Online Coding Class about Web Crawler

2022.05.15

소프트웨어공학개론 41
TEAM 2 (CrawlLearn)

Team Leader	이민영
Team Member	김진환
Team Member	배지현
Team Member	이정우
Team Member	정미서

CONTENTS

List of Figures

List of Tables

1. Introduction

이 프로젝트는 성균관대학교 학생들을 위한 파이썬 교육 프로그램을 만드는 목적을 가지고 있다. 이 교육 프로그램은 웹을 통하여 제공될 예정이며 비전공자 학생들의 눈높이에 맞는 교육이 이루어질 것이다. 파이썬의 기초 문법을 익힌 성균관대학교 신입생들이 알고리즘 교육을 통해 기른 컴퓨팅 사고력을 계속 유지하고 파이썬을 활용한 웹 크롤링을 익힘으로써 추후 실무에 나가서도 코딩 교육을 통해 배운 내용들을 활용할 수 있도록 하는 것이 이 교육 프로그램의 지향점이다. 이 디자인 명세서는 이 프로젝트 시행을 위해 사용된 혹은 사용될 디자인에 관한 설명을 담고 있다. 디자인은 이전에 쓰여진 요구사항 명세서(Software Requirements Specifications document)의 내용을 토대로 만들어졌다.

1.1. Objectives

이 장에서는 디자인 단계에서 우리가 사용한 다양한 기능들과 다이어그램에 대한 내용을 설명할 것이다.

1.2. Applied Diagrams

1.2.1. UML

UML 은 Unified Modeling Language 의 약자이다. 간단히 말해서, UML 은 소프트웨어를 모델링하고 문서화하는 현대적 접근 방식이다. 실제로 이것은 가장 인기 있는 비즈니스 프로세스 모델링 기술 중 하나이다. UML 은 비즈니스 분석가, 소프트웨어 설계자 및 개발자들이 기존의 혹은 새로운 비즈니스 프로세스의 구조 및 동작을 기술, 지정, 설계 및 문서화하는데 사용하는 공통 언어이다. UML 은 다양한 응용 분야에 적용될 수 있는데 그 예시로 은행, 금융, 인터넷, 항공우주, 의료 등이 있다. 소프트웨어 구성 요소의 도식 표현을 기반으로 하여 모든 주요 객체, 컴포넌트 소프트웨어 개발 방법 및 다양한 구현 플랫폼(예: J2EE, .NET)에 사용할 수 있다. "사진 한 장이 천 마디의 말보다 낫다"는 옛 속담에 있듯이 우리는 시각적 표현을 사용함으로써 소프트웨어나 비즈니스 프로세스에서 발생할 수 있는 결함이나 오류를 더 잘 이해할 수 있다.

1.2.2. Use Case Diagram

시스템의 cornerstone 은 시스템이 이행하는 functional requirement 이다. Use case Diagram 은 시스템의 high-level requirement 를 분석하는 데 사용된다. 이러한 요구사항은 다른 use case 를 통해 표현된다. UML 다이어그램의 세 가지 주요 구성 요소를 볼 수 있다. Functional requirements – use case 로 표현됨; 동작을 설명하는 동사. Actors – actor 는 시스템과 상호 작용한다. Actor 는 인간, 조직 또는 내부 또는 외부 응용 프로그램이 될 수 있다. Actor 와 use case 간의 관계 – 직선 화살표를 사용하여 나타낸다.

1.2.3. Sequence Diagram

Sequence diagram 은 컴퓨터 공학 커뮤니티뿐만 아니라 비즈니스 애플리케이션 개발을 위한 설계 수준 모델로서도 가장 중요한 UML 다이어그램이다. 최근에는 시각적으로 설명이 가능한 특성 때문에 비즈니스 프로세스를 묘사하는데 있어서도 인기가 있다. 이름에서 알 수 있듯이 Sequence diagram 은 행위자와 객체 간에 발생하는 메시지와 상호 작용의 순서를 설명한다. 모든 의사소통은 연대순으로 표현되며 Actor 또는 객체는 다른 객체가 그들과 통신하고자 할 때 등의 필요한 경우에만 활성화될 수 있다.

1.2.4. Class Diagram

UML 의 Class diagram 은 시스템의 클래스, 속성, 연산 및 객체 간의 관계를 보여줌으로써 시스템의 구조를 설명하는 정적 구조 다이어그램의 일종이다. 클래스가 객체의 구성 요소인 것처럼 Class diagram 은 UML 의 구성 요소이다. Class diagram 의 다양한 구성 요소는 실제로 프로그래밍 될 주 객체를 표현하거나 클래스와 객체 간의 상호 작용을 나타낼 수 있다. 클래스 모양 자체는 세 개의 행이 있는 직사각형으로 구성된다. 맨 위 행은 클래스의 이름을 포함하고, 가운데 행은 클래스의 속성을 포함하며, 맨 아래 섹션은 클래스에서 사용할 수 있는 method 또는 연산을 나타낸다. 클래스와 하위 클래스는 각 개체 간의 정적 관계를 보여주기 위해 함께 그룹화된다.

1.2.5. Context Diagram

Context diagram(레벨 0 DFD 라고도 함)은 data flow diagram 에서 highest level 이며 전체 의 context 와 경계를 설정하는 하나의 프로세스만 포함한다. 시스템과 외부 entity(Actor) 간의 정보 흐름을 식별한다. Context diagram 은 일반적으로 requirement

specification 에 포함된다. 이 명세서는 프로젝트의 모든 stakeholder 가 읽어야 하므로 stakeholder 들이 항목을 이해할 수 있도록 쉬운 언어로 작성되어야 한다. Context diagram 의 목적은 시스템의 요구 사항과 제약을 식별하는데 고려되어야 하는 외부 요인 및 사건에 주목하는 것이다. Context diagram 은 프로젝트 초기에 조사 대상의 범위를 결정하기 위해 종종 사용된다. 따라서, 문서 내에서, Context diagram 은 시스템과 상호 작용할 수 있는 모든 외부 entity 를 나타낸다. 전체 소프트웨어 시스템은 단일 프로세스로 표시되며 이 다이어그램에서 시스템은 중앙에 배치되고 (내부 구조에 대한 세부 정보는 없이) 모든 외부 entity, 상호 작용하는 시스템 및 환경에 의해 둘러싸여 있다.

1.2.6. Entity Relationship Diagram

Entity relationship diagram 은 데이터베이스에 저장된 entity set 의 관계를 보여준다. 이 entity 는 데이터의 구성 요소인 객체이고 entity set 는 유사한 entity 의 집합이다. 이러한 entity 에는 해당 속성을 정의하는 속성이 있을 수 있다. ER 다이어그램은 entity 와 속성을 정의하고 entity 간의 관계를 표시함으로써 데이터베이스의 논리적 구조를 보여준다. Entity relationship diagram 은 데이터베이스의 설계를 스케치하는 데 사용된다.

1.3. Applied tools

1.3.1. Microsoft PowerPoint

시스템의 cornerstone 은 시스템이 이행하는 functional requirement 이다. Use case Diagram 은 시스템의 high-level requirement 를 분석하는 데 사용된다. 이러한 요구사항은 다른 use case 를 통해 표현된다. UML 다이어그램의 세 가지 주요 구성 요소를 볼 수 있다. Functional requirements – use case 로 표현됨; 동작을 설명하는 동사. Actors – actor 는 시스템과 상호 작용한다. Actor 는 인간, 조직 또는 내부 또는 외부 응용 프로그램이 될 수 있다. Actor 와 use case 간의 관계 – 직선 화살표를 사용하여 나타낸다.

1.3.2. ERD cloud

Entity Relationship Diagram 을 빠르고 쉽게 만들 수 있는 웹 기반 데이터베이스 모델링 도구이다.

1.3.3. Figma

웹 제작 시 Frontend 디자인을 빠르고 쉽게 할 수 있도록 도와주는 웹 기반 모델링 도구이다.

1.4. References

이 디자인 명세서의 작성자는 다음과 같은 자료를 참조하였다.

- Team 2, 2020 Spring. Software Design Document, SKKU.

2. System Architecture – Overall

2.1. Objectives

이 장에서는 시스템의 구조를 나타낼 것이다. 이 프로젝트의 frontend 를 시작으로 backend 까지 상세히 설명할 예정이다.

2.2. System Organization

이 서비스는 클라이언트 - 서버 모델을 적용하여 설계되었으며, frontend 애플리케이션은 사용자와의 모든 상호작용을 담당하며, frontend 애플리케이션과 backend 애플리케이션은 JSON 기반의 HTTP 통신을 통해 데이터를 주고 받는다. Backend 애플리케이션은 유저의 요청을 frontend 에서 컨트롤러로 배포하고, 데이터베이스에서 필요한 객체 정보를 가져오고, 데이터베이스에서 이를 처리하여 JSON 형식으로 전달한다.

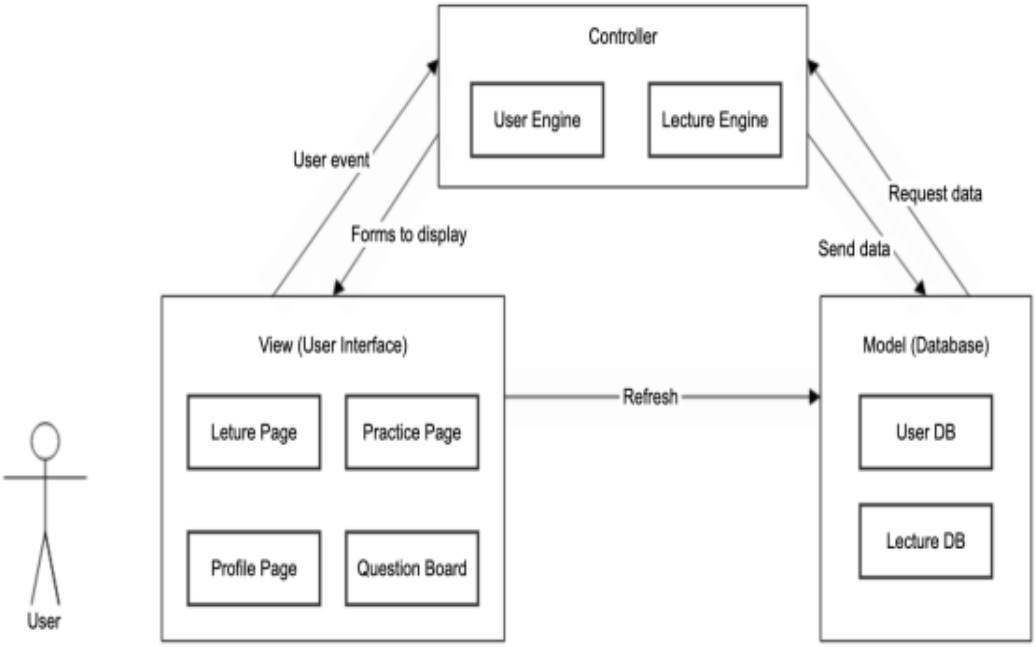


Figure 1 Overall System Architecture

2.2.1. Context Diagram

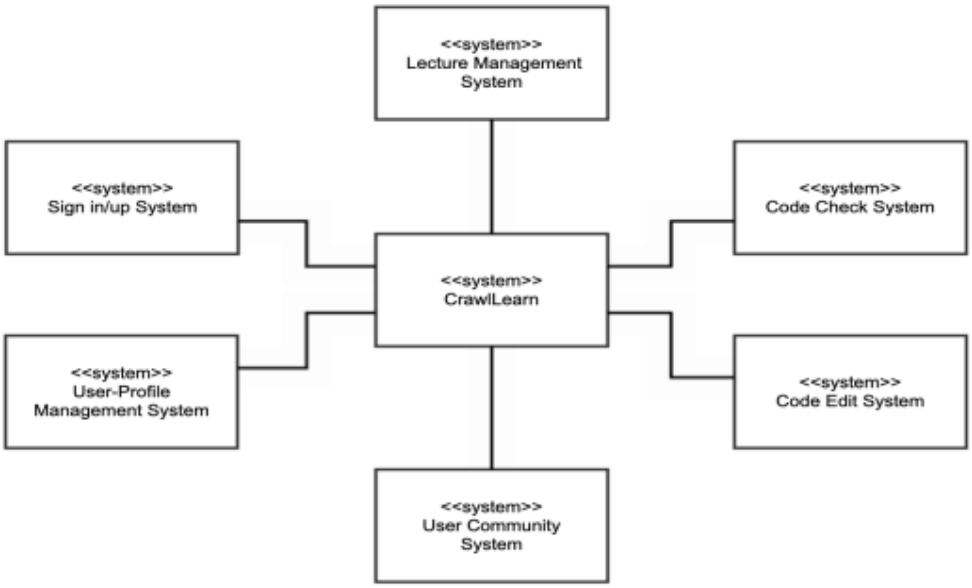


Figure 2 Overall Context Diagram

2.2.2. Sequence Diagram

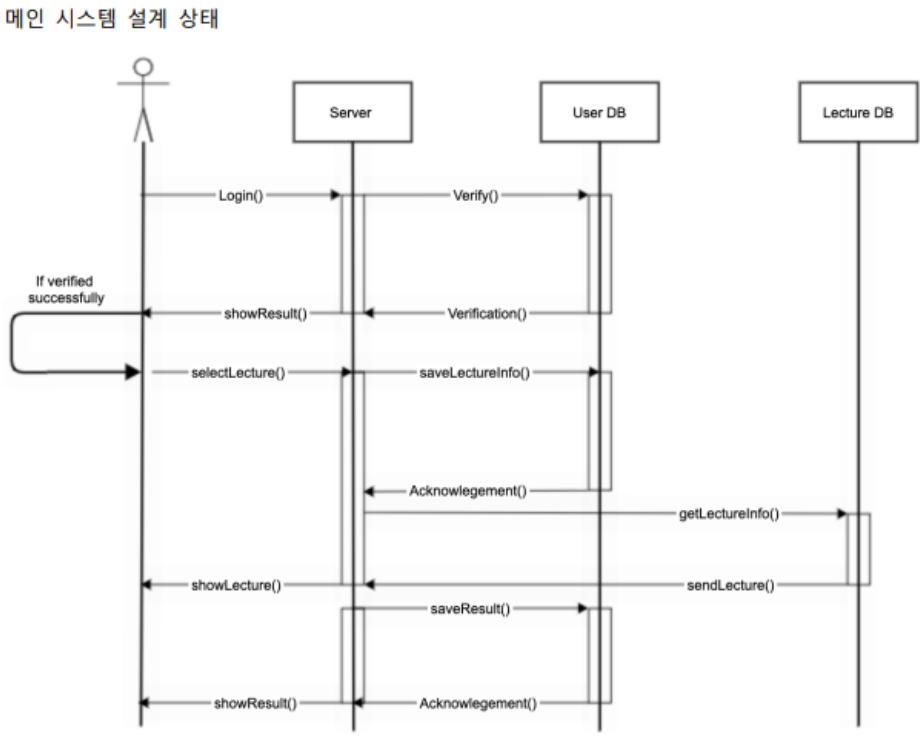


Figure 3 Overall Sequence Diagram

2.2.3. Use Case Diagram

Figure 4 Overall Use Case Diagram

3. System Architecture – Frontend

3.1. Objectives

이 장에서는 프론트 엔드 시스템의 구조, 속성 및 기능에 관한 설명을 기술하고, 시스템을 구성하는 각 요소들의 관계를 설명한다.

