Online Coding Test Web Platform

2022.11.13

소프트웨어공학개론
TEAM 7

Team Leader	김호진
Team Member	김양선
Team Member	변영민
Team Member	이주빈
Team Member	임세아
Team Member	전윤태

CONTENTS

1. Introduction 10

1.1. Objectives 10

1.2. Applied Diagrams 10

1.2.1. UML 10

1.2.2. Use Case Diagram 11

1.2.3. Sequence Diagram 11

1.2.4. Class Diagram 11

1.2.5. Context Diagram 11

1.2.6. Entity Relationship Diagram 12

1.3. Applied tools 12

1.3.1. Microsoft PowerPoint 12

1.3.2. Diagrams 12

1.3.3. ERD cloud 12

1.3.4. Figma 12

1.4. References 12

2. System Architecture – Overall 13

2.1. Objectives 13

2.2. System Organization 13

2.2.1. Context Diagram 14

2.2.2. Sequence Diagram 14

2.2.3. Use Case Diagram 15

3. System Architecture – Frontend 16

3.1. Objectives 16

3.2. Subcomponents 16

3.2.1. Profile 16

3.2.1.1. Attributes 16

3.2.1.2. Methods 16

3.2.1.3. Class Diagram 17

3.2.1.4. Sequence Diagram 17

3.2.2. Login, Logout 18

3.2.2.1. Attributes 18

3.2.2.2. Methods 18

3.2.2.3. Class Diagram 18

3.2.2.4. Sequence Diagram 19

3.2.3. Class 19

3.2.3.1. Attributes 19

3.2.3.2. Methods 20

3.2.3.3. Class Diagram 20

3.2.3.4. Sequence Diagram 20

3.2.4. Problem 21

3.2.4.1. Attributes 21

3.2.4.2. Methods 21

3.2.4.3. Class Diagram 22
3.2.4.4. Sequence Diagram 22
3.2.5. Code 23
3.2.5.1. Attributes 23
3.2.5.2. Methods 23
3.2.5.3. Class Diagram 23
3.2.5.4. Sequence Diagram 23
3.2.6. Terminal 24
3.2.6.1. Attributes 24
3.2.6.2. Methods 24
3.2.6.3. Class Diagram 25
3.2.6.4. Sequence Diagram 25
4. System Architecture – Backend 26
4.1. Purpose 26
4.2. Overall Architecture 26
4.3. Subcomponents 27
4.3.1. User system 27
4.3.1.1. User System Class Diagram 27
4.3.1.2. User Sequence Diagram 28
4.3.2. Lecture System 29
4.3.2.1. Lecture System Class Diagram 29
4.3.2.2. Lecture System Sequence Diagram 30
4.3.3. Problem & Code 31
4.3.3.1. Problem & Code System Class Diagram 31

4.3.3.2. Problem & Code System Sequence Diagram 32

5. Protocol Design 33

5.1. Objectives 33

5.2. JSON 33

5.3. CSRF Token 33

5.4. Authentication 34

5.4.1. Login 34

5.4.2. Logout 34

5.5. Classes 35

5.5.1. Classes List 35

5.5.2. Class Enrollment 36

5.5.3. Class Information 37

5.6. Problem 38

5.6.1. Problem Information 38

5.7. Code 38

5.7.1. Code Execution 38

5.7.2. Code Grading 39

5.7.3. Code Submission 40

5.7.4. Code Submission Results 41

5.7.5. Code Store 41

5.7.6. Code Load 42

6. Database Design 43

6.1. Objectives 43

6.2. EER Diagram 43

6.2.1. Table 44

6.2.1.1. User 44

6.2.1.2. Class 44

6.2.1.3. Enrollment 45

6.2.1.4. Problem 45

6.2.1.5. Submission 46

6.2.1.6. Storage 46

6.3. SQL DDL 47

6.3.1. User 47

6.3.2. Class 47

6.3.3. Enrollment 47

6.3.4. Problem 48

6.3.5. Storage 48

6.3.6. Submission 48

7. Testing Plan 49

7.1. Objectives 49

7.2. Testing Policy 49

7.2.1. Development Testing 49

7.2.1.1. Performance 49

7.2.1.2. Reliability 49

7.2.1.3. Security 50

7.2.2. User Testing 50

7.2.3. User Testing 50

7.2.4. Testing Case 50

8. Development Plan 51

8.1. Objectives 51

8.2. Frontend Environment 51

8.2.1. Figma 51

8.2.2. vscode 51

8.3. Backend Environment 52

8.3.1. Github 52

8.3.2. Django 52

8.3.3. Django REST framework 53

8.3.4. SQLite 53

8.4. Constraints 54

8.5. Assumptions and Dependencies 54

9. Supporting Information 55

9.1. Software Design Specification 55

9.2. Document History 55

List of Figures

FIGURE 1 OVERALL SYSTEM ARCHITECTURE 13
FIGURE 2 OVERALL CONTEXT DIAGRAM 14
FIGURE 3 OVERALL SEQUENCE DIAGRAM 14
FIGURE 4 OVERALL USE CASE DIAGRAM 15
FIGURE 5 CLASS DIAGRAM - PROFILE 17
FIGURE 6 SEQUENCE DIAGRAM - PROFILE 17
FIGURE 7 CLASS DIAGRAM – LOGIN AND LOGOUT 18
FIGURE 8 SEQUENCE DIAGRAM - LOGIN AND LOGOUT 19
FIGURE 9 CLASS DIAGRAM – CLASS 20
FIGURE 10 SEQUENCE DIAGRAM – CLASS 20
FIGURE 11 CLASS DIAGRAM - PROBLEM 22
FIGURE 12 SEQUENCE DIAGRAM - PROBLEM 22
FIGURE 13 CLASS DIAGRAM - CODE 23
FIGURE 14 SEQUENCE DIAGRAM - CODE 24
FIGURE 15 CLASS DIAGRAM – TERMINAL 25
FIGURE 16 SEQUENCE DIAGRAM - TERMINAL 25
FIGURE 17 OVERALL ARCHITECTURE 26
FIGURE 18 CLASS DIAGRAM – USER SYSTEM 27
FIGURE 19 SEQUENCE DIAGRAM – USER SYSTEM 28
FIGURE 20 CLASS DIAGRAM - LECTURE SYSTEM 29
FIGURE 21 SEQUENCE DIAGRAM – LECTURE SYSTEM 30
FIGURE 22 CLASS DIAGRAM – PROBLEM AND CODE SYSTEM 31
FIGURE 23 SEQUENCE DIAGRAM – PROBLEM AND CODE SYSTEM 32
FIGURE 24 EER DIAGRAM 43
FIGURE 25 EER DIAGRAM - USER 44
FIGURE 26 EER DIAGRAM - CLASS 44
FIGURE 27 EER DIAGRAM - ENROLLMENT 45
FIGURE 28 EER DIAGRAM - PROBLEM 45
FIGURE 29 EER DIAGRAM - SUBMISSION 46
FIGURE 30 EER DIAGRAM - STORAGE 46
FIGURE 31 SQL DDL - USER 47
FIGURE 32 SQL DDL - CLASS 47
FIGURE 33 SQL DDL - ENROLLMENT 47
FIGURE 34 SQL DDL – PROBLEM 48
FIGURE 35 SQL DDL - STORAGE 48
FIGURE 36 SQL DDL - SUBMISSION 48
FIGURE 37 FIGMA LOGO 51
FIGURE 38 VSCODE LOGO 51
FIGURE 39 GITHUB LOGO 52
FIGURE 40 DJANGO LOGO 52
FIGURE 41 DJANGO REST FRAMEWORK LOGO 53
FIGURE 42 SQLITE LOGO 53

List of Tables

TABLE 1 TABLE OF LOG-IN REQUEST 34
TABLE 2 TABLE OF LOG-IN RESPONSE 34
TABLE 3 TABLE OF LOG-OUT REQUEST 34
TABLE 4 TABLE OF LOG-OUT RESPONSE 35
TABLE 5 TABLE OF CLASSES LIST REQUEST 35
TABLE 6 TABLE OF CLASSES LIST RESPONSE 35
TABLE 7 TABLE OF CLASS ENROLLMENT REQUEST 36
TABLE 8 TABLE OF CLASS ENROLLMENT RESPONSE 36
TABLE 9 TABLE OF CLASS INFORMATION REQUEST 37
TABLE 10 TABLE OF CLASS INFORMATION RESPONSE 37
TABLE 11 TABLE OF PROBLEM INFORMATION REQUEST 38
TABLE 12 TABLE OF PROBLEM INFORMATION RESPONSE 38
TABLE 13 TABLE OF CODE EXECUTION REQUEST 38
TABLE 14 TABLE OF CODE EXECUTION RESPONSE 39
TABLE 15 TABLE OF CODE GRADING REQUEST 39
TABLE 16 TABLE OF CODE GRADING RESPONSE 39
TABLE 17 TABLE OF CODE SUBMISSION REQUEST 40
TABLE 18 TABLE OF CODE SUBMISSION RESPONSE 40
TABLE 19 TABLE OF CODE SUBMISSION RESULTS REQUEST 41
TABLE 20 TABLE OF CODE SUBMISSION RESULTS RESPONSE 41
TABLE 21 TABLE OF CODE STORE REQUEST 41
TABLE 22 TABLE OF CODE STORE RESPONSE 42
TABLE 23 TABLE OF CODE LOAD REQUEST 42
TABLE 24 TABLE OF CODE LOAD RESPONSE 42
TABLE 25 DOCUMENT HISTORY 55

1. Introduction

본 서비스는 성균관대학교 학생들에게 웹 기반의 코딩 테스트 사이트를 제공하기 위해서 고안되었다. 코딩 테스트는 개발자로 취업을 하기 위해 필수적으로 거쳐야 하는 관문으로 많은 회사에서 학력, 나이, 스펙보다 순수한 실력과 현실적인 역량을 확인하고자 코딩 테스트를 통해 업무 수행 능력을 평가하고 있다. 이를 대비하기 위한 다양한 코딩 연습 사이트들이 현존하고 있지만, 수많은 문제 중에 필요한 것을 선정해 학습하는 것은 쉬운 일이 아니다. 따라서 이번 프로젝트를 통해 성균관대 학생에게 적합한 새로운 코딩 사이트를 제작하고자 한다. 실제로 학생들이 수강하는 과목에서 요구하는 과제를 직접 수행, 제출, 채점함은 물론 관련된 정보를 획득할 수 있도록 웹 사이트를 제작함으로써 성균관대 학생들이 편리하게 코딩 연습을 행할 수 있도록 도와주는 것이 본 프로젝트의 목표이다. 본 디자인 명세서는 프로젝트 시행을 위해 사용되거나 사용될 디자인에 관한 설명을 담고 있으며, 이전 단계에서 작성된 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

UML은 Unified Modeling Language의 약자이다. 간단히 말해서, UML 은 소프트웨어를 모델링하고 문서화하는 현대적 접근 방식이다. 실제로 이것은 가장 인기 있는 비즈니스 프로세스 모델링 기술 중 하나이다. UML은 비즈니스 분석가, 소프트웨어 설계자 및 개발자들이 기존의 혹은 새로운 비즈니스 프로세스의 구조 및 동작을 기술, 지정, 설계 및 문서화하는데 사용하는 공통 언어이다. UML은 은행, 금융, 인터넷, 항공우주, 의료 등 다양한 응용 분야에서 적용될 수 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. 시각적 표현을 사용함으로써 소프트웨어나 비즈니스 프로세스에서 발생할 수 있는 결함이나 오류를 더 잘 이해할 수 있다.

1.2.2. Use Case Diagram

시스템의 functional requirement을 파악하는 일은 시스템 구현에 매우 중요하다. Use Case Diagram은 시스템의 high-level requirement를 분석하는데 사용되며, 다양한 use case를 통해 요구사항이 표현된다. UML 다이어그램의 세 가지 주요 구성 요소는 다음과 같다. Functional requirements는 use case로 표현되고, 동작을 설명하는 동사로 나타난다. Actors는 시스템과 상호작용하며, 인간이나 조직, 내부 혹은 외부 응용 프로그램이 될 수 있다. 마지막은 actor와 user case 간의 관계로 직선 화살표를 통해 나타난다.

