는 시간

+ conditions: 피모집자의 조건

+ department: 학부

+ gender: 원하는 피모집자의 성별

+ recruitNumber: 원하는 피모집자 수

+ education: 원하는 피모집자의 학력

+ detailDescription: 개설하는 project에 대한 상세 설명

Sequence Diagram

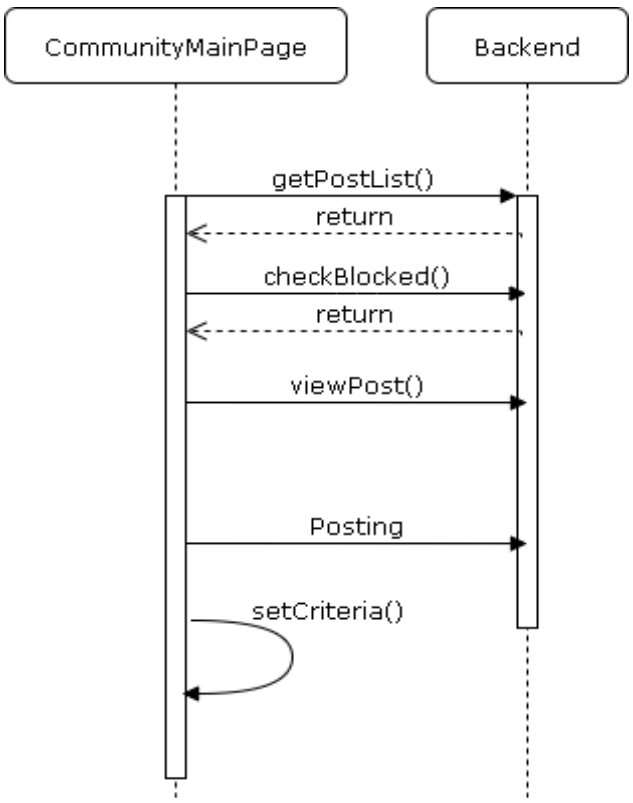

Figure 14. Community System Sequence Diagram

D. Rating & Report System

Class Diagram

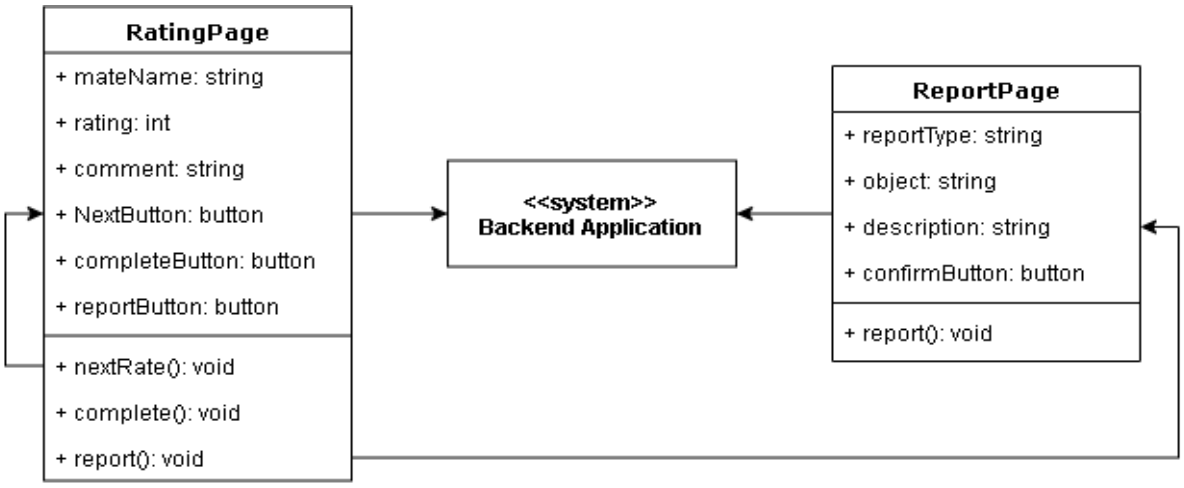

Figure 15. Rating & Report System Class Diagram

1) RatingPage

A. Attributes

+ mateName: 평가하는 대상의 이름(팀내 별명)

+ rating: 대상에게 부여할 평점

+ comment: 대상에게 남기고 싶은 말

+ NextButton: 다음 팀원 평가 페이지로 이동

+ completeButton: 평가 완료 버튼

+ reportButton: 해당 팀원 신고 버튼

B. Methods

+ nextRate(): 다음 팀원의 평가 페이지를 백엔드에 요청한다.

+ complete(): 진행한 모든 평가의 정보를 백엔드에 전송한다.

+ report(): 해당 팀원에 대한 신고 페이지로 이동한다.

2) ReportPage

A. Attributes

+ reportType: 신고의 종류

+ object: 신고의 대상

+ description: 신고하는 구체적인 사유

+ confirmButton: 신고 완료 버튼

B. Methods

+ report(): 해당 신고 내역을 백엔드에 전송한다.

Sequence Diagram

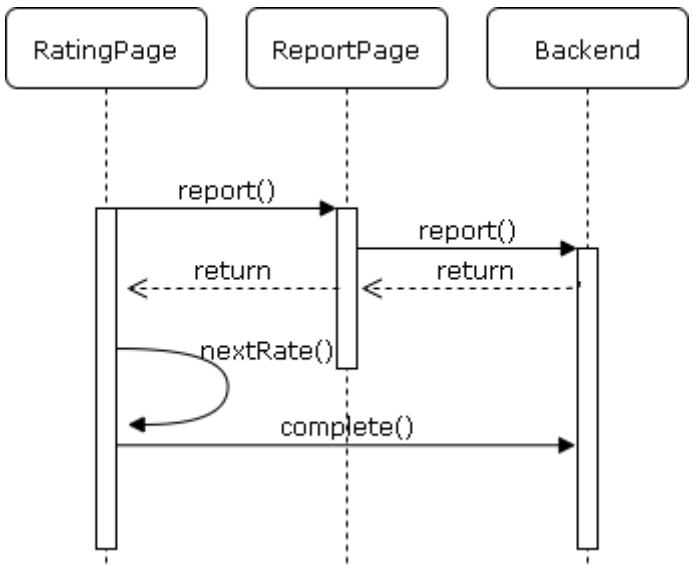

Figure 16. Rating & Report System Sequence Diagram

5. System Architecture - Backend

5.1. Objective

이번 챕터에서는 User와의 직접적인 상호작용을 담당하는 Frontend에서 요청을 받아 처리하는 Backend 시스템의 각 컴포넌트 구조에 대해 설명한다.