3.2. SubComponents

3.2.1. Profile

Profile 클래스는 사용자 개인 정보와 활동 내역을 관리하는 객체이다. 개인 정보에는 사용자 별명(nickname)과 계정 정보가 있다. 사용자 계정은 변경이 불가능하며 사용자 이름만 변경 가능하다. 활동 내역에는 수강한 강의, 작성한 질문, 좋아요 표시한 강의 정보가 표시된다.

3.2.1.1. Attributes

이 클래스가 가지는 속성은 다음과 같다.

- user_email: 사용자 계정 정보인 이메일 주소이다
- user_seq: 사용자 학번이다. 로그인 가입은 성균관대학교 학생만 가능하다.
- user_name: 사용자 계정 정보에 있는 이름이다.
- nickname: 이 사이트에서 사용자가 사용하는 별명이다.
- registered_lecture: list<lecture_card>: 사용자가 수강한 강의 리스트이다.
- registered_QA: list<QA_card>: 사용자가 작성 또는 댓글을 작성한 질문 리스트이다.

liked_lecture: list<lecture_card>: 사용자가 좋아요 표시간 강의 리스트이다

3.2.1.2. Methods

이 클래스가 가지는 메서드는 다음과 같다.

- SetId(): 사용자 별칭을 바꾸는 함수이다.

- GetProfile(): 사용자 프로파일 정보를 가져오는 함수이다.
- ShowProfile(): 가져온 프로파일 정보를 보여주는 함수이다.

3.2.1.3. Class Diagram

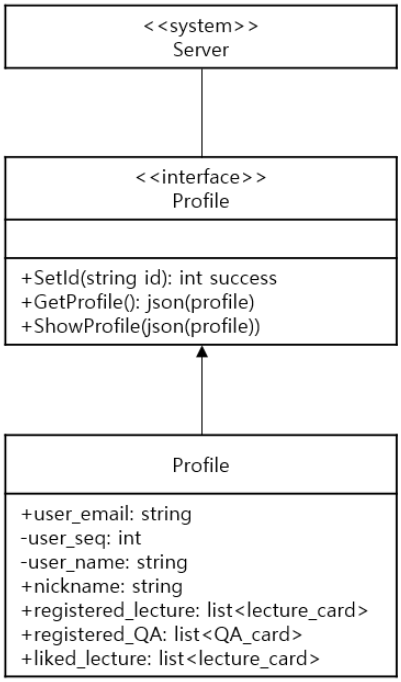


Figure 5 Class Diagram - Profile

3.2.1.4. Sequence Diagram

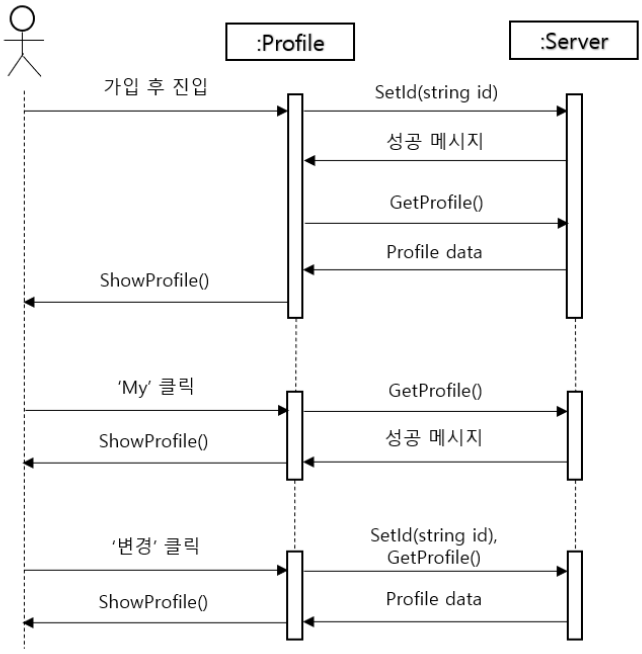


Figure 6 Sequence Diagram - Profile

3.2.2. LogSignin

이 클래스는 로그인과 가입 그리고 세션 토큰을 관리하는 함수이다. 구글 API 를 사용한다.

3.2.2.1. Attributes

이 클래스가 가지는 속성은 다음과 같다.

- user_email: 사용자 계정인 이메일 주소이다.
- user_seq: 사용자 계정에 있는 사용자 학생 번호이다.
- user_name: 사용자 계정에 있는 사용자 이름 정보이다.

session: 로그인 시 결과값으로 받은 토큰으로 사용자의 세션을 관리한다.

3.2.2.2. Methods

이 클래스가 가지는 메서드는 다음과 같다.

- GoogleLogin(): 구글 API를 사용한 로그인 함수이다.
- GoogleSignin(): 구글 API를 사용한 가입 함수이다.

3.2.2.3. Class Diagram

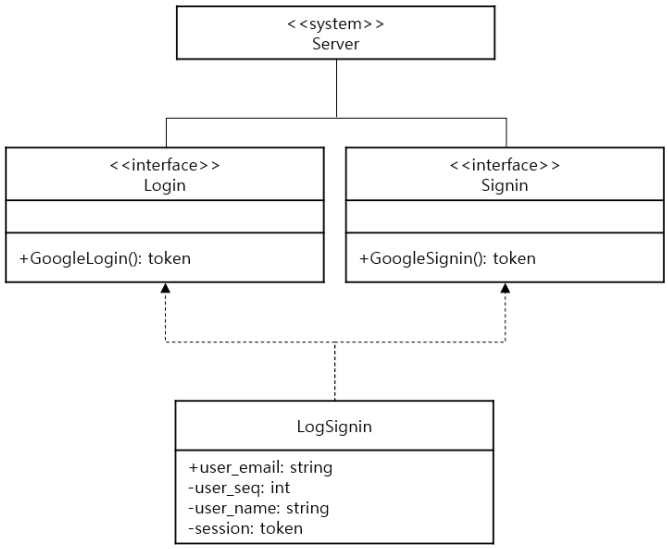


Figure 7 Class Diagram – Login and Signin

3.2.2.4. Sequence Diagram

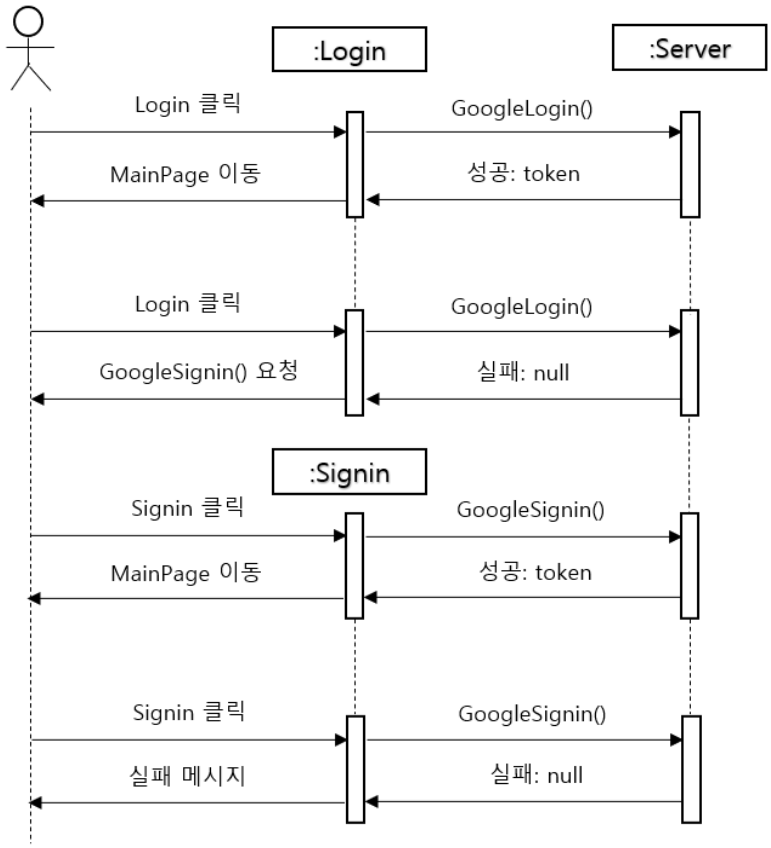


Figure 8 Sequence Diagram - Login and Signin

3.2.3. Main

메인 페이지를 관리하는 클래스이다. 크게 강의 관리하는 객체와 질문을 관리하는 객체를 속성으로 가진다. 이 장에서는 강의 부분을 주로 설명하고 다음 장에서 질문 부분을 설명한다.

3.2.3.1. Attributes

Main 클래스가 가지는 속성은 다음과 같다.

- lecture: 강의를 관리하는 Lecture 객체이다
- qa: 질문을 관리하는 QA 객체이다. 다음 장에 기술되어 있다.

lecture 클래스가 가지는 속성은 다음과 같다.

- lecture_list: 강의 대분류를 나타내는 문자열 리스트이다.
- lecture_table: 강의 데이터를 관리하는 객체이다.

- navigator: 사용자 로그인과 페이지 이동을 담당하는 객체이다.

lecture_table 클래스가 가지는 속성은 다음과 같다.

- lectures: lecture_card 객체의 배열이다.
- page: 현재 표시되는 페이지이다.

lecture_card 클래스가 가지는 속성은 다음과 같다.

- lecture_seq: 강의를 구분하는 유일한 id값이다.
- lecture_title: 강의 제목이다.
- lecture_description: 강의에 관한 짧은 설명 글이다.
- difficulty: 어려움을 표시한 값이다.
- like: 사용자가 좋아요를 표시한 값이다.

nevigator 클래스가 가지는 속성은 다음과 같다.

- login: LogSignin 객체이다.
- my: Profile 객체이다

3.2.3.2. Methods

lecture 클래스가 가지는 메서드는 다음과 같다.

- order(): 사용자에게 표시된 강의 정렬 순서를 바꾸는 함수이다.
- show_lecture_list(): 강의 대분류에 따라 DB로부터 강의 정보를 받아서 사용자에게 보여주는 함수이다.
- lecture_search(): 사용자가 입력한 강의 제목으로 DB에 쿼리를 보내 해당 강의를 받아서 사용자에게 보여주는 함수이다.

navigator 클래스가 가지는 메서드는 다음과 같다.

- login(): 로그인 관리를 하는 함수이다.
- logout(): 사용자 세션을 종료하여 로그아웃을 하는 함수이다.
- mypage(): 사용자 프로필을 보여주는 함수이다.
- show(int type): 사용자 입력을 받아 강의 또는 질문을 보여주는 함수이다.

lecture_table 클래스가 가지는 메서드는 다음과 같다.

- pagination(): DB로부터 받은 강의 데이터를 페이지로 나눠 표시하는 함수이다.