1.2.3. Sequence Diagram

Sequence Diagram은 컴퓨터 공학 커뮤니티 이외에 비즈니스 애플리케이션 개발을 위한 설계 수준 모델에서도 매우 중요한 UML 다이어그램이다. 시각적인 설명이 가능하다는 특성으로 인해 비즈니스 프로세스를 묘사하는데 있어 최근 인기를 끌고 있다. 이름에서 알 수 있듯이 Sequence diagram 은 행위자와 객체 간에 발생하는 메시지와 상호 작용의 순서를 설명한다. 모든 의사소통은 연대순으로 표현되며 Actor나 객체는 다른 객체가 그들과 통신하는 상황과 같이 필요한 경우에서만 활성화될 수 있다.

1.2.4. Class Diagram

UML의 Class Diagram은 시스템의 클래스, 속성, 연산 및 객체 간의 관계를 보여줌으로써 시스템의 구조를 설명하는 정적 구조 다이어그램의 일종이다. 클래스가 객체의 구성 요소인 것처럼 Class diagram 은 UML의 구성 요소이다. Class diagram의 다양한 구성 요소는 실제로 프로그래밍 될 주요 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)은 Data Flow Diagram의 highest level이다. 전체 context와 경계를 설정하는 하나의 프로세스만을 포함하며, 시스템과 외부 actor(entity) 사이의 정보 흐름을 식별한다. Context Diagram은 일반적으로 requirement specification에 포함되는데, 해당 명세서는 프로젝트의 모든 stakeholder가 읽어야 하므로 이들이 항목을 이해할 수 있도록 쉬운 언어로 작성되어야 한다. Context diagram의 목적은 시스템의 요구 사항과 제약을 식별하는데 고려되어야 하는 외부 요인 및 사건에 주목하는 것이다.

Context diagram 은 프로젝트 초기에 조사 대상의 범위를 결정하기 위해 사용되기도 한다. 따라서, Context Diagram은 시스템과 상호 작용할 수 있는 모든 외부 entity를 나타낸다. 전체 소프트웨어 시스템은 단일 프로세스로 표시되며 시스템은 다이어그램 상의 중앙에 배치되고 (내부 구조에 대한 세부 정보는 없이) 모든 외부 entity를 비롯한 상호 작용하는 시스템 및 환경에 의해 둘러싸여 있다.

1.2.6. Entity Relationship Diagram

Entity Relationship Diagram은 데이터베이스에 저장된 entity set의 관계를 보여준다. Entity는 데이터의 구성 요소인 객체이고, entity set는 유사한 entity의 집합이다. 이러한 entity에는 해당 속성을 정의하는 속성이 있을 수 있다. ER Diagram은 entity와 속성을 정의하고 entity 간의 관계를 표시함으로써 데이터베이스의 논리적 구조를 보여준다. Entity Relationship Diagram은 데이터베이스의 설계를 스케치하는 데 사용된다.

1.3. Applied tools

1.3.1. Microsoft PowerPoint

다양한 Diagram을 쉽고 빠르게 만들 수 있는 웹 기반 도구이다.

1.3.2. Diagrams

HTML5 및 JavaScript로 개발된 무료 오픈 소스 소프트웨어로, UML Diagram을 쉽고 빠르게 만들 수 있다.

1.3.3. ERD cloud

Entity Relationship Diagram을 쉽고 빠르게 만들 수 있는 웹 기반 데이터베이스 모델링 도구이다.

1.3.4. Figma

Frontend 디자인을 쉽고 빠르게 할 수 있도록 도와주는 웹 기반 모델링 도구이다.

1.4. References

본 디자인 명세서는 다음 자료를 참조하였다.

- Team 2, 2022 Spring. Software Design Document, SKKU.

2. System Architecture – Overall

2.1. Objectives

이 장에서는 시스템의 구조를 나타낼 것이다. 본 프로젝트의 frontend부터 backend까지 상세히 설명할 예정이다.

2.2. System Organization

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

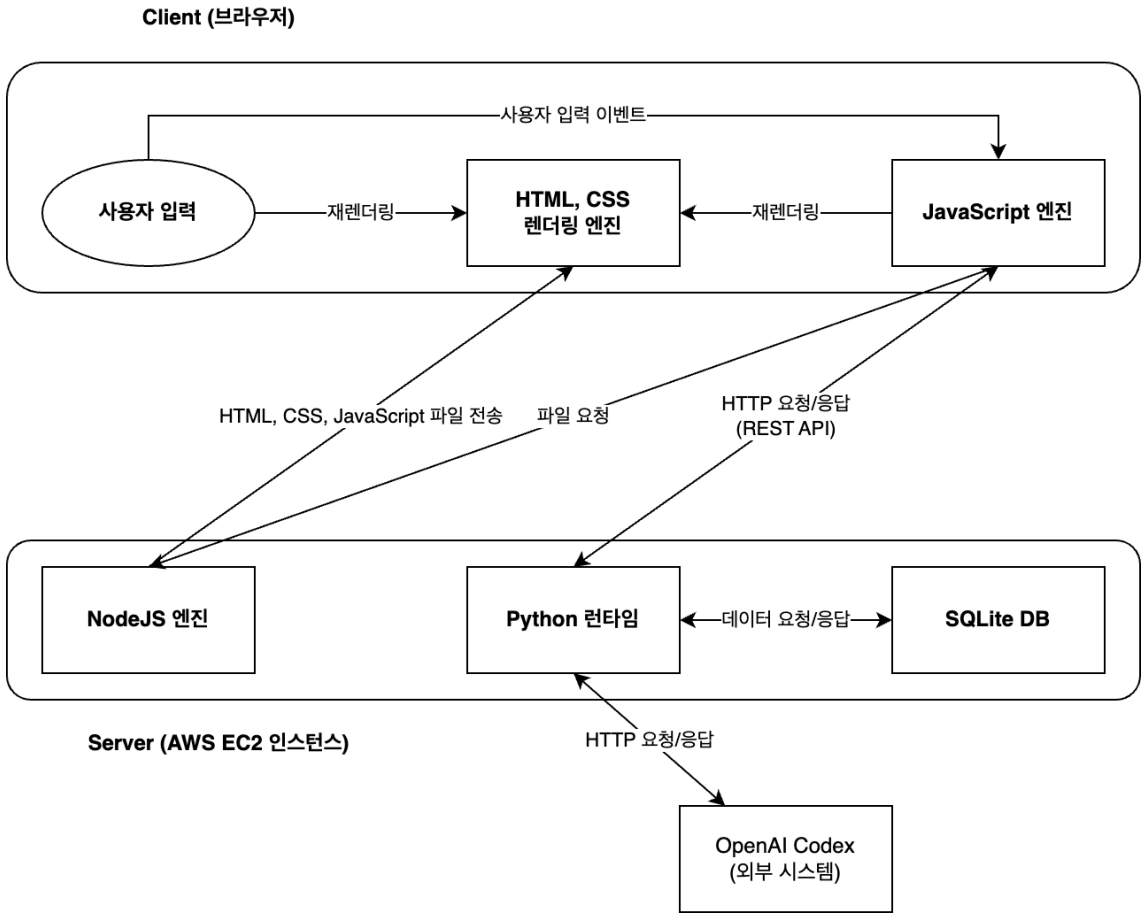

Figure 1 Overall System Architecture

2.2.1. Context Diagram

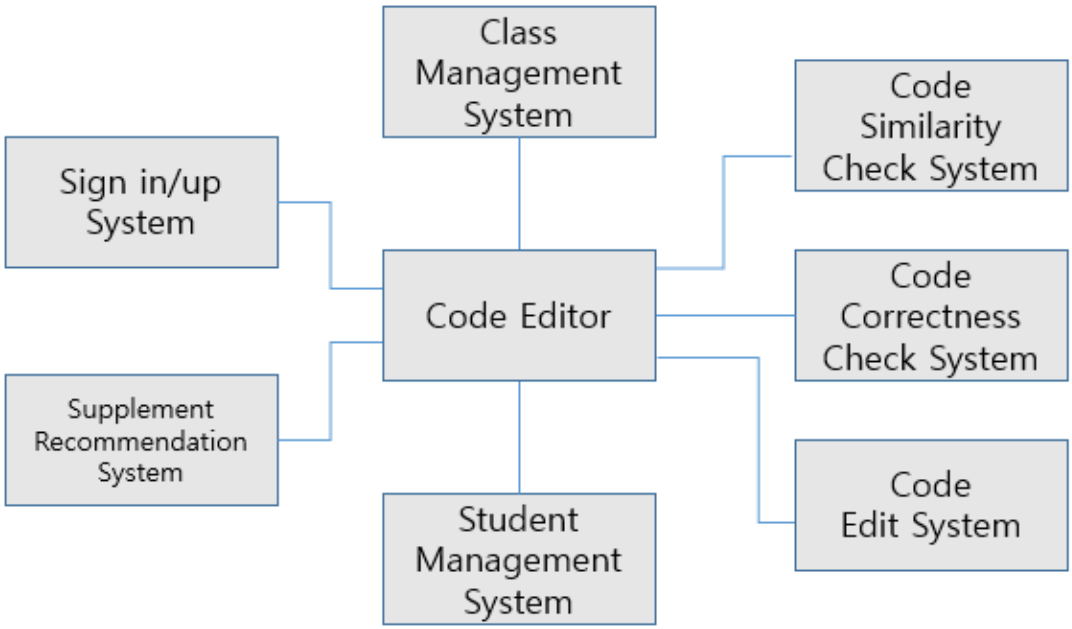

Figure 2 Overall Context Diagram

2.2.2. Sequence Diagram

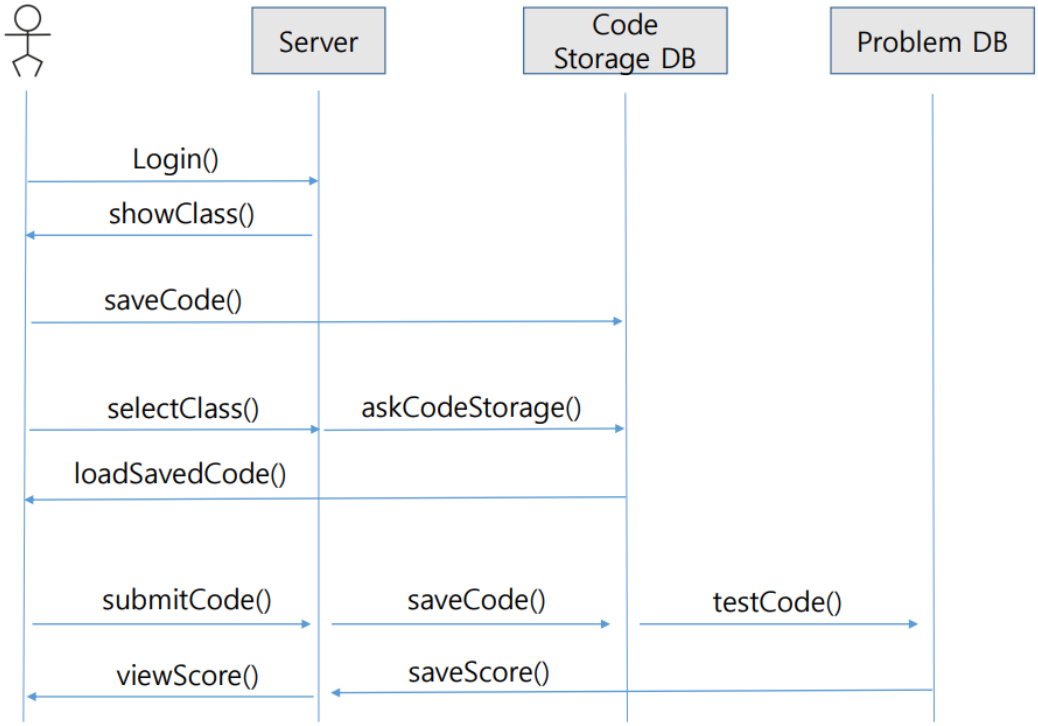

Figure 3 Overall Sequence Diagram

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 계정 정보를 관리하는 객체이다. 개인 정보에는 사용자의 이름과 학번, 그리고 아이디와 비밀번호가 저장된다. 계정 정보에는 해당 사용자가 수강하는 강의의 목록과, 강의 별 주어진 과제 그리고 제출한 과제의 기록이 저장된다.