5.2. Components

5.2.1. Controller

Class Diagram

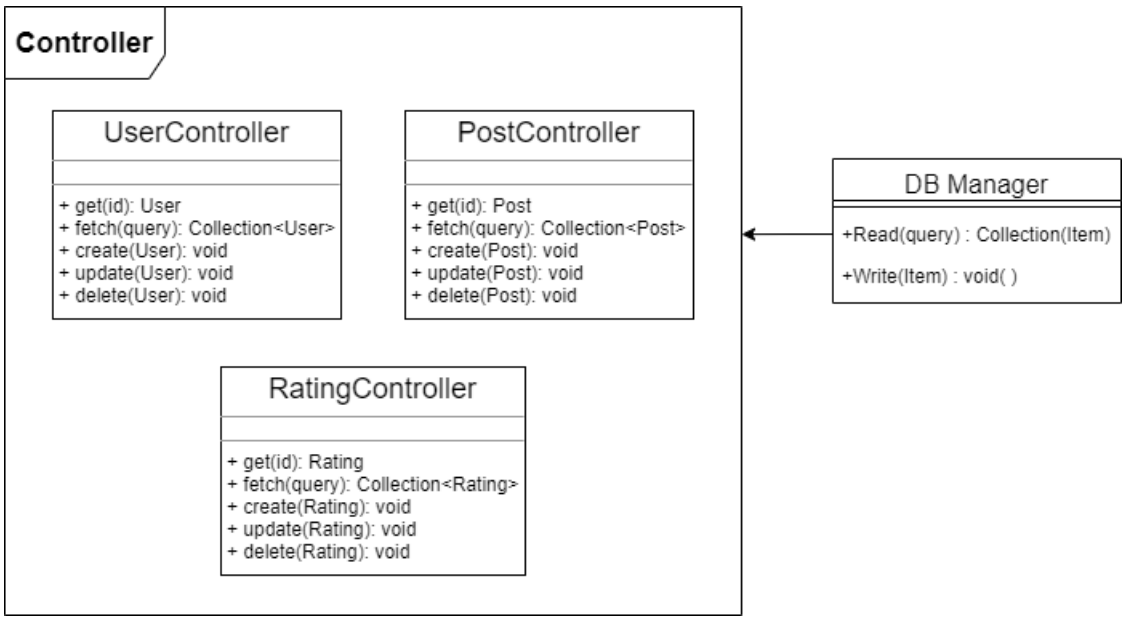

Figure 17. Controller Class Diagram

A. Method Description

get[entity](entity_id) : 개체의 정보를 받아온다

fetch[entity](query) : 쿼리 조건에 맞는 개체 리스트를 받아온다

create[entity](entity_id) : 개체를 DB에 추가하는 역할을 한다

update[entity](entity_id) : 개체의 정보를 수정한다

delete[entity](entity_id) : 개체를 DB에서 삭제한다

5.2.2. Rating System

Class Diagram

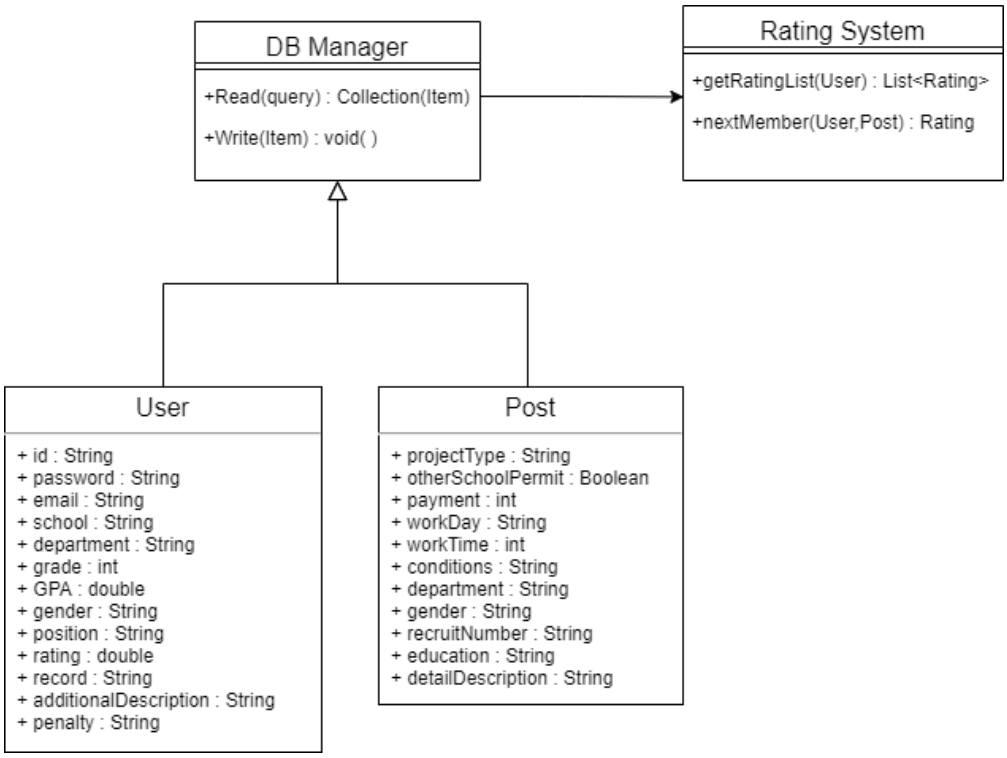

Figure 18. Rating System Class Diagram

A. Method Description

getRatingList(User) : 초기에 User가 평가해야할 Rating 목록을 받아온다

nextMember(User, Post) : 프론트에서 평가를 마치면 다음 평가해야할 팀원의 정보를 불러온다

Sequence Diagram

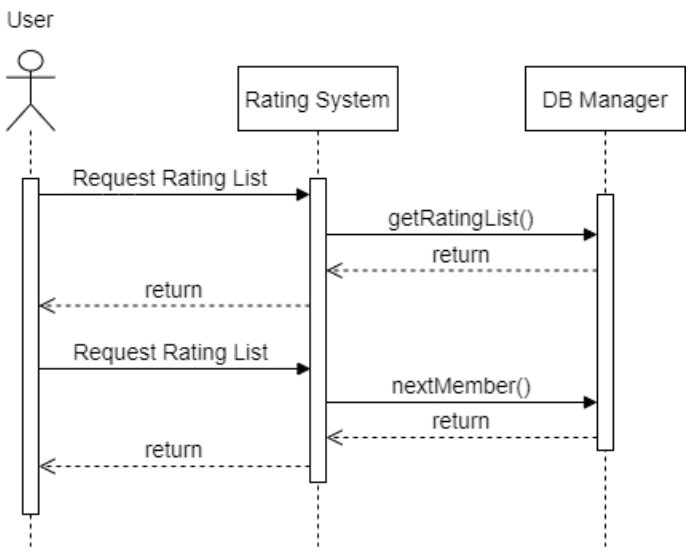

Figure 19. Rating System Sequence Diagram

5.2.3. User Authentication System

인증메일을 통해서 User가 소속대학의 구성원임을 인증해주는 시스템

Class Diagram

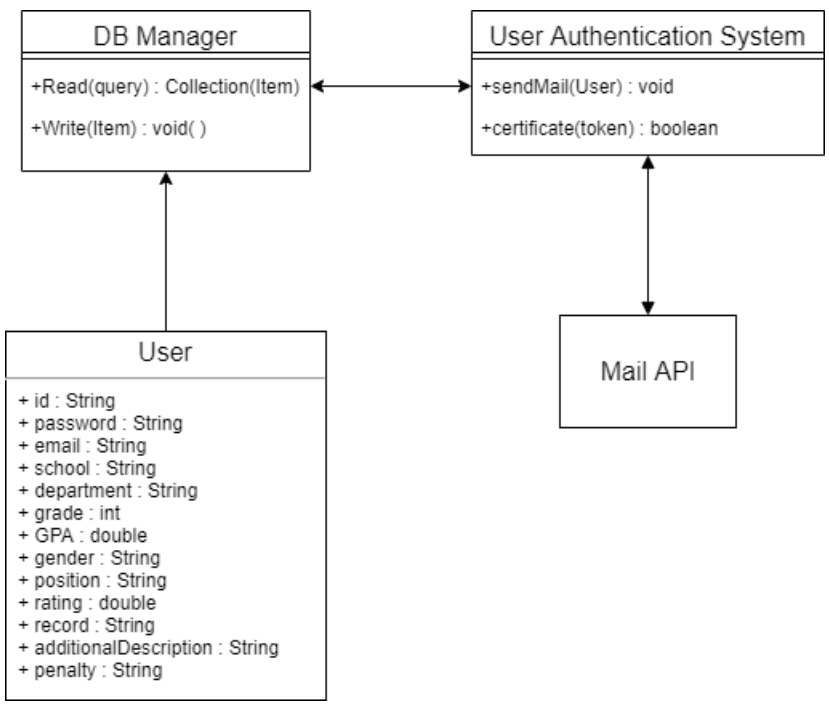

Figure 20. User Authentication System Class Diagram

A. Method Description

sendMail(User) : User의 email을 받아 Mail API에 랜덤으로 생성한 인증토큰을 전송하도록 요청한다

certificate(User) : 프론트에서 User가 email로 받은 토큰을 검증한다

