Online Judge Platform

Design Specification

2022.11.13

Introduction to Software Engineering 41

TEAM 10

팀장 최시열

팀원 김도환

김준석

박주현

윤민철

이수민

목차

1. Introduction

본 문서는 “Online Judge” 플랫폼(이하 ‘프로젝트’)에 대한 디자인 명세서이다. 본 문서의 내용은 프로젝트의 요구사항 명세서를 기반으로 하며 요구사항 명세서 작성 이후의 변경 사항들을 반영하여 작성되었다.

1.1. Objectives

이 장에서는 시스템 설계에 사용한 다이어그램과 개발 툴을 소개하며 개발팀에서 목표로 하는 프로젝트를 설명한다.

1.2. Applied Diagrams

1.2.1. UML

UML(Unified Modeling Language, 통합 모델링 언어)이란 소프트웨어 공학에서 사용되는 표준화된 범용 모델링 언어이다. UML은 시스템을 모델링하기 위해 시각적으로 표현할 때 사용하며 UML을 사용함으로써 시스템에 대한 이해도를 높일 수 있다.

UML 다이어그램은 UML을 사용하여 그린 도면을 의미하며 13 종류가 있다. 본 문서에서는 프로그램을 모델링하기 위해 UML 다이어그램 중 deployment diagram, use case diagram, sequence diagram, class diagram을 사용한다.

1.2.1.1. Deployment Diagram

Deployment diagram은 시스템을 구성하는 하드웨어 컴포넌트의 연결 관계와 각 하드웨어에서 실행되는 소프트웨어 컴포넌트의 배치를 나타내는 도면이다. 하드웨어 컴포넌트는 직사각형의 노드로 표현하며 소프트웨어 컴포넌트는 하드웨어 컴포넌트 안에 중첩해서 표현한다.

1.2.1.2. Use case Diagram

Use case 다이어그램은 actor와 시스템의 상호작용을 나타낸다. Use case 다이어그램은 시스템의 여러 actor와 이들이 시스템을 사용하는 다양한 케이스를 보여주기 때문에 use case 다이어그램을 작성하기 위해서는 actor와 use case가 필요하며 actor는 사람의 모습으로, use case는 원 또는 타원으로 표현한다. Use case 다이어그램의 actor는 시스템의 사용자와 관리자 등 사람뿐만 아니라 개발하고자 하는 시스템의 서브시스템과 상호작용하는 다른 시스템도 포함한다.

1.2.1.3. Sequence Diagram

Sequence 다이어그램은 객체 간의 상호작용을 시간 순서대로 나타낸 도면이다. 시스템을 구성하는 객체와 이들이 서로 교환하는 메시지를 통해 표현된다. 메시지를 통해 데이터의 흐름을 파악하기 용이하며 use case 다이어그램에서 표현된 기능이 시스템 내에서 어떻게 동작하는지 구체적으로 표현할 수 있다. 또한 시간 순으로 표현되기 때문에 시스템이 동작하는 시나리오를 확인하기에도 편리하다.

1.2.1.4. Class Diagram

Class 다이어그램은 시스템의 클래스, 속성, 동작, 객체 간의 관계를 보여주는 도면으로, 시스템의 구조를 정적으로 표현한 것이다. 클래스는 클래스의 이름, 속성(변수), 메소드를 작성하여 나타낸다. Class 다이어그램을 통해 클래스 간의 상속 관계와 연결 관계를 확인하여 각각의 서브시스템이 어떻게 구성되어 있는지를 파악할 수 있다.

1.2.2. Entity Relationship Diagram

ER 다이어그램(Entity-Relationship Diagram, 개체-관계 다이어그램)은 데이터 모델링 방법 중 하나인 ER 모델링(Entity-Relationship Modelling, 개체-관계 모델링)을 통해 만들어지는 도면을 뜻하며 entity(개체)와 entity 사이의 relationship(관계)으로 데이터를 표현한다. 데이터 모델링이란 논리적인 데이터 모델을 구성하는 작업을 의미하며 이를 물리적인 데이터베이스 모델로 환원하여 시스템의 데이터베이스에 반영하게 된다.
본 문서에서는 ER 다이어그램과 함께 relational schema를를 사용하여 데이터베이스 디자인을 설명한다.

1.3. Applied Tools(개발에 사용한 툴 작성하는 파트. 1.3.1.은 지난 학기 팀들이 작성한 예시)

1.3.1. Visual Studio Code

마이크로소프트에서 개발한 IDE이다. Windows, Linux, Mac 모든 환경에서 활용가능하며 extentsion을 통해 코드 컴파일, 개발 환경 세팅 등을 할 수 있다. 다른 IDE에 비해 가벼우며 파일 관리 또한 편히 할 수 있기에 프론트엔드/백엔드 개발에 사용했다.

1.3.2. Figma

웹의 UI/UX 디자인을 만들기 위해 활용하는 도구이다. 별도의 독립된 환경에서 디자인하기에 웹 개발과 parallel하게 진행된다.

1.3.3 Chrome

프론트엔드 개발에서, 구글 크롬의 개발자모드를 이용하여 여러 액션에 대한 에러 및

유효성 검증을 하였다.

1.4. System Overview

본 서비스는 컴퓨터 과학을 공부하는 학생들이 코딩 테스트를 실습할 수 있도록 고안되었다. 제한된 개발 기간에 인해 python 코드만 지원을 하도록 하였으며, 학생들이 코드를 작성할 시에 결과값을 볼 수 있도록 구현하였다. 또한 교수자가 선정한 공개된 test case에 대한 오류를 확인할 수 있으며, 최종 제출 시 모든 test case에 대한 정답여부를 확인할 수 있다. 또한 코드의 정답여부 이외에 가독성, 표절률, 효율성과 같은 점수도 제공한다. 다른 점수를 제공함으로 학생들의 높은 목적성 및 실력 향상을 기대한다.

본 서비스는 웹 서비스를 제공하기 위해 java기반 프레임워크인 react를 활용하였으며 libarary를 활용하여 backend 서버로 부터 데이터를 제공받는다. 즉 클라이언트 side에서 action을 취하면 http통신을 통해 server side로부터 request를 받아 유저는 이에 대한 내용을 볼 수 있다.

