Software Design Specification

for

Web Coding Test

2017312425 고유안
2018314384 배재현
2015314136 조규상
2018312790 최민석
2017311028 최주은
2018310619 황선진

소프트웨어공학개론 6 조

2022. 11. 13

목차

그림 목차

표 목차

1. Preface

1.1. Objective

본 문서의 독자층을 정의하고, 문서를 구성하는 항목과 각 항목에 대해 기술한다.

1.2 . Readership

본 문서는 독자층을 코딩 제출 플랫폼을 개발 및 유지보수하는 소프트웨어 엔지니어, 시스템의 구조를 설계하는 시스템 아키텍처 등으로 상정하여 요구사항과 구성 요소를 이해하기 쉽도록 돕기 위해 작성되었다.

1.3 . Document Structure

1.3.1 Introduction

이 장에서는 전반적인 시스템의 구조에 대하여 기술한다. 각 서브 시스템 구조에 대한 소개와 함께 전체 시스템 구조에 대한 소개를 통해 서브 시스템 간의 상호작용 관계를 나타낸다.

1.3.2 Overall System Architecture

이 장에서는 시스템 설계에 사용된 다이어그램과 도구를 소개하고, 시스템의 개발 배경이 되는 문제의식과 본 시스템의 활용 방안을 기술한다.

1.3.3 System Architecture - Frontend

이 장에서는 프론트 엔드 시스템의 구조, 속성 및 기능, 시스템을 구성하는 각 요소들의 관계를 기술한다.

1.3.4 System Architecture - Backend

이 장에서는 백 엔드 시스템의 구조를 시각화하여 제시하고, 각 구성요소들의 속성과 관계에 대해 기술한다.

1.3.5 Protocol Design

이 장에서는 서브 시스템 간의 상호작용에 필요한 프로토콜에 관해 설명하고, 통신 과정에서 전달되는 메시지의 형식과 용도를 설명한다.

1.3.6 Database / SQL

이 장에서는 데이터베이스 다이어그램과 값들의 속성 및 자료형에 대한 기술을 통해 본 시스템의 데이터베이스를 설명한다.

2 Introduction

2.1. Applied Diagram

State diagram, sequence diagram, class diagram 을 중점적으로 사용하여 시스템을 표현하였다.

- State Diagram

  State Diagram 을 이용하여 시스템이 내/외부적 이벤트에 어떻게 반응하는지 표현하였다. 본 명세서에서는 주로 frontend 에 속하는 attribute 와 method 들이 event 에 따라 어떤 방식으로 작동하는지 시각적으로 표현하기 위해 사용하였다.

- Sequence Diagram

  Sequence diagram 을 사용함으로써 시스템을 구성하는 component 간 어떤 상호작용이 어떤 순서로 이루어지는지를 시각화하고 모델링한다. 특히 backend 부분에서 sequence diagram 을 중점적으로 사용하였는데, 이 diagram 을 통해 API 의 use-case 를 파악하고 API 호출 등의 logic 을 모델링하여 표현할 수 있었다.

- Class Diagram

  Class Diagram 은 시스템의 class object 간의 포함 관계와 상호 연결 관계 보여 주는 다이어그램으로 시스템의 정적 구조와 논리적 기능들을 표현하기 위해 작성된다. 하나의 클래스는 이름(Name), 속성 (Attribute), 기능이나 함수(Method)을 가질 수 있다. 본 문서에서는 주로 여러 API 의 기능과 API 와 DB 간 관계를 서술하기 위해 사용되었다.

2.2 System Overview

본 서비스 “Web Coding Test”는 다양한 난이도의 코딩 문제를 제공함으로써 전공자는 물론 비전공자의 소프트웨어 컴퓨팅 역량을 키울 수 있도록 하기 위해 개발된다. 코딩 학습에 대한 사람들의 니즈 증가에 발맞춰 문법이 간단하고 가독성이 좋아 접근성이 높은 프로그래밍 언어인 python 을 사용하는 문제들이 본 서비스에 포함된다. 다양한 이유로 코딩 학습을 원하는 사람들은 그 중 그들 자신이 원하는 문제를 골라 학습할 수 있다. 또한 사용자는 해당 문제와 제약사항에 따라 코드를 작성하고, 그 코드를 저장, 제출할 수 있으며 결과를 검증 받을 수 있다. 코드 제출 후에 사용자에게 제공되는 정보는 사용자 코드에 대해 다측면으로 계산된 점수와 관련 자료 등이 있다. 따라서 사용자는 단순히 코딩 문제를 풀어 정답의 오답 유무를 확인하는 것에서 그치지 않고 본 서비스에서 제공하는 여러 정보로 말미암아 보다 효과적인 코딩 학습을 경험할 수 있다.