Sequence Diagram

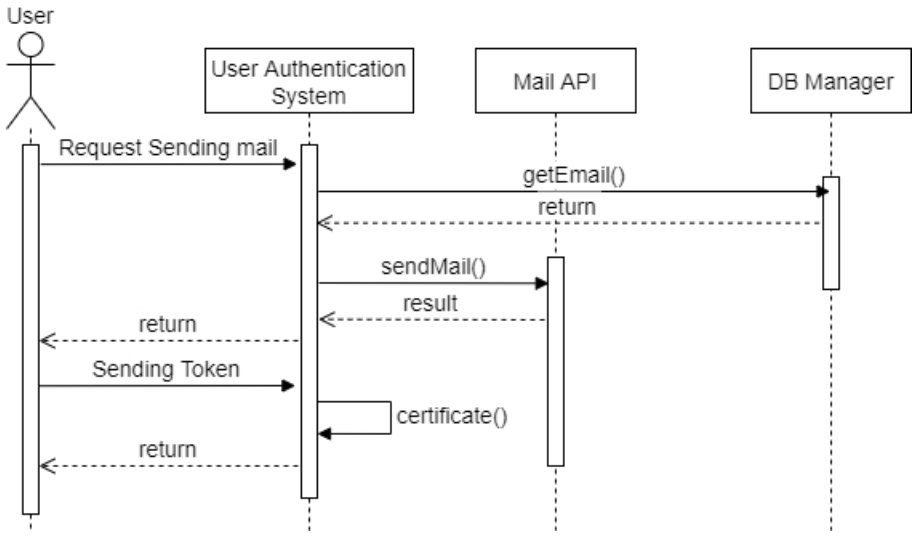

Figure 21. User Authentication System Sequence Diagram

5.2.4. Penalty System

User의 Report 건수를 파악해 누적 건수에 비례하여 정지등의 제재를 부과하는 시스템

Class Diagram

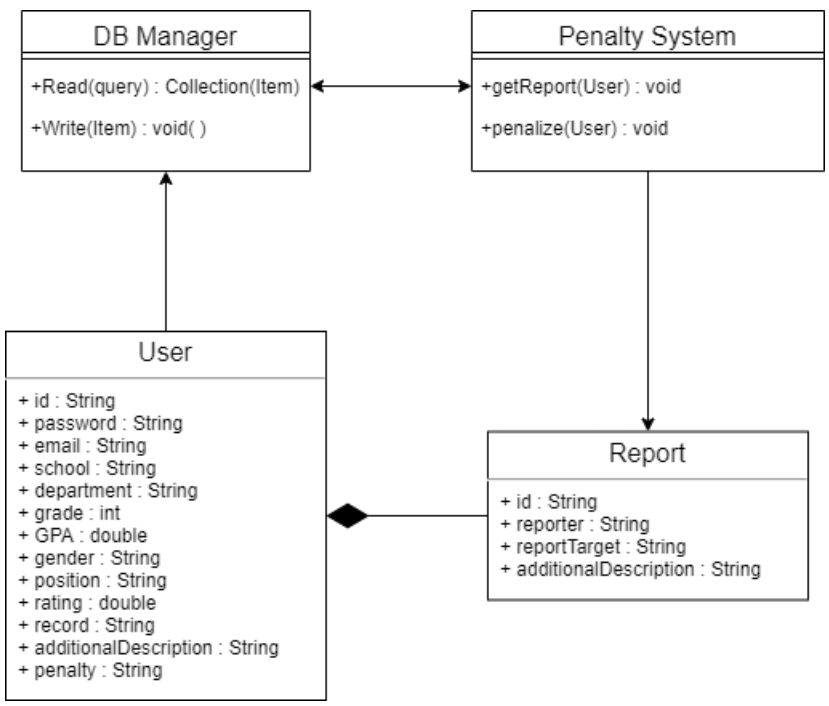

Figure 22. Penalty System Class Diagram

A. Method Description

getReport(User) : User의 Report를 받아 신고내역을 작성하도록 DB에 요청한다

penalize(User) : User가 Report된 횟수를 판단하여 사용정지 등의 제재를 적용한다

Sequence Diagram

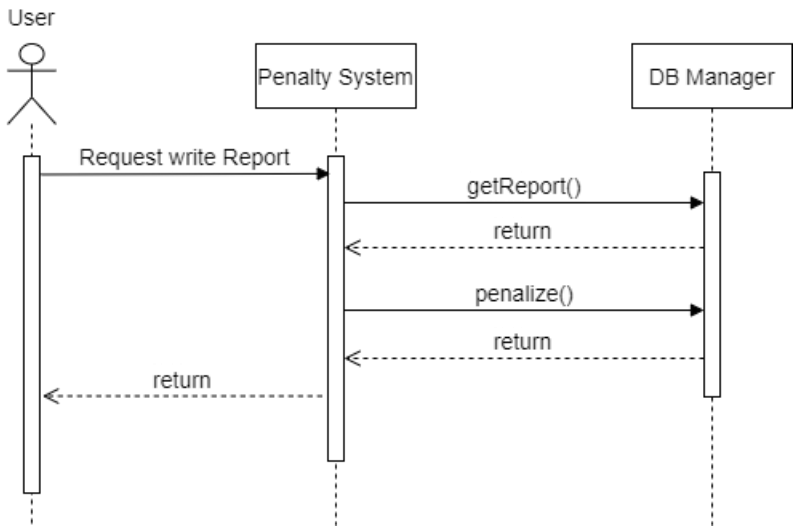

Figure 23. Penalty System Sequence Diagram

5.2.5. Post Search System

Class Diagram

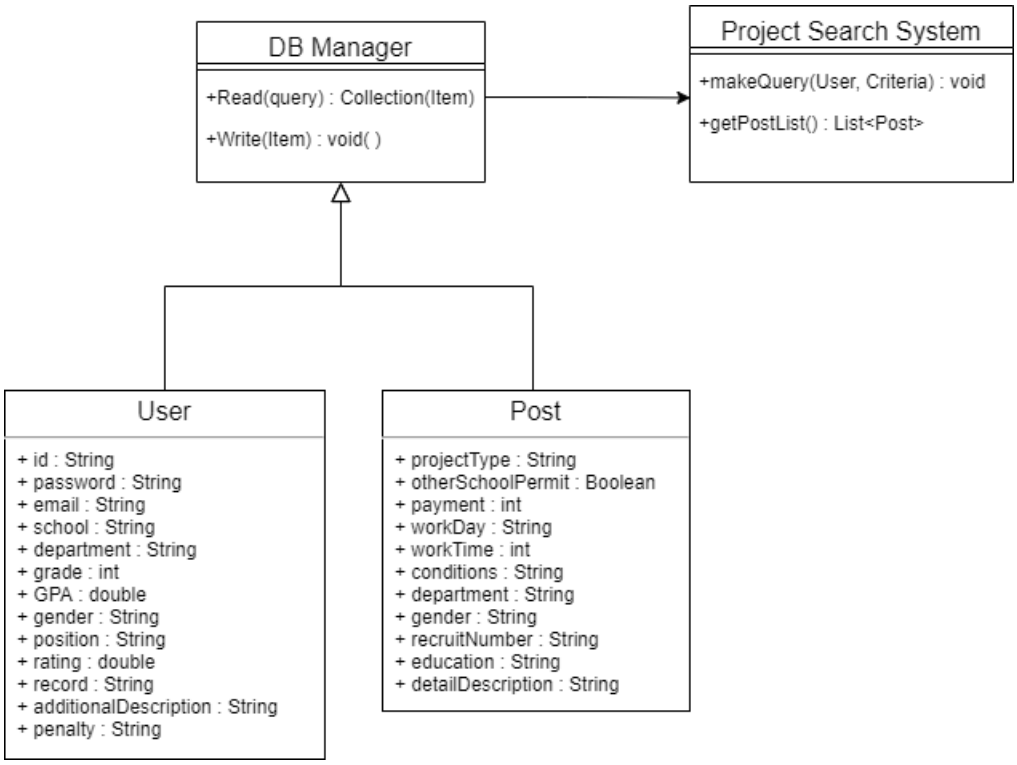

Figure 24. Project Search System Class Diagram

A. Method Description

makeQuery(User,Criteria) : User와 기준을 프론트엔드로 부터 받아서 쿼리를 작성한다

getPostList() : 만들어진 쿼리를 기반으로 sort하여 Post의 리스트를 DB에 요청한다

Sequence Diagram

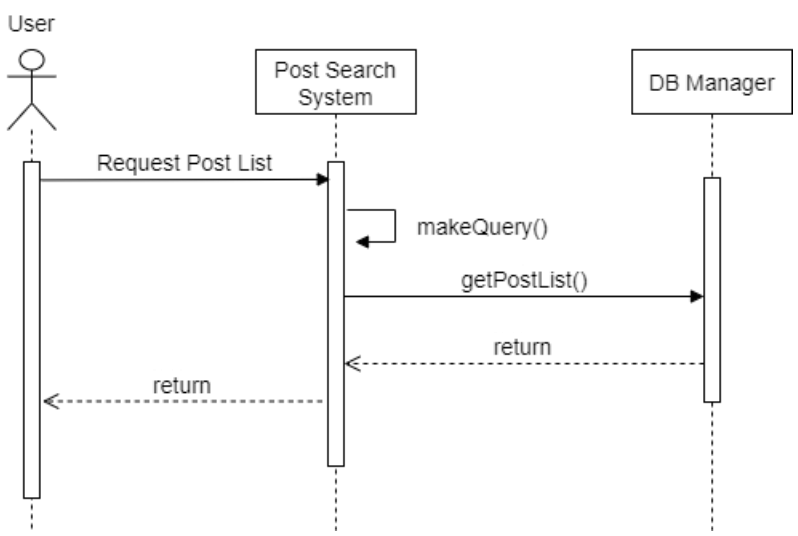

Figure 25. Project Search System Sequence Diagram

6. Protocol Design

6.1 Objectives

이 챕터에서는 각 서브시스템 간의 상호작용, 특히 Frontend와 Backend system간 상호작용에 사용되는 프로토콜에 어떠한 구조가 이용되는지 설명하고, 각각의 인터페이스가 어떻게 정의되는지 기술한다.