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.
- registered_lecture: list<lecture _card>: 사용자가 수강한 강의 리스트이다.
- registered_assignment: list<assignment _card>: 사용자가 수행해야 하는 과제 리스트이다.
- conducted_assignment: list<assignment _card>: 사용자가 제출한 과제의 리스트이다.

3.2.1.2. Methods

이 클래스가 가지는 메서드는 다음과 같다.

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

3.2.1.3. Class Diagram

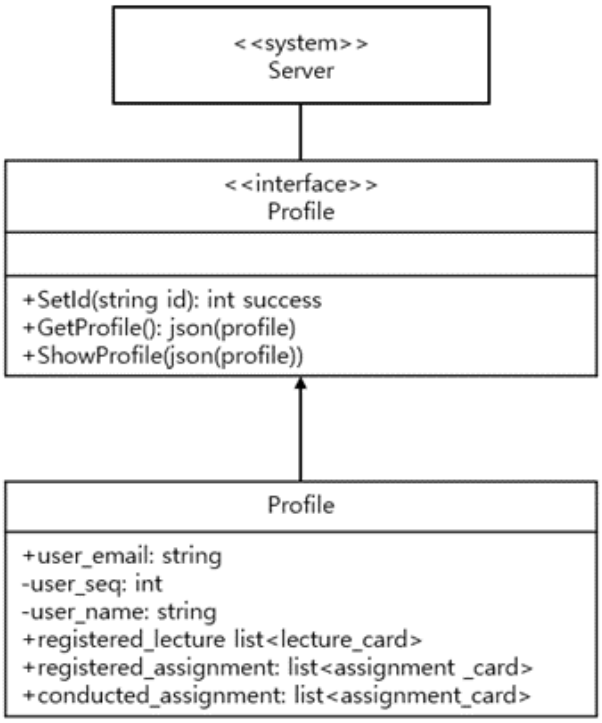

Figure 5 Class Diagram – Profile

3.2.1.4. Sequence Diagram

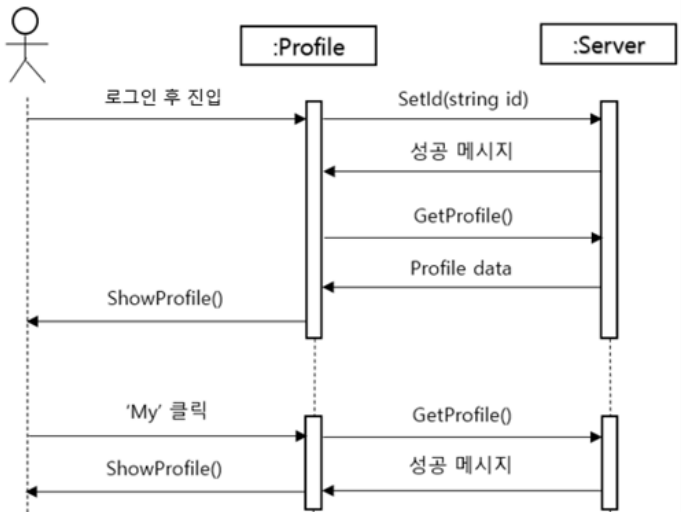

Figure 6 Sequence Diagram - Profile

3.2.2. Login, Logout

이 클래스는 로그인과 로그아웃을 관리하는 함수이다.

3.2.2.1. Attributes

이 클래스가 가지는 속성은 다음과 같다.

- user_id: 사용자 계정에 있는 사용자 학생 번호(학번)이다.
- user_password: 사용자 계정에 있는 사용자 비밀번호 정보이다.
- session_id: 로그인 시 결과값으로 받은 토큰(csrftoken)으로 사용자의 세션을 관리한다.

3.2.2.2. Methods

이 클래스가 가지는 메서드는 다음과 같다.

- Login(): 로그인 함수이다.
- Logout(): 로그아웃 함수이다.

3.2.2.3. Class Diagram

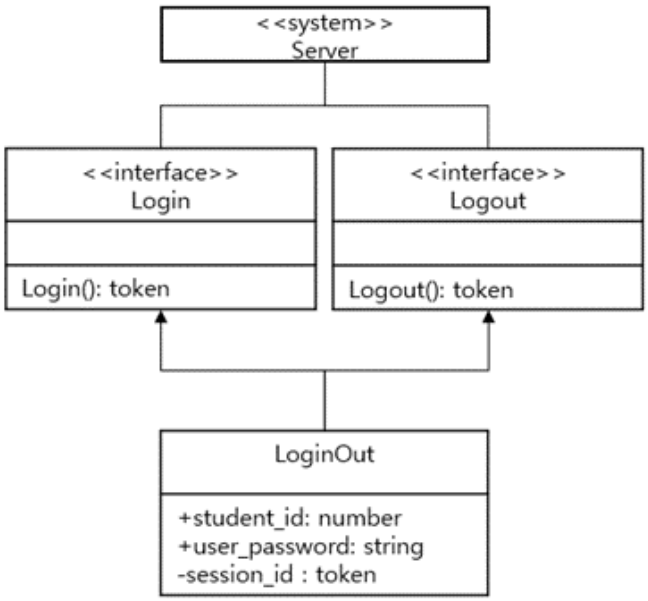

Figure 7 Class Diagram – Login and Logout

3.2.2.4. Sequence Diagram

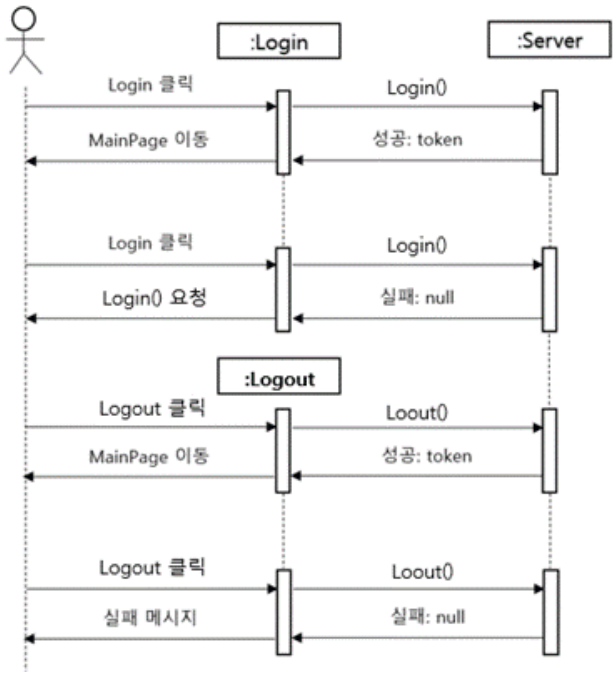

Figure 8 Sequence Diagram - Login and Logout

3.2.3. Class

Class 클래스는 사용자가 수강 중인 강의에 대한 정보를 관리하는 객체이다. 강의 정보에는 강의 이름, 과제의 마감 기한, 강의에 해당하는 문제, 제출 가능 횟수, 저장 가능 횟수 정보가 있다.

3.2.3.1. Attributes

이 클래스가 가지는 속성은 다음과 같다.

- class_id: 강의의 고유한 id값이다.
- class_name: 강의의 이름이다.
- class_deadline: 과제의 마감일이다.
- submission_capacity: 과제를 제출할 수 있는 횟수이다.
- storage_capacity: 코드를 중간 저장할 수 있는 횟수이다.
- problems: list<problem>: 강의에 포함된 문제 리스트이다.

3.2.3.2. Methods

이 클래스가 가지는 메서드는 다음과 같다.

- GetClass(): 강의 정보를 가져오는 함수이다.
- ShowClass(): 가져온 강의 정보를 보여주는 함수이다.

3.2.3.3. Class Diagram

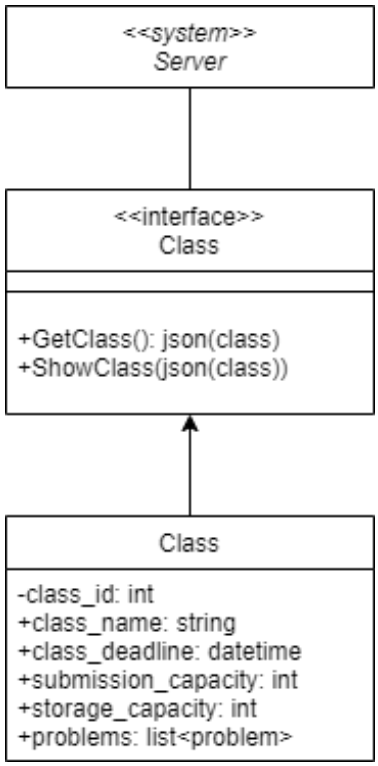

Figure 9 Class diagram – Class

3.2.3.4. Sequence Diagram

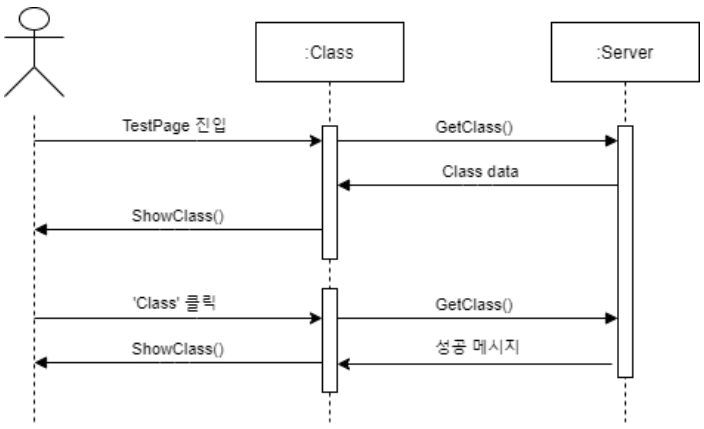

Figure 10 Sequence diagram – Class

3.2.4. Problem

이 클래스는 사용자가 풀어야 할 문제에 대한 정보를 관리하는 객체이다. 문제 정보에는 문제 설명, 참조사항, 테스트 케이스, 스켈레톤 코드, 정답 코드, 관련 자료 정보가 있다.

3.2.4.1. Attributes

이 클래스가 가지는 속성은 다음과 같다.

- problem_id: 문제의 고유한 id값이다.
- class_id: 문제가 속한 강의의 id값이다.
- explanation: 문제에 대한 설명이다.
- reference: 문제의 참조/제약 사항이다.
- testcases: list<testcase>: 문제에 해당하는 테스트 케이스 리스트이다.
- skeleton_code: 문제의 스켈레톤 코드이다.
- answer_code: 문제의 정답 코드이다.

3.2.4.2. Methods

이 클래스가 가지는 메서드는 다음과 같다.

- GetProblem(): 문제 정보를 가져오는 함수이다.
- ShowProblem(): 가져온 문제 정보를 보여주는 함수이다. 문제 설명, 참조/제약사항, 테스트 케이스 리스트를 보여준다.
- RunTestcase(): 사용자가 선택한 테스트 케이스를 검증하는 함수이다.
- ShowTestcaseResult(): 테스트 케이스 검증 결과를 보여주는 함수이다.