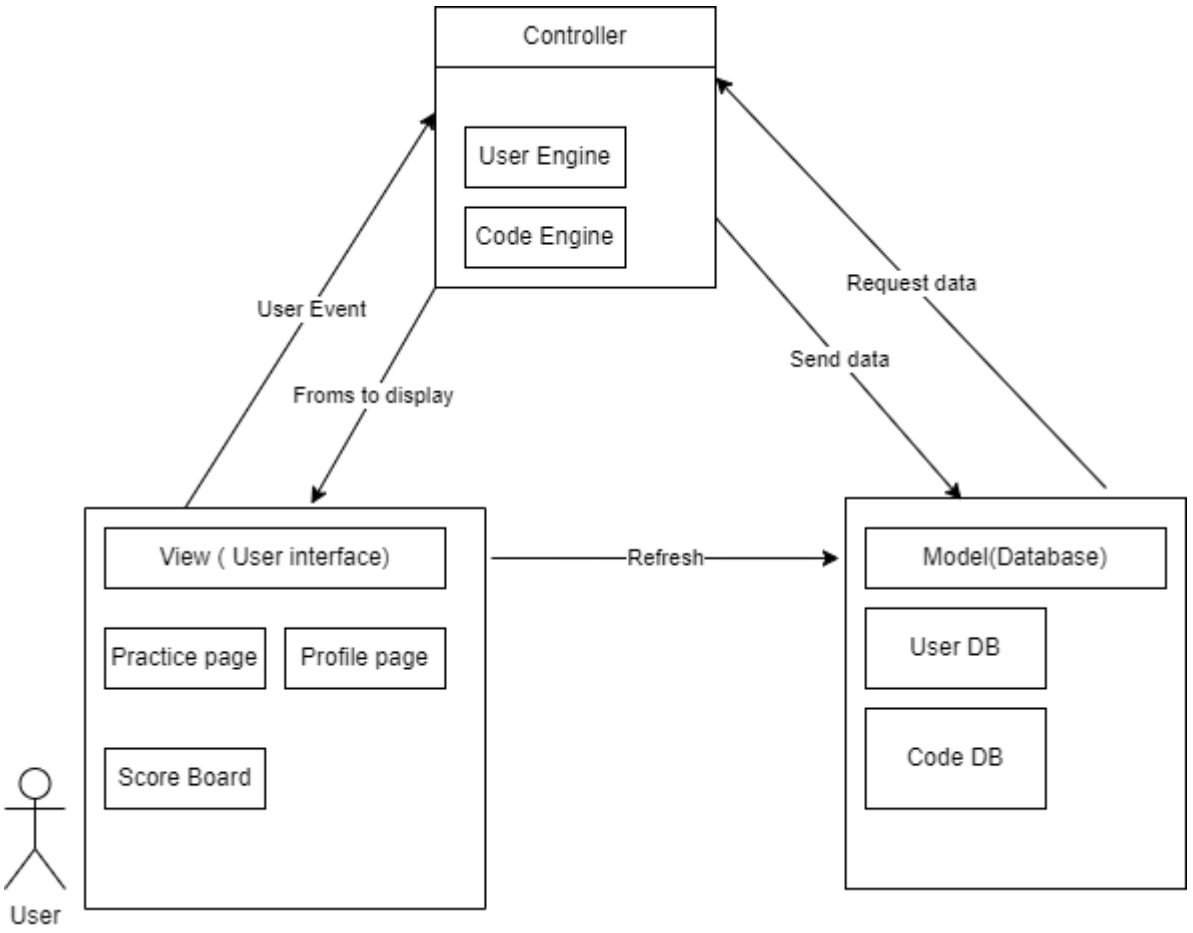
2. System Architecture – Overall

2.1. Objectives

이 장에서는 시스템의 구조를 나타낼 것이다. 이 프로젝트의 frontend를 시작으로 backend까지 상세히 설명할 예정이다.

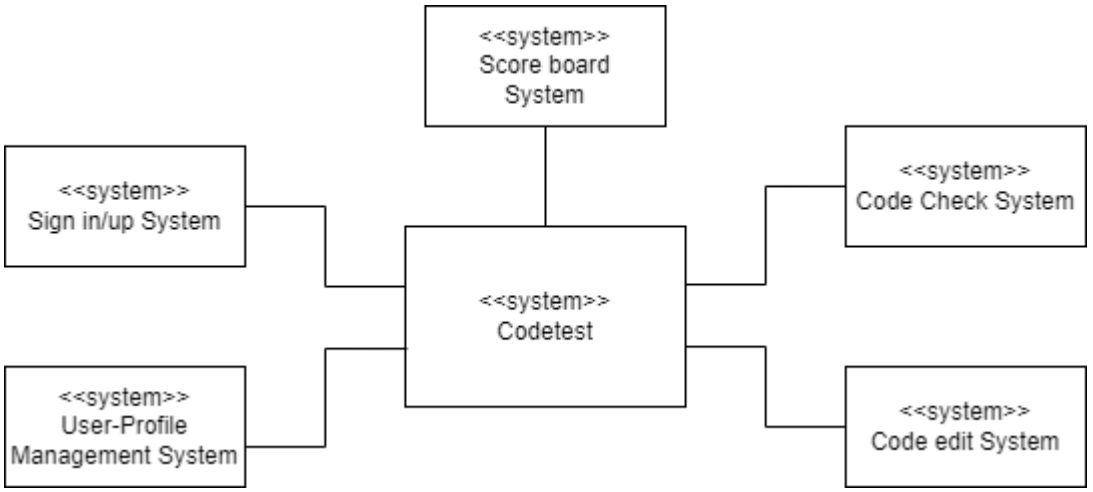
2.2. System Organization

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



[Diagram 1] Overall System Architecture

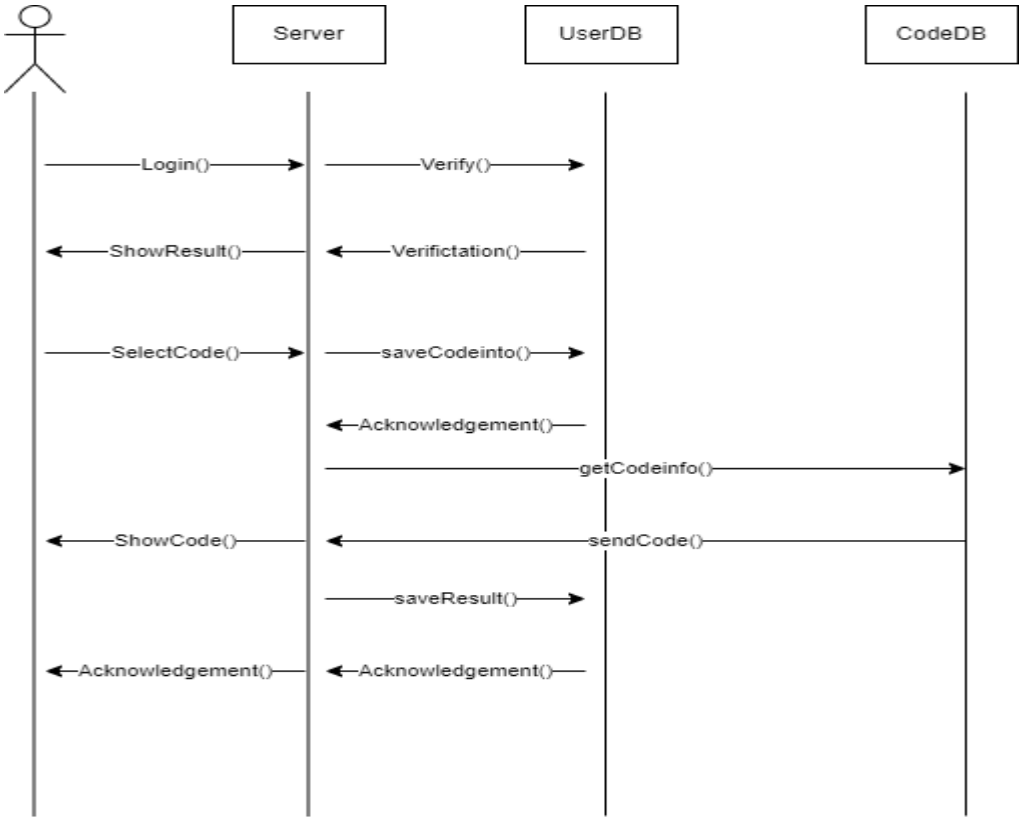
2.2.1. Context Diagram



[Diagram 2] Overall Context Diagram

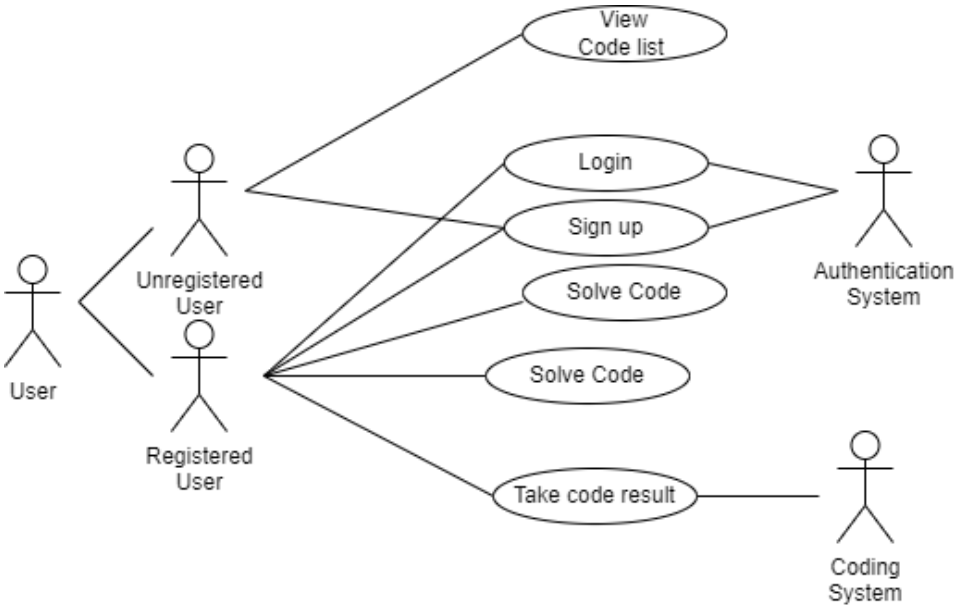
2.2.2. Sequence Diagram

메인 시스템 설계 상태



[Diagram 3] Overall Sequence Diagram

2.2.3. Use Case Diagram



[Diagram 4] Overall Use case Diagram

3. System Architecture – Front-end

3.1. Objectives

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 이름과 계정 정보가 있다. 사용자 계정에 학적 정보는 변경 할 수 없으며 사용자 이름만 변경 가능하다. 문제 내역에는 사용자에게 배정된 문제와 관련된 제출 여부를 성공과 지각 미제출로 표시된다.