6.2 REST API

본 시스템의 Frontend와 Backend system 사이의 통신에는 HTTP웹 인터페이스를 사용하며, 요청과 응답의 형식은 REST API형식을 따른다. REST API란 Representational State Transfer의 약자로 서버에 저장되어 있는 각각의 자원을 이름으로 구분하여 해당 자원의 상태를 주고받는 API의 설계 형식을 의미한다. REST API는 다음의 장점을 가진다.

1) REST API 의 장점

a. client - server 구조

REST API 시스템에서, 서버는 API제공, 클라이언트는 사용자 인증이나 Context 등을 직접 관리하는 구조로 각각의 역할이 구분되어 있다. 따라서 서버와 클라이언트에서 개발해야 하는 내용의 구분이 명확하고, 서로 간의 의존성이 줄어든다.

b. Uniform Interface

Uniform Interface는 URI로 지정한 리소스에 대한 조작을 일관성 있게 통일된 인터페이스로 수행하게 하는 아키텍처 스타일을 가진다.

c. Stateless

REST는 작업을 위한 상태정보를 따로 저장하고 관리하지 않기 때문에, API 서버는 들어오는 요청만 단순히 처리한다. 따라서 시스템 서비스가 자유로워질 수 있고, 서버에서는 불필요한 정보를 관리할 필요가 없기 때문에 구현이 단순하다.

d. Cacheable

REST API의 가장 큰 장점은 기존에 존재하는 HTTP 형식을 사용하기 때문에 HTTP가 가진 캐싱 기능이 적용 가능하다. HTTP 프로토콜 표준에서 사용하는 Last-Modified태그나 E-Tag를 이용하면 캐싱 구현이 가능하다.

e. Self-descriptiveness

REST API의 메시지만 보고도 이를 쉽게 이해할 수 있는 자체 표현 구조로 되어있다.

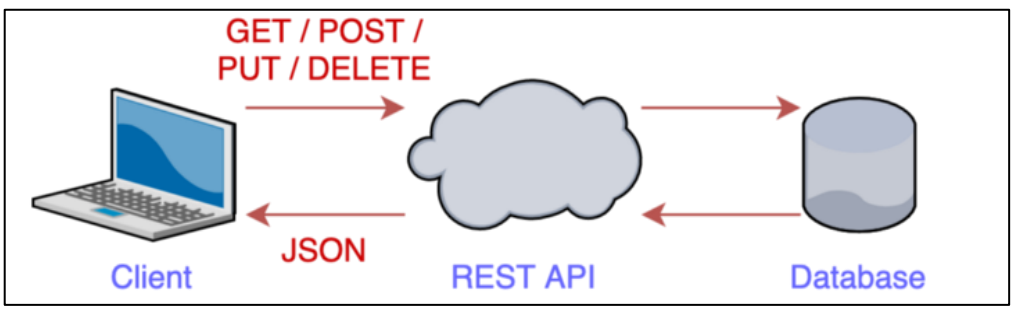

Figure 26. REST API diagram

REST API는 다음의 세 부분으로 구성된다.

2) REST API 의 구성

a. 자원 (Resource) : URL

서버가 보관하고 있는 데이터를 뜻하며, 각 자원은 고유한 URL을 가진다.

b. 행위 (Verb) : HTTP Method

서버의 자원에 접근해 상태를 조작하기 위한 요청을 뜻하며, 각 조작 행위는 HTTP Method를 통해 표현된다.

c. 표현 (Representation) : JSON

클라이언트의 요청에 대한 서버의 응답 형식을 뜻하며 주로 JSON이 사용된다.

- ❏ JSON 표기법은 인간이 읽을 수 있도록 데이터 교환용으로 설계된 경량 텍스트 기반 개방형 표준 형식이다. JSON 은 프로그램 작성에서 많이 사용하는 C, C++, Python, JAVA 등 많은 프로그래밍 언어에서 이용할 수 있다.

6.3 Details

1) Authorization

a. Signup

- Request

Method	POST	
URI	authorization/signup	
Parameters	ID	사용자 ID
	Password	사용자 비밀번호
	Name	사용자 이름
	University	소속 학교
	Phone-number	연락처
	E-mail	이메일 주소

Table 1. authorization/signup Request

- Response

Success Code	200 OK	
Failure Code	400 Bad Request (회원 정보를 잘못 입력했을 때)	
Success Response Body	URL	이메일 인증 URL
Failure Response Body	message	실패한 이유 출력

Table 2. authorization/signup Response

b. Login

- Request

Method	GET	
URI	authorization/login	
Parameters	ID	사용자 ID
	Password	사용자 비밀번호

Table 3. authorization/login Request

- Response

Success Code	200 OK	
Failure Code	400 Bad Request (회원 목록에 등록되어 있지 않는 사용자 정보)	
Success Response Body	URL	소속 인증 URL
Failure Response Body	message	실패한 이유 출력

Table 4. authorization/login Response

2) User

a. My Page

- Request

Method	GET	
URI	user/mypage/:id	
parameters	-	-
Header	Authorization	사용자 인증 토큰

Table 5. User/mypage Request

- Response

Success Code	200 OK	
Failure Code	404 Not Found	
Success Response Body	user data	사용자 상세 정보
Failure Response Body	message	실패한 이유 출력

Table 6. User/mypage Response

b. Edit My Page

- Request

Method	PUT	
URI	user/mypage/edit	
parameters	ID	사용자 ID
	Password	사용자 비밀번호
	Name	사용자 이름
	University	소속 학교
	Phone-number	연락처
	E-mail	이메일 주소
Header	Authorization	사용자 인증 토큰

Table 7. User/mypage/edit Request

- Response

Success Code	200 OK
Failure Code	400 Bad Request (잘못된 형식의 정보를 입력)

Success Response Body	user data	변경한 사용자 상세 정보
Failure Response Body	message	실패한 이유 출력

Table 8. User/mypage/edit Response

3) Post

a. Search

- Request

Method	GET	
URI	post/search	
parameters	-	-
Header	Authorization	사용자 인증 토큰

Table 9. Post/search Request

- Response

Success Code	200 OK	
Failure Code	404 Not Found (등록된 post가 없을 때)	
Success Response Body	post data	공고글 정보
Failure Response Body	message	실패한 이유 출력

Table 10. Post/search Response

b. Enroll post

- Request

Method	POST	
URI	post/enroll	
	Recruitment-information	모집 정보

parameters	Details	상세 정보
	Recruiter-information	모집자 정보
Header	Authorization	사용자 인증 토큰

Table 11. Post/enroll Request

- Response

Success Code	200 OK	
Failure Code	400 Bad Request (잘못된 post 정보를 입력)	
Success Response Body	-	-
Failure Response Body	message	실패한 이유 출력

Table 12. Post/enroll Response

c. Recommend post

- Request

Method	GET	
URI	post/recommend	
parameters	-	-
Header	Authorization	사용자 인증 토큰

Table 13. Post/recommend Request

- Response

Success Code	200 OK	
Failure Code	404 Not Found (추천 post가 없을 때)	
Success Response Body	Recommend post data	사용자에게 적합한 추천 공고글 정보

Failure Response Body	message	실패한 이유 출력

Table 14. Post/recommend Response

d. matching

- Request

Method	POST	
URI	post/matching	
parameters	Details	공고글 매칭 상세 정보
Header	Authorization	사용자 인증 토큰

Table 15. Post/matching Request

- Response

Success Code	200 OK	
Failure Code	400 Bad Request	
Success Response Body	-	-
Failure Response Body	message	실패한 이유 출력

Table 16. Post/matching Response

4) Apply history

a. Get

- Request

Method	GET	
URI	apply history/get	
parameters	-	-

Header	Authorization	사용자 인증 토큰

Table 17. applyhistory/get Request

- Response

Success Code	200 OK	
Failure Code	404 Not Found (apply history가 없을 때)	
Success Response Body	apply history data	지원 상세 정보
Failure Response Body	message	실패한 이유 출력

Table 18. applyhistory/get Response

5) Search history

a. Get

- Request

Method	GET	
URI	search history/get	
parameters	-	-
Header	Authorization	사용자 인증 토큰