3.2.3.3. Class Diagram

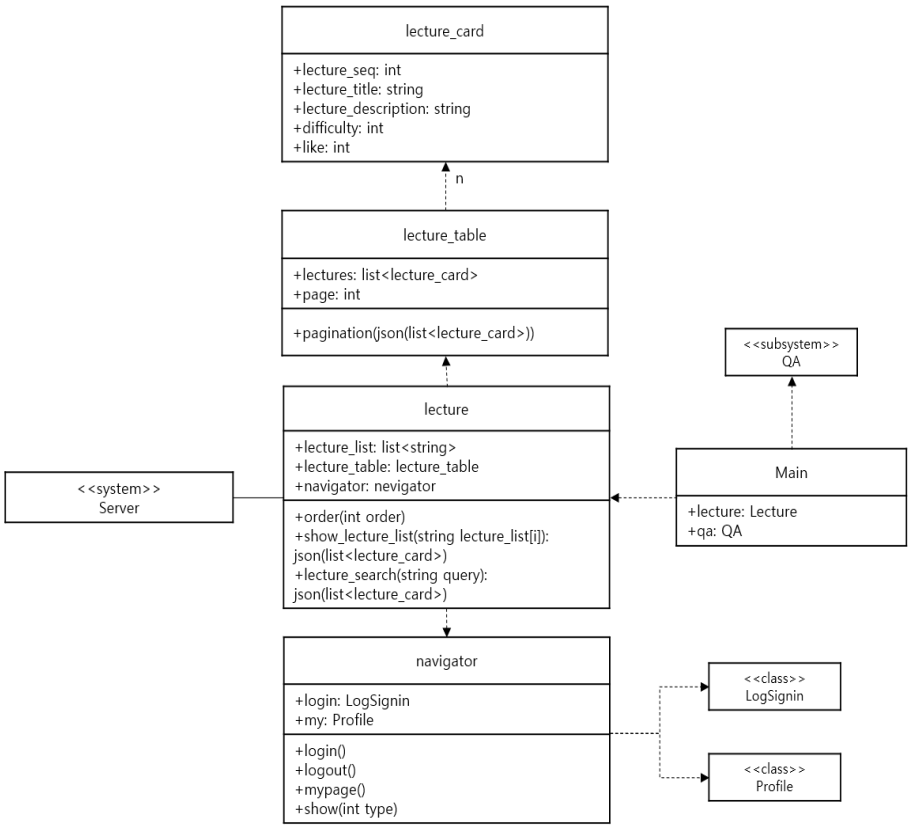


Figure 9 Class diagram – main

3.2.3.4. Sequence Diagram

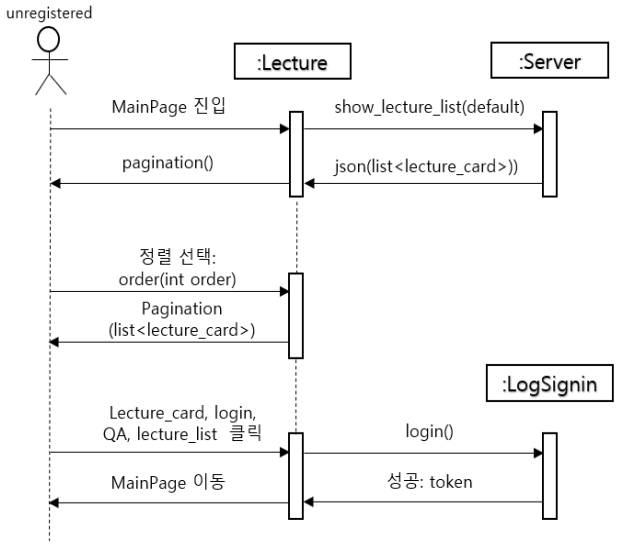


Figure 10 Sequence diagram – unregistered main

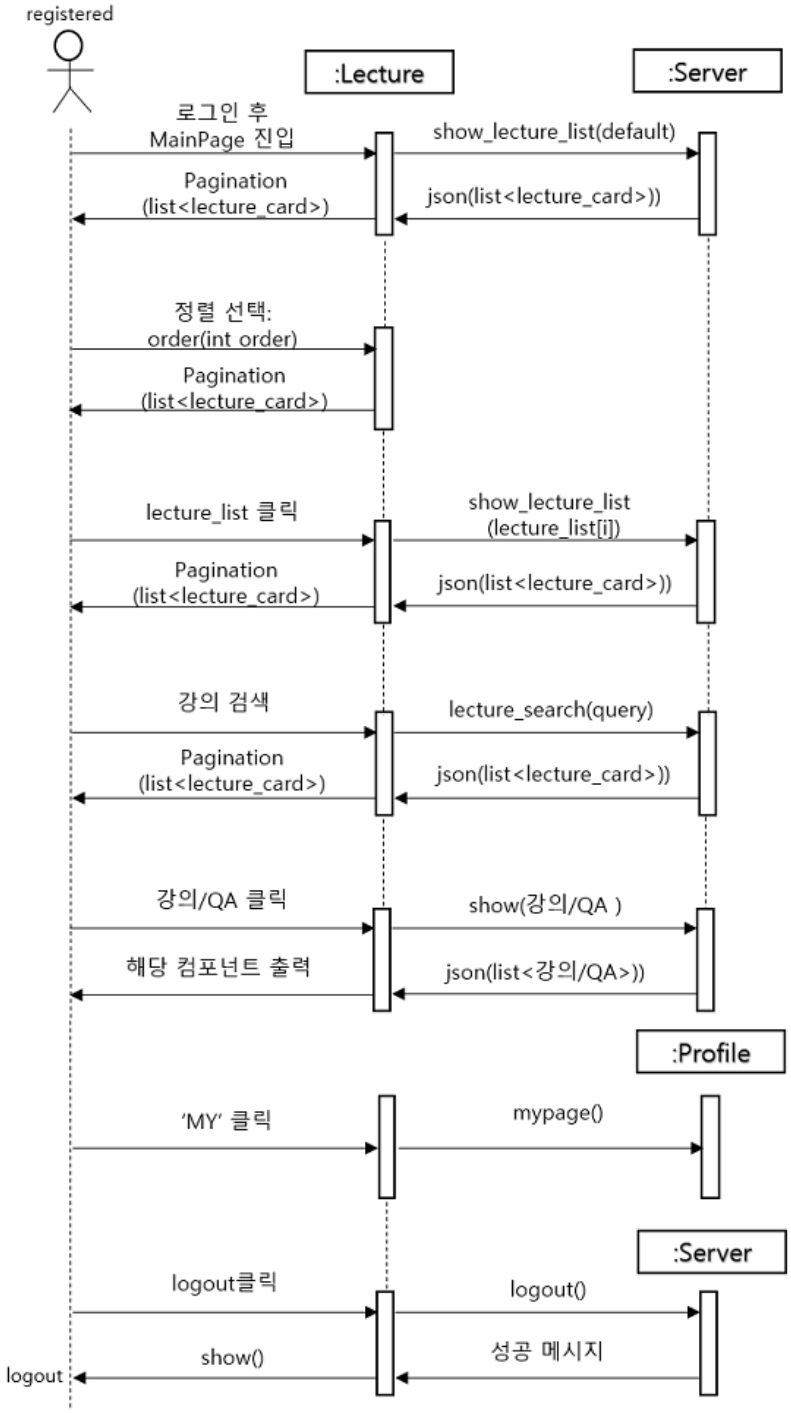


Figure 11 Sequence diagram – registered main

3.2.4. Lecture-content

lecture-content 클래스는 강의의 내용을 보여주는 객체이다. lecture card 를 통해 사용자가 원하는 강의를 선택하면 lecture-intro 를 통해 강의를 소개해주는 글을 보여주고, 최종적으로 강의 듣기 버튼을 누르면 보여지는 클래스이다.

3.2.4.1. Attributes

Lecture-content object가 갖는 attributes이다.

- Lecture_content_seq: 세부 강의 번호이다.
- Lecture_content: 강의 내용이다.
- Create_time: 강의 생성 날짜이다.
- Modification_time: 강의 변경 및 수정 날짜이다.
- User_code: 해당 강의를 듣고 사용자가 적는 코드이다.
- Lecture_answer: 해당 강의의 실습 결과란이다.

3.2.4.2. Methods

- ShowResult(lecture_content_seq)
- RequestLectureContent(lecture_content_seq)
- RequestAnswer(lecture_content_seq, user_code)

3.2.4.3. Class Diagram

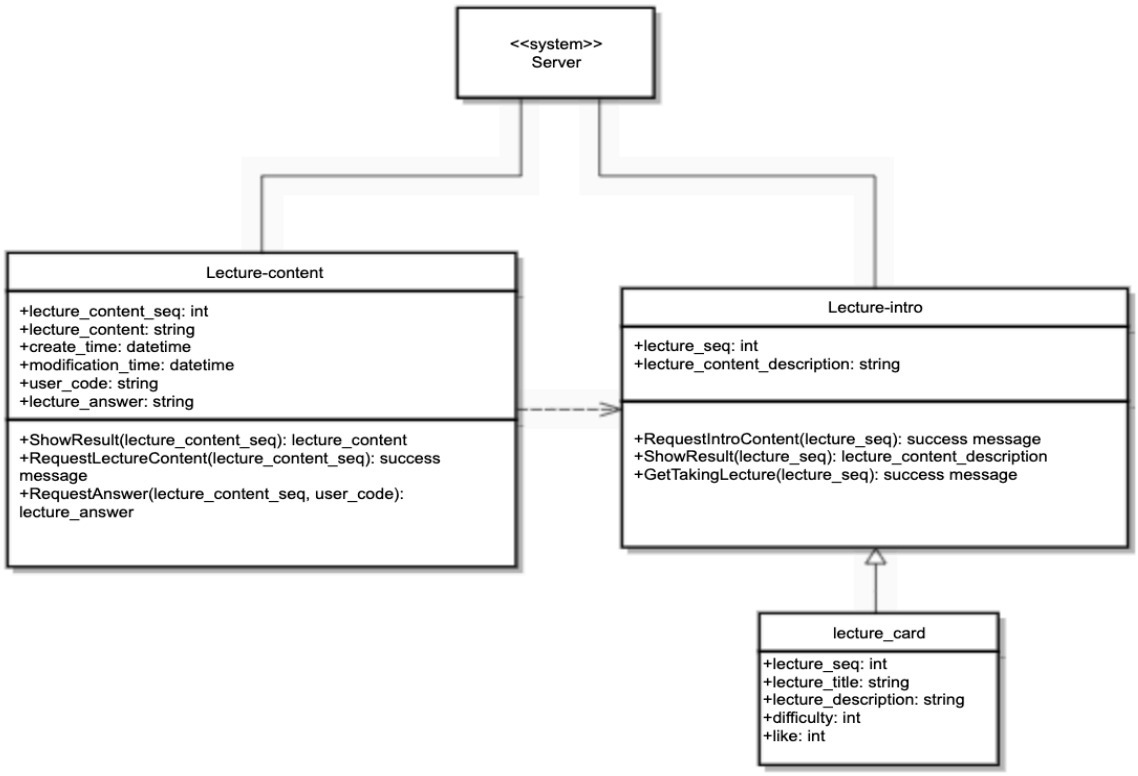


Figure 12 Class Diagram - Lecture-content

3.2.4.4. Sequence Diagram

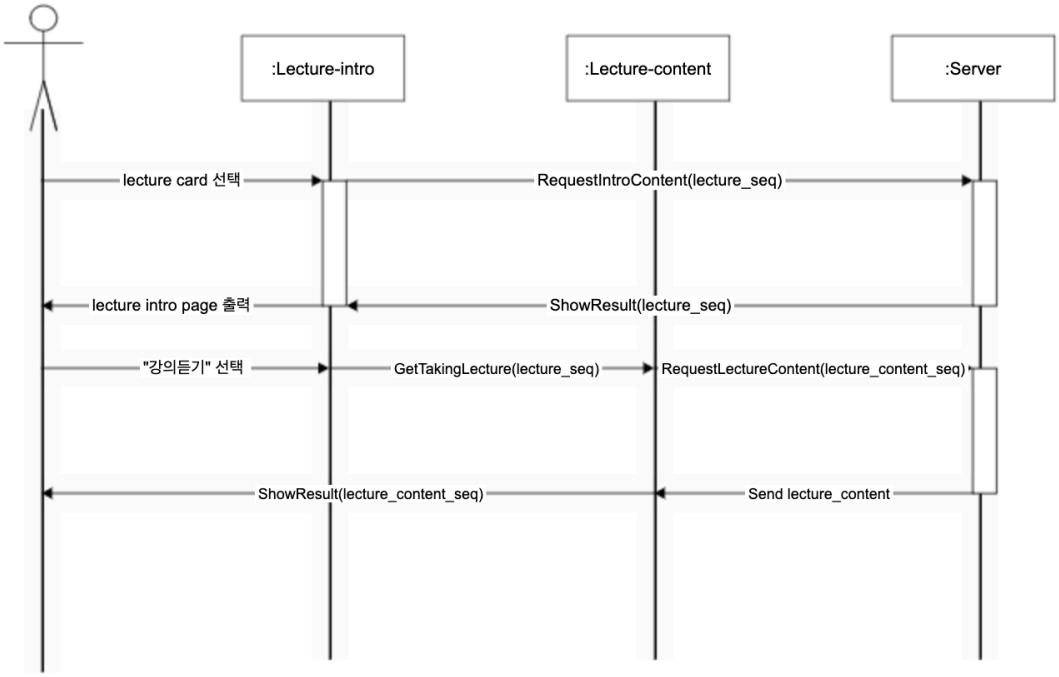


Figure 13 Sequence Diagram - Lecture-content

3.2.5. Code

사용자가 실습 코드를 입력하고 그 코드를 정답 코드와 비교하는 과정들을 담당하는 컴포넌트이다.

3.2.5.1. Attributes

Code object가 갖는 attributes이다.

- lecture_content_seq: 세부 강의 번호이다.
- code_seq: 코드 식별 번호이다.
- code_content: 코드 내용이다.

3.2.5.2. Methods

- showResult(lecture_content_seq)

3.2.5.3. Class Diagram

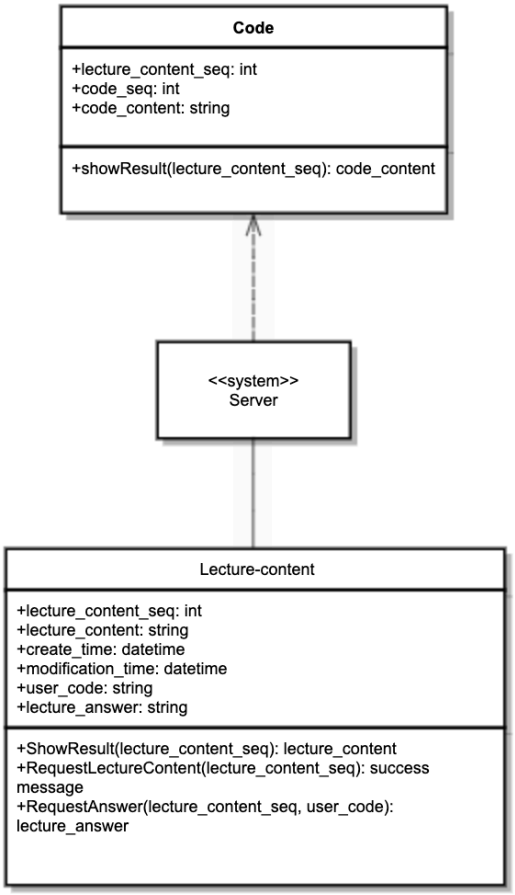


Figure 14 Class Diagram - Code

3.2.5.4. Sequence Diagram

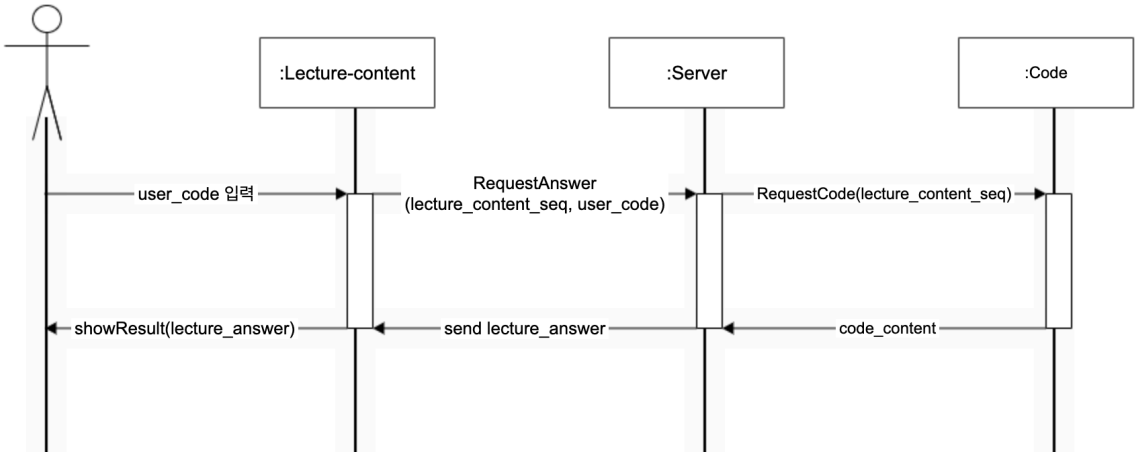


Figure 15 Sequence Diagram - Cod

3.2.6. QA

Q&A 작성과 보여주는 페이지를 모두 관리하는 컴포넌트이다. 크게 질문을 관리하는 객체와 댓글을 관리하는 객체를 속성으로 가진다.

3.2.6.1. Attributes

- QA 클래스가 갖는 attributes는 다음과 같다.

- qa_list: 질문의 분류를 나타내는 문자열 리스트이다.
- qa_table: 질문 데이터를 관리하는 객체이다.

- qa_table 클래스가 가지는 속성은 다음과 같다.

- qas: lecture_card 객체의 배열이다.
- page: 현재 표시되는 페이지이다.

- qa_card 클래스가 가지는 속성은 다음과 같다.

- qa_seq: question 을 구분하는 식별자 id 값이다.
- user_email: 작성자의 이메일이다.
- create_time: question 생성 날짜이다.
- qa_title: question 제목이다.

- Comment 클래스가 가지는 속성은 다음과 같다.

- user_email: 작성자의 이메일이다.
- comment_seq: 댓글 식별자 id 값이다.
- comment_content: 댓글 내용이다.
- qa_createtime: 댓글 생성 날짜이다.
- qa_seq: 해당 댓글의 question 식별자 id 값이다.