3.2.4.3. Class Diagram

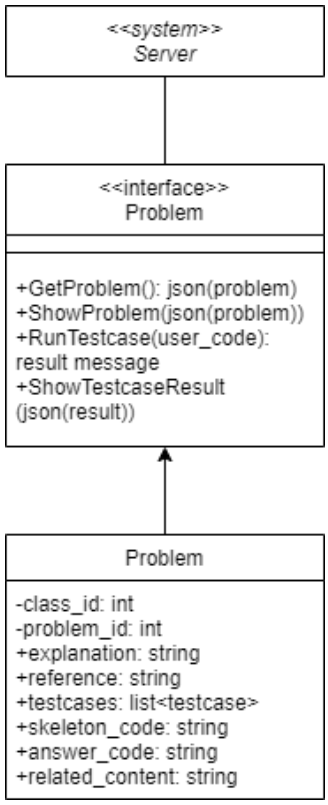

Figure 11 Class Diagram - Problem

3.2.4.4. Sequence Diagram

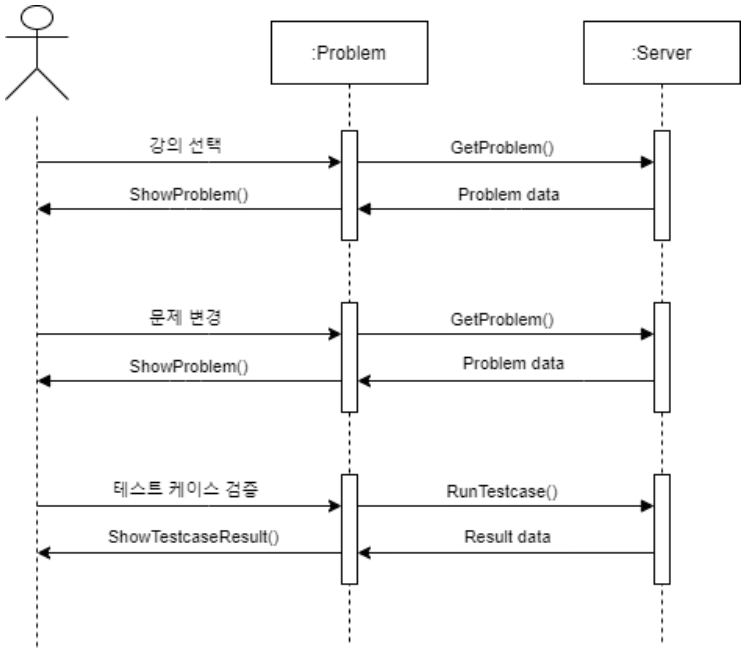

Figure 12 Sequence Diagram - Problem

3.2.5. Code

사용자가 problem에 대한 실습 코드를 작성하는 컴포넌트이다. 실습 코드 작성을 돕기 위해 불러오기, 저장, 초기화 등을 제공한다.

3.2.5.1. Attributes

Code 클래스가 가지는 attributes는 다음과 같다.

- problem_id: 문제의 고유한 id값이다.
- class_id: 문제가 속한 강의의 id값이다.
- skeleton_code: 문제의 스켈레톤 코드이다.

3.2.5.2. Methods

이 클래스가 가지는 메서드는 다음과 같다.

- Storages_call(): 임시 저장한 코드 중 선택한 내용을 불러오는 함수이다.
- Storages_save(): 작성한 코드를 임시 저장하는 함수이다.

3.2.5.3. Class Diagram

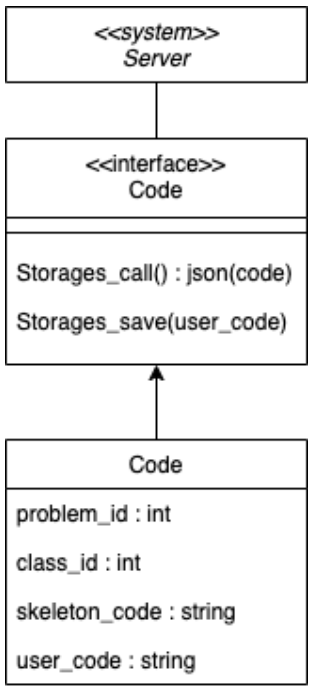

Figure 13 Class Diagram - Code

3.2.5.4. Sequence Diagram

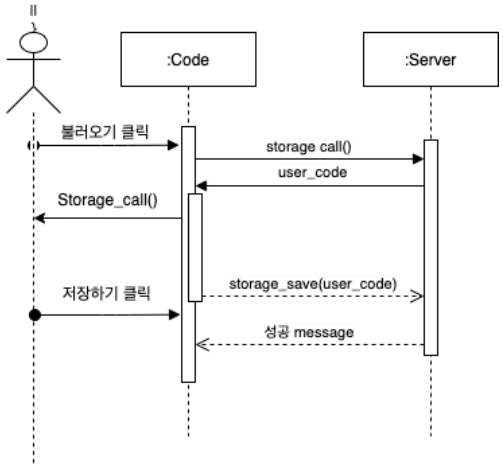

Figure 14 Sequence Diagram - Code

3.2.6. Terminal

사용자가 입력한 실습 코드를 실행하고 그 코드를 정답과 비교하는 과정을 담당하는 컴포넌트이다.

3.2.6.1. Attributes

Terminal 클래스가 가지는 attributes는 다음과 같다.

- user_code: 사용자가 작성해 실행할 실습 코드이다.
- input: 실습 코드를 실행할 때 입력하는 input 값이다.
- result: 실습 코드를 실행한 결과값이다.
- is_passed: 테스트 케이스로 실습 코드를 검증한 결과이다.
- error: 실습 코드 실행 시 에러가 발생할 경우 그 내용을 반환한다.

3.2.6.2. Methods

이 클래스가 가지는 메서드는 다음과 같다.

- RunCode(): 실습 코드를 실행하여 그 결과를 반환하는 함수이다.
- TestCode(): 테스트 케이스로 작성한 실습 코드를 검증하는 함수이다.
- SubmitCode(): 테스트 케이스로 실습 코드를 채점하고 표절, 기능, 효율, 가독성을 분석하는 함수이다.

3.2.6.3. Class Diagram

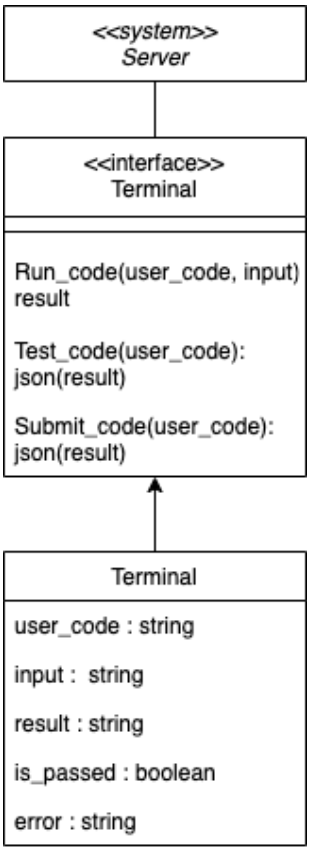

Figure 15 Class Diagram – Terminal

3.2.6.4. Sequence Diagram

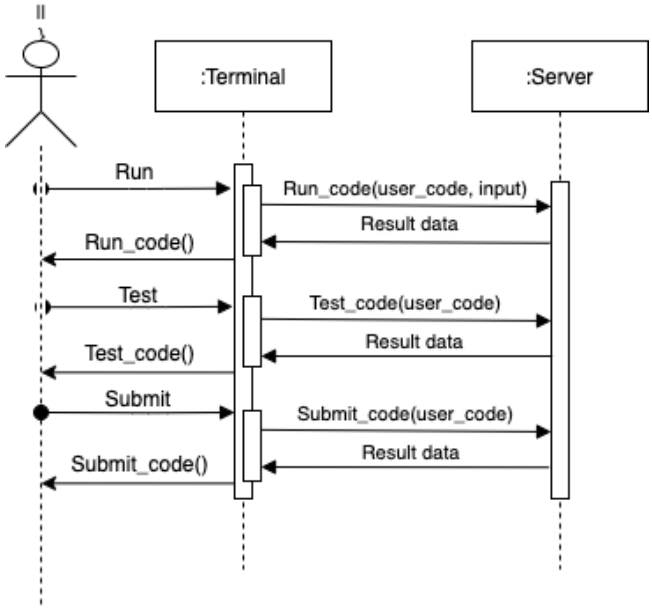

Figure 16 Sequence Diagram - Terminal

4. System Architecture – Backend

4.1. Purpose

이 장에서는 DB 와 서버의 구조를 설명한다.

4.2. Overall Architecture

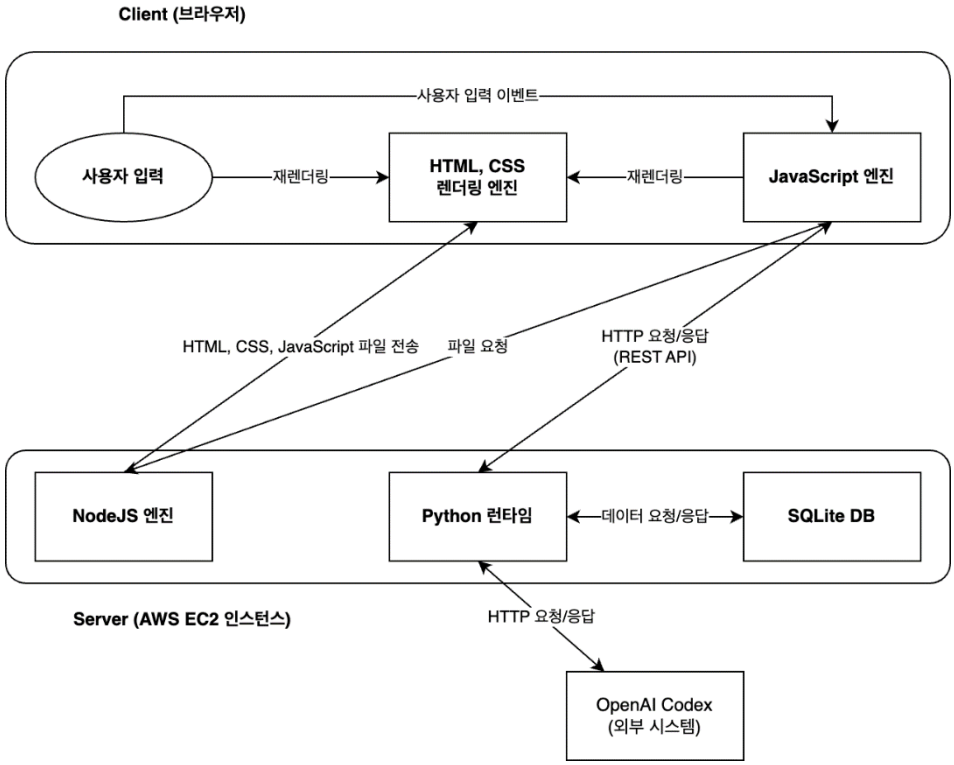

Figure 17 overall architecture

전체적인 아키텍처는 크게 클라이언트와 서버로 나눌 수 있다. 클라이언트는 웹 브라우저로, 유저의 상호작용에 따라 프론트엔드 서버인 NodeJS 엔진과 정적 파일을 주고받으며 웹페이지를 렌더링한다. 이때 JavaScript 라이브러리인 React의 기능을 활용한다. 비즈니스 데이터가 필요한 경우에는 axios 라이브러리를 활용하여 서버 측으로 API를 호출한다. 서버는 클라이언트로부터 받은 데이터를 해석하고 필요한 데이터를 DB 또는 OpenAI Codex와 같은 외부 시스템에 요청하여 획득하여 가공한 뒤 다시 클라이언트로 반환한다. DB와 상호작용하는 로직은 주로 유저 정보, 강의 정보, 문제 정보, 저장된 코드 등을 불러오거나 저장하는 작업에서 사용한다. 그 외의 핵심 작업으로는 파이썬 스크립트 실행, 검증 및 분석 작업이 있는데 이는 요청 또는 DB에 존재하는 파이썬 스크립트 데이터를 별도의 프로세스로 실행하도록 설계되었으며 출력 결과는 단위 테스트 라이브러리를 활용해 검증된다. 분석에는 OpenAI Codex와 같은 외부 시스템과 다양한 파이썬 분석 툴이 활용되고 해당 내용은 DB에 다시 저장된다.

4.3. Subcomponents

4.3.1. User system

User와 관련된 작업들이 공통적으로 묶여 있는 System이다.