3.2.1.1 Attributes

이 클래스가 가지는 속성은 다음과 같다.

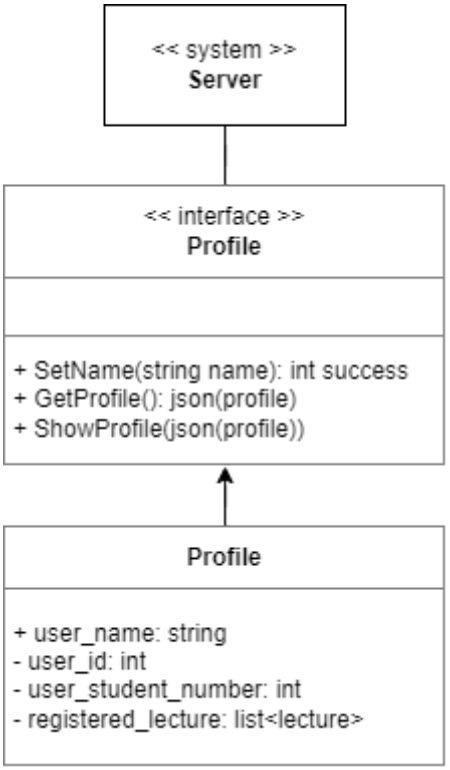
- user_student_number: 학생의 학번이다.
- user_name: 학생의 이름이다.
- registered_lecture: list<lecture>: 사용자가 수강하는 강의 리스트다.

3.2.1.2. Methods

이 클래스가 가지는 메서드는 다음과 같다.

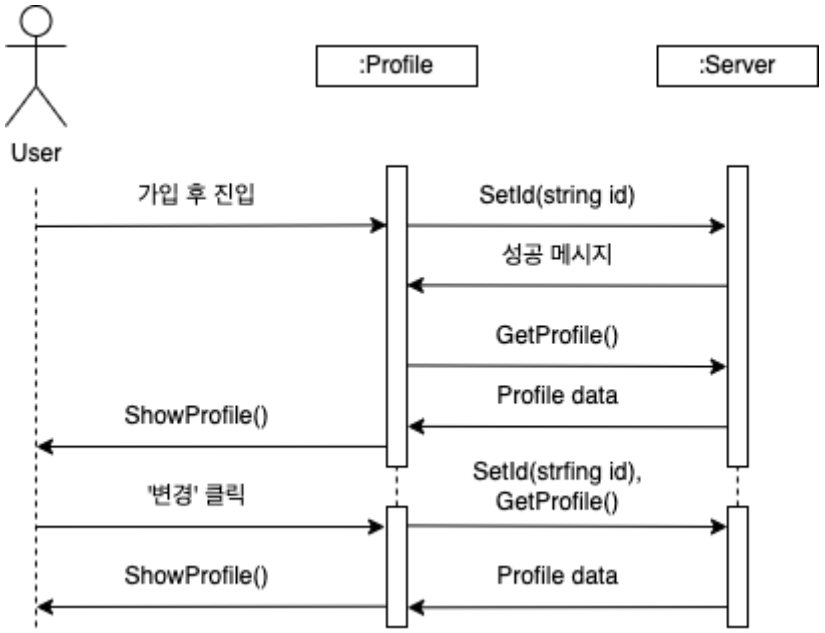
- SetName(): 사용자 이름을 바꾸는 함수이다.
- GetProfile(): 사용자 프로파일 정보를 가져오는 함수이다.
- ShowProfile(): 가져온 프로파일 정보를 보여주는 함수이다.

3.2.1.3 Class Diagram



[Diagram 5] Class Diagram - Profile

3.2.1.4 Sequence Diagram



[Diagram 6] Sequence Diagram - Profile

3.2.2. Main

메인 페이지를 관리하는 클래스이다. 메인 페이지는 수강하는 강의와

제출해야할 과제 객체로 크게 나뉜다.

3.2.2.1 Attribute

Main 클래스가 가지는 속성은 다음과 같다.

- lecture: 강의를 관리하는 lecture 객체이다.

Lecture 클래스가 가지는 속성은 다음과 같다.

- lecture_list: 강의의 대분류를 나타내는 문자열 리스트이다.
- task: 과제를 관리하는 객체이다.

Task 클래스가 가지는 속성은 다음과 같다.

- task: list<task_card>: 강의가 가진 과제 리스트이다.
- lecture_id: 과제를 갖는 강의의 아이디 값이다.

Task_card 클래스가 가지는 속성은 다음과 같다.

- task_title: 과제의 이름이다.
- task_id: 과제를 구분하는 id값이다.
- testcase: 과제의 테스트케이스를 관리하는 객체이다.
- task_sub_num: 과제를 제출한 횟수이다.

Testcase 객체의 속성들이다.

- open_testcase_list: list<values>공개된 테스트 케이스의 리스트이다.
- hidden_testcase_list: list<values>비공개 테스트 케이스의 리스트이다.

3.2.2.2 Methods

Lecture 클래스가 가지는 메서드는 다음과 같다.

- get_lecture_list(): 서버에서 사용자가 듣는 강의 리스트를 json으로 받아오는 함수다.
- show_lecture_list(lecture_list): 사용자가 대분류에 따라 듣는 강의를 DB에서 받아와서 보여주는 함수이다.
- show_lecture(string lecture_list[i]): 사용자가 강의를 선택했을 때 해당 강의 정보를 json으로 넘겨주는 함수다.
- show_task(son(list<task>)): 사용자에게 task들을 보여주는 함수다.

Task 클래스가 가진 메서드들이다.

- show_task_list(task[i]): 주어진 과제의 정보를 가져오는 함수다.
- show_task_card(): 특정 과제를 보여주는 함수다.