Table 19. searchhistory/get Request

- Response

Success Code	200 OK	
Failure Code	404 Not Found (search history가 없을 때)	
Success Response Body	search history data	검색 상세 정보

Failure Response Body	message	실패한 이유 출력

Table 20. searchhistory/get Response

6) Enroll history

a. Get

- Request

Method	GET	
URI	enroll history/get	
parameters	-	-
Header	Authorization	사용자 인증 토큰

Table 21. enrollhistory/get Request

- Response

Success Code	200 OK	
Failure Code	404 Not Found (enroll history가 없을 때)	
Success Response Body	enroll history data	공고글 등록 상세 정보
Failure Response Body	message	실패한 이유 출력

Table 22. enrollhistory/get Response

7) Report

a. Enroll report

- Request

Method	POST	
URI	report/enroll	
parameters	Detail	신고 내용

Header	Authorization	사용자 인증 토큰

Table 23. report/enroll Request

- Response

Success Code	200 OK	
Failure Code	400 Bad Request (잘못된 형식의 신고를 입력)	
Success Response Body	-	-
Failure Response Body	message	실패한 이유 출력

Table 24. report/enroll Response

8) Rating

a. Enroll rate

- Request

Method	POST	
URI	Rating/enroll	
parameters	Rating	사용자 평가
Header	Authorization	사용자 인증 토큰

Table 25. rating/enroll Request

- Response

Success Code	200 OK	
Failure Code	400 Bad Request (잘못된 형식의 사용자평가)	
Success Response Body	-	-
Failure Response Body	message	실패한 이유 출력

Table 26. rating/enroll Request

b. Get

- Request

Method	GET	
URI	Rating/get	
parameters	-	-
Header	Authorization	사용자 인증 토큰

Table 27. rating/enroll Request

- Response

Success Code	200 OK	
Failure Code	404 Not Found (등록된 사용자 평가가 없을 때)	
Success Response Body	Rating data	사용자 평가 정보
Failure Response Body	message	실패한 이유 출력

Table 28. rating/enroll Response

7. Database Design

7.1. Objectives

이 파트에서는 시스템 데이터 구조와 이러한 구조가 데이터베이스에 표시되는 방법에 대해 설명한다. 먼저 개체-관계 다이어그램 (ER-diagram)을 통해 개체와 해당 관계를 식별한 다음 관계형 스키마 및 SQL DDL (Data Description Language)를 생성한다.

7.2. ER-Diagram

이 시스템은 User, Post, Search History, Enroll History 및 Sign-Up History의 5 개 항목으로 구성된다. ER-다이어그램은 각 개체를 직사각형으로, 관계를 마름모로 표현한다. 한 개체가 다른 개체와 여러 관계를 가질 때 삼지창 모양의 세 줄을 사용하여 이를 나타낸다. 한 개체가 다른 개체와 하

나의 관계만 있는 경우 십자선)을 사용하여 이를 나타낸다. 개체의 속성은 타원으로 표현하였다. 개체를 고유하게 식별하는 고유 속성에는 밑줄이 표시하였다. 엔티티에 동일한 속성이 여러 개 있는 경우 해당 속성은 이중 경계선이 있는 타원으로 표시하였다.

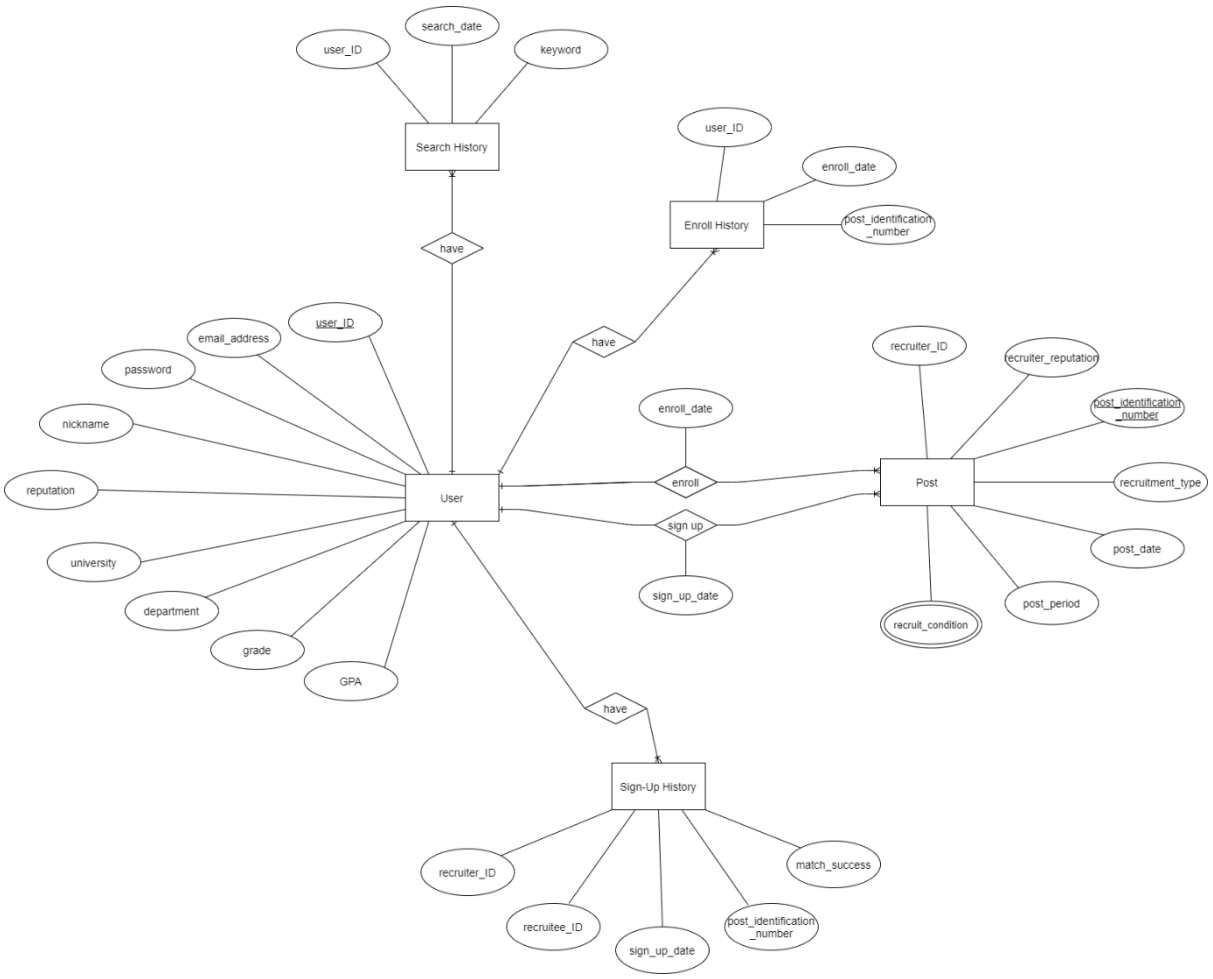

Figure 27. ER-diagram

7.2.1. Entities

7.2.1.1. User

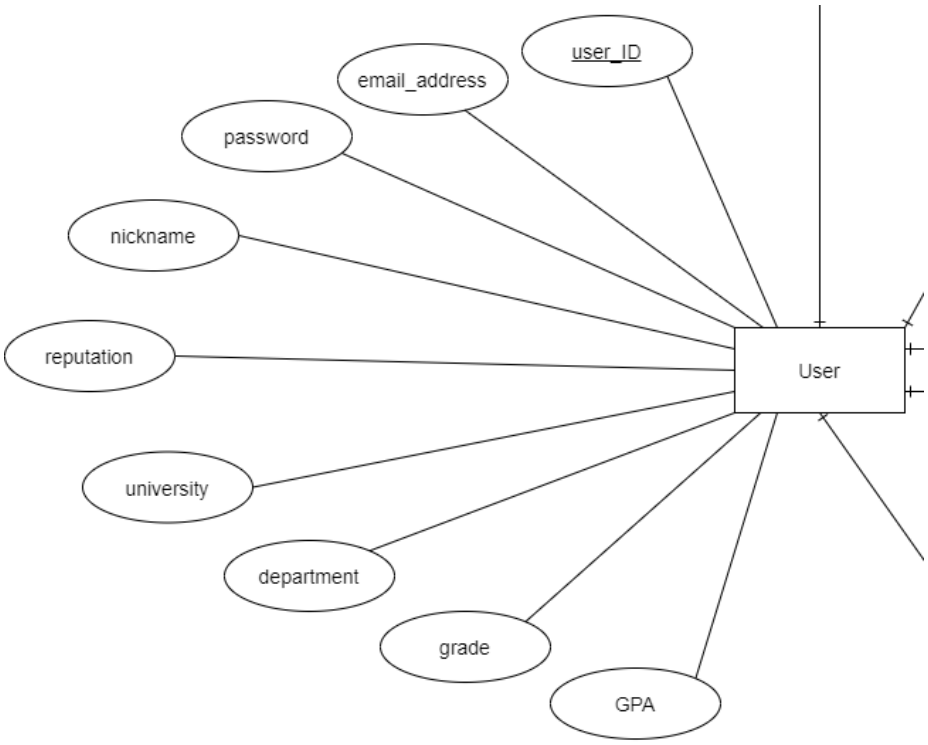

Figure 28. ER diagram, Entity, User

User 개체는 Connect-U service의 사용자를 나타낸다. User 개체의 특성은 user_ID, email_address, password, nickname, reputation, university, department, grade, 그리고 GPA로 구성된다. user_id 속성은 기본키(primary key)이다. User 개체에는 여러 Search History, Enroll History 및 Sign-Up History 개체들과 관계를 가질 수 있다. 또한 Post 개체와 등록 / 가입 관계가 있다.