4.3.1.1. User System Class Diagram

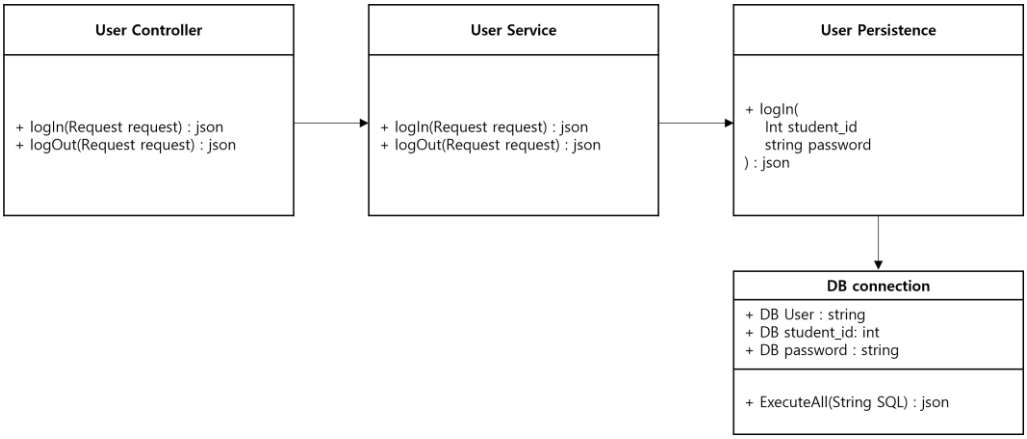

Figure 18 Class Diagram – User System

4.3.1.1.1. User Controller Class

WAS로부터 받은 Request에서 Parameter를 받아서 Service Layer를 호출하는 역할을 가지고 있다. 해당 클래스는 2가지의 메소드가 존재한다. 첫번째는 로그인과 관련된 logIn이며, 두번째는 로그아웃과 관련된 logOut이다.

4.3.1.1.2. User Service Class

Controller로부터 받은 데이터에서 필요한 정보를 추출하고 Persistence를 호출하는 역할을 한다. 이후 Persistence Layer를 호출해서 DB에 필요한 정보를 저장하고 불러오고 검색하는 역할을 한다. 또한 Persistence Layer를 호출하고 나서는 해당 Layer에서 반환된 값을 Controller에 넘겨주는 역할을 한다. 해당 클래스에는 2개의 메소드가 존재하는데, 첫번째는 로그인과 관련된 logIn이며, 두번째는 로그아웃과 관련된 logOut이다. 이때 첫번째 메소드에서는 Set-Cookie 헤더로 오는 session id와 csrftoken을 저장하고 인증이 필요한 모든 요청의 쿠키와 X-CSRFToken 헤더에 csrftoken 값을 넣어서 전송한다. 두번째 메소드에서는 쿠키에 있는 session id를 이용해서 데이터베이스에 존재하는 세션을 제거하고 해당 처리 이후 보유하고 있는 쿠키가 무효화되도록 한다.

4.3.1.1.3. User Persistence Class

Service Layer에서 받은 parameter를 바탕으로 SQL 문을 만들어서 DB Connection에서 넘겨준다. 또한, DB Connection에서 DB 로부터 받은 값을 Service Layer에 넘겨주는 역할을 한다. 해당 클래스에는 logIn 메소드가 존재한다.

4.3.1.1.4. DB Connection

DB 랑 연결을 담당하는 부분으로 Persistence Layer 로부터 받은 SQL을 DB에 넘겨 결과값을 받는 역할을 한다. 이후 결과를 받으면 Persistence Layer 로 넘겨준다.

4.3.1.2. User Sequence Diagram

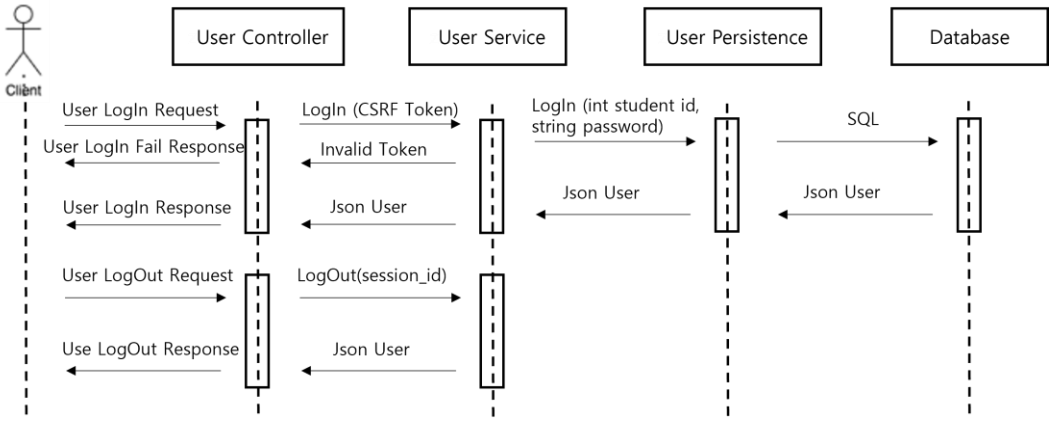

Figure 19 Sequence Diagram – User System

4.3.2. Lecture System

Lecture(class)와 관련된 작업들이 공통적으로 묶여 있는 System이다.

4.3.2.1. Lecture System Class Diagram

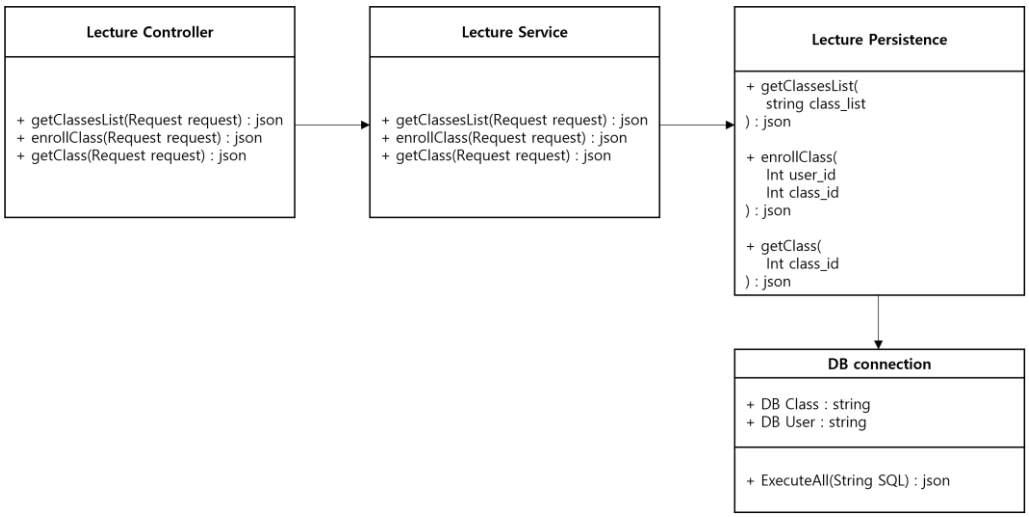

Figure 20 Class Diagram - Lecture System

4.3.2.1.1. Lecture Controller Class

WAS로부터 받은 Request에서 Parameter를 받아서 Service Layer를 호출하는 역할을 가지고 있다. 또한, 해당 Service Layer의 결과를 json으로 만들어 response의 body에 들어갈 내용을 만든다. Lecture Controller Class에는 3가지의 메소드가 존재한다. 첫번째는 강의 목록을 불러오는 getClassesList이다. 두번째는 강의를 등록하는 enrollClass이다. 마지막은 강의에 대한 정보를 불러오는 getClass이다.

4.3.2.1.2. Lecture Service Class

Controller로부터 받은 데이터에서 필요한 정보를 추출하고 Persistence를 호출하는 역할을 한다. 이후 Persistence Layer를 호출해서 DB에 필요한 정보를 저장하고 불러오고 검색하는 역할을 한다. 또한 Persistence Layer를 호출하고 나서는 해당 Layer에서 반환된 값을 Controller에 넘겨주는 역할을 한다. Lecture Service Class의 메소드는 총 3가지가 존재한다. 첫번째는 강의 목록을 불러오는 getClassesList이다. 두번째는 강의를 등록하는 enrollClass이다. 마지막은 강의에 대한 정보를 불러오는 getClass이다.

4.3.2.1.3. Lecture Persistence Class

Service Layer에서 받은 parameter를 바탕으로 SQL 문을 만들어서 DB Connection에서 넘겨준다. 또한 DB Connection에서 DB 로부터 받은 값을 Service Layer에 넘겨주는 역할을 한다. 해당 클래스에는 강의 목록을 불러오는 getClassesList와 강의를 등록 처리하는 enrollClass, 강의에 대한 정보를 불러오는 getClass가 존재한다.

4.3.2.2. Lecture System Sequence Diagram

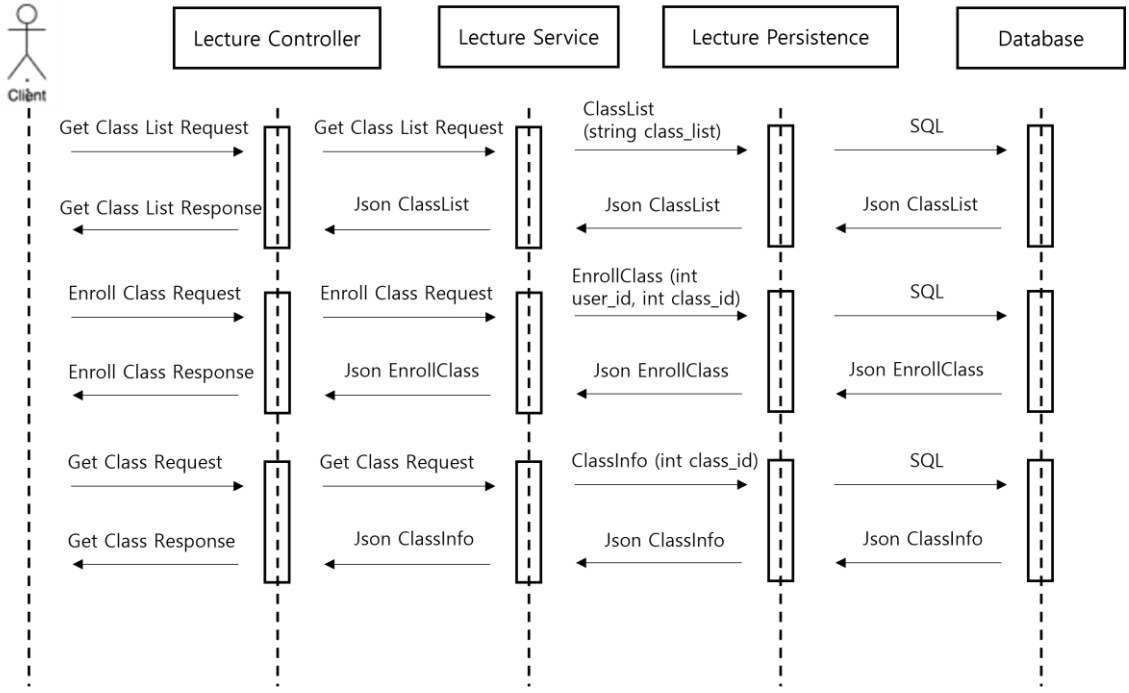

Figure 21 Sequence Diagram – Lecture System

4.3.3. Problem & Code

문제 선택 시 해당 문제에 대한 정보와 더불어 사용자가 해당 문제에 대하여 작성한 코드와 관련자료를 불러오는 System이다.