사용자의 입장에서 이러한 서비스 기능을 직관적으로 사용할 수 있도록 UX 를 설계하고 그에 맞춰 UI 를 디자인, 구현한다. 목표는 사용자로 하여금 별도의 교육이나 배경지식을 요구하지 않고, 이러한 서비스를 처음 접한 사람도 단시간 내에 쉽게 시스템을 파악하고 여러 기능을 사용할 수 있도록 하는 것이다. 그러기

위해서 논리적이지 않은 인터페이스 배치나 기능 구조 등 UX 에 걸림돌이 될 수 있는 요소들을 포함하지 않는다.

본 서비스는 시스템의 완성 이후 이루어질 수 있는 확장의 가능성을 배제하지 않는다. 그 예시로 python 외의 프로그래밍 언어 코딩 테스트 서비스로도 확장될 가능성을 염두에 두고 설계를 진행한다. 확장의 과정에서 발생 가능한 코드 변동에 예비하여 코드 구조를 명확히 하고, frontend, server, DB 간 이루어지는 상호작용들을 체계적으로 관리한다.

3 System Architecture – Overall

3.1 System Organization

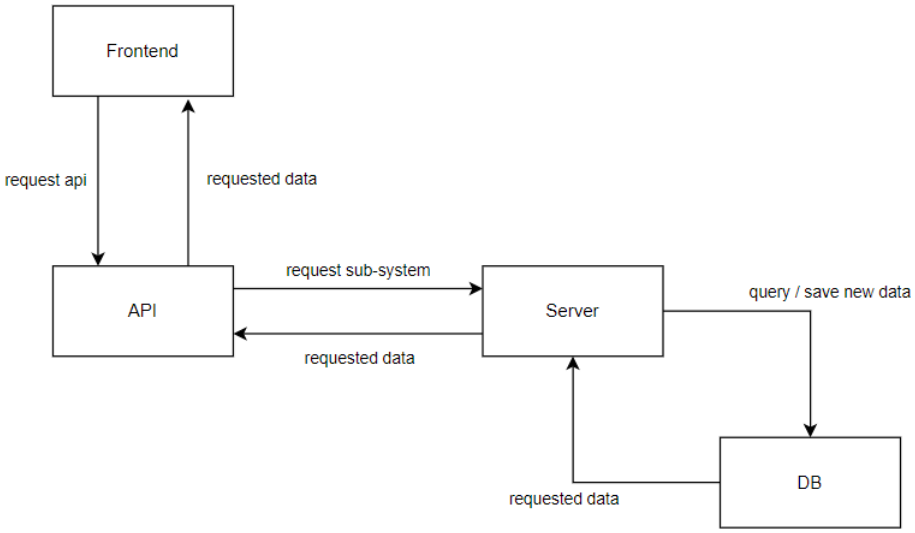

Figure 1 System Organization

본 서비스의 시스템은 크게 frontend 와 backend(Web-server, DB) 부분으로 이루어져 있다. 사용자는 frontend 와 상호작용하며 본 서비스의 기능을 이용할 수 있다. Frontend 는 server 에 정보를 입력하거나 받아올 수 있다. Backend 의 server 를 구성하는 sub-system 들은 작성된 REST API 를 통해 frontend 에 의해 호출된다. Server 는 DB 와 연결되어 있으며 필요시 요청된 정보는 DB 에서 server 측으로 전송된다.