3.2.6.2. Methods

- QA 클래스가 갖는 method 는 다음과 같다.

- order(int order)
- showQaList(string qa_list[i]): json(list<qa_card>)
- requestQaWritePage(lecture_content_seq): success message

- Qa_table 클래스가 갖는 method 는 다음과 같다.

- pagination: json(list<qa_card>)

3.2.6.3. Class Diagram

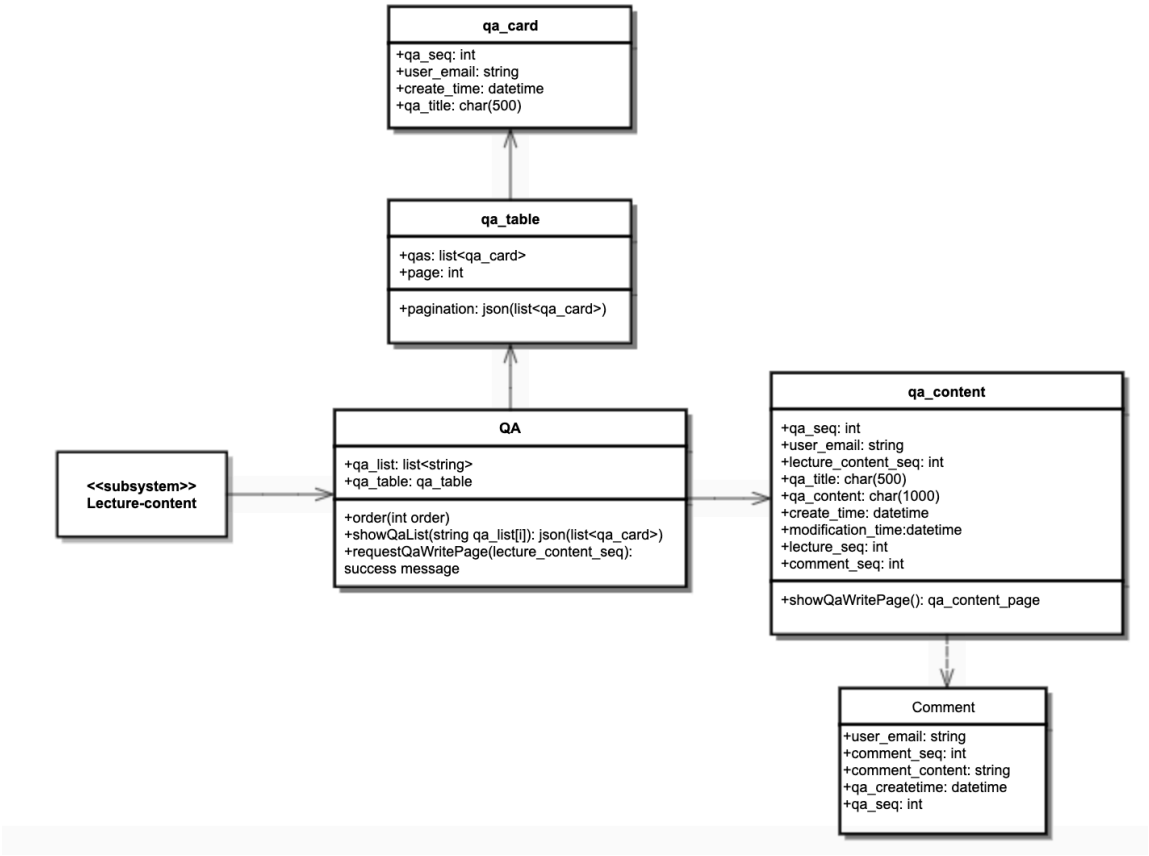


Figure 16 Class Diagram – QA

3.2.6.4. Sequence Diagram

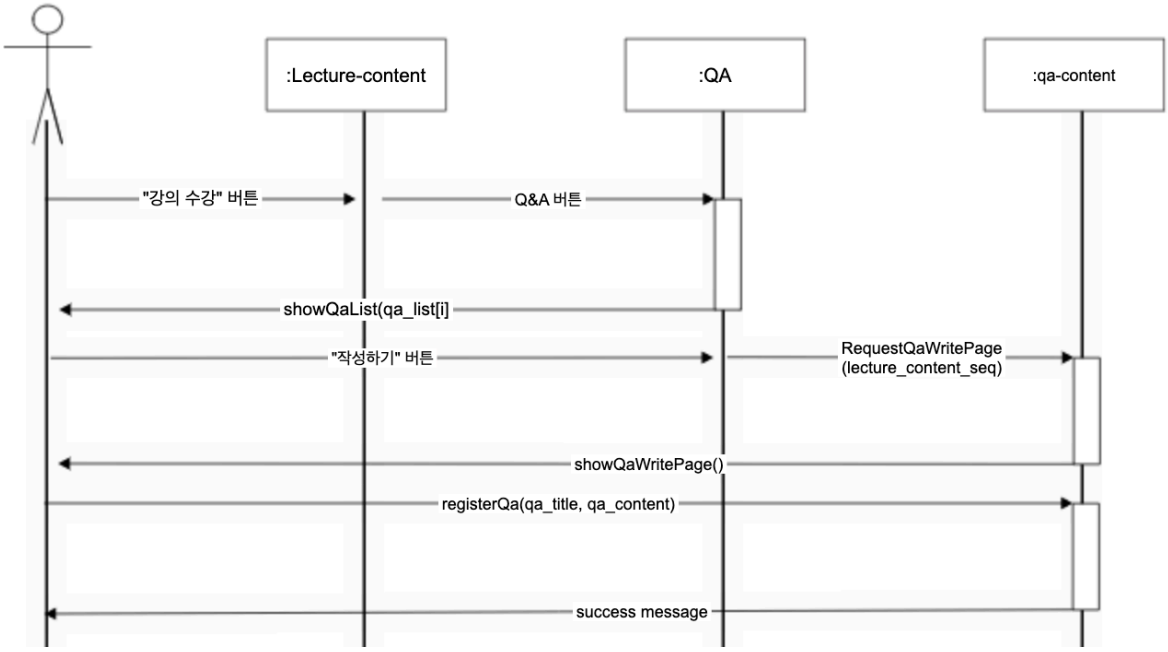


Figure 17 Sequence Diagram - QA

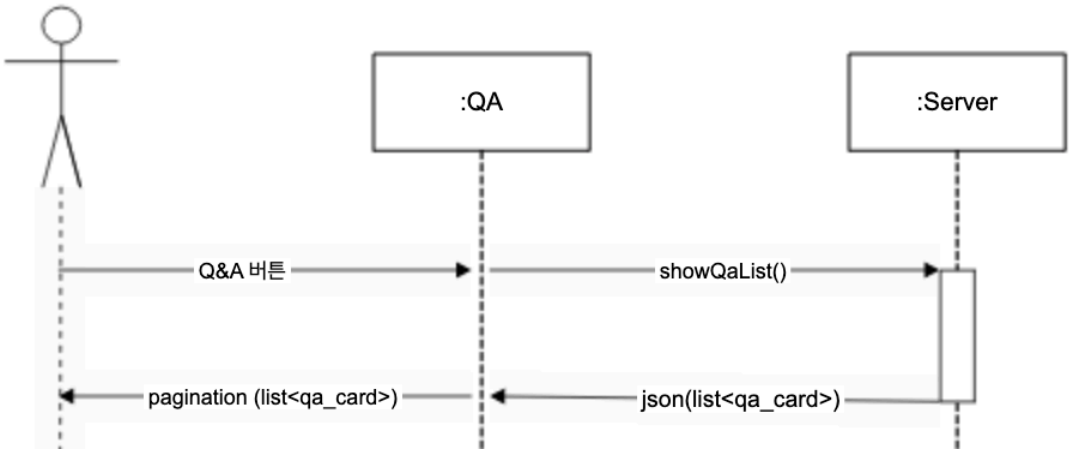


Figure 18 Sequence Diagram - QA2

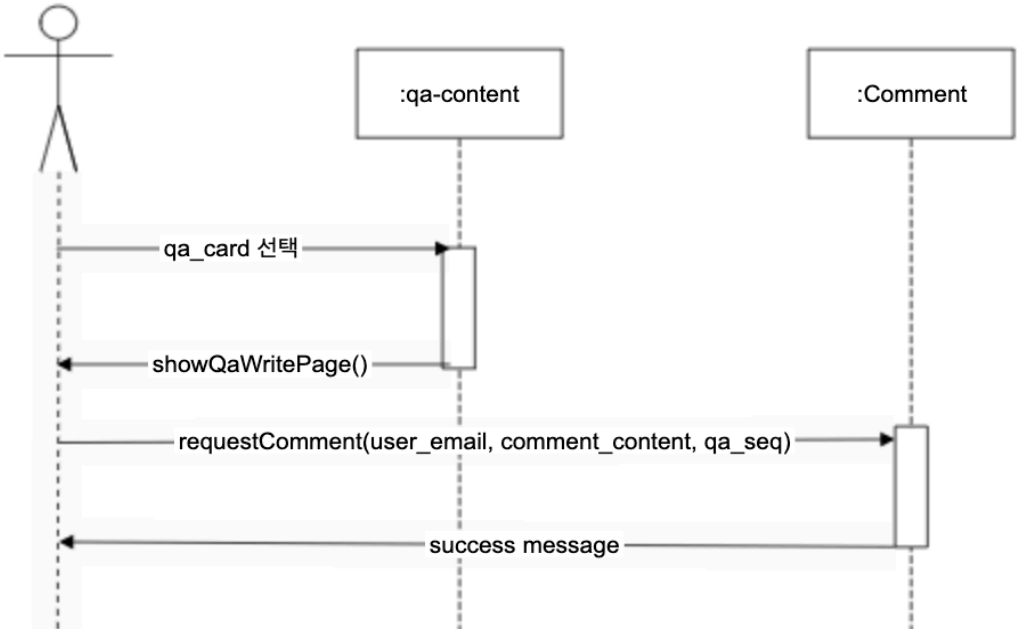


Figure 19 Sequence Diagram - QA3

4. System Architecture – Backend

4.1. Purpose

이번 장에서는 DB 와 서버의 구조를 설명한다.

4.2. Overall Architecture

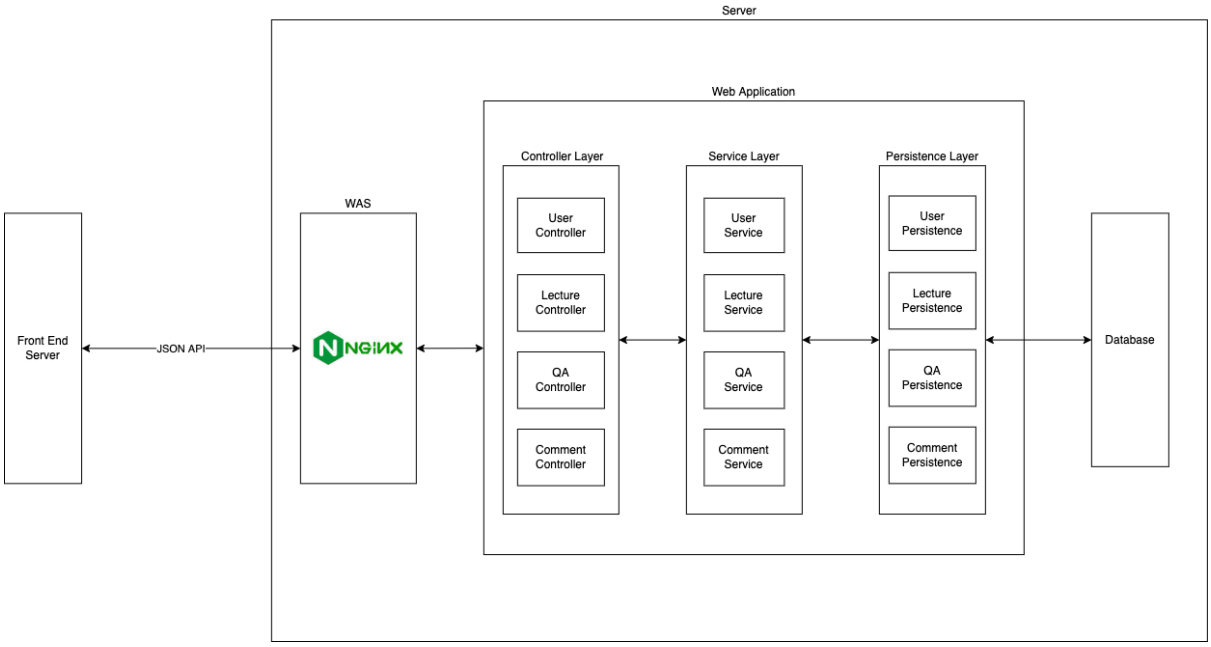


Figure 20 overall architecture

서버의 전체 구조는 다음과 같다. 먼저 Front End 에서 API 를 호출을 하면 NGINX 로 만들어진 WAS 를 통해 Web Application 에서 기능이 호출이 된다. Web Application 은 Layered Architecture 로 구성되어있다. Request 와 Response 를 처리하는 Controller layer, 비지니스 로직을 처리하는 Service Layer, 마지막으로 DB 와의 통신을 담당하는 Persistence Layer 로 구성되어있다.

4.3. Subcomponents

4.3.1. User system

User 와 관련된 작업들이 공통적으로 묶여있는 System 이다.

4.3.1.1. User System Class Diagram

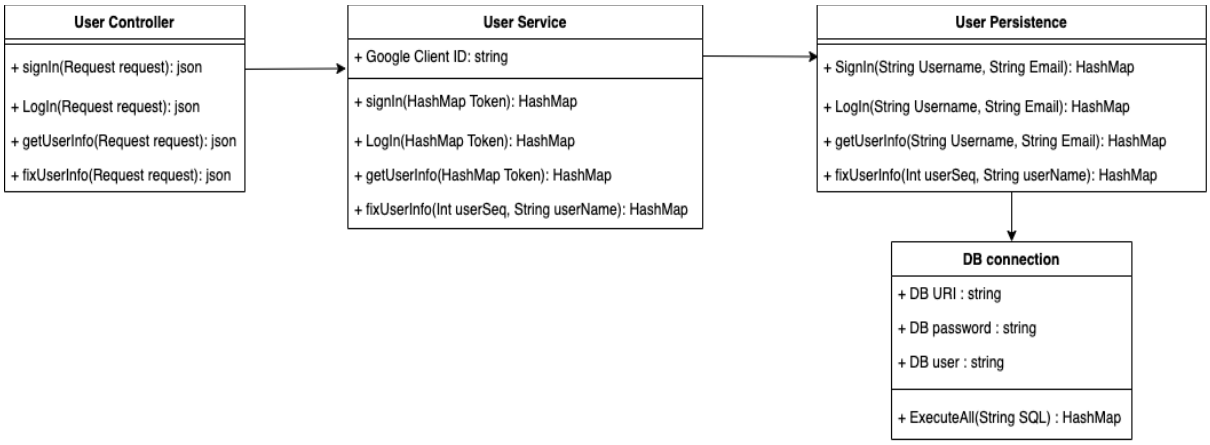


Figure 21 Class Diagram – User System

4.3.1.1.1. User Controller Class

Was 로 부터 받은 Request 에서 Parameter 를 받아서 이를 HashMap 으로 parsing 한 다음 Service Layer 를 호출하는 역할을 가지고 있다. 해당 클래스는 4 가지의 메소드가 존재한다. 첫번째는 회원가입과 관련된 SignIn 이 존재한다. 두번째는 로그인과 관련된 LoginIn 이다. 세번째는 사용자의 정보를 가져오는 getUserInfo 이다. 마지막은 개인정보 중 이름을 수정할 수 있는 fixUserInfo 이다. 해당 메소드들은 모두 json 을 was 에 반환되게 되어있다.