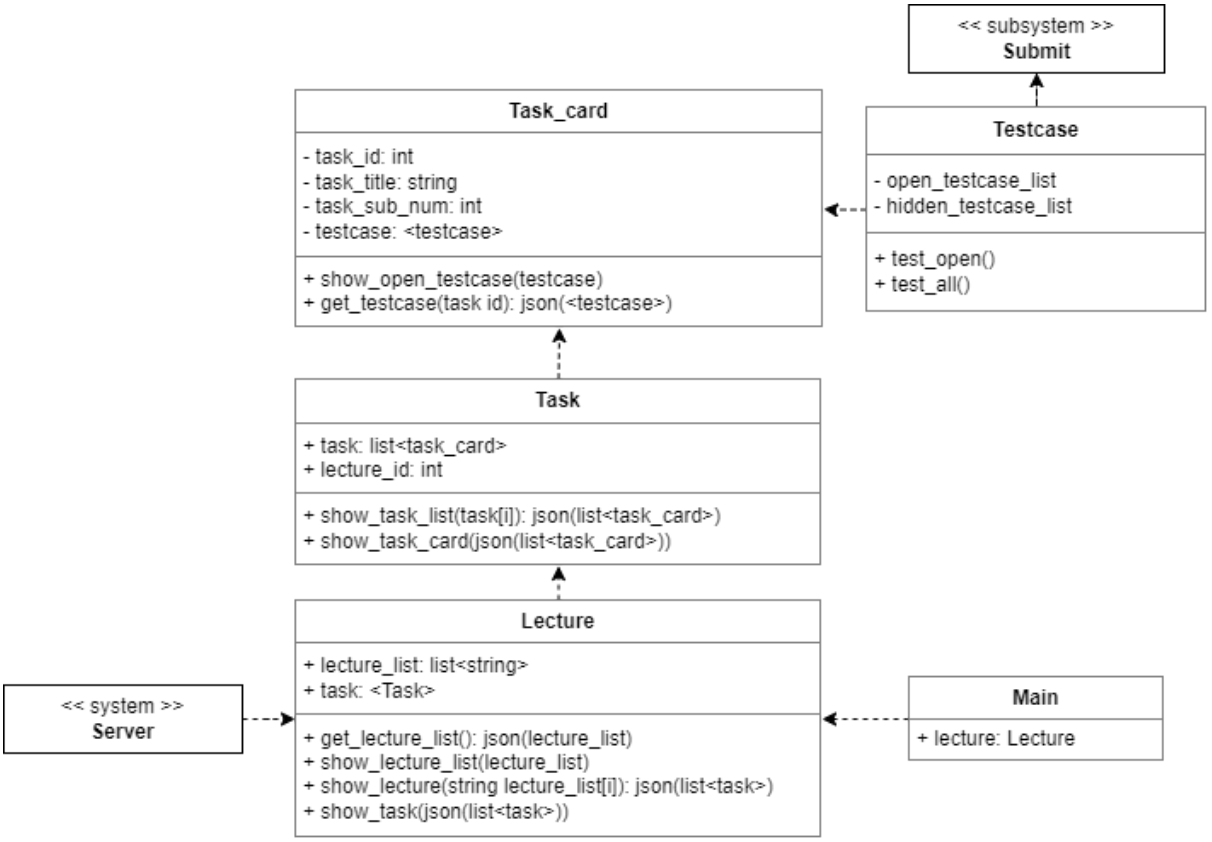
Task_card 클래스가 가진 메서드들이다.

- show_open_testcase(): testcase들을 보여주는 함수다.
- get_testcase(): 서버에서 testcase를 받아오는 함수다.

Testcase 객체가 가진 메서드는 다음과 같다.

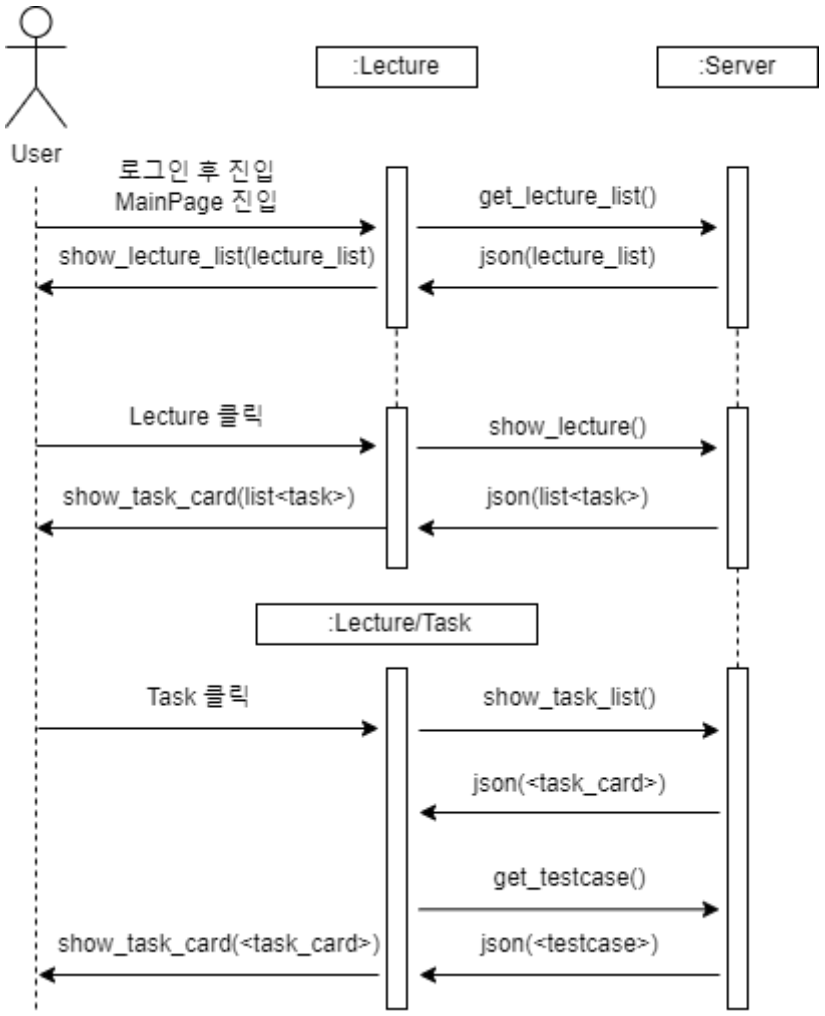
- test_open(): 공개 테스트 케이스만 테스트하는 함수다.
- test_all(): 모든 테스트 케이스를 테스트하는 함수다.

3.2.2.3 Class Diagram



[Diagram 7] Class Diagram - Main

3.2.2.4 Sequence Diagram



[Diagram 8] Sequence Diagram - Main

3.2.3. Code

사용자가 실습 코드를 입력하고 그 코드를 제출하여 결과 점수를 확인하는 과정을 담당하는 컴포넌트다.

3.2.3.1 Attribute

Code Object가 갖는 attribute이다.

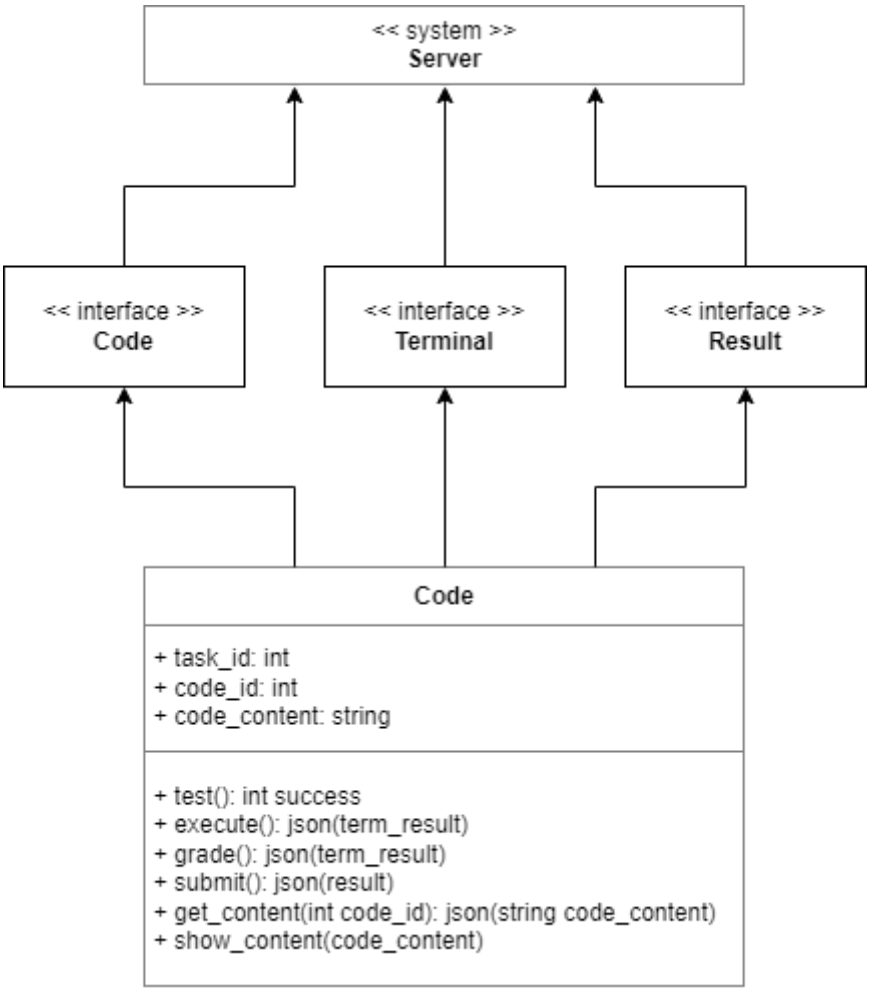
- task_id: 작성한 실습 코드의 대상 task id 값이다.
- code_id: 작성한 코드의 id값이다.
- code_content: 코드 내용이다.