3.2. System Architecture – Frontend Application

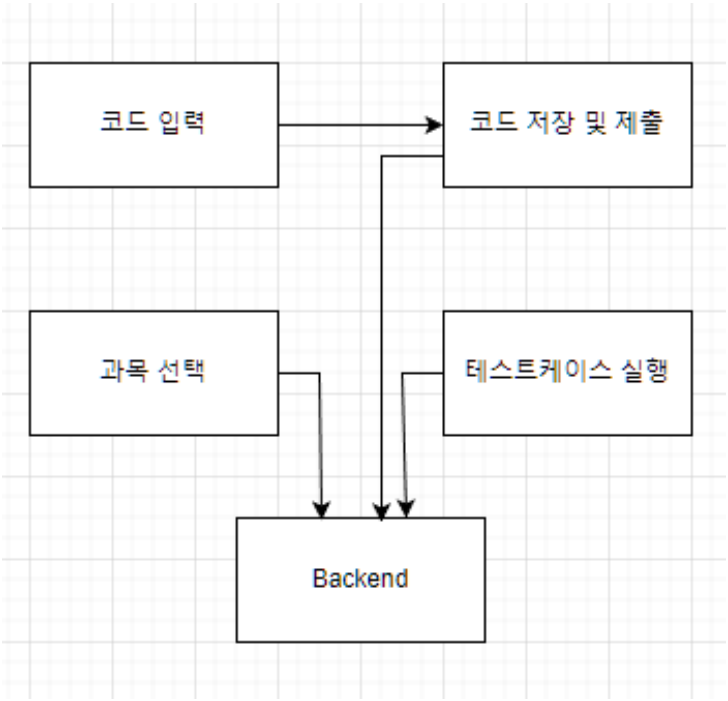

Figure 2 Overall Frontend Architecture

프론트 엔드는 서버와 상호작용 하는 방식으로 작동한다. 사용자가 웹 브라우저를 이용해 HTML, CSS, Javascript 로 구성된 화면에 접근하며, 과목 조회, 테스트케이스 실행, 코드 저장, 코드 제출 등의 기능을 요청할 수 있다. 서버에 데이터를 전달하거나 서버로부터 받아온 데이터를 화면에 나타낼 수 있다.

3.3. System Architecture – Backend

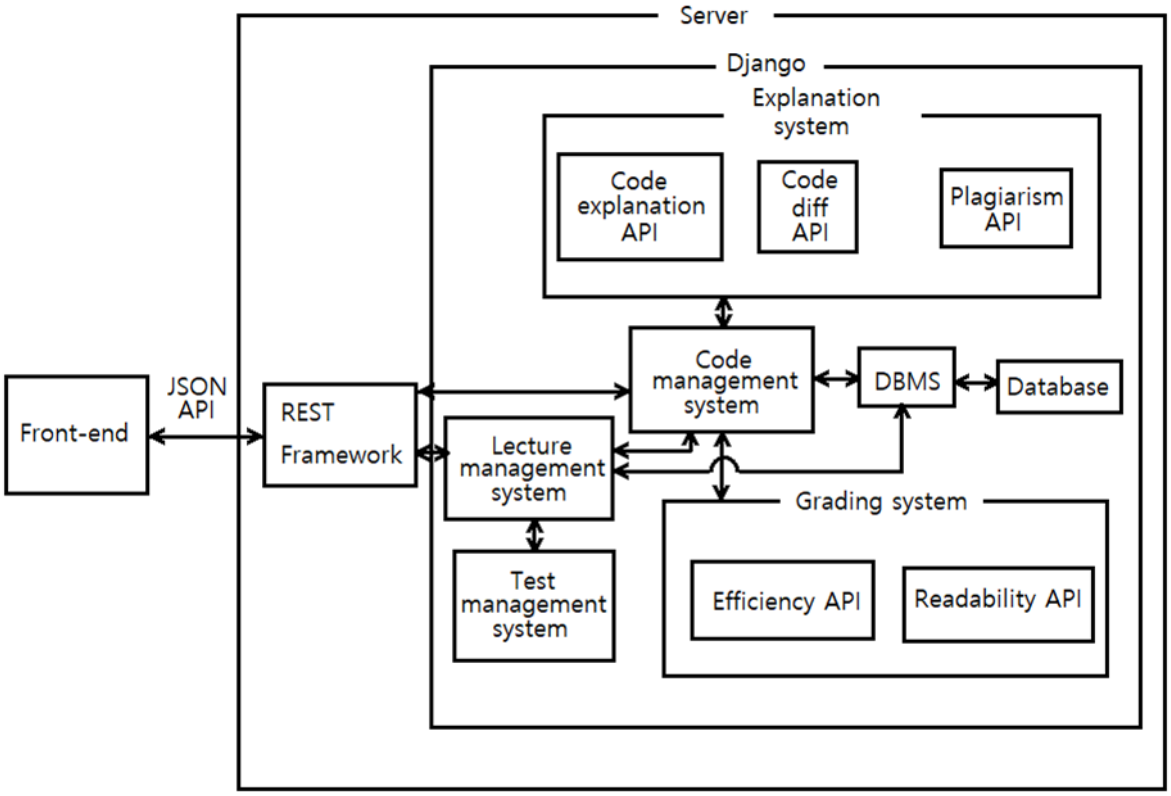

Figure 2. Overall Backend Architecture

Backend 의 Django 는 크게 다섯 개의 sub-system 으로 구성되어 있다. Lecture management system, Code management system, Test management system, Grading system, Explanation system 이 그 다섯 sub-system 에 해당한다.