4.3.1.1.2. User Service Class

Controller 로 부터 받은 인자를 parsing 하고 Persistence Layer 를 호출하고 나서는 해당 Layer 에서 반환된 값을 Controller 에 넘겨주는 역할을 한다. 해당 서비스에는 총 4 가지 메소드가 존재한다. 첫번째는 회원가입과 관련된 SignIn 이 존재한다. 두번째는 로그인과 관련된 LoginIn 이다. 세번째는 사용자의 정보를 가져오는 getUserInfo 이다. 마지막은 개인정보 중 이름을 수정할 수 있는 fixUserInfo 이다. 첫번째, 두번째, 세번째 메소드가 받는 인수는 Token 이다. 해당 Token 은 구글 OAuth 서버로 부터 받아온

것이다. 마지막 메소드는 개인의 데이터 베이스상 번호와 바꾸고자 하는 이름을 받게 된다. 이들은 아래 레이어에서 받은 Hashmap 을 controller 에게 넘겨준다.

4.3.1.1.3. User Persistence Class

Service Layer 에서 받은 parameter 를 바탕으로 SQL 문을 만들어서 DB Connection 에서 넘겨준다. 또한 DB Connection 에서 DB 로 부터 받은 값을 HashMap 으로 Service Layer 에 넘겨주는 역할을 한다. 해당 서비스에는 총 4 가지 메소드가 존재한다. 첫번째는 회원가입과 관련된 SignIn 이 존재한다. 두번째는 로그인과 관련된 LoginIn 이다. 세번째는 사용자의 정보를 가져오는 getUserInfo 이다. 마지막은 개인정보 중 이름을 수정할 수 있는 fixUserInfo 이다.

4.3.1.1.4. DB Connection

DB 랑 연결을 담당하는 부분으로 Persistence Layer 로 부터 받은 SQL 을 DB 에 넘겨 결과값을 받는 역할을 한다. 이후 결과를 받으면 Persistence Layer 로 넘겨준다.

4.3.1.2. User Sequence Diagram

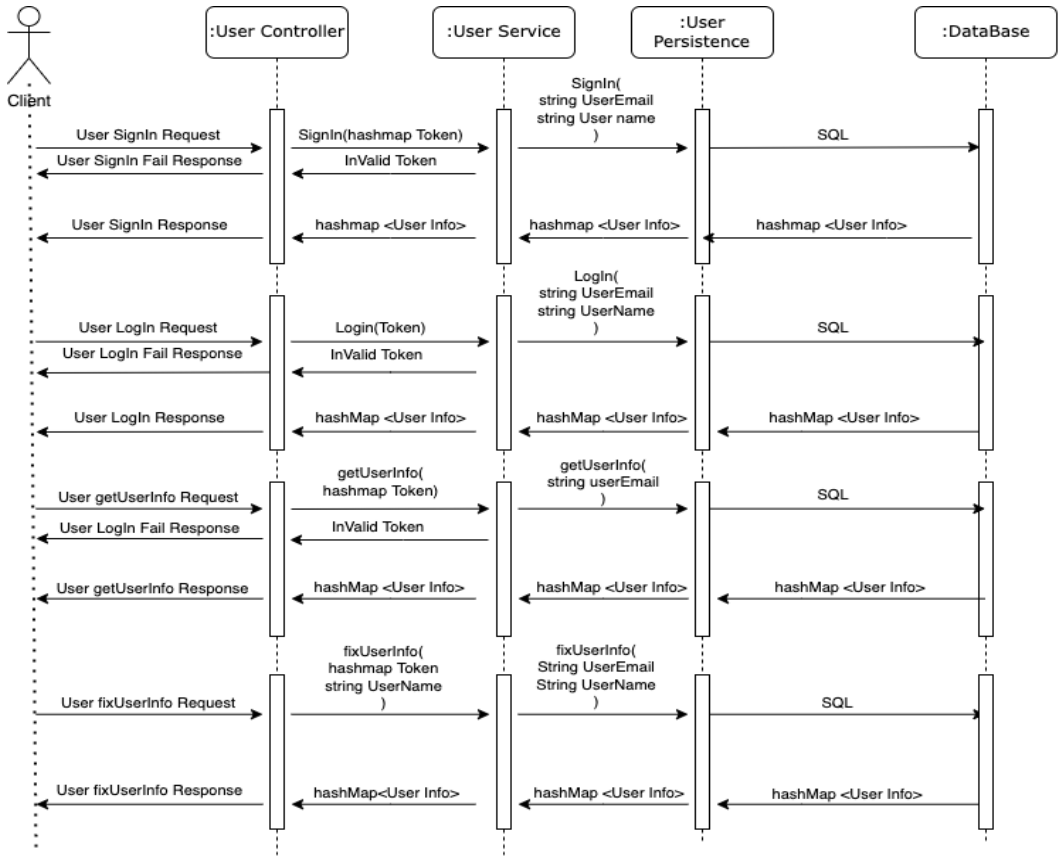


Figure 22 User System Sequence Diagram

4.3.2. Lecture System

Lecture 와 관련된 작업들이 공통적으로 묶여있는 System 이다.

4.3.2.1. Lecture System Class Diagram

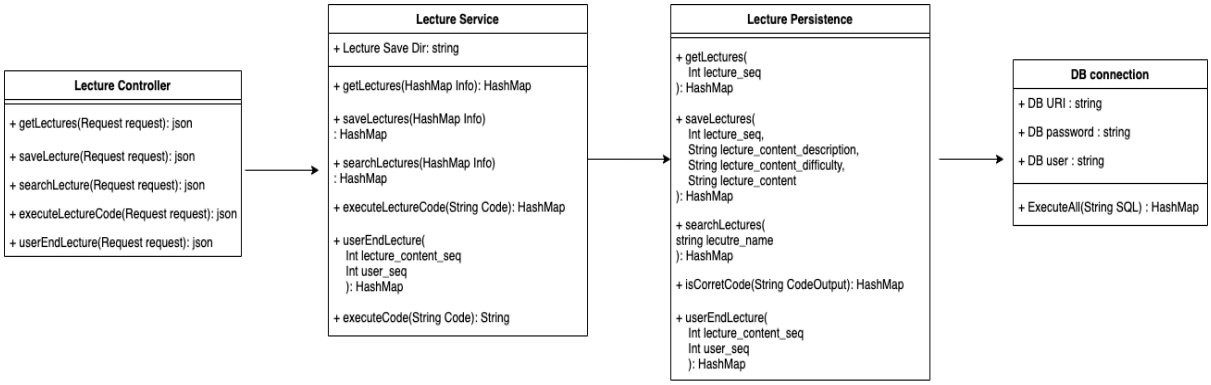


Figure 23 Class Diagram - Lecture System

4.3.2.1.1. Lecture Controller Class

Was 로 부터 받은 Request 에서 Parameter 를 받아서 이를 HashMap 으로 parsing 한 다음 Service Layer 를 호출하는 역할을 가지고 있다. 해당 Service Layer 의 결과를 json 으로 만들어 response 의 body 에 들어갈 내용을 만드는 역할을 한다. Lecture Controller 에는 총 5 가지 존재한다. 첫번째는 강의 내용을 가져오는 getLectures 이다. 두번째는 강의를 저장하는 saveLecture 이다. 세번째는 강의를 검색하는 searchLecture 이다. 네번째는 실습강의에서 작성한 코드를 실행하는 executeLectureCode 이다. 마지막은 사용자가 강의를 완료했을 때 해당 내용을 데이터베이스에 반영하고자 하는 userEndLecture 이다.

4.3.2.1.2. Lecture Service Class

Controller 로 부터 받은 HashMap 에서 필요한 정보를 추출하고 Persistence 를 호출하는 역할을 한다. 이후 Persistence Layer 를 호출해서 DB 에 필요한 정보를 저장하고 불러오고 검색하는 역할을 한다. 또한 Persistence Layer 를 호출하고 나서는 결과 HashMap 으로 Controller 에 넘겨주는 역할을 한다. Lecture Service 에는 총 6 가지 존재한다. 첫번째는 강의 내용을 가져오는 getLectures 이다. 두번째는 강의를 저장하는 saveLecture 이다. 세번째는 강의를 검색하는 searchLecture 이다. 네번째는 실습강의에서 작성한 코드를 실행하기 위한 전처리 단계인 executeLectureCode 이다. 다섯번째는 사용자가 강의를 완료했을 때 해당 내용을 데이터베이스에 반영하고자 하는

userEndLecture 이다. 마지막은 작성한 코드를 실행하기 위한 메소드인 executeCode 이다.

4.3.2.1.3. Lecture Persistence Class

Service Layer 에서 받은 parameter 를 바탕으로 SQL 문을 만들어서 DB Connection 에서 넘겨준다. 또한 DB Connection 에서 DB 로 부터 받은 값을 HashMap 으로 Service Layer 에 넘겨주는 역할을 한다. Lecture Persistence Class 에는 총 5 가지 메소드가 존재한다. 첫번째는 강의 내용을 가져오는 getLectures 이다. 두번째는 강의를 저장하는 saveLecture 이다. 세번째는 강의를 검색하는 searchLecture 이다. 네번째는 강의 결과물이 원하는 결과와 맞는지 검사하는 IsCorrectCode 이다. 마지막은 사용자가 강의를 완료했을 때 해당 내용을 데이터베이스에 반영하고자 하는 userEndLecture 이다.

4.3.2.2. Lecture System Sequence Diagram

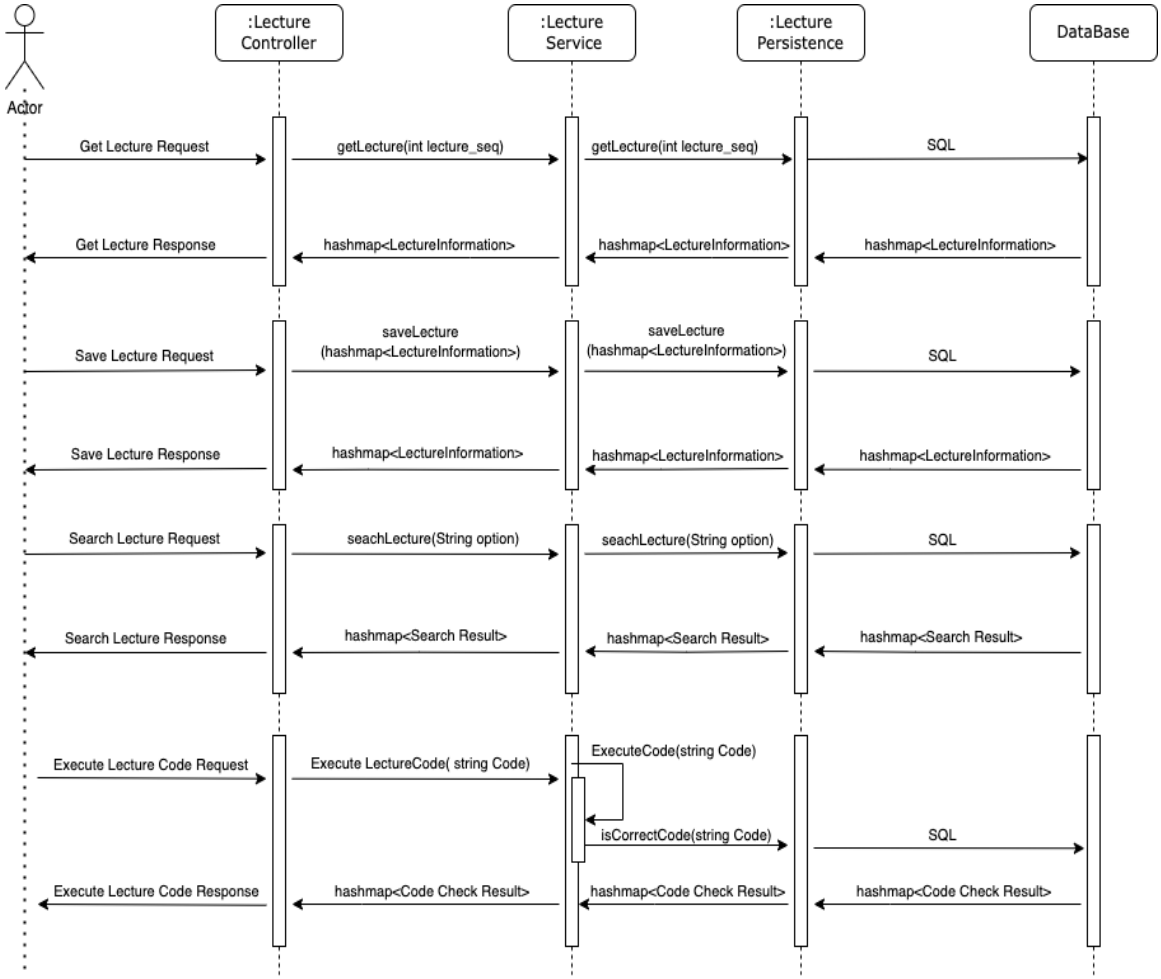


Figure 24 Lecture System Sequence Diagram

4.3.3. QA System

질문을 작성하고 조회하는 작업들이 묶여 있는 System 이다.