4.3.3.1. Problem & Code System Class Diagram

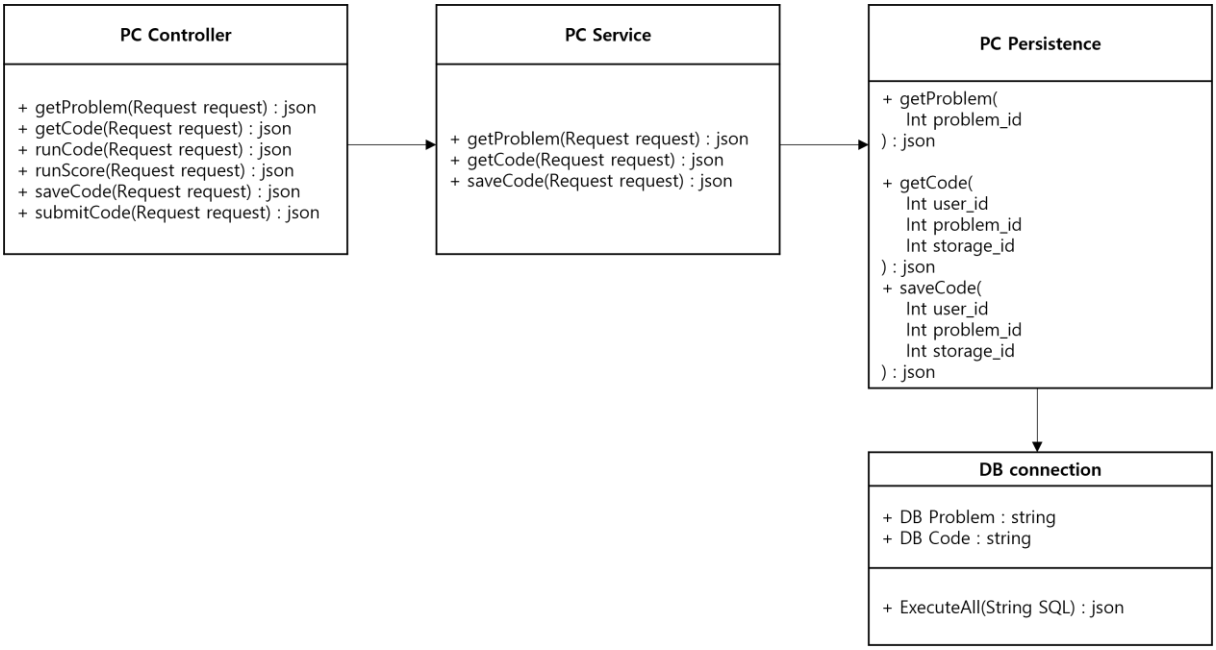

Figure 22 Class Diagram – Problem and Code System

4.3.3.1.1. Program & Code controller class

WAS로부터 받은 Request에서 Parameter를 받아서 Service Layer를 호출하는 역할을 가지고 있다. 해당 클래스는 6가지의 메소드가 존재한다. 첫번째는 문제를 불러오는 getProblem이 존재한다. 두번째는 사전에 저장된 코드가 있다면 이를 불러오는 getCode이다. 세번째는 사용자가 작성한 코드를 실행하는 runCode가 존재한다. 네번째는 사용자가 작성한 코드에 대해 유사도 검사 및 채점을 해주는 runScore가 존재한다. 다섯번째로는 사용자가 작성한 코드를 저장하는 saveCode가 존재한다. 마지막으로는 사용자가 작성한 코드를 최종 과제에 제출하는 submitCode가 존재한다.

4.3.3.1.2. Problem & Code Service Class

Controller로부터 받은 데이터에서 필요한 정보를 추출하고 Persistence를 호출하는 역할을 한다. 이후 Persistence Layer를 호출해서 DB에 필요한 정보를 저장하고 불러오고 검색하는 역할을 한다. 또한 Persistence Layer를 호출하고 나서는 해당 Layer에서 반환된 값을 Controller에 넘겨주는 역할을 한다. 해당 클래스에는 3개의 메소드가 존재하는데, 첫번째는 문제를 불러오는 getProblem이 존재한다. 두번째는 사전에 저장된 코드가 있다면 이를 불러오는 getCode이다. 세번째는 사용자가 작성한 코드를 저장하는 saveCode가 존재한다.

4.3.3.1.3. Problem & Code Persistence Class

Service Layer에서 받은 parameter를 바탕으로 SQL 문을 만들어서 DB Connection에서 넘겨준다. 또한 DB Connection에서 DB 로부터 받은 값을 Service Layer에 넘겨주는 역할을 한다. 해당 클래스에는 3 가지 메소드가 존재하는데 문제를 불러오는 getProblem 와 사전에 저장된 코드가 있다면 이를 불러오는 getCode, 사용자가 작성한 코드를 저장하는 saveCode가 존재한다.

4.3.3.2. Problem & Code System Sequence Diagram

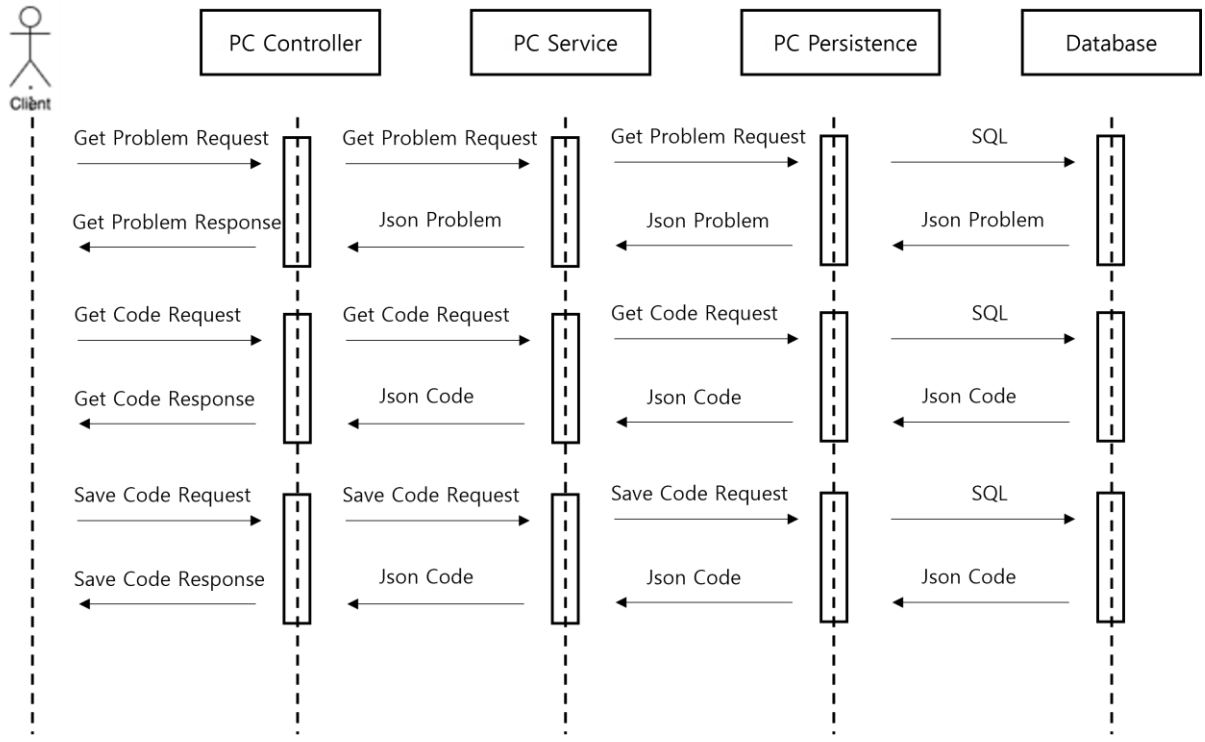

Figure 23 Sequence Diagram - Program and Code System

5. Protocol Design

5.1. Objectives

이 장에서는 하위 시스템 간의 상호 작용에 사용되는 프로토콜의 구조를 표현하고, 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 사용되는 데이터 형식으로, AJAX 시스템에서 XML을 대체하는 역할을 하는 등 다양한 응용 프로그램을 가진다.

5.3. CSRF Token

CSRF Token은 서버에 들어온 요청이 실제 서버에서 허용한 요청이 맞는지 확인하기 위해 사용되는 토큰이다. 서버는 뷰 페이지를 발행할 때 랜덤으로 생성된 Token을 함께 제공하며, 이를 사용자 세션에 저장한다. 추후 사용자가 서버로 작업을 요청하면, 페이지에 숨겨진 token 값이 서버로 전송되는데 서버는 해당 Token 값이 세션에 저장된 값과 일치하는지 확인하여 요청이 위조된 게 아니라는 것을 확인한다. 일치 여부가 확인된 Token은 즉시 폐기하고, 새로운 뷰 페이지를 발행할 때마다 새롭게 생성한다.

5.4. Authentication

5.4.1. Login

- Request

Table 1 Table of login request

Attribute	Detail
Method	POST
URI	http: /login
Parameter	X

- Response

Table 2 Table of login response

Attribute	Detail	
Server Behavior	- 사이트에서 입력된 학번, 비밀번호를 검증하여 이상이 없다면 비밀번호를 제외한 유저 정보를 반환 - Set-Cookie 헤더로 오는 session id와 csrftoken을 저장하고 인증이 필요한 모든 요청의 쿠키와 X-CSRFToken 헤더에 csrftoken 값을 넣어서 전송	
Success Code	HTTP status code = 200 (OK)	
Failure Code	HTTP status code = 400 (Bad Request)	
Success Response Body	Message	Success message
Failure Response Body	Message	Fail message

5.4.2. Logout

- Request

Table 3 Table of logout request

Attribute	Detail
Method	GET
URI	http: /logout
Parameter	X

- Response

Table 4 Table of logout response

Attribute	Detail	
Server Behavior	- 쿠키에 있는 session id를 이용해서 데이터베이스에 존재하는 세션을 제거 - 해당 처리 이후, 보유하고 있는 쿠키 무효화	
Success Code	HTTP status code = 200 (OK)	
Failure Code	HTTP status code = 400 (Bad Request)	
Success Response Body	Message	Success message
Failure Response Body	Message	Fail message

5.5. Classes

5.5.1. Classes List

– Request

Table 5 Table of classes list request

Attribute	Detail
Method	GET
URI	http: /classes
Parameter	X

- Response

Table 6 Table of classes list response

Attribute	Detail
Server Behavior	- 데이터베이스에 있는 모든 강의 고유번호와 이름 목록 반환 - 단, 마감기한이 지난 강의는 제외
Success Code	HTTP status code = 200 (OK)
Failure Code	HTTP status code = 400 (Bad Request)

Success Response Body	Message	Success message
Failure Response Body	Message	Fail message

5.5.2. Class Enrollment

- Request

Table 7 Table of class enrollment request

Attribute	Detail
Method	GET
URI	http: /classes/:id/enroll
Parameter	class_id

- Response

Table 8 Table of class enrollment response

Attribute	Detail	
Server Behavior	- 데이터베이스에서 class_id로 강의 검색 - 데이터베이스에 미등록 된 class_id는 failure response 보내기 - 마감기한을 확인하여 강의 수강 가능 여부를 검사 - 강의 수강이 가능하면 수강 처리 진행 (기록이 존재하지 않으면 새로 생성하고, 기록이 존재하면 기존 데이터 로딩)	
Success Code	HTTP status code = 200 (OK)	
Failure Code	HTTP status code = 400 (Bad Request)	
Success Response Body	Message	Success message
Failure Response Body	Message	Fail message

5.5.3. Class Information

- Request

Table 9 Table of class information request

Attribute	Detail
Method	GET
URI	http: /classes/:id
Parameter	class_id

- Response

Table 10 Table of class information response

Attribute	Detail	
Server Behavior	- 데이터베이스에서 class_id로 강의 검색 - 수강 기록 유무 검사하여 강의 수강 여부를 검사 - 수강하지 않는 강의일 경우, failure response 보내기 - 수강하고 있는 강의일 경우, class 테이블의 모든 칼럼 및 연결된 모든 문제의 고유번호와 이름 목록 반환	
Success Code	HTTP status code = 200 (OK)	
Failure Code	HTTP status code = 400 (Bad Request)	
Success Response Body	Message	Success message
Failure Response Body	Message	Fail message

5.6. Problem

5.6.1. Problem Information

- Request

Table 11 Table of problem information request

Attribute	Detail
Method	GET
URI	http: /problems/:id
Parameter	problem_id

- Response

Table 12 Table of problem information response

Attribute	Detail	
Server Behavior	- 데이터베이스에서 problem_id로 문제 검색 - 문제의 히든 테스트케이스, 관련 자료, 정답 코드 정보는 필터링 - 필터링한 내용에 저장된 코드와 제출 코드 고유번호 목록을 포함하여 반환	
Success Code	HTTP status code = 200 (OK)	
Failure Code	HTTP status code = 400 (Bad Request)	
Success Response Body	Message	Success message
Failure Response Body	Message	Fail message

5.7. Code

5.7.1. Code Execution

- Request

Table 13 Table of code execution request

Attribute	Detail
Method	POST
URI	http: /execute
Parameter	X

- Response

Table 14 Table of code execution response

Attribute	Detail	
Server Behavior	- 제출 코드를 실행한 뒤 출력 결과 반환 - 기본적으로 최대 10초의 time out 부여 (코드 실행에 10초 이상이 걸리면 timeout error 반환)	
Success Code	HTTP status code = 200 (OK)	
Failure Code	HTTP status code = 400 (Bad Request)	
Success Response Body	Message	Success message
Failure Response Body	Message	Fail message