- Lecture system: lecture 와 question 에 대한 정보를 담당한다.
- Code management system: code 의 저장 및 제출을 담당한다.
- Test management system: testcase 정보와 그것을 이용해 code 를 검증하는 일을 담당한다.
- Grading System: code 의 효율성 및 가독성 채점을 담당한다.
- Explanation system: 기타 code 에 대한 분석 및 설명을 담당한다.

이렇게 구성된 서버는 REST API 를 이용해 frontend 로부터 접근 가능하다. Frontend 에서 서버에 API 를 호출하면 Django 에서 작성된 API 에 따라 해당하는 sub-system 이 호출된다.

4. System Architecture – Frontend

4.1. Objectives

이 장에서는 시스템의 웹 페이지를 실행 상태에 따라 나누어 속성 및 기능에 대해 설명하고, 사용자의 요청과 동작을 기준으로 각 페이지의 state diagram 을 그려 Front-end Architecture 을 설명한다.

4.2. Main page

Web Coding Test 의 메인 페이지로 처음 시스템에 접속하면 볼 수 있는 페이지이다. 이 페이지에서는

과목 선택과 과제 선택 기능을 제공하고, 사용자가 선택한 과제에 대한 정보를 표시하여 사용자가 코드 에디터를 통해 해당 과제에 맞는 코드를 작성할 수 있도록 한다. 또한 초기 메인 페이지로 돌아가는 홈 기능, 설정 기능, 문제의 공개 테스트케이스에 대한 검증 기능, 작성한 코드의 저장, 초기화, 복사, 다운로드 기능 등을 제공한다. 마지막으로 작성한 코드를 실행, 채점, 제출하는 기능을 제공하여 사용자가 코드를 실행하고, 테스트케이스에 대해 점수를 확인하고 코드를 제출할 수 있도록 한다.

4.2.1. Attributes

메인 페이지는 크게 아래의 네 가지 부분으로 구성되어 있다.

- Head: 메인 페이지의 head 부분으로, 시스템을 관리한다.
- QuestionInfo: 과제 정보를 표시한다.
- CodeEditor: 사용자가 코드를 작성할 수 있는 코드 에디터이다.
- Result: 사용자의 코드 실행, 채점, 제출 결과를 표시하고 관리한다.

(1) Head 는 아래의 속성을 가지고 있다.

Home: 홈 버튼이다.

LectureName: 강의 제목이다.

QuestionName: 과제 제목이다.

Deadline: 과제 제출 기한이다.

Settings: 설정 버튼이다.

(2) QuestionInfo 는 아래의 속성을 가지고 있다.

ProblemExplain: 과제 내용에 대해 설명한다.

Requirements: 과제의 참조/제약사항에 대해 설명한다.

Testcase: 과제에 속한 testcase 들을 표시한다.

Test: testcase 를 검증하는 버튼이다.

(3) CodeEditor 는 아래의 속성을 가지고 있다.

SkeletonCode: 과제에 주어지는 기본 코드이다.

Code: 사용자가 작성하는 코드이다.

(4) Result 는 아래의 속성을 가지고 있다.

Result: 사용자가 코드를 실행, 채점, 제출한 결과를 표시한다.

4.2.2. Methods

메인 페이지가 가지고 있는 method 를 파트에 따라 나누면 다음과 같다.

(1) Head 가 가지고 있는 method 는 아래와 같다.

clickHome(): 사용자가 Home 버튼을 눌렀을 때 초기 상태의 메인 페이지로 돌아간다.

setLectureList(): Lecture 데이터 리스트를 가져와 사용자에게 LectureName 을 보여준다.

setSelectedLectureId(): 사용자가 선택한 Lecture 에 속하는 데이터를 가져온다.

setQuestionList(): 사용자가 선택한 Lecture 에 해당하는 Question 데이터 리스트를 가져와 사용자에게 QuestionName 을 보여준다.

setSelectedQuestionId(): 사용자가 선택한 Question 에 속하는 데이터를 가져온다.

clickSettings(): 사용자가 설정 버튼을 눌렀을 때 메인 페이지의 설정을 변경할 수있도록 한다.