4.3.3.1. QA System Class Diagram

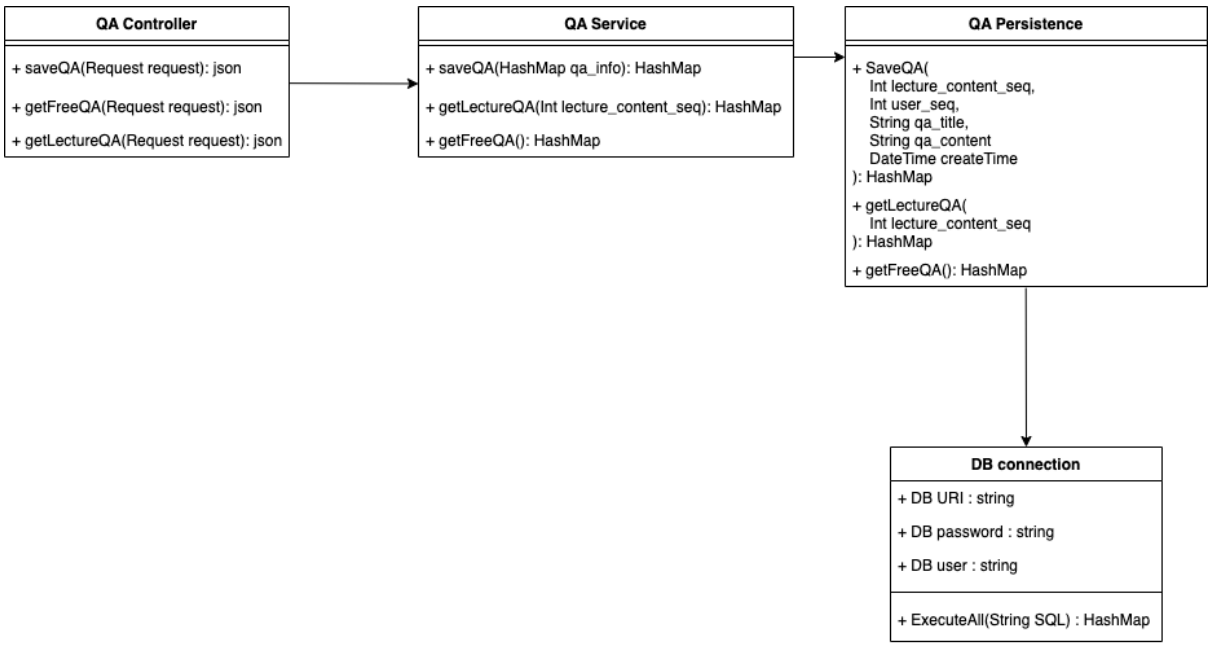


Figure 25 Class Diagram – QA System

4.3.3.1.1. QA controller

Was 로 부터 받은 Request 에서 Parameter 를 받아서 이를 HashMap 으로 parsing 한 다음 Service Layer 를 호출하는 역할을 가지고 있다. 해당 Service Layer 의 결과를 json 으로 만들어 WAS 에게 넘겨주는 역할을 한다. 해당 클래스에는 3 가지 메소드가 존재하는데 QA 를 저장하는 request 를 받는 saveQA, 자유질문 관련한 request 를 받는 getFreeQA, 강의 관련한 질문을 남기는 getLectureQA 가 있다.

4.3.3.1.2. QA Service Class

Controller 로 부터 받은 HashMap 에서 필요한 정보를 추출하고 Persistence 를 호출하는 역할을 한다. 이후 Persistence Layer 를 호출해서 DB 에 필요한 정보를 저장하고 불러오고 검색하는 역할을 한다. 또한 Persistence Layer 를 호출하고 나서는 해당 Layer 에서 반환된 값을 Controller 에 넘겨주는 역할을 한다. 해당 클래스에는

3 가지 메소드가 존재하는데 QA 를 저장하는 request 를 받는 saveQA, 자유질문 관련한 request 를 받는 getFreeQA, 강의 관련한 질문을 남기는 getLectureQA 가 있다.

4.3.3.1.3. QA Persistence Class

Service Layer 에서 받은 parameter 를 바탕으로 SQL 문을 만들어서 DB Connection 에서 넘겨준다. 또한 DB Connection 에서 DB 로 부터 받은 값을 Service Layer 에 넘겨주는 역할을 한다. 해당 클래스에는 3 가지 메소드가 존재하는데 QA 를 저장하는 request 를 받는 saveQA, 자유질문 관련한 request 를 받는 getFreeQA, 강의 관련한 질문을 남기는 getLectureQA 가 있다.

4.3.3.2. QA System Sequence Diagram

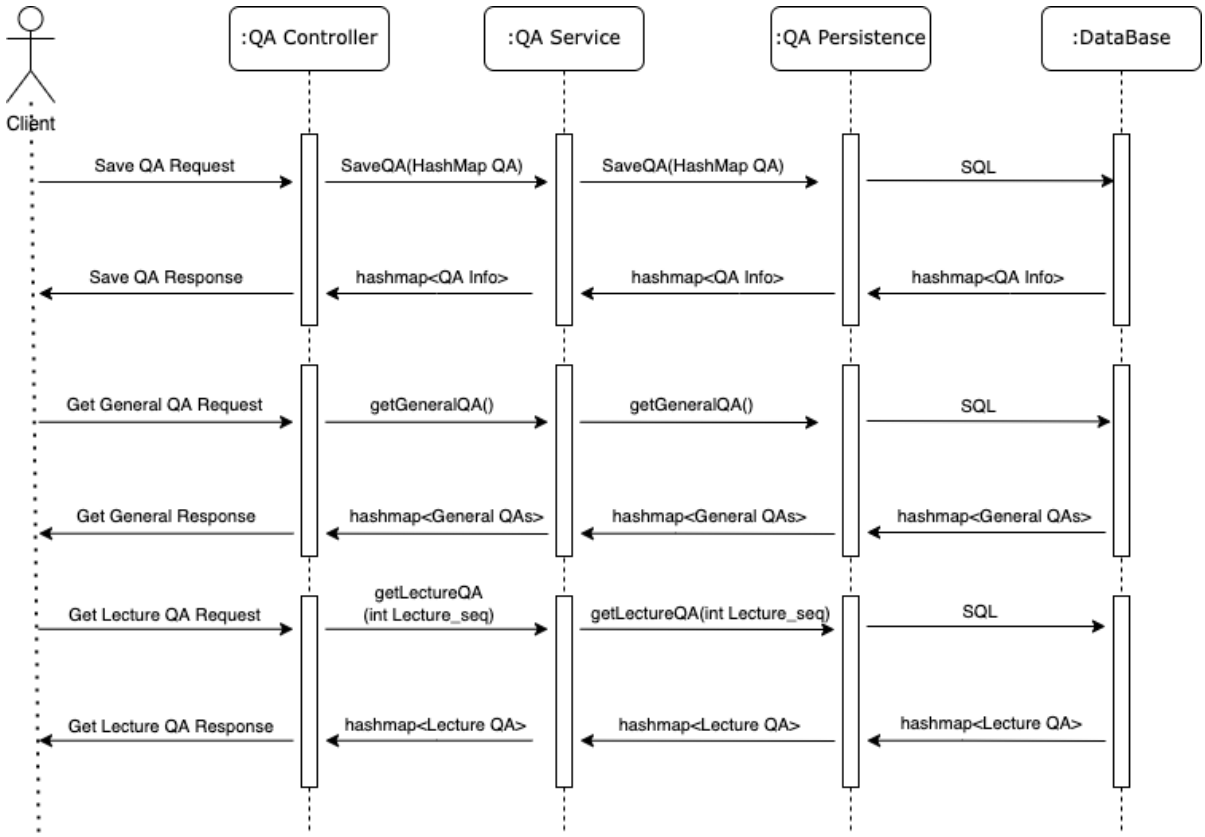


Figure 26 QA System Sequence Diagram

4.3.4. Comment System

댓글과 관련된 작업들이 묶여 있는 System 이다.

4.3.4.1. Comment System Class Diagram

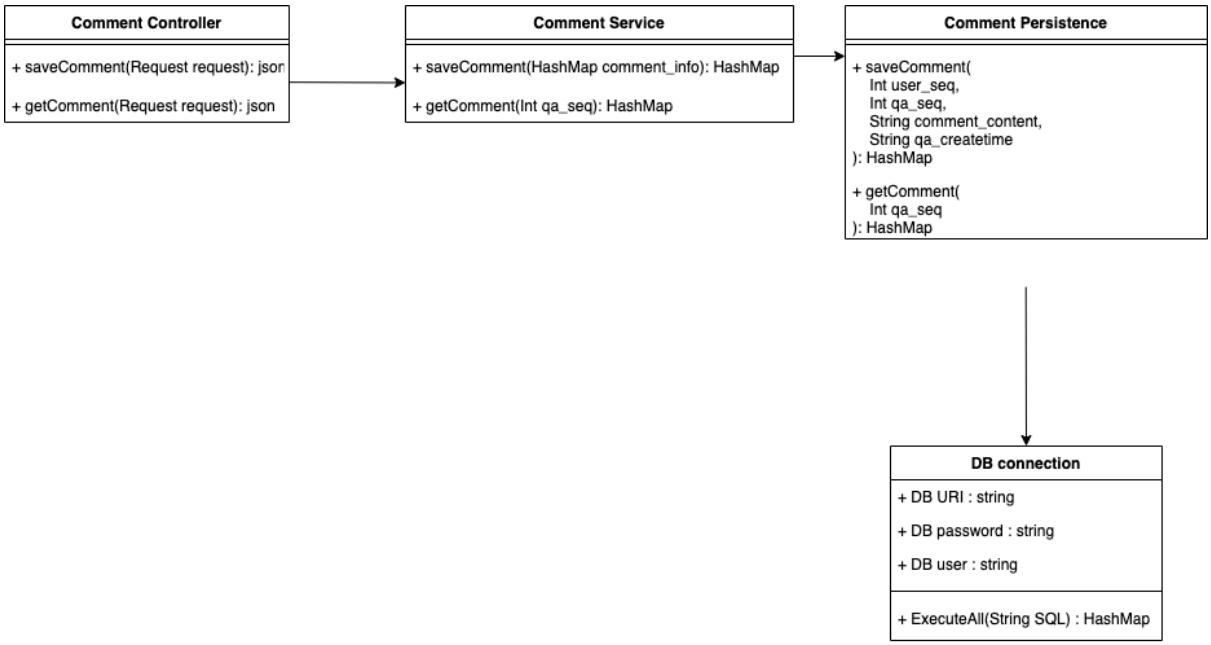


Figure 27 Class Diagram - Comment System

4.3.4.1.1. Comment controller

Was 로 부터 받은 Request 에서 Parameter 를 받아서 이를 HashMap 으로 parsing 한 다음 Service Layer 를 호출하는 역할을 가지고 있다. 해당 Service Layer 의 결과를 json 으로 WAS 로 넘겨주는 역할을 한다. 해당 클래스에는 2 가지 메소드가 존재하는데 Comment 를 저장하는 request 를 받는 saveComment, 특정 질문에 달려있는 댓글을 가져오는 request 를 받는 getComment 가 있다.

4.3.4.1.2. Comment Service Class

Controller 로 부터 받은 HashMap 에서 필요한 정보를 추출하고 Persistence Layer 를 호출하는 역할을 한다. 이후 Persistence Layer 를 호출해서 DB 에 필요한 정보를 저장하고 불러오고 검색하는 역할을 한다. 또한 Persistence Layer 를 호출하고 나서는 해당 Layer 에서 반환된 값을 Controller 에 넘겨주는 역할을 한다. 해당 클래스에는 2 가지 메소드가 존재하는데 Comment 를 저장하는 request 를 받는 saveComment, 특정 질문에 달려있는 댓글을 가져오는 request 를 받는 getComment 가 있다.

4.3.4.1.3. Comment Persistence Class

Service Layer 에서 받은 parameter 를 바탕으로 SQL 문을 만들어서 DB Connection 에서 넘겨준다. 또한 DB Connection 에서 DB 로 부터 받은 값을 Service Layer 에 넘겨주는 역할을 한다. 해당 클래스에는 2 가지 메소드가 존재하는데 Comment 를 저장하는 request 를 받는 saveComment, 특정 질문에 달려있는 댓글을 가져오는 request 를 받는 getComment 가 있다.

4.3.4.2. Comment System Sequence Diagram

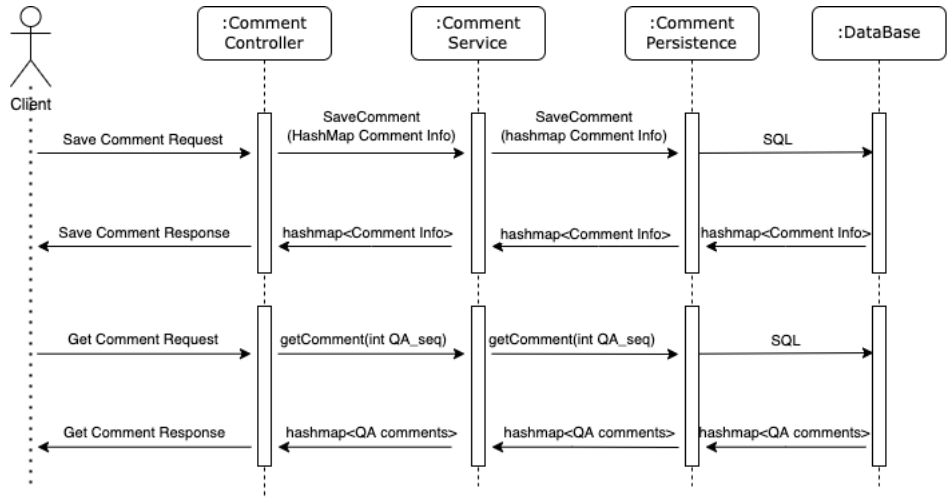


Figure 28 Comment System Sequence Diagram

5. Protocol Design

5.1. Objectives

해당 장에서는 각 하위 시스템 간의 상호 작용에 사용되는 프로토콜에 어떤 구조가 사용되는지 설명한다. 또한 각 인터페이스가 어떻게 정의되는지 설명한다.

5.2. JSON

JSON(JavaScript Object Notation)은 사람이 읽을 수 있는 텍스트를 사용하여 속성-값 쌍과 배열 데이터 유형(또는 기타 직렬화 가능한 값)으로 구성된 데이터 객체를 저장하고 전송하는 개방형 표준 파일 형식이다. 보편적으로 쓰이는 데이터 형식이며, AJAX 시스템에서 XML 을 대체하는 역할을 하는 등 다양한 응용 프로그램을 가지고 있다.

5.3. OAuth

OAuth 는 인터넷 사용자들이 비밀번호를 부여하지 않고 다른 웹사이트의 자신의 정보에 대한 접근 권한을 부여하기 위한 방법으로 일반적으로 사용되는 접근 위임에 대한 개방형 표준이다. 이 메커니즘은 아마존, 구글, 페이스북, 마이크로소프트 그리고 트위터와 같은 회사들이 사용자들이 제 3 자 애플리케이션이나 웹사이트와 그들의 계정에 대한 정보를 공유할 수 있도록 하기 위해 사용된다.