5.7.2. Code Grading

- Request

Table 15 Table of code grading request

Attribute	Detail
Method	POST
URI	http: /problems/:problem_id/grade/:testcase_num
Parameter	problem_id, testcase_num

- Response

Table 16 Table of code grading response

Attribute	Detail
Server Behavior	- 코드를 실행하여 출력 결과 확인 - 출력 결과를 problem_id에 맞는 테스트케이스와 검증 - 출력 결과와 검증 결과 반환 - 단, 공개/비공개 테스트케이스 구분
Success Code	HTTP status code = 200 (OK)
Failure Code	HTTP status code = 400 (Bad Request)

Success Response Body	Message	Success message
Failure Response Body	Message	Fail message

5.7.3. Code Submission

- Request

Table 17 Table of code submission request

Attribute	Detail
Method	POST
URI	http: /submissions?problem_id=:id/
Parameter	problem_id

- Response

Table 18 Table of code submission response

Attribute	Detail	
Server Behavior	- 데이터베이스에서 problem_id로 문제 검색 - 찾은 문제에 대한 제출 가능 여부 검사 (최대 제출 횟수 확인) - 제출이 가능하면, 제출 코드에 대한 기록 생성 - 제출 코드의 검증 및 분석 처리 - 생성된 제출 코드의 고유번호 반환	
Success Code	HTTP status code = 200 (OK)	
Failure Code	HTTP status code = 400 (Bad Request)	
Success Response Body	Message	Success message
Failure Response Body	Message	Fail message

5.7.4. Code Submission Results

- Request

Table 19 Table of code submission results request

Attribute	Detail
Method	GET
URI	http: /submissions/:id
Parameter	submission_id

- Response

Table 20 Table of code submission results response

Attribute	Detail	
Server Behavior	- 데이터베이스에서 submission_id로 제출 코드 검색 - 제출 코드의 상태 검사 (분석 완료 여부 확인) - 제출 코드에 해당하는 강의 마감 여부 검사 - 마감기한이 지난 경우, 제출 결과에 대한 정보 반환	
Success Code	HTTP status code = 200 (OK)	
Failure Code	HTTP status code = 400 (Bad Request)	
Success Response Body	Message	Success message
Failure Response Body	Message	Fail message

5.7.5. Code Store

- Request

Table 21 Table of code store request

Attribute	Detail
Method	POST
URI	http: /storages/:order
Parameter	storage_order

- Response

Table 22 Table of code store response

Attribute	Detail	
Server Behavior	- 데이터베이스 내 storage_order에 해당하는 저장소에 사용자의 코드를 저장 (덮어쓰기)	
Success Code	HTTP status code = 200 (OK)	
Failure Code	HTTP status code = 400 (Bad Request)	
Success Response Body	Message	Success message
Failure Response Body	Message	Fail message

5.7.6. Code Load

- Request

Table 23 Table of code load request

Attribute	Detail
Method	GET
URI	http: /storages/:order
Parameter	storage_order

- Response

Table 24 Table of code load response

Attribute	Detail	
Server Behavior	- 데이터베이스에서 storage_order로 코드 검색 - 찾은 코드를 반환	
Success Code	HTTP status code = 200 (OK)	
Failure Code	HTTP status code = 400 (Bad Request)	
Success Response Body	Message	Success message
Failure Response Body	Message	Fail message

6. Database Design

6.1. Objectives

이 장에서는 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-diagram은 모델의 테이블 간의 관계를 시각적으로 표현한다. 테이블이 다른 테이블과 여러 관계가 있는 경우 이를 나타내기 위해 닻 모양이 사용된다. 테이블이 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.

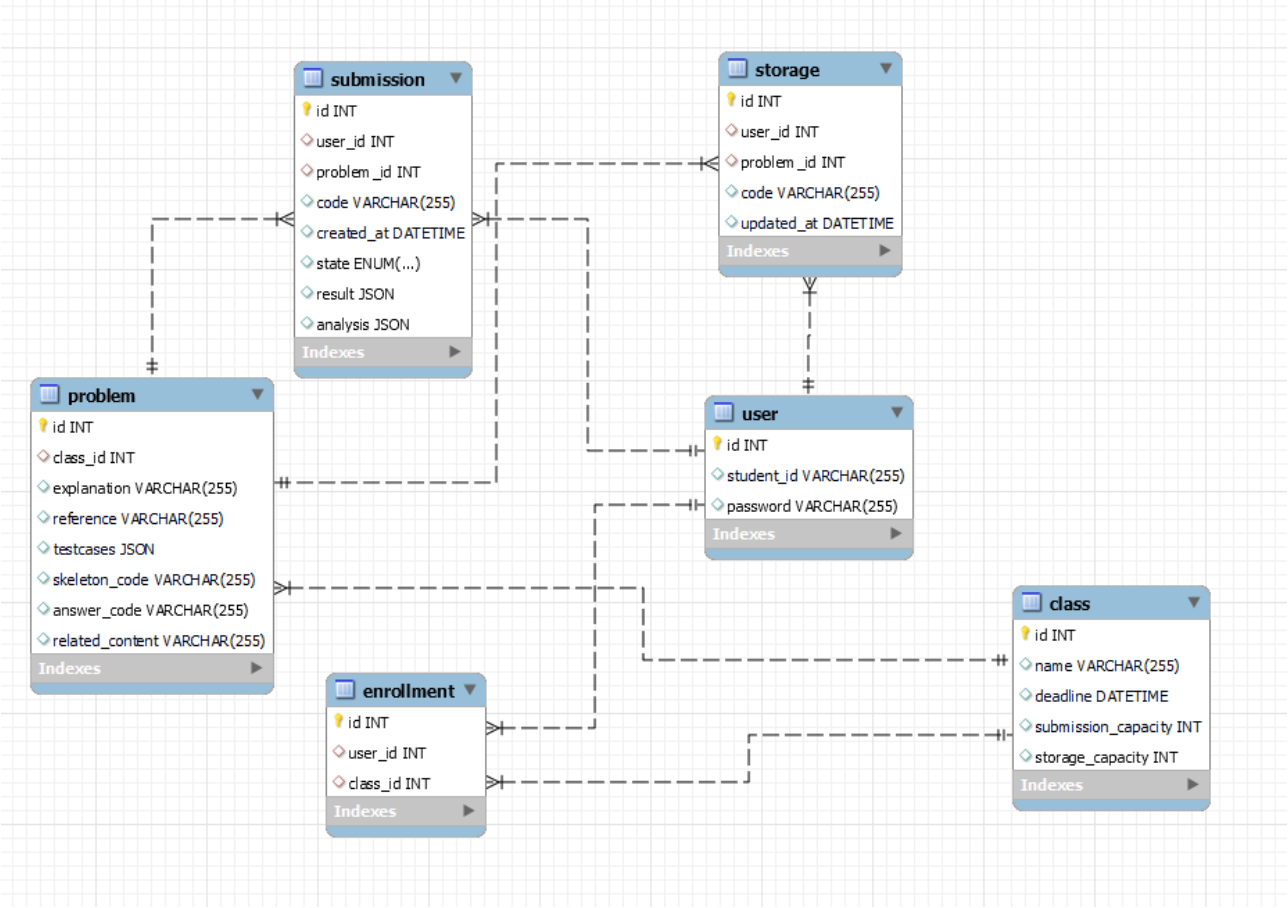

Figure 24 EER diagram

6.2.1. Table

6.2.1.1. User

User 테이블은 코드 에디터를 사용하는 학생 사용자를 나타낸다. Id, student_id, passeword으로 구성되어 있고, id가 primary key이다. id 속성을 통해 enrollment, submission, storage 테이블과 일대다 관계를 맺는다.

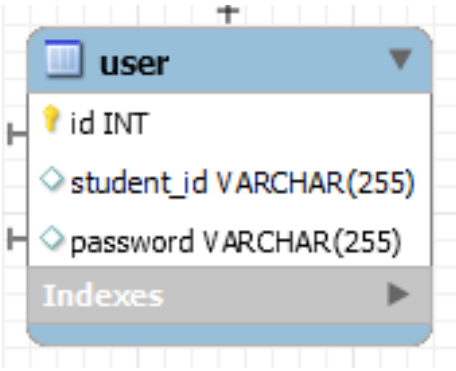

Figure 25 EER diagram - User

6.2.1.2. Class

class 테이블은 등록되어 있는 강좌의 목록 및 특성을 나타낸다. id, name, deadline, submission_capacity, storage_capacity로 구성되어 있고 id가 primary key이다. Id 속성을 통해 problem, enrollment 테이블과 일대다 관계를 맺는다.

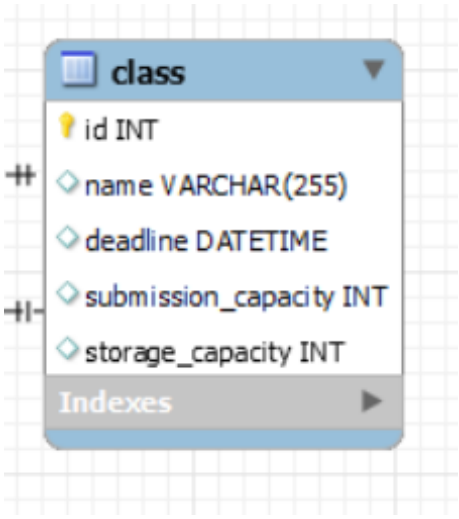

Figure 26 EER diagram - Class

6.2.1.3. Enrollment

enrollment 테이블은 등록되어 있는 강좌와 사용자를 연결시켜주는 역할을 한다. Id, user_id, class_id 로 구성되어 있고 id가 primary key다. User_id 속성을 통해 user 테이블과 관계를 다대일 관계를 맺는다. Class_id 속성을 통해 class 테이블과 관계를 다대일 관계를 맺는다.

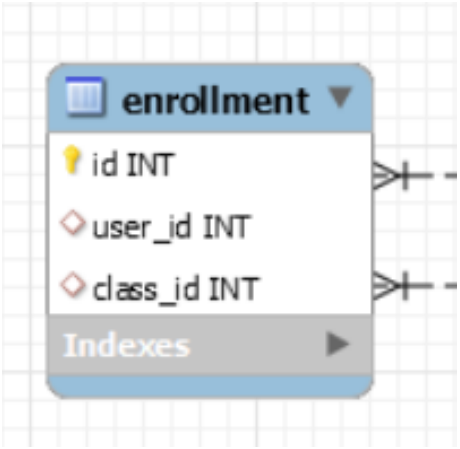

Figure 27 EER diagram - Enrollment

6.2.1.4. Problem

problem 테이블은 강의에 등록되어 있는 문제를 나타낸다. Id, class_id, explanation, reference, testcase, skeleton_code, answer_code, related_content로 구성되어 있고 id 가 primary key이다. Id 속성을 통해 storage, submission 테이블과 일대다 관계를 맺는다. Class_id 속성을 통해 class 테이블과 다대일 관계를 맺는다.

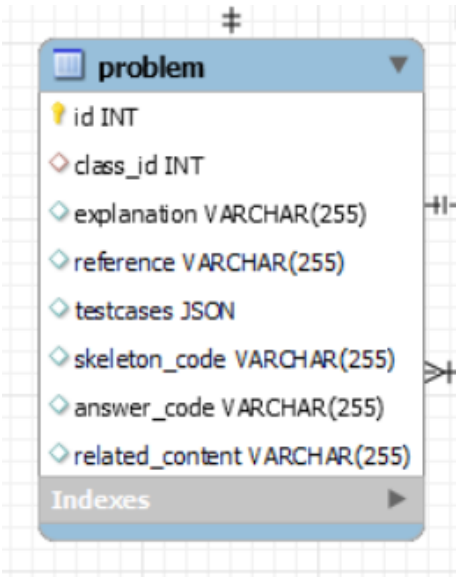

Figure 28 EER diagram - Problem

6.2.1.5. Submission

submission 테이블은 제출된 코드를 나타낸다. Id, user_id, problem_id, code, created_at, state, result, analysis로 구성되어 있고 id가 primary key이다. User_id 속성을 통해 user 테이블과 다대일 관계를 맺는다. Problem_id 속성을 통해 problem 테이블과 다대일 관계를 맺는다.