(2) QuestionInfo 가 가지고 있는 method 는 아래와 같다.

setProblemExplain(): 사용자가 선택한 QuestionId 에 속하는 ProblemExplain 내용을 가져와 표시한다.

setRequirements(): 사용자가 선택한 QuestionId 에 속하는 Requirements 내용을 가져와 표시한다.

setTestcase(): 사용자가 선택한 QuestionId 에 속하는 공개 Testcase 내용을 가져와 표시한다.

testTestcase(): 사용자가 주어진 공개 testcase 의 검증 버튼을 누르면 사용자가 작성한 코드에 해당 testcase input 값을 넣어 결과를 알려준다.

(3) CodeEditor 가 가지고 있는 method 는 아래와 같다.

runCode(): 사용자가 코드 에디터의 실행 버튼을 누르면 작성중인 코드를 실행하여 실행 결과창에 그 결과가 표시되도록 한다.

gradeCode(): 사용자가 코드 에디터의 채점 버튼을 누르면 작성중인 코드를 과제의 testcase 로 채점하여 채점 결과창에 그 결과가 표시되도록 한다.

submitCode(): 사용자가 코드 에디터의 제출 버튼을 누르면 작성중인 코드를 제출하고, 제출 결과창에 그 결과가 표시되도록 한다.

saveCode(): 사용자가 코드 에디터의 저장 버튼을 누르면 작성중인 코드를 저장한다.

loadCode(): 사용자가 코드 에디터의 불러오기 버튼을 누르면 사용자의 컴퓨터에서 파일을 불러올 수 있도록 한다.

refreshCode(): 사용자가 코드 에디터의 초기화 버튼을 누르면 작성중인 코드가 사라지고 초기 상태의 코드 에디터로 돌아가 SkeletonCode 를 표시한다.

copyCode(): 사용자가 코드 에디터의 복사 버튼을 누르면 작성중인 코드를 클립보드에 복사할 수 있도록 한다.

downloadCode(): 사용자가 코드 에디터의 다운로드 버튼을 누르면 작성중인 코드를 사용자의 컴퓨터에 다운로드할 수 있도록 한다.

(4) Result 가 가지고 있는 method 는 아래와 같다.

showResult(): 사용자가 작성한 코드를 실행, 채점, 제출한 결과가 화면에 표시되도록 한다.

4.2.3. State Diagram

아래는 사용자가 Web Coding Test 를 사용할 때에 해당 메인 페이지의 상태를 그린 state diagram 이다.