5.4. Authenticatoin

5.4.1. Register

- Request

Table 1 Table of register request

Attribute	Detail	
Protocol	OAuth	
Request Body	Email	User email
	Request Token	Token for OAuth
	User	Basic information of the user

- Response

Table 2 Table of register response

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

5.4.2. Login

- Request

Table 3 Table of log-in request

Attribute	Detail	
Protocol	OAuth	
Request Body	Email	User email
	Request Token	Token for OAuth
	User	Basic information of the user

- Response

Table 4 Table of log-in response

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 400	
Success Response Body	Access Token	Token for access
	Message	Success message
Failure Response Body	Message	Fail message

5.5. Lecture

5.5.1. Store Lecture

– Request

Table 5 Table of store lecture request

Attribute	Detail
Method	POST
URI	http: /lecture
Parameter	lecture_seq, lecture_content_description, lecture_content_difficulty, lecture_content, lecutre_content_answer

- Response

Table 6 Table of store lecture response

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 400	
Success Response Body	Access Token	Token for access
	Message	Success message
Failure Response Body	Message	Fail message

5.5.2. Get Lecture

- Request

Table 7 Table of get lecture request

Attribute	Detail
Method	GET
URI	http:/lectures?lecture_seq=1
Parameter	lecture_seq

- Response

Table 8 Table of get lecture response

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 400	
Success Response Body	Access Token	Token for access
	Message	Success message
Failure Response Body	Message	Fail message

5.5.3. Set Listening Lecture Complete

- Request

Table 9 Table of set listening lecture complete request

Attribute	Detail
Method	POST
URI	http: /lectures/1/lectureContent/2/userSeq/2/qa
Parameter	qa_title, qa_content

- Response

Table 10 Table of set listening lecture complete response

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 400	
Success Response Body	Access Token	Token for access
	Message	Success message
Failure Response Body	Message	Fail message

5.5.4. Search Lecture

- Request

Table 11 Table of set listening lecture complete request

Attribute	Detail
Method	GET
URI	http:/lectures/3/search
Parameter	search_option

- Response

Table 12 Table of set listening lecture complete response

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 400	
Success Response Body	Access Token	Token for access
	Message	Success message
Failure Response Body	Message	Fail message

5.6. Code

5.6.1. Test Code (F8)

- Request

Table 13 Table of test code request

Attribute	Detail
Method	GET
URI	http:/testcode

- Response

Table 14 Table of test code response

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 400	
Success Response Body	Access Token	Token for access
	Message	Success message
Failure Response Body	Message	Fail message

5.7. Comments

5.7.1. Write Comments

- Request

Table 15 Table of write comments request

Attribute	Detail
Method	POST
URI	http: /qa/1/comment
Parameter	comment_content, user_email

- Response

Table 16 Table of write comments response

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 400	
Success Response Body	Access Token	Token for access
	Message	Success message
Failure Response Body	Message	Fail message

5.7.2. Get Comments

- Request

Table 17 Table of get comments request

Attribute	Detail
Method	GET
URI	http: /qa/1/comments
Parameter	comment_content

- Response

Table 18 Table of get comments response

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 400	
Success Response Body	Access Token	Token for access
	Message	Success message
Failure Response Body	Message	Fail message

5.8. Question

5.8.1. Write Question

- Request

Table 19 Table of write question request

Attribute	Detail
Method	POST
URI	http: /lectures/1/lectureContent/2/userSeqs/2/qa
Parameter	qa_title, qa_content

- Response

Table 20 Table of write question response

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 400	
Success Response Body	Access Token	Token for access
	Message	Success message
Failure Response Body	Message	Fail message

5.8.2. Get Question

- Request

Table 21 Table of get question response

Attribute	Detail
Method	GET
URI	http: /lectures/1/lectureContent/2/userSeqs/2/qa/1
Parameter	qa_seq

- Response

Table 22 Table of get question response

Attribute		Detail
Success Code	200 OK	
Failure Code	HTTP error code = 400	
Success Response Body	Access Token	Token for access
	Message	Success message
Failure Response Body	Message	Fail message

6. Database Design

6.1. Objectives

해당 절에서는 시스템 데이터 구조와 데이터 구조를 데이터베이스에 표시하는 방법을 설명한다. 우선 ER-diagram(Entity Relationship diagram)을 통해 엔티티와 그 관계를 식별한다. 이후 관계형 스키마 및 SQL DDL(Data Description Language) 명세서를 생성한다.

6.2. EER Diagram

EER-diagram 은 모델의 테이블 간의 관계를 시각적으로 표현한다. 테이블이 다른 테이블과 여러 관계가 있는 경우 이를 나타내기 위해 닻 모양이 사용된다. 테이블가 다른 테이블과 하나의 관계만 있는 경우, 십자가 모양을 사용하여 이를 나타낸다. 테이블의 속성은 테이블 안에 표현된다. 테이블을 고유하게 식별하는 키 속성은 속성의 왼쪽에 키 모양으로 표시되어 있다.

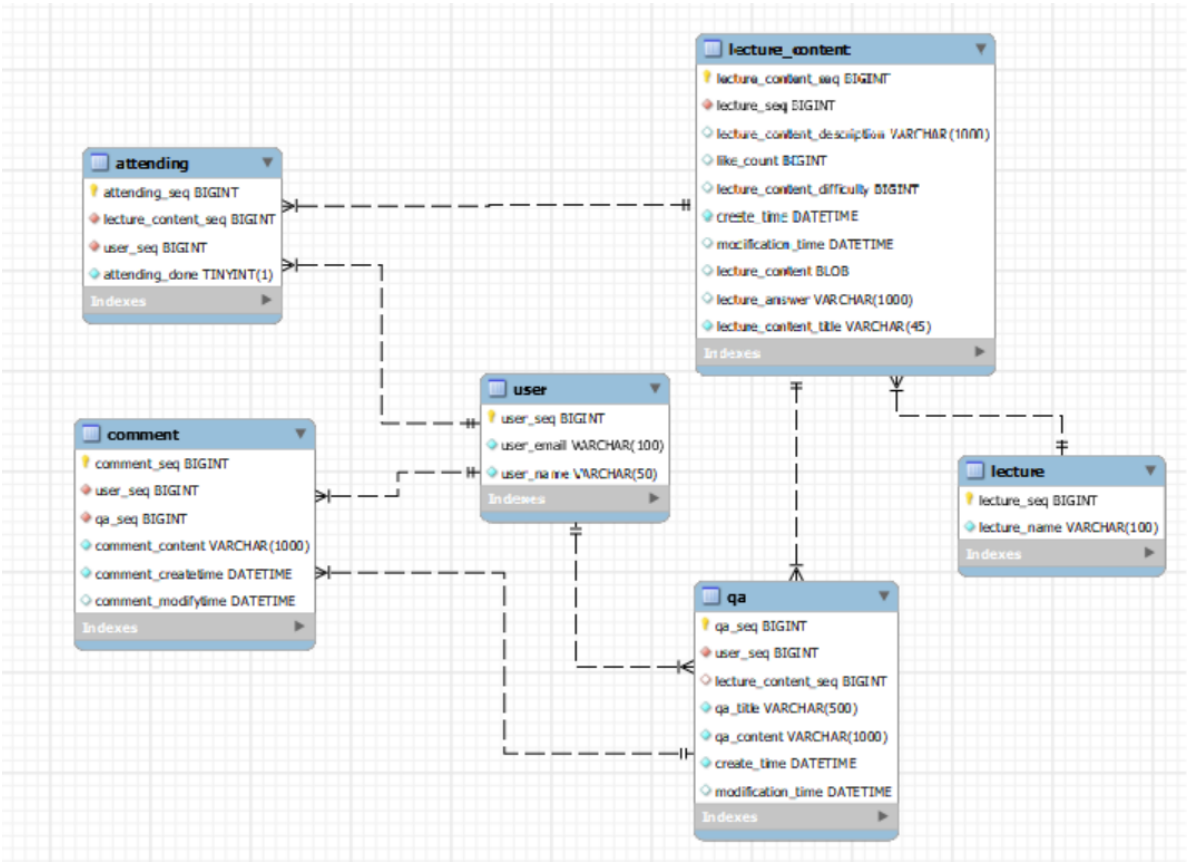


Figure 29 EEE-diagram

6.2.1. Table

6.2.1.1. user

User 테이블은 Clawlearn 의 사용자를 나타낸다. user_seq, user_email, user name 으로 구성되어 있고, user_seq 이 주키이다. user_seq 속성을 통해 attending, comment, qa 테이블과 일대다 관계를 맺는다.

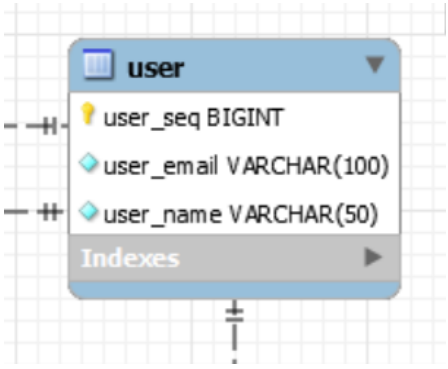


Figure 30 EEE diagram, user

6.2.1.2. attending

attending 테이블은 Clawlearn 사용자의 참여를 나타낸다. attending_seq, lecture_content_seq, user_seq, attending_done 으로 구성되어 있고 attending_seq 이 주키이다. lecture_content_seq 속성을 통해 lecture_content 테이블과 다대일 관계를 맺는다. user_seq 속성을 통해 user 테이블과 관계를 다대일 관계를 맺는다.

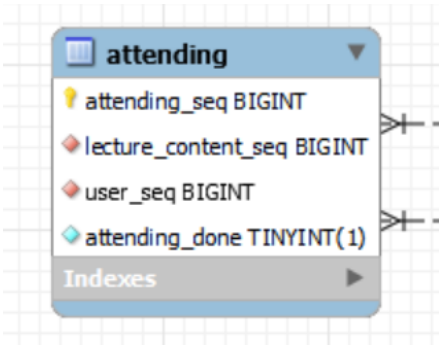
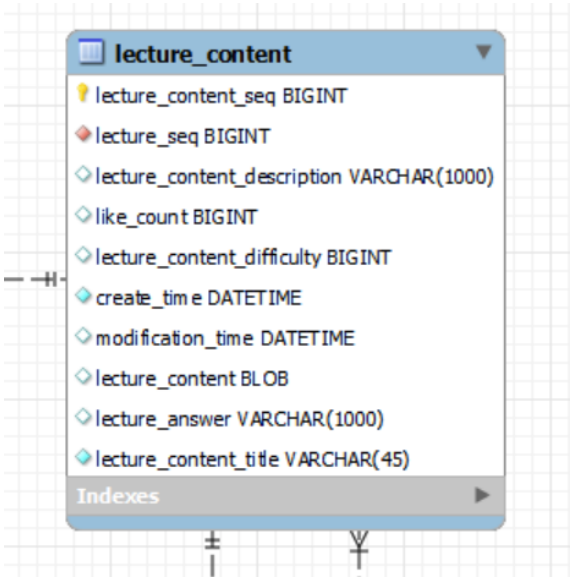


Figure 31 EER diagram, attending

6.2.1.3. lecture_content

lecture_content 테이블은 Clawlearn 의 강의 내용을 나타낸다. lecture_content_seq, lecture_seq, lecture_content_description, like_count, create_time, modification_time,

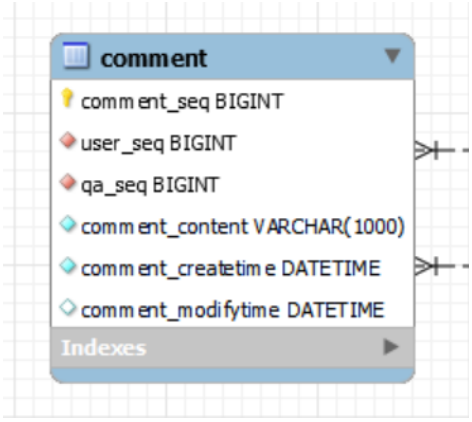
lecture_content, lecture_answer, lecture_content_title 으로 구성되어 있고 lecture_content_seq 이 주키이다. lecture_content_seq 속성을 통해 lecture_content 테이블과 관계를 다대일 관계를 맺는다. user_seq 속성을 통해 user 테이블과 관계를 일대다 관계를 맺고, lecture_seq 속성을 통해 lecture 테이블과 다대일 관계를 맺는다. lecture_content_seq 속성을 통해 qa 테이블과 일대다 관계를 맺는다.



[Figure 28] EER diagram, lecture_content

6.2.1.4. comment

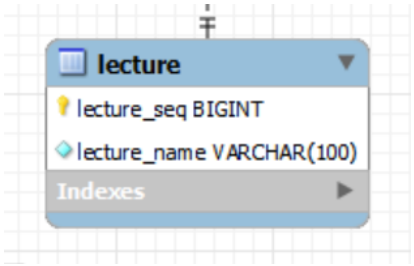
comment 테이블은 댓글을 나타낸다. comment_seq, user_seq, qa_seq, comment_content, comment_createtime, comment_modifytime 으로 구성되어 있고 comment_seq 이 주키이다. lecture_content_seq 속성을 통해 lecture_content 테이블과 관계를 다대일 관계를 맺는다. user_seq 속성을 통해 user 테이블과 다대일 관계를 맺고, qa_seq 속성을 통해 qa 테이블과 다대일 관계를 맺는다.



[Figure 28] EER diagram, comment

6.2.1.5. lecture

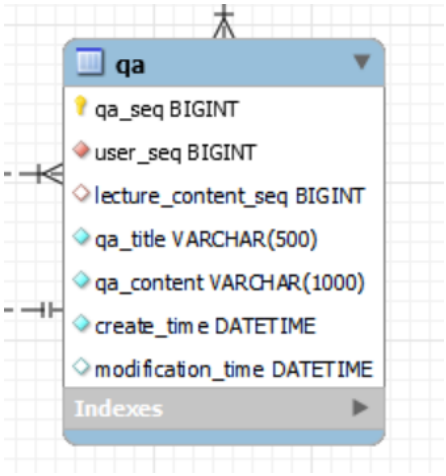
lecture 테이블은 Clawlearn 강의를 나타낸다. lecture_seq, lecture_name 으로 구성되어 있고 lecture_seq 이 주키이다. lecture_seq 속성을 통해 lecture_content 테이블과 일대다 관계를 맺는다.



[Figure 28] EER diagram, lecture

6.2.1.6. qa

qa 테이블은 qa 를 나타낸다. qa_seq, user_seq, lecture_content_seq qa_title, qa_content, create_time, modification_time 으로 구성되어 있고 qa_seq 이 주키이다. user_seq 속성을 통해 user 테이블과 다대일 관계를 맺는다. qa_seq 속성을 통해 comment 테이블과 관계를 일대다 관계를 맺는다.