7.2.1.2 Search History

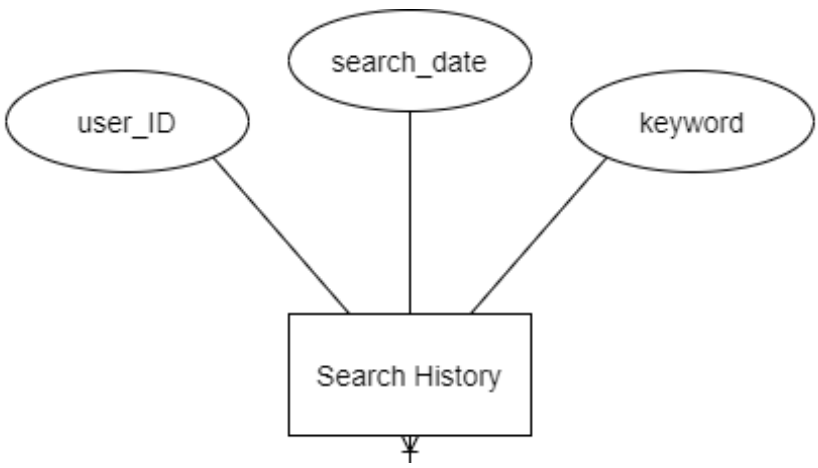

Figure 29. ER diagram, Entity, Search History

Search History 개체는 사용자의 검색 기록을 나타낸다. 따라서 Search History 개체는 User 개체와 관계를 가질 수 있다. Search History 개체의 여러 특성들 중 하나인 search_date는 사용자가 키워드를 검색 한 날짜를 나타낸다. Search History 개체는 User 개체와는 다르게 기본키가 없다. 대신, user_ID를 포함하는 복합키(composite key)로 찾을 수 있다.

7.2.1.3 Enroll Histrory

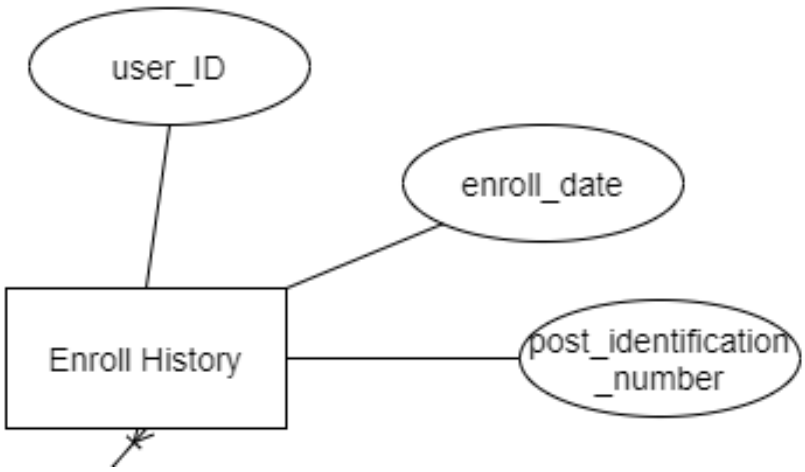

Figure 30. ER diagram, Entity, Enroll History

Enroll History는 사용자의 게시물 등록 내역을 나타낸다. Enroll History 개체 또한 User 개체와 관계를 가질 수 있다. Enroll History 개체의 여러 특성들 중 하나인 enroll_date는 사용자가 모집 게시물을 등록한 날짜를 나타낸다. 또한, post_identification_number는 다른 게시물과 구별되는 모집 게시물의 번호를 나타낸다. Search History 개체와 마찬가지로 Enroll History 개체 또한 기본키가 없고 user_ID를 포함하는 복합 키로 찾을 수 있다.

7.2.1.4 Sign-Up History

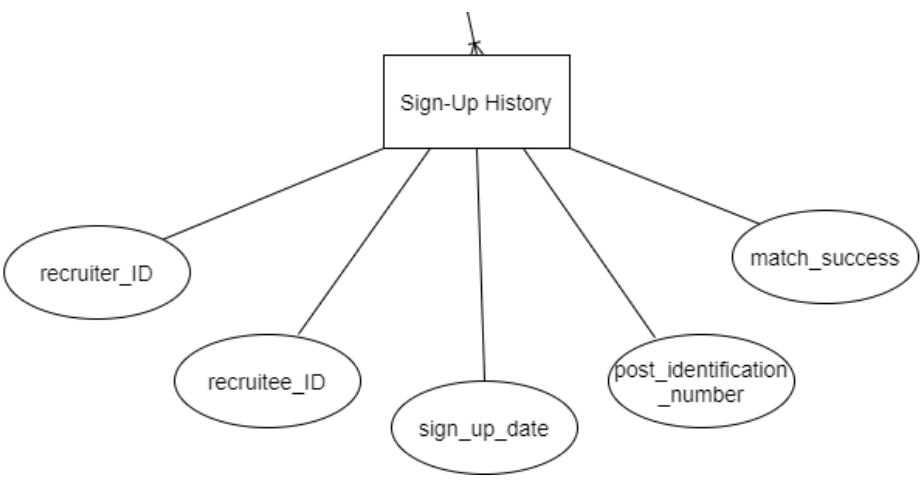

Figure 31. ER diagram, Entity, Sign-Up History

Sign-Up History 사용자의 게시물 지원 내역을 나타낸다. Sign-Up History 개체 또한 User 개체와 관계가 있을 수 있다. Sign-Up History 개체의 여러 특성들 중 하나인 sign_up_date는 사용자가 채용 게시물에 지원한 날짜를 나타낸다. 그리고, post_identification_number는 다른 게시물과 구별되는 모집 게시물의 번호를 나타낸다. recruiter_ID는 해당 글을 등록하여 게시한 사용자의 ID를 나타내고, recruitee_ID는 해당 게시물에 지원한 사용자의 ID를 나타낸다. match_succes는 recruiter와 recruitee가 성공적으로 매칭이 되었는지 여부를 나타낸다. 마찬가지로 Sign-Up History 개체 또한 기본키가 없고 user_ID를 포함하는 복합키로 찾을 수 있다.

7.2.1.5 Post

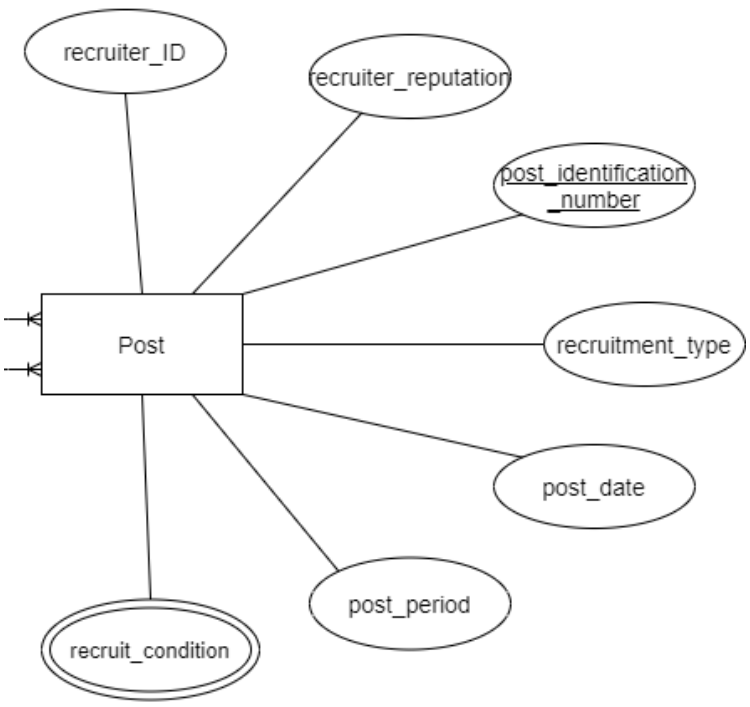

Figure 32. ER diagram, Entity, Post

Post 개체에는 일자리-인력 매칭에 필요한 정보가 포함되어 있다. Post 개체의 특성인 post_identification_number는 이 개체의 기본키이다. 이 특성으로 게시물을 식별한다. recruiter_ID는 해당 모집글을 게시한 사용자의 ID이다. recruiter_reputation 특성은 recruiter의 평점을 나타낸다. recruit_type 특성은 연구, 프로젝트, 스터디 또는 공모전과 같은 모집 유형을 나타낸다. post_date는 recruiter가 공지글을 게시 한 날짜를 나타내고 post_period는 해당 글을 게시할 수 있는 기간을 나타낸다. 모집 조건을 나타내는 Post 개체의 recruit_condition 여러 특성을 가질 수 있다.