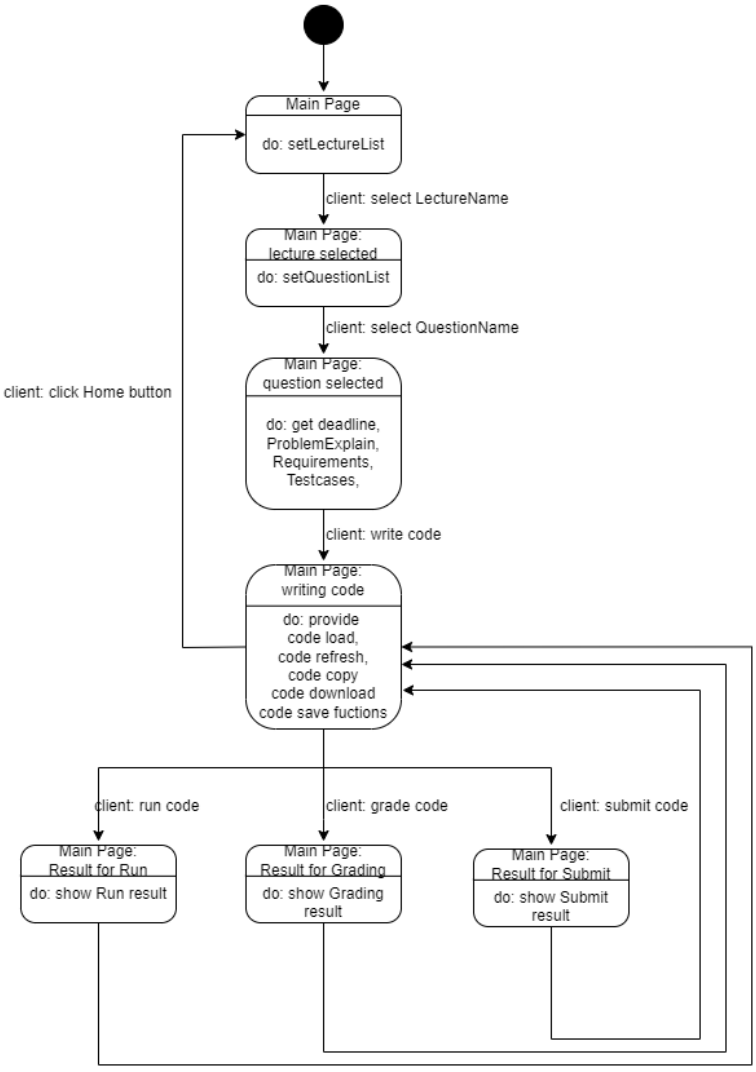

Figure 2. State Diagram of Main Page

사용자가 Web Coding Test 에 접속하면 Main Page 는 Lecture List 를 가져와 사용자가 Lecture Name 을 선택할 수 있도록 한다. 사용자가 Lecture Name 을 선택하면 그 Lecture 에 속해 있는 Question List 를 가져오고, 사용자가 Question Name 을 선택하면 그 Question Id 에 속한 Deadline, Problem

Explain, Requirements, Testcase, Skeleton code 를 가져와 화면에 표시한다.

Question Name 까지 선택되어 과제 정보가 표시되면 사용자는 코드 에디터에 과제에 대한 코드를 작성할 수 있게 된다. 이는 사용자가 코드를 작성중인 writing code 상태로 코드 에디터는 사용자에게 코드 저장, 불러오기, 초기화, 복사, 다운로드 기능을 제공하며 사용자는 또한 코드를 실행, 채점, 제출할 수 있게 된다.

사용자가 작성한 코드를 실행, 채점, 제출하면 해당하는 결과가 Main Page 의 결과창에 표시되며 다시 writing code 상태로 돌아가 기능들을 사용할 수 있도록 한다.

4.3. Main page – Result for Run

사용자가 메인 페이지의 코드 에디터에서 코드를 작성하고 실행 버튼을 눌렀을 때 표시되는 페이지이다. 사용자가 작성한 코드를 실행한 결과를 메인 페이지의 오른쪽 실행 결과 창에서 보여준다. 실행 상태의 결과창의 속성과 method 는 아래와 같다.

4.3.1. Attributes

Result: 사용자가 작성한 코드를 실행한 결과이다.

ErrorMessage: 사용자가 작성한 코드를 실행하여 에러가 발생했을 때 해당 에러의 에러 메시지이다.

4.3.2. Methods

showResult(): 실행한 결과를 실행 결과창에 표시한다.

showError(): 에러가 발생하였을 때 그 위치와 에러 메시지를 표시한다.

4.3.3. State Diagram

아래는 사용자가 코드를 작성하고 실행 버튼을 눌렀을 때의 메인 페이지 상태를 그린 state diagram 이다.

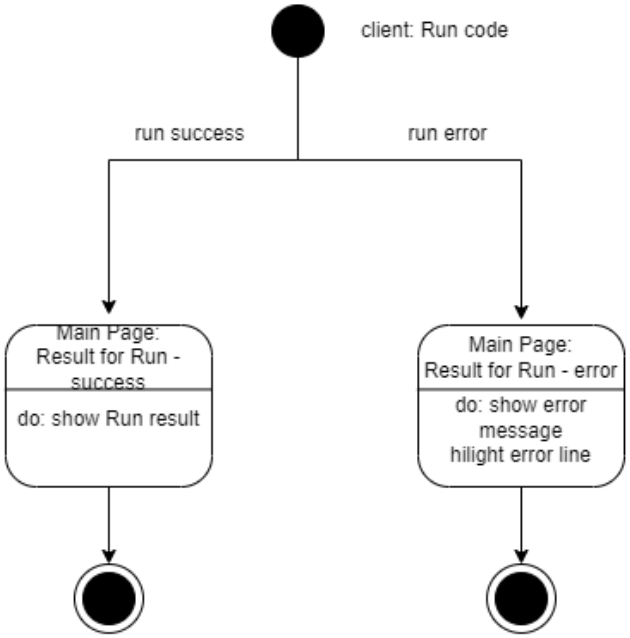

Figure 2. State Diagram of Main Page – Run Code

사용자가 코드 실행 버튼을 누르면 작성한 코드를 실행하여 성공적으로 실행될 경우 그 결과를, 에러가 발생할 경우 error message 를 제공하고 에러가 발생한 코드 라인을 하이라이트 표시한다. 이처럼 실행에 대한 결과를 표시한 후 실행 상태가 종료되고 다시 코드 작성 상태가 된다.