3.2.3.2 Methods

- test(): 선택된 testcase로 실행하는 함수다.
- execute(): 코드를 실행하는 함수다.
- grade(): 코드를 전체 testcase에 대하여 채점하는 함수다.
- submit(): 과제를 제출하고 해당 결과를 보여주는 함수이다.

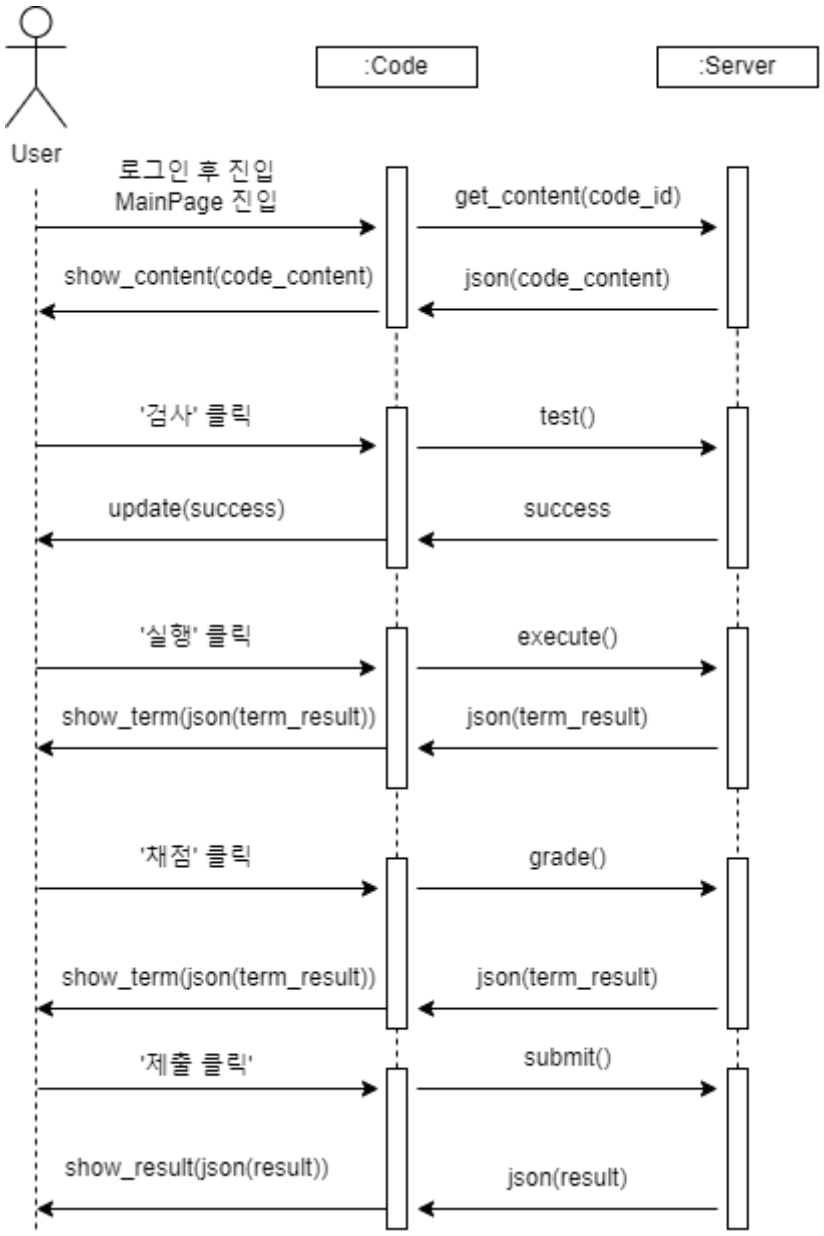
- update(success): success 값을 보여주는 함수다.
- show_term(json(term_result)): 터미널에 결과 값을 보여주는 함수다.
- show_result(json(result)): 결과창에 제출 결과를 보여주는 함수다.
- show_content(code_id): 마지막으로 저장된 코드를 보여주는 함수다.

3.2.3.3 Class Diagram



[Diagram 9] Class Diagram - Code

3.2.3.4 Sequence Diagram



[Diagram 10] Sequence Diagram - Code

4. System Architecture – Back-end

4.1. Purpose

이번 장에서는 DB와 서버의 구조를 설명한다

4.2. Overall Architecture

DB는 사용자의 정보와 코드의 정보를 불러오는 용도로 사용이 된다.

Request와 Response를 처리하는 Controller layer, 비즈니스 로직을 처리하는 Service

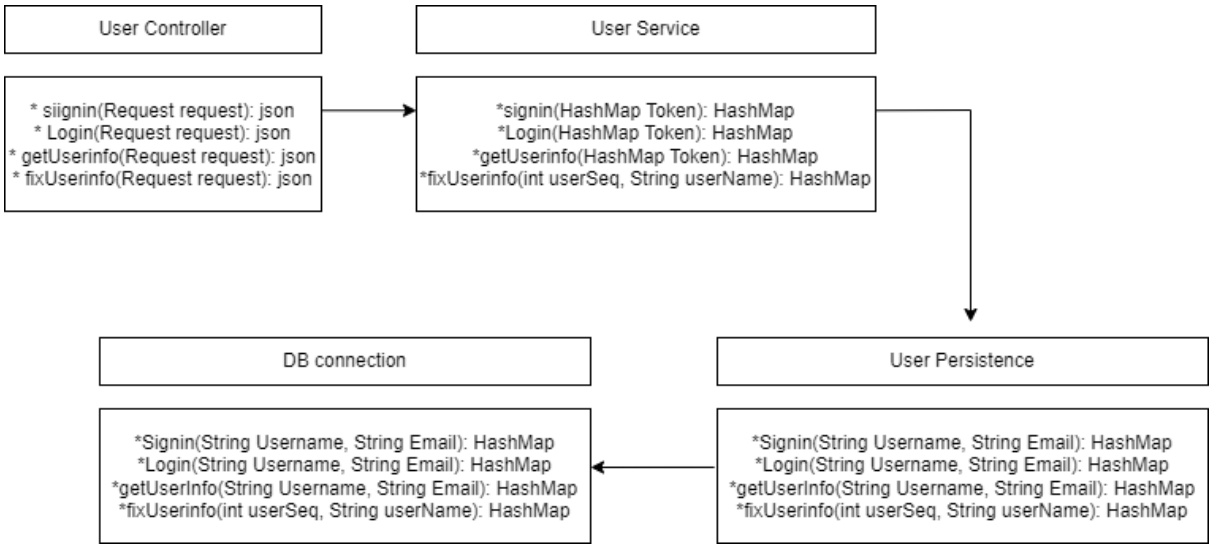
layer, 마지막으로 DB와의 통신을 담당하는 Persistence layer로 구성되어 있다.

4.3. SubComponents

4.3.1. User system

User와 관련된 작업들이 공통적으로 묶여있는 System이다.