7.3 Relational Schema

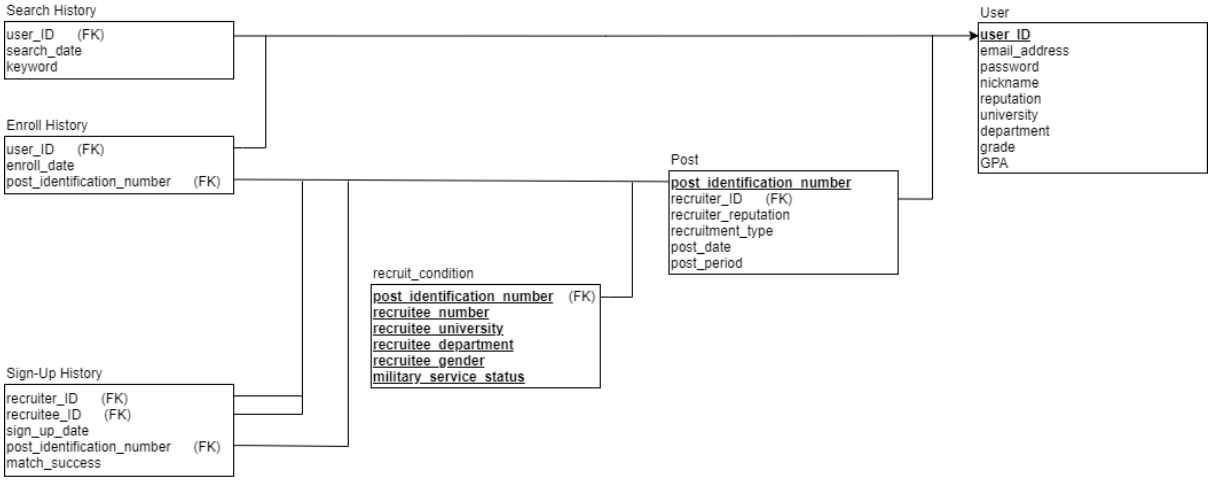

Figure 33. Relational Schema

7.4 SQL DDL

7.4.1 User

```
Create TABLE User
(
    user_ID INT NOT NULL,
    email_address INT NOT NULL,
    password INT NOT NULL,
    nickname INT NOT NULL,
    reputation INT NOT NULL,
    university INT NOT NULL,
    department INT NOT NULL,
    grade INT NOT NULL,
    GPA INT NOT NULL,
    PRIMARY KEY (user_ID)
);
```

7.4.2 Search History

```
Create TABLE Seach_History
(
      user_ID INT NOT NULL,
      search_date INT NOT NULL,
      keyword INT NOT NULL,
      FOREIGN KEY (user_ID) REFERNCES User(user_ID)
);
```

7.4.3 Enroll History

```
Create TABLE Enroll_History
(
      user_ID INT NOT NULL,
      enroll_date INT NOT NULL,
      post_identification_number INT NOT NULL,
      FOREIGN KEY (user_ID) REFERNCES User(user_ID),
      FOREIGN KEY (post_identification_number) REFERNCES Post(post_identification_number)
);
```

7.4.4 Sign-Up History

```
Create TABLE Sign_Up_History
(
      recruiter_ID INT NOT NULL,
      recruitee_ID INT NOT NULL,
      sign_up_date INT NOT NULL,
      post_identification_number INT NOT NULL,
      match_success INT NOT NULL,
      FOREIGN KEY (recruiter_ID) REFERNCES User(user_ID),
      FOREIGN KEY (recruitee_ID) REFERNCES User(user_ID),
      FOREIGN KEY (post_identification_number) REFERNCES Post(post_identification_number)
);
```

7.4.5 Post

```
Create TABLE Post
(
      recruiter_ID INT NOT NULL,
      recruiter_reputation INT NOT NULL,
      post_identification_number INT NOT NULL,
      recruitment_type INT NOT NULL,
      post_date INT NOT NULL,
      post_period INT NOT NULL,
      PRIMARY KEY (post_identification_number),
      FOREIGN KEY (recruiter_ID) REFERNCES User(user_ID)
);
```

7.4.6 recruit_condition

```
Create TABLE recruit_condition
(
      post_identification_number INT NOT NULL,
      recruitee_number INT NOT NULL,
      recruitee_university INT NOT NULL,
      recruitee_department INT NOT NULL,
      recruitee_gender INT NOT NULL,
      military_service_status INT NOT NULL,
      PRIMARY KEY (post_identification_number, recruitee_number, recruitee_university,
recruitee_department, recruitee_gender, military_service_status),
      FOREIGN KEY (post_identification_number) REFERNCES Post(post_identification_number)
);
```

8. Testing Plan

8.1. Objectives

이 장에서는 본 시스템을 개발 완료하고 배포하기 전에 시스템이 기능적, 비기능적 요구사항들을 충족하였는지 그리고 배포 이후에 발생될 수 있는 문제점들을 확인하기 위한 Development testing, Release testing, User testing에 대한 계획을 기술한다.

8.2. Testing Policy

8.2.1. Development Testing

Development testing은 개발하는 과정이나 실제 배포 시 발생할 다양한 문제들을 찾아내고 해결하기 위한 테스트 과정이다. 이 단계에서는 소프트웨어가 충분한 테스트를 거치지 않아 불안정할 수 있기 때문에 static code analyzing, peer code review, unit testing, data flow analyzing 등을 수행해야 한다. 이 과정을 거치면서 시스템이 사용자가 원하는 수준의 performance, reliability, security를 맞추는 데에 목표를 하고 있다.

A. Performance

Connect-U 시스템에 많은 수의 동시 접속자가 있을 수 있고, 많은 양의 사용자 정보, 프로젝트 정보가 Database에서 front-end로 호출되어야 할 것이다. 따라서 많은 수의 동시 접속자가 있어도 시스템이 느려지지 않게 하는 것이 중요하다. 또한 다양한 접속자가 많은 양의 정보를 원함으로 Database 구조의 최적화에 대한 평가가 필요할 것이다. 마지막으로 recommendation algorithm을 최적화하여 2초 내에 사용자에게 추천 목록을 제공할 수 있게 해야 한다.

B. Reliability

시스템 사용자가 시스템을 사용할 때 시스템이 다운되거나 오류가 생기지 않아야 한다. 따라서 시스템 컴포넌트 개발 단계부터 integration단계까지 차례대로 unit test, integrate test를 통해 system reliability를 증가시켜야 한다.

C. Security

이 시스템은 대학교 인증을 해야만 시스템 사용이 가능하다. 따라서 대학교 인증을 할 때에 사용자의 대학교가 정확히 맞는지 확인하는 인증 시스템을 사용해야 한다. 또한 사용자의 개인정보는 데이터베이스에 저장되며 이 때 비밀번호와 같은 민감한 정보는 암호화되어 저장된다. 이렇게 암호화된 정보는 DBA에게만 접근 권한을 주어 시스템의 security를 증가시킨다.

8.2.2. Release Testing

기능적으로 뛰어난 시스템이더라도 release를 잘못된 방식으로 한다면 사용자의 요구를 만족하지 못할 수 있다. 따라서 시스템과 시장의 연결을 위해서 release test가 필요하다. 시스템의 새로운 버전이 출시되어 적용시켜야 할 때, 새로운 버전의 시스템이 기존의 시스템과 충돌이 없으며 각 요구사항들을 잘 충족하는지 테스트한다.