[Figure 28] EER diagram, qa

6.3. SQL DDL

6.3.1. User

```
CREATE TABLE `user` (
  `user_seq` bigint NOT NULL AUTO_INCREMENT COMMENT 'PK',
  `user_email` varchar(100) NOT NULL COMMENT '구글 이메일',
  `user_name` varchar(50) NOT NULL COMMENT '학생 이름',
  PRIMARY KEY (`user_seq`)
) ENGINE=InnoDB AUTO_INCREMENT=7 DEFAULT CHARSET=utf8mb3;
```

6.3.2. Attending

```
CREATE TABLE `attending` (
  `attending_seq` bigint NOT NULL AUTO_INCREMENT COMMENT 'PK',
  `lecture_content_seq` bigint NOT NULL COMMENT 'FK',
  `user_seq` bigint NOT NULL COMMENT 'FK',
  `attending_done` tinyint(1) NOT NULL COMMENT '수강 완료 여부',
  PRIMARY KEY (`attending_seq`),
  KEY `lecture_content_seq` (`lecture_content_seq`),
  KEY `user_seq` (`user_seq`),
  CONSTRAINT `attending_ibfk_1` FOREIGN KEY (`lecture_content_seq`)
            REFERENCES `lecture_content` (`lecture_content_seq`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `attending_ibfk_2` FOREIGN KEY (`user_seq`) REFERENCES `user` (`user_seq`)
            ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=9 DEFAULT CHARSET=utf8mb3;
```

6.3.3. Lecture_content

```
CREATE TABLE `lecture_content` (
  `lecture_content_seq` bigint NOT NULL AUTO_INCREMENT COMMENT 'PK',
  `lecture_seq` bigint NOT NULL COMMENT 'FK',
  `lecture_content_description` varchar(1000) DEFAULT NULL COMMENT '강좌 설명',
  `like_count` bigint DEFAULT NULL COMMENT '강좌 좋아요 갯수',
  `lecture_content_difficulty` bigint DEFAULT NULL,
  `create_time` datetime NOT NULL COMMENT '강좌 생성 날짜',
  `modification_time` datetime DEFAULT NULL COMMENT '강좌 수정 날짜',
  `lecture_content` blob COMMENT '강좌 내용',
  `lecture_answer` varchar(1000) DEFAULT NULL COMMENT '실습 강좌 정답',
  `lecture_content_title` varchar(45) NOT NULL,
  PRIMARY KEY (`lecture_content_seq`),
  KEY `lecture_seq` (`lecture_seq`),
  CONSTRAINT `lecture_content_ibfk_1` FOREIGN KEY (`lecture_seq`) REFERENCES `lecture` (`lecture_seq`)
        ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=11 DEFAULT CHARSET=utf8mb3;
```

6.3.4. Comment

```
CREATE TABLE `comment` (
  `comment_seq` bigint NOT NULL AUTO_INCREMENT COMMENT 'PK',
  `user_seq` bigint NOT NULL COMMENT 'FK',
  `qa_seq` bigint NOT NULL COMMENT 'FK',
  `comment_content` varchar(1000) NOT NULL COMMENT '댓글 내용',
  `comment_createtime` datetime NOT NULL COMMENT '댓글 작성 날짜',
  `comment_modifytime` datetime DEFAULT NULL COMMENT '댓글 수정 날짜',
  PRIMARY KEY (`comment_seq`),
  KEY `user_seq` (`user_seq`),
  KEY `qa_seq` (`qa_seq`),
  CONSTRAINT `comment_ibfk_1` FOREIGN KEY (`user_seq`) REFERENCES `user` (`user_seq`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `comment_ibfk_2` FOREIGN KEY (`qa_seq`) REFERENCES `qa` (`qa_seq`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=utf8mb3;
```

6.3.5. Lecture

```
CREATE TABLE `lecture` (
  `lecture_seq` bigint NOT NULL AUTO_INCREMENT COMMENT 'PK',
  `lecture_name` varchar(100) NOT NULL COMMENT '강의 이름(기초파이썬, 웹페이지 구조..)',
  PRIMARY KEY (`lecture_seq`)
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8mb3;
```

6.3.6. Qa

```
CREATE TABLE `qa` (
  `qa_seq` bigint NOT NULL AUTO_INCREMENT COMMENT 'PK',
  `user_seq` bigint NOT NULL COMMENT 'FK',
  `lecture_content_seq` bigint DEFAULT NULL COMMENT 'FK',
  `qa_title` varchar(500) NOT NULL COMMENT '질문 제목',
  `qa_content` varchar(1000) NOT NULL COMMENT '질문 내용',
  `create_time` datetime NOT NULL COMMENT '작성 날짜',
  `modification_time` datetime DEFAULT NULL COMMENT '수정 날짜',
  PRIMARY KEY (`qa_seq`),
  KEY `user_seq` (`user_seq`),
  KEY `lecture_content_seq` (`lecture_content_seq`),
  CONSTRAINT `qa_ibfk_1` FOREIGN KEY (`user_seq`) REFERENCES `user` (`user_seq`)
            ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `qa_ibfk_2` FOREIGN KEY (`lecture_content_seq`)
            REFERENCES `lecture_content` (`lecture_content_seq`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=6 DEFAULT CHARSET=utf8mb3;
```

7. Testing Plan

7.1. Objectives

이 장에서는 Development testing, Release testing, User testing 의 세 가지 주요 하위 그룹으로 구성된 testing plan 에 대해서 설명한다. 이러한 테스트는 프로젝트의 잠재적인 오류와 결함을 감지함으로써 서비스를 완벽하게 작동시켜 안정적인 출시를 할 수 있게 하는 매우 중요한 개발 과정이다.

7.2. Testing Policy

7.2.1. Development Testing

Development testing 은 주로 소프트웨어 개발의 잠재적 위험을 줄이고 시간과 비용의 절약을 목적으로 광범위한 결함 예방 및 탐지 전략의 동기화된 적용을 위해 수행된다. 이 단계에서는 소프트웨어가 충분한 테스트를 거치지 않았기 때문에 불안정할 수 있으며 구성 요소가 서로 충돌할 수 있다. 따라서 이 단계에서 정적 코드 분석, 데이터 흐름 분석, 피어 코드 검토, unit test 가 수행되어야 한다. 이러한 프로세스를 통해 1) 소프트웨어의 정체성을 정의하는 성능, 2) 안전 및 무장애 작동 보장을 위한 신뢰성, 3) 보안 실현에 중점을 두고 있다.

7.2.1.1. Performance

코드 테스트 알고리즘은 우리 시스템에서 가장 정확성을 요구하는 작업이기 때문에 개발자가 코딩 결과를 검토하고 사용자에게 정답 여부를 제시하는데 있어서 정확성을 높이는 것이 중요하다. 우리는 다양한 학습 알고리즘에 대한 실습 사례를 준비하고 코드 테스트 결과의 정확성을 평가하고, 사용자가 작성한 코드 테스트 페이지와 서버간의 통신에 관한 코드의 흐름을 개선할 것이다. 또한 사용자의 로그인 기록, 수강 완료 기록 등의 사용자 정보와 DB 와의 의존성도 충족되어야 하므로 이를 중점적으로 검토할 예정이다.

7.2.1.2. Reliability

시스템이 고장 없이 안전하게 작동하려면 먼저 시스템을 구성하는 하위 구성 요소 및 장치가 올바르게 작동하고 연결되어야 한다. 따라서 우리는 unit 개발 단계부터 개발 테스트를 하고 각 unit 이 시스템에 통합되어 있는 동안 반복적으로 오류 여부를 점검할 것이다.

7.2.1.3. Security

사용자와 시스템의 정보 보안은 개발자가 처리해야 할 중요한 문제이다. 정보의 가치에 무관하게 시스템이 원치 않는 방문자로부터 정보를 보호해야 한다. 시스템 보안을 위해 거의 완성된 버전의 웹에 접속하여 보안 문제를 파악하고 수동 코드 검토를 통해 정보 보안을 수행할 것이다. 또한, 우리는 Maltego, BackTrack 등이 제공하는 다른

웹 보안 테스트 서비스를 이용할 것이며, 이를 통해 개발자들이 웹 개발에 있어서 간과한 취약점을 확인할 수 있다.

7.2.2. Release Testing

제품을 시장과 고객에게 출시하는 것은 소프트웨어 개발에 있어 중요한 부분을 차지한다. 기술적으로 좋은 소프트웨어도 잘못된 출시 방식으로 인해 잘못될 수 있다. 따라서 제품과 시장 간 연결을 더 잘하기 위해서는 Release Testing 이 필수적이다. Release Testing 은 새 버전의 소프트웨어와 어플리케이션을 테스트하여 소프트웨어가 완벽하게 출시될 수 있으므로 운영에 아무런 결함이 없는지 확인하는 것이다. 해당 작업은 수명이 다하기 전에 수행되어야 한다. 소프트웨어 출시 수명 주기에 따라 테스트는 일반적으로 소프트웨어의 기본 구현이 완료된 'Alpha' 버전부터 시작된다. 알파 버전으로 개발 테스트를 시작하고 베타 버전을 출시하여 사용자 및 릴리스 테스트를 포함한 추가 테스트를 진행할 계획이다. 베타 버전이 출시되면 실제 사용자로부터도 피드백을 받을 수 있다.

7.2.3. User Testing

우리는 필요한 사용자 테스트를 진행할 수 있는 가능한 시나리오와 현실적인 상황을 설정해야 한다. 우리는 Crawlearn 의 사용자가 100 명이라고 가정한다. 이러하 상황에서 Crawlearn 의 베타 버전을 배포하고 자체 유스 케이스 테스트를 진행하면서 사용자 검토한다.

7.2.4. Testing Case

Test case 는 기능, 성능, 보안의 세 가지 기본 측면에 따라 설정될 것이다. 기능은 주로 상호작용 측면에서 test 된다. 각 측면에서 5 건의 테스트 케이스를 설정하고 해당 테스트를 바탕으로 결과지를 작성할 계획이다.

8. Development Plan

8.1. Objectives

이 장에서는 응용 프로그램 개발을 위한 기술과 환경을 설명합니다.

8.2. Frontend Environment

8.2.1. Figma

Figure 32 Figma logo

Figma 는 기본적으로 웹 기반 인 벡터 그래픽 편집기 및 프로토타이핑 도구이며 macOS 및 Windows 용 데스크톱 응용 프로그램에서 추가 오프라인 기능을 사용할 수 있다. Figma 의 기능 세트는 실시간 협업에 중점을 두고 사용자 인터페이스 및 사용자 경험 디자인에서의 사용에 중점을 둔다. 이러한 도구를 기반으로 웹 사이트 프로토타입을 디자인하며, 공통된 css 정보를 얻어낼 수 있었다.

8.2.2. vscode

Figure 33 vscode logo

비주얼 스튜디오 코드(영어: Visual Studio Code) 또는 코드(code)는 마이크로소프트가 마이크로소프트 윈도우, macOS, 리눅스용으로 개발한 소스 코드 편집기이다. 디버깅

지원과 Git 제어, 구문 강조 기능 등이 포함되어 있으며, 사용자가 편집기의 테마와 단축키, 설정 등을 수정할 수 있다.

비주얼 스튜디오 코드는 깃허브가 개발한 일렉트론 프레임워크를 기반으로 구동된다.

협업을 위한 깃과 여러 기능 등이 포함되어 있으며 웹사이트 개발에 많이 사용되는 편집기이므로 클라이언트, 서버 구축을 위해 vscode 를 사용하였다.

8.3. Backend Environment

8.3.1. Github

Figure 34 Github logo

깃허브는 분산 버전 관리 툴인 깃 저장소 호스팅을 지원하는 웹서비스이다. 이를 통해 같은 작업 폴더에서 작업을 하더라도 각자의 브랜치에서 작업을 하고 이를 깃허브에 푸시를 해놓기 때문에 장소의 제약 없이 팀원들의 코드를 볼 수 있다는 장점을 가지고 있다. 따라서 우리는 프로젝트 코드 관리를 위한 저장소로 깃허브를 사용했다.

8.3.2. Flask

Figure 35 Flask logo

Flask 는 파이썬으로 작성된 마이크로 웹 프레임 워크의 하나로 장고에 비해 프레임워크 자체가 가볍고 자유도가 높은 프레임워크이다. 플라스크는 다른 파이썬 웹 프레임워크와 달리 규칙이 엄격하지 않아 개발자가 개발하기 원할한 웹 프레임 워크 이다.

8.3.3. Postman

Figure 36 Postman logo

Postman 은 API 개발을 도와주는 플랫폼이다. Postman 을 통해 서버가 개발 API 를 테스트 할 수 있고 더 나아가 변수 및 환경, request 설명, 테스트 및 사전 요청에 필요한 스크립트 작성등을 할 수 있는 프레임 워크이다.

8.3.4. MySQL

Figure 37 MySQL logo

MySQL 은 오픈소스 관계형 데이터 베이스 관리 시스템이다. 이들은 다중 스레드, 다중 사용자 형시의 구조 질의어 형식의 데이터 베이스 관리 시스템으로 오라클에서 관리 및 지원하고 있다.

8.4. Constraints

해당 시스템은 이 문서에 언급된 내용을 기반으로 설계 및 구현된다. 기타 세부사항은 개발자의기술을 반영하여 설계 및 구현하되, 다음 사항을 준수한다.

- 검증된 기술을 사용한다.
- 별도의 라이선스가 필요하거나 비용을 지불해야 하는 기술이나 소프트웨어는 사용하지 않는다. (시스템에 필요한 유일한 기술 또는 소프트웨어인 경우 합의 하에 이 조항을 제외한다)
- 전반적인 시스템 성능 향상을 지향한다.
- 사용자 친화적 시스템을 지향한다.
- 오픈 소스 소프트웨어 사용을 지향한다.
- 시스템 자원 낭비를 지양하며 이를 위해 최적화 작업을 지향한다.
- 시스템 비용 및 유지보수 비용을 고려한다.
- 시스템의 가용성과 확장성을 고려한다.
- 반응 시간이 4초를 넘지 않는다.
- 모든 데이터 인코딩은 UTF-8을 따른다.
- Node.js version 17.5, Python version 3.7 환경에서 개발한다
- 브라우저는 최소 Chrome version 83을 지원하며 Chrome version 100를 대상으로 개발한다.
- 테스트는 Chrome version 100을 대상으로 한다.

8.5. Assumptions and Dependencies

이 문서에 기술된 모든 시스템은 네트워크 통신이 원활하고 브라우저가 react 를 지원한다는 가정 하에 작성되었다. 따라서 모든 콘텐츠는 Chrome version 100, Microsoft Edge version 98, Safari version 16, Firefox version 96 이상의 브라우저를 기준으로 작성되었으며 다른 브라우저나 버전에는 동작하지 않을 수 있다.

9. Supporting Information

9.1. Software Design Specification

이 소프트웨어 디자인 명세서는 IEEE 권장 사항에 따라 작성되었다 (IEEE Recommended Practice for Software Design Description, IEEE-Std-1016).

9.2. Document History

Date	Version	Description	Writer
2022/05/13	0.1	1, 2	이정우
2022/05/13	0.1	3.1-3.2.3	배지현
2022/05/13	0.1	3.2.4-3.2.7	이민영
2022/05/13	0.1	4	김진환
2022/05/13	0.1	5, 6	정미서
2022/05/14	0.2	7	이정우, 정미서
2022/05/14	0.2	8	이민영, 김진환, 배지현
2022/05/14	0.2	9	배지현
2022/05/15	1.1	통합	이민영