4.3.1.1. User System Class Diagram



[Diagram 11] Class Diagram - User System

4.3.1.1.1. User Controller Class

Request에서 Parameter를 받아서 이를 HashMap으로 parsing한 다음 Service Layer를 호출하는 역할을 가지고 있다. 해당 클래스는 4가지의 메소드가 존재한다. 첫 번째는 회원가입과 관련된 Singin이 존재한다. 두번째는 로그인과 관련된 Login이다. 세번째는 사용자의 정보를 가져오는 getUserinfo이다. 마지막은 개인정보 중 이름을 수정할 수 있는 fixUserinfo이다.

4.3.1.1.2. User Service Class

Controller로부터 받은 인자를 parsing하고 Persistence Layer를 호출하고 나서는 해당 Layer에서 반환된 Controller에 넘겨주는 역할을 한다. 해당 서비스에는 총 4가지 메소드가 존재한다. 첫번째는 회원가입과 관련된 Signin이 존재한다. 두번째는 로그인과 관련된 Loginin이다. 세번째는 사용자의 정보를 가져오는 getUserInfo이다.

마지막은 개인정보 중 이름을 수정할 수 있는 fixUserInfo이다.

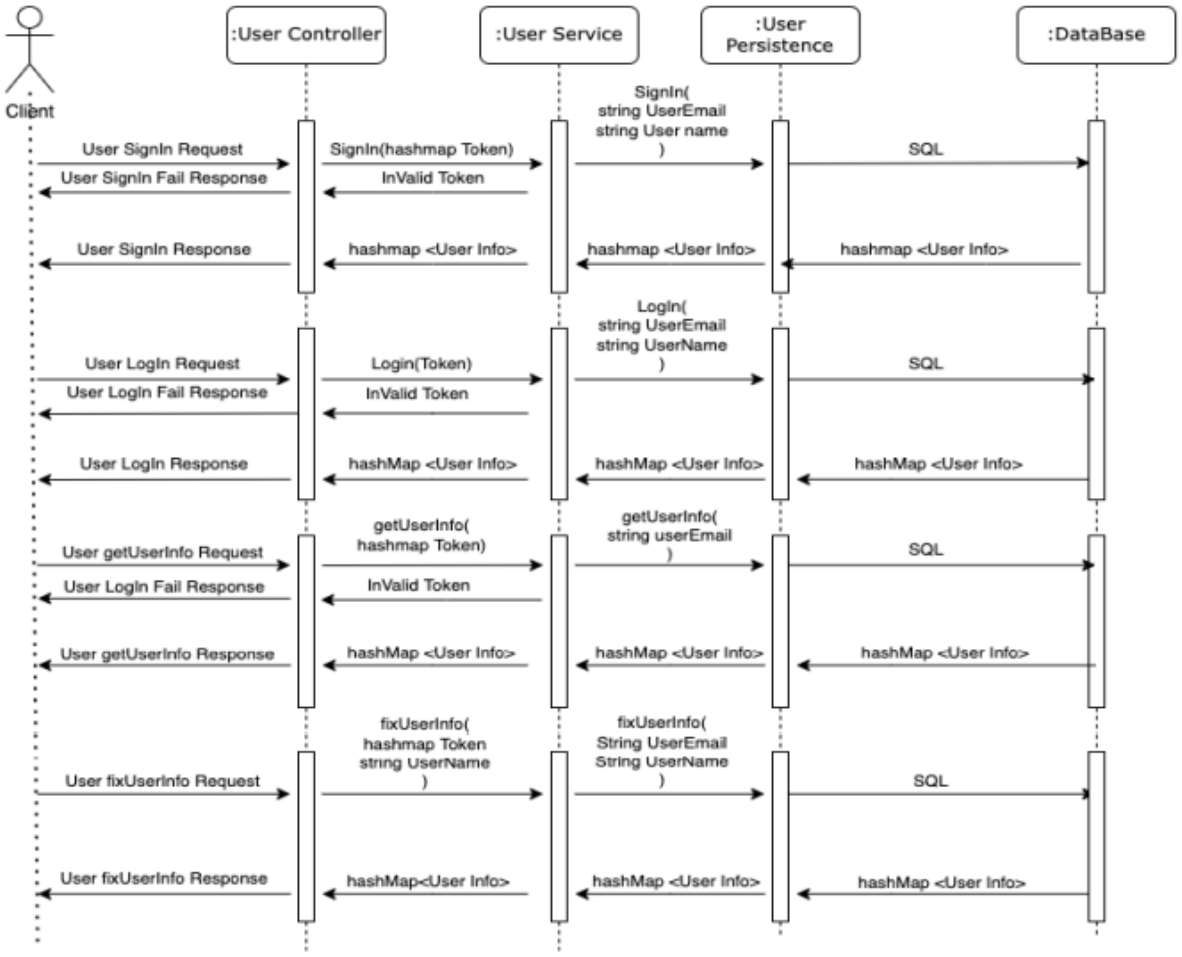
4.3.1.1.3. User Persistence Class

Service Layer에서 받은 parameter를 바탕으로 SQL문을 만들어서 DB connection에서 넘겨준다. 또한 DB connnection에서 DB로부터 받은 값을 HashMap으로 Service Layer에 넘겨주는 역할을 한다. 해당 서비스에는 총 4가지 메소드가 존재한다. 첫번째는 회원가입과 관련된 Signin이 존재한다. 두번째는 로그인과 관련된 Login이다. 세번째는 사용자의 정보를 가져오는 getUserinfo이다. 마지막은 개인정보 중 이름을 수정할 수 있는 fixUserinfo이다.

4.3.1.1.4. DB Connection

DB랑 연결을 담당하는 부분으로 Persistence Layer로부터 받은 SQL을 DB에 넘겨 결과값을 받는 역할을 한다. 이후 결과를 받으면 Persistence Layer로 넘겨준다.

4.3.1.2. User Sequence Diagram

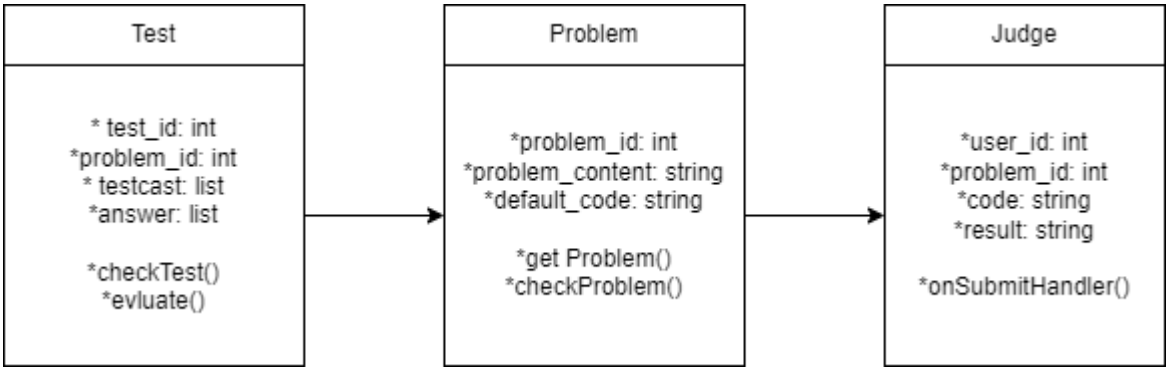


[Diagram 12] User Sequence Diagram

4.3.2. Code System

Code와 관련된 작업들이 공통적으로 묶여있는 시스템이다.

4.3.2.1. Code System Class Diagram



[Diagram 13] Code System Class Diagram

4.3.2.1.1. Problem Class

각 문제마다 식별 번호와 문제 설명이 존재한다. 그리고 필요에 따라 기본 코드(스켈레톤 코드)가 존재한다. setProblem()을 통해 문제를 가져오고, checkProblem()을 통해 문제 채점 결과를 반환한다.