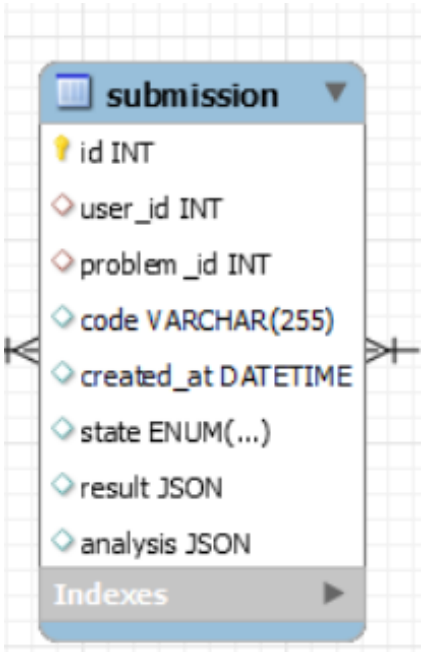

Figure 29 EER diagram - Submission

6.2.1.6. Storage

storage 테이블은 저장된 코드를 나타낸다. Id, user_id, problem_id, code, updated_at으로 구성되어 있고 id가 primary key이다. Problem_id 속성을 통해 problem 테이블과 다대일 관계를 맺는다. User_id 속성을 통해 user 테이블과 관계를 다대일 관계를 맺는다.

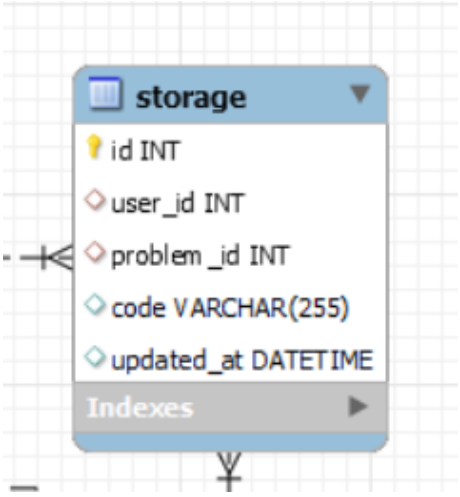

Figure 30 EER diagram - Storage

6.3. SQL DDL

6.3.1. User

```
CREATE TABLE `user` (
  `id` int PRIMARY KEY AUTO_INCREMENT,
  `student_id` varchar(255),
  `password` varchar(255)
);
```

Figure 31 SQL DDL - User

6.3.2. Class

```
CREATE TABLE `class` (
  `id` int PRIMARY KEY AUTO_INCREMENT,
  `name` varchar(255),
  `deadline` datetime,
  `submission_capacity` int,
  `storage_capacity` int
);
```

Figure 32 SQL DDL - Class

6.3.3. Enrollment

```
CREATE TABLE `enrollment` (
  `id` int PRIMARY KEY AUTO_INCREMENT,
  `user_id` int,
  `class_id` int,
  FOREIGN KEY (`user_id`) REFERENCES `user` (`id`),
  FOREIGN KEY (`class_id`) REFERENCES `class` (`id`)
);
```

Figure 33 SQL DDL – Enrollment

6.3.4. Problem

```
CREATE TABLE `problem` (
  `id` int PRIMARY KEY AUTO_INCREMENT,
  `class_id` int,
  `explanation` varchar(255),
  `reference` varchar(255),
  `testcases` json,
  `skeleton_code` varchar(255),
  `answer_code` varchar(255),
  `related_content` varchar(255),
  FOREIGN KEY (`class_id`) REFERENCES `class` (`id`)
);
```

Figure 34 SQL DDL – Problem

6.3.5. Storage

```
CREATE TABLE `storage` (
  `id` int PRIMARY KEY AUTO_INCREMENT,
  `user_id` int,
  `problem_id` int,
  `code` varchar(255),
  `updated_at` datetime,
  FOREIGN KEY (`user_id`) REFERENCES `user` (`id`),
  FOREIGN KEY (`problem_id`) REFERENCES `problem` (`id`)
);
```

Figure 35 SQL DDL – Storage

6.3.6. Submission

```
CREATE TABLE `submission` (
  `id` int PRIMARY KEY AUTO_INCREMENT,
  `user_id` int,
  `problem_id` int,
  `code` varchar(255),
  `created_at` datetime,
  `state` enum('채점중', '분석중', '완료'),
  `result` json,
  `analysis` json,
  FOREIGN KEY (`user_id`) REFERENCES `user` (`id`),
  FOREIGN KEY (`problem_id`) REFERENCES `problem` (`id`)
);
```

Figure 36 SQL DDL - Submission

7. Testing Plan

7.1. Objectives

이 장에서는 Development testing, Release testing, User testing의 세 가지 주요 하위 그룹으로 구성된 testing plan에 대해서 설명한다. 해당 테스트들은 프로젝트의 잠재적인 오류와 결함을 감지함으로써 서비스를 완벽하게 작동시키고 안정적인 출시를 할 수 있게 만드는 매우 중요한 개발 과정이다.

7.2. Testing Policy

7.2.1. Development Testing

Development testing은 주로 소프트웨어 개발의 잠재적 위험을 줄이고 시간과 비용의 절약을 목적으로 광범위한 결함 예방 및 탐지 전략의 동기화된 적용을 위해 수행된다. 해당 단계에서 소프트웨어는 충분한 테스트를 거치지 않아 불안정할 수 있으며 구성 요소가 서로 충돌할 수 있다. 따라서 이 단계에서 정적 코드 분석, 데이터 흐름 분석, 피어 코드 검토, unit test가 수행되어야 한다. 해당 프로세스는 소프트웨어의 정체성을 정의하는 성능, 안전 및 정상 작동 보장을 위한 신뢰성, 보안 실현에 중점을 둔다.

7.2.1.1. Performance

코드 채점 알고리즘은 본 시스템에서 가장 정확성을 요구하는 작업이기 때문에 개발자가 코드의 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 서버 간의 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 테스트를 진행하고, 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. User Testing

완성된 제품을 시장에 출시하여 고객에게 배포하는 것은 소프트웨어 개발에 있어 중요한 부분을 차지한다. 기술적으로 좋은 소프트웨어라도 잘못된 출시 방식으로 인해 문제가 발생할 수 있다. 따라서 제품과 시장 간의 연결을 원활하게 하기 위해서 Release Testing이 필수적이다. Release Testing은 새 버전의 소프트웨어와 어플리케이션을 테스트하여 소프트웨어가 완벽하게 출시될 수 있게 돕고, 운영상의 결함이 없는지 확인한다. 따라서 해당 작업은 제품의 수명이 다하기 전에 수행되어야 한다. 소프트웨어 출시 수명 주기에 따라 테스트는 소프트웨어의 기본적인 구현이 완료된 'Alpha' 버전부터 시작한다. 알파 버전으로 개발 테스트를 시작하고 베타 버전을 출시하여 사용자 및 릴리스 테스트를 포함한 추가 테스트를 진행할 계획이다. 베타 버전이 출시되면 실제 사용자로부터 피드백을 받을 수 있다.

7.2.3. User Testing

사용자 테스트를 진행할 수 있는 시나리오와 현실적인 상황을 설정해야 한다. 우리는 온라인 코딩 테스트 플랫폼의 사용자가 100명 정도라고 가정한다. 이러한 상황에서 온라인 코딩 테스트 플랫폼의 베타 버전을 배포하고 자체적으로 Use Case Test를 진행한다.

7.2.4. Testing Case

Testing case는 기능, 성능, 보안의 세 가지 기본 측면에 따라 설정될 것이다. 기능은 주로 상호작용 측면에서 테스트된다. 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 설명한다.

8.2. Frontend Environment

8.2.1. Figma

Figure 37 Figma logo

Figma는 웹 기반 UI/UX 디자인 및 프로토타이핑 협업 툴이다. 웹 기반 툴이기에 별다른 설치 없이 브라우저에서 바로 실행할 수 있으며, 운영체제와 기기에 상관없이 사용 가능하다. 웹 기반이기에 피그마 계정 소유자가 링크를 공유하여 여러 명이 아트보드를 확인하며 동시에 온라인으로 실시간 작업을 진행할 수 있어 협업에 특화되어있다. 또한, 정보를 참조할 수 있는 '개발 툴 바'가 있기에, 마우스 툴 바로 각 디자인된 요소들의 수치 값을 확인할 수 있어, 별도의 가이드 라인 없이도 빠른 작업이 가능하다. 이러한 특징들을 활용하여, 수렴한 조원들의 디자인 아이디어를 바탕으로 제작된 전체적인 디자인을 피그마로 구성하였다. 이후 프론트 엔드를 제작할 때, 툴 바를 통해 미리 정해진, 규격화된 css 정보를 쉽게 확인하여 이용할 수 있었다.

8.2.2. vscode

Figure 38 vscode logo

비주얼 스튜디오 코드(영어: Visual Studio Code) 또는 코드(code)는 마이크로소프트가 마이크로소프트 윈도우, macOS, 리눅스용으로 개발, 판매중인 통합 개발 환경이자 패키지로, 소스 코드 편집기로 사용되었다. 디버깅 지원과 Git 제어, 구문 강조 기능 등이 포함되어 있으며, 사용자가 편집기의 테마와 단축키, 설정 등을 수정할 수 있다. 마이크로소프트의 IDE기에

윈도우에서는 표준 개발 툴로서 사용되고, 텍스트 자동완성 기능과 디버깅이 매우 뛰어나기에 웹사이트 개발에 사용하였다. 또한 깃 허브가 개발한 일렉트론 프레임워크를 기반으로 구동되며, 협업을 위한 깃과 여러 기능 등이 포함되어 있으므로 클라이언트, 서버 구축을 위해 vscode를 사용하였다.

8.3. Backend Environment

8.3.1. Github

Figure 39 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 프로젝트 코드 관리를 위한 저장소로 깃허브를 사용했다.

8.3.2. Django

Figure 40 Django logo

Django는 파이썬으로 작성된 오픈 소스 웹 프레임워크로, Model-Template-View(MTV) 패턴을 따르고 있다. 고도의 데이터베이스 기반 웹사이트를 작성하는데 요구되는 수고를 덜어주고, component 재사용성(reusability)와 플러그인화 가능성(pluggability), 빠른 개발 등에 용이하게 사용된다.

8.3.3. Django REST framework

Figure 41 Django REST Framework logo

DRF(Django REST Framework)란 Django 내에서 RESTful API 서버를 쉽게 구축할 수 있도록 도와주는 오픈소스 라이브러리다. 웹 브라우저 API는 범용성이 크기 때문에 개발을 용이하게 하고, 시리얼라이즈나 인증 정책과 같은 부가적인 기능을 제공한다.

8.3.4. SQLite

Figure 42 SQLite logo

SQLite는 MySQL과 같은 데이터베이스 관리 시스템이지만, 서버가 아닌 응용 프로그램에 넣어 사용하는 상대적으로 가벼운 데이터베이스로 장고에서 기본적으로 제공한다. API는 라이브러리만을 호출하며, 데이터를 저장하는데 하나의 파일만을 사용하는 것이 특징이다.

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.
- 오픈 소스 소프트웨어 사용을 지향하며 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.
- 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.
- JavaScript 버전 16 이상, Python 버전 3.9 이상, 3.10 이하의 환경에서 개발한다.
- 브라우저는 최소 Chrome 버전 83을 지원하며 Chrome 버전 100 이상 환경에서 개발 및 테스트한다.

8.5. Assumptions and Dependencies

이 문서에 기술된 모든 시스템은 네트워크 통신이 원활하고 브라우저가 react를 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

본 소프트웨어 디자인 명세서는 IEEE 권장사항에 따라 작성되었다. (IEEE Recommended Practice for Software Design Description, IEEE-Std-1016).

9.2. Document History

Table 25 Document History

Date	Version	Description	Writer
2022/11/02	0.1	1, 2, 7	김호진
2022/11/07	0.1	5	변영민, 김호진
2022/11/11	0.1	6	전윤태
2022/11/12	0.1	4	전윤태, 변영민, 김호진
2022/11/13	0.1	3, 8	김양선, 이주빈, 임세아
2022/11/13	1	통합	Team 7