8.2.3. User Testing

시스템을 시장에 release 하기 전에 특정 사용자들에게 먼저 배포해서 사용자의 입장에서 일어날 수 있는 오류들을 검토하여야 한다.

8.2.4. Testing Case

시스템 사용자가 처음 시스템 회원 가입을 하고, 인증을 받고, 프로젝트를 생성하거나 프로젝트에 등록하는 일련의 과정에 대한 테스트가 진행된다.

9. Development Plan

9.1. Objectives

이 장에서는 시스템을 구현하는 데 사용되는 개발 도구와 프로그래밍 언어, 라이브러리, 플랫폼 등의 개발 환경에 대해 설명한다.

9.2. Frontend Environment

A. Adobe Photoshop

Figure 34. Adobe Photoshop

어도비 포토샵(Adobe Photoshop)은 미국의 어도비 시스템즈사에서 개발한 레스터 그래픽 편집기이다. 이 프로그램을 통해 시스템의 레이아웃과 아이콘을 생성해 시스템에 미적인

디자인을 추가할 수 있다.

B. Zeplin

Figure 35. Zeplin

제플린(Zeplin)은 디자이너 및 개발자를 위한 공동 작업 응용 프로그램이다. 제플린은 스케치 또는 포토샵과 연동하여 자동으로 작업한 결과물을 이미지 파일 Asset과 디자인 가이드로 생성해 준다.
제플린은 가이드를 생성하고 요소의 크기를 확인하는 이 모든 과정을 자동화하여 시간과 노력을 절약해준다 또한 모바일 해상도별 가이드를 일일이 수 작업할 필요도 없다. 특히 스케치와 포토샵과도 연동할 수 있기 때문에 IOs와 Windows 사용자가 사용할 수 있다.

C. Flutter with Android Studio

Figure 36. Flutter with Android Studio

Flutter는구글이 개발한 오픈소스 모바일 UI 프레임워크로 동일한 코드 기반에서 native-looking Android 및 iOS 애플리케이션을 구축하는 데 사용할 수 있다. Flutter는 자바나 코틀린 언어 대신,

Dart 프로그래밍 언어를 사용한다. 마지막으로 hot reload를 지원함으로써 빠른 개발 환경을 제공한다.

9.3. Backend Environment

A. Github

Figure 37. Github

깃허브(GitHub)는 분산 버전 컨트롤 소프트웨어 깃(Git)을 기반으로 소스 코드를 호스팅 하고, 협업 지원 기능들을 지원하는 Microsoft의 웹서비스이다. 깃허브를 이용해 여러 팀원들이 프로젝트를 함께 개발할 수 있고, 구성요소들을 쉽게 합칠 수 있다. 또한 이것은 시스템의 형상 관리도 용이하게 하여 업데이트 된 시스템을 release하거나 rollback하는데에도 도움을 준다.

B. Firebase

Figure 38. Firebase

Firebase 는구글에서제공하는 Native 및 PWA 개발플랫폼이다. 이것은 **인증, 데이터베이스, 푸시 알람, 스토리지, API 등 모든 것을** 프로젝트 구축 시, 자동적으로 만들어 준다. 또한 서버를

구축하기 위해서 리눅스 명령어를 알 필요도 없고, 도메인을 구입할 필요도 없으며 개발하는 동안에는 서버를 구입할 필요도 없다. **FireBase** 는 **백엔드 기능**을 클라우드 서비스 형태로 제공하기 때문에 **서버리스 애플리케이션 개발이 가능**하다.

9.4. Constraints

시스템은 이 문서에 언급된 내용을 기반으로 설계 및 구현된다. 하지만 기타 세부사항은 개발자별로 선호하는 방식들로 구현할 수 있다. 그럼에도 불구하고, 개발자는 다음 사항을 준수해야 한다.

- 널리 검증된 기술을 사용해야 한다.
- 시스템 퍼포먼스가 향상되는 방향으로 개발해야 한다.
- 가능하면 open source기반의 소프트웨어를 reuse해야 한다.
- 개발자 친화적이 아닌 사용자 친화적인 방향으로 개발해야 한다.
- 시스템 개발 비용뿐만 아니라 유지보수 비용도 고려해야 한다.
- 시스템 자원 낭비를 줄이는 방향으로 소스 코드를 최적화해야 한다.
- Android와 IOs 운영체제 기반으로 개발해야 한다.

9.5. Assumptions and Dependencies

이 문서의 모든 시스템은 Android 및 IOs 장치를 기반으로 설계되고 구현된다는 가정 하에 작성되었다. 따라서 다른 운영체제에서는 시스템이 작동되지 않을 수 있다.

10. Supporting Information

10.1. Software Design Specification

이 문서는 IEEE Recommendation (IEEE Recommended Practice for Software Design Description, IEEE-Std-1016) 형식에 맞추어 작성되었다.

10.2. Figures

Figure 1. Use case diagram의 예.....p.10

Figure 2. Sequence diagram의 예.....p.11

Figure 3. Class diagram의 예 p.11

Figure 4. ER diagram의 예 p.12

Figure 5. Power Point p.12

Figure 6. Draw.io p.13

Figure 7. Frontend context diagram p.14

Figure 8. Architecture Diagram - Backend p.15

Figure 9. Registration & Authentication System Class Diagram p.16

Figure 10. Registration & Authentication System Sequence Diagram p.19

Figure 11. Profile System Class Diagram p.20

Figure 12. Profile System Sequence Diagram p.21

Figure 13. Community System Class Diagram p.22

Figure 14. Community System Sequence Diagram p.24

Figure 15. Rating & Report System Class Diagram p.25

Figure 16. Rating & Report System Sequence Diagram p.26

Figure 17. Controller Class Diagram p.27

Figure 18. Rating System Class Diagram p.28

Figure 19. Rating System Sequence Diagram p.28

Figure 20. User Authentication System Class Diagram p.29

Figure 21. User Authentication System Sequence Diagram p.30

Figure 22. Penalty System Class Diagram p.30

Figure 23. Penalty System Sequence Diagram p.31

Figure 24. Project Search System Class Diagram p.32

Figure 25. Project Search System Sequence Diagram p.32

Figure 26. REST API diagram p.34

Figure 27. ER-diagram p.45

Figure 28. ER diagram, Entity, User p.46

Figure 29. ER diagram, Entity, Search History....p.46
Figure 30. ER diagram, Entity, Enroll History....p.47
Figure 31. ER diagram, Entity, Sign-Up History....p.47
Figure 32. ER diagram, Entity, Post....p.48
Figure 33. Relational Schema....p.49
Figure 34. Adobe Photoshop....p.53
Figure 35. Zeplin....p.54
Figure 36. Flutter with Android Studio....p.54
Figure 37. Github....p.55
Figure 38. Firebase....p.55

10.3. Tables

Table 1. authorization/signup Request....p.35
Table 2. authorization/signup Response....p.35
Table 3. authorization/login Request....p.36
Table 4. authorization/login Response....p.36
Table 5. User/mypage Request....p.36
Table 6. User/mypage Response....p.37
Table 7. User/mypage/edit Request....p.37
Table 8. User/mypage/edit Response....p.38
Table 9. Post/search Request....p.38
Table 10. Post/search Response....p.38
Table 11. Post/enroll Request....p.39
Table 12. Post/enroll Response....p.39
Table 13. Post/recommend Request....p.39
Table 14. Post/recommend Response....p.40

Table 15. Post/matching Request.....p.40

Table 16. Post/matching Response.....p.40

Table 17. applyhistory/get Request.....p.41

Table 18. applyhistory/get Response.....p.41

Table 19. searchhistory/get Request.....p.41

Table 20. searchhistory/get Response.....p.42

Table 21. enrollhistory/get Request.....p.42

Table 22. enrollhistory/get Response.....p.42

Table 23. report/enroll Request.....p.43

Table 24. report/enroll Response.....p.43

Table 25. rating/enroll Request.....p.43

Table 26. rating/enroll Request.....p.43

Table 27. rating/enroll Request.....p.44

Table 28. rating/enroll Response.....p.44

10.4. Document History

Date	Version	Description	Writer
2021/5/9	0.1	Style and overview	최영우, 김연재
2021/5/12	1.0	Part 3, 4 추가	최영우
2021/5/12	2.0	Part 6 추가	김연재
2021/5/12	3.0	Part 1,2,8,9 추가	이인수
2021/5/13	4.0	Part 3, 5 추가	이광호
2021/5/15	5.0	Part 7 추가	김도현
2021/5/16	6.0(Final)	리뷰 및 최종 수정	이인수, 이광호, 김도현