4.3.2.1.2. Test Class

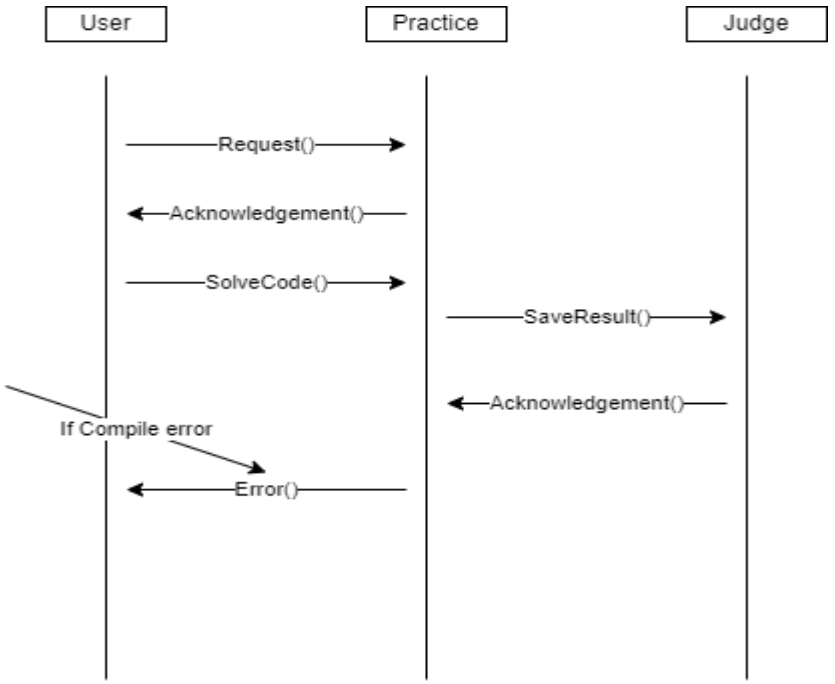
테스트마다 test_id(식별번호)가 존재하고 problem_id도 존재한다. 그리고 각 문제마다 testcase가 존재하고 answer도 존재한다. checkTest()를 통해 테스트케이스를 돌려보며 확인하고 evaluate()를 통해 테스트케이스 별 정답을 확인한다.

4.3.2.1.3. Judge Class

user_id(사용자 식별 변호), problem_id(문제 식별 번호)가 존재하며, code, result를 통해 제출한 코드와 코드 실행 결과를 확인한다.

onSubmitHandler()를 통해 코드를 서버로 보내 결과를 가져온다.

4.3.2.2. Code System Sequence Diagram



[Diagram 14] Code System Sequence Diagram

5. Protocol Design

5.1. Objectives

해당 장에서는 서버와 클라이언트의 통신에 사용되는 프로토콜을 명시한다. 또한 각 인터페이스의 정의를 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[1]. 이 형식을 통해 서버와 클라이언트 간의 데이터 교환이 이루어진다.

[1]https://ko.wikipedia.org/wiki/JSON

5.3. Protocol Description

HyperText Transfer Protocol, 즉 HTTP 프로토콜을 통해 서버와 클라이언트가 통신한다. REST API를 사용하여 주소와 HTTP 요청 메소드를 명시하여 서버로부터 동작을 요청할 예정이다.

5.4 Interface

5.4.1 Get Lecture list

-Request

Attribute	Detail
Method	Get
Url	/lectures

[Table 1] Get Lecture List Table

- Response

Attribute	Detail	
Success Code	200	
Failure Code	HTTP error code = 400	
Success Response Body	Lectures	List of Lectures
	Lecture= {key: ..., name: …}	
Failure Response Body	Message	Fail message

[Table 2] Response Lecture List Table

5.4.2 Get Problem list

-Request

Attribute	Detail
Method	Get
Url	/lectures/:lecture_id/problems

Parameter	Lecture key	Key of selected lecture

[Table 3] GET Problem List Table

- Response

Attribute	Detail	
Success Code	200	
Failure Code	HTTP error code = 400	
Success Response Body	Problems	List of Problems
	Problem= {key: ..., name: …}	
Failure Response Body	Message	Fail message

[Table 4] Response Problem List Table

5.4.3 Get Problem Detail

-Request

Attribute	Detail	
Method	Get	
Url	/lectures/:lecture_id/problems/:problem_id	
Parameter	Lecture key	Key of selected lecture
	Problem key	Key of selected problem

[Table 5] GET Problem Detail Table

- Response

Attribute	Detail	
Success Code	200	
Failure Code	HTTP error code = 400	
Success Response Body	text	the contents of the problem
	constrain	the constraints of the problem
	skeleton	the skeleton code of the problem
	testcases	the list of test case
	testcase={input:..., output:...}	
Failure Response Body	Message	Fail message

[Table 6] Response Problem Detail Table

5.4.4 Save code to server

-Request

Attribute	Detail	
Method	Post	
Url	/lectures/:lecture_id/problems/:problem_id/save/:num	
Parameter	Lecture key	Key of selected

		lecture
	Problem key	Key of selected problem
	Code	Code created by user
	num	the number of the saving

[Table 7] Request Save Code

- Response

Attribute	Detail	
Success Code	200	
Failure Code	HTTP error code = 400	
Success Response Body	Message	Success message
Failure Response Body	Message	Fail message

[Table 8] Response Save Code

5.4.5 Run test case

-Request

Attribute	Detail	
Method	Post	
Url	/lectures/:lecture_id/problems/:problem_id/test/:num	
Parameter	Lecture key	Key of selected

		lecture
	Problem key	Key of selected problem
	Code	Code created by user
	num	the number of the test case

[Table 9] Request Run Test Case

- Response

Attribute	Detail	
Success Code	200	
Failure Code	HTTP error code = 400	
Success Response Body	Output	Output from the user code by the test case
Failure Response Body	Message	Fail message

[Table 10] Response Run Test Case

5.4.6 Grade

-Request

Attribute	Detail
Method	Post
Url	/lectures/:lecture_id/problems/:problem_id/grade

Parameter	Lecture key	Key of selected lecture
	Problem key	Key of selected problem
	Code	Code created by user

[Table 11] Request Grading

- Response

Attribute	Detail	
Success Code	200	
Failure Code	HTTP error code = 400	
Success Response Body	Results	Results of all test cases
	Result={result:..., output:...}	
Failure Response Body	Message	Fail message

[Table 12] Response Grading

5.4.7 Submit

-Request

Attribute	Detail
Method	Post
Url	/lectures/:lecture_id/problems/:problem_id/submit

Parameter	Lecture key	Key of selected lecture
	Problem key	Key of selected problem
	Code	Code created by user

[Table 13] Request Submitting

- Response

Attribute	Detail	
Success Code	200	
Failure Code	HTTP error code = 400	
Success Response Body	Plagiarism	Plagiarism rate of the user code
	Results	Same as 5.4.6
	Efficiency	Efficiency score of the user code
	Readability	Readability score of the user code
	Description	Description about the problem or answer
	Morelearn	Link to learning materials related to the problem

Failure Response Body	Message	Fail message

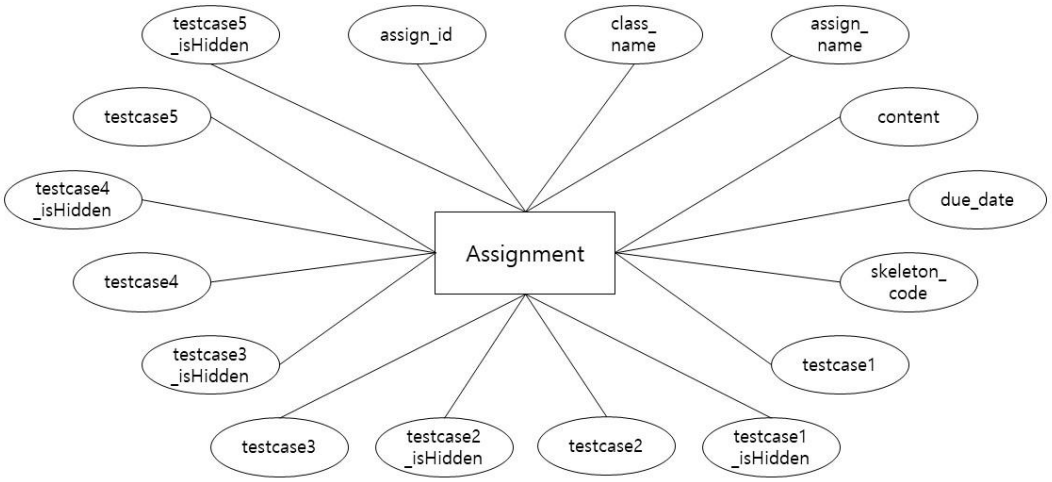
[Table 14] Response Submitting

6. Database Design

6.1. Objectives

본 장에서는 프로젝트의 데이터베이스 디자인을 설명한다. 요구사항 명세서에서 작성된 data dictionary와 data flow diagram을 바탕으로 작성되며 변경된 요구사항을 반영하였다. ‘Testcase’ 테이블의 attribute가 ‘Assignment’ 테이블에 추가되고 ‘Assignment’ 테이블 이외의 테이블은 삭제되었다.

6.2. ER Diagram



[Diagram 15] ER Diagram(Assignment Entity)

6.3. Relational Schema

테이블 이름	Assignment
테이블 설명	과제 정보
PRIMARY KEY	assign_id
FOREIGN KEY	
INDEX	

NO	PK	AI	FK	NULL	칼럼 이름	TYPE	설명	참조 테이블
1	Y	Y			assign_id	INTEGER	과제 식별 번호	
2					class_name	TEXT	수업명	
3					assign_name	TEXT	과제명	
4					content	TEXT	과제 설명	
5					due_date	TEXT	마감일	
6					skeleton_code	TEXT	스켈레톤 코드	
7					testcase1	TEXT	테스트케이스1	
8					testcase1_isHidden	INTEGER	테스트케이스1의 히든 여부	
9					testcase2	TEXT	테스트케이스2	
10					testcase2_isHidden	INTEGER	테스트케이스2의 히든 여부	
11					testcase3	TEXT	테스트케이스3	
12					testcase3_isHidden	INTEGER	테스트케이스3의 히든 여부	

13					testcase4	TEXT	테스트케이스4	
14					testcase4_isHidden	INTEGER	테스트케이스4의 히든 여부	
15					testcase5	TEXT	테스트케이스5	
16					testcase5_isHidden	INTEGER	테스트케이스5의 히든 여부	

[Table 15] Assignment Relational Schema

7. Testing Plan

7.1. Objectives

Testing plan 에서는 요구사항 명세서에 기술한 사용자 및 관리자의 시나리오를 바탕으로 전체 시스템이 그에 맞게 잘 실행되는지를 확인한다. Development Testing, Release Testing, User Testing의 세 가지 하위 그룹으로 구성된 Testing Plan에 대해서 설명하며 해당 테스트들은 프로젝트의 잠재적인 결함과 오류를 미리 감지함으로써 서비스의 안정성과 완성도를 높인다.

7.2. Testing Policy

7.2.1. Development Testing

Developement testing은 소프트웨어 개발 과정에서 발생할 수 있는 잠재적 위협 요소를 미리 확인해 비용과 시간을 절약하는 목적으로 수행되며 광범위적 결함 예방 및 탐지 전략의 동기화된 적용을 위해 수행된다. 크게 Performance, Reliability, Security의 측면에서 수행되며 그를 위해 정적 코드 분석, 데이터 흐름 분석, 피어 코드 리뷰, 및 단위 테스트가 진행 된다.

7.2.2. Release Testing

Release tseting은 새 버전의 소프트웨어와 어플리케이션을 테스트하여 소프트웨어가

조금 더 완성도 있게 출시 될 수 있도록 운영에 결함이 없는지 확인하는 과정이다. 출시 수명 주기에 따라 기본 기능만 구현된 “Alpha” 버전으로 부터 사용자로부터 피드백을 받을 수 있는 “Beta”버전 등을 거쳐 최종 배포 버전으로 발전해 나가게 된다. 이 과정에서 시스템이 사용자의 요구사항을 모두 충족하는지, 개발 의도에 맞게 작성되었는지, 서비스들이 의도된 사항대로 작동하는지 등이 확인된다.

7.2.3. User Testing

실제 사용 환경에서 시스템을 테스트 하게 되는 과정이다. 이 과정에서 실제 사용자들이 테스트를 진행하면서, product의 내용에 대해 평가하고, product의 출시 여부를 결정하게된다.

7.2.4. Test Case

A. Student ID 확인

구성요소	설명
테스트 이름	Valid User ID
사전 조건	시작 화면에서 유저가Student ID를 submit 하였다
수행 절차	1. 사용자가 String형태의 Student ID를 제출한다. 2. 서버는 ID가 Non-null임을 Assure한다. 3. Non-Null 임이 확인되면 다음 화면으로 넘어간다
기대 결과	Student ID를 확인할 수 있다.

[Table 16] Test Case: Student ID 확인

B. 문제 풀이 확인

구성요소	설명
테스트 이름	isSolved
사전 조건	사용자가 메인 페이지에서 종료를 시도하려한다.

수행 절차	1. 사용자가 코드 제출을 하지 않고 시험을 종료하려함 2. Submit status가 submitted로 확인되지 않으면 종료시 warning message를 생성 3. 코드 실행만 하고 제출은 하지 않았을 경우 이를 확인해 경고 메시지를 생성함
기대 결과	비정상 종료 및 실수로 인한 종료를 방지할 수 있음

[Table 17] Test Case: isSolved

C. 위험 코드 감지

구성요소	설명
테스트 이름	Toxic Code Detect
사전 조건	사용자가 본인의 코드를 서버에 제출함
수행 절차	1. 코드 속에 존재하는 불필요한 Import Library를 감지해 이가 존재한다면 코드를 실행하지 않음 2. import sys 등 시스템 환경변수를 변경시킬 수 있는 코드의 실행을 미연에 방지 3. is in 기능 (python 내장 기능)을 활용해 import sys라는 string이 있는지 확인해 존재 시 실행 방지
기대 결과	유저의 코드가 시스템 설정을 망치는 것을 방지할 수 있음

[Table 18] Test Case: Toxic Code Detect

D. Multi-Request 대비

구성요소	설명
테스트 이름	Extreme_Submit
사전 조건	다수의 사용자가 동시다발적으로 서버에 request를 보냄
수행 절차	1. PostMan 등의 라이브러리를 활용해 여러 ip에서

	동시다발적으로 request를 보내 어느 만큼의 request까지 동시 처리가 가능한지 확인 2. 다수의 리퀘스트 발생 시 사전 정의된 우선순위에 따라 할당함 3. 사전 정의된 threshold를 넘지 않는 request들에 서버가 과부하가 된다면 test실패로 간주
기대 결과	시스템 과부하 및 서버 다운의 상황을 방지할 수 있음

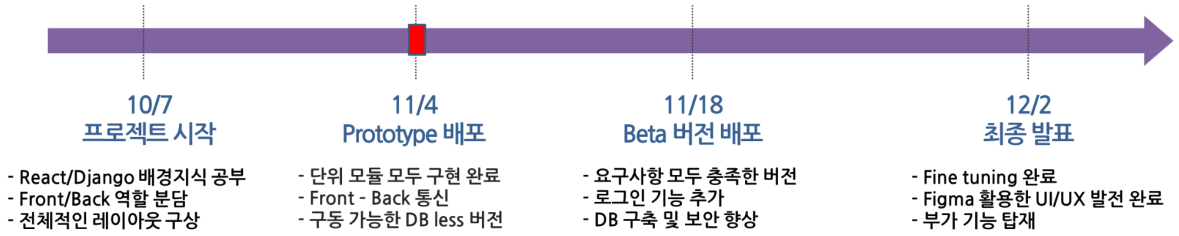
[Table 19] Test Case: Extreme Submit

8. Development Plan

8.1 Objective

프로젝트의 Milestone들을 기술하며 전체적인 개발 계획과 완성 목표일등을 명시해 현재까지의 개발 현황을 설명한다.

8.2 Project Milestone



[Figure 1] Project Milestone

9. Index

9.1 Diagram Index

9.2 Table Index

9.3 Figure Index