4.4. Main Page – Result of Grading

사용자가 메인 페이지의 코드 에디터에 코드를 작성하고 채점 버튼을 눌렀을 때 표시되는 페이지이다. 메인 영역 중 우측에 풀이 코드를 채점한 결과를 출력한다. 채점 결과는 총점과 각 테스트케이스에 대한 실행 결과로 구성되어 있다. 각 테스트케이스에 대한 실행 결과는 Pass/Fail 여부로 제공되고, 이때 테스트케이스는 오픈, 히든 테스트케이스를 모두 사용한다. 오픈 테스트케이스가 Fail 한 경우 테스트케이스 정보를 제공하고, 히든 테스트케이스가 Fail 한 경우 테스트케이스 정보를 제공하지 않는다. 채점 결과 창의 속성과 메소드는 다음과 같다.

4.4.1. Attributes

Score: 사용자가 작성한 코드를 채점한 결과 중 총점에 해당한다.

Result: 각 테스트케이스에 대한 실행 결과에 해당한다. 오픈 테스트케이스가 Fail 한 경우 테스트케이스 정보를 함께 저장한다.

ErrorMessage: 사용자가 작성한 코드를 채점하기 위해 실행하는 도중 에러가 발생한 경우 저장되는 에러 메시지에 해당한다.

4.4.2. Methods

showScore(): 총점을 채점 결과 창에 표시한다.

showResult(): 각 테스트케이스에 대한 실행 결과를 채점 결과 창에 표시한다.

showError(): 에러가 발생했을 때 에러 메시지를 표시한다.

4.4.3. State Diagram

다음은 사용자가 코드를 작성하고 채점 버튼을 눌렀을 때의 메인 페이지 상태를 그린 state diagram 이다.

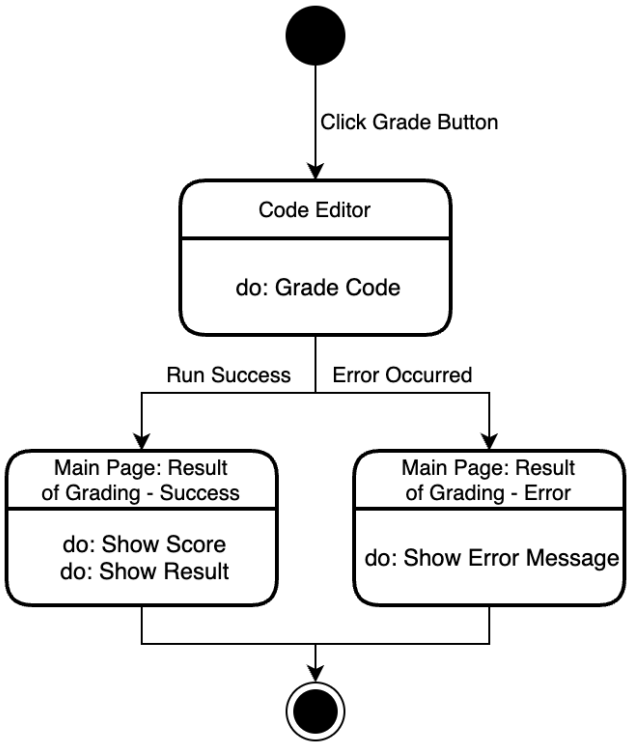

Figure 2. State Diagram of Main Page – Result of Grading

사용자가 채점 버튼을 누르면 작성한 코드를 실행하고, 채점 결과인 총점과 각 테스트케이스에 대한 실행 결과를 생성한다. 채점에 성공한 경우 그 결과를, 에러가 발생한 경우 error message 를 제공한다. 채점을 마치면 다시 코드 작성 상태로 복귀한다.

4.5. Main Page – Result of Submit

사용자가 메인 페이지의 코드 에디터에 코드를 작성하고 제출 버튼을 눌렀을 때 표시되는 페이지이다. 메인 영역 중 우측에 제출한 풀이 코드를 분석한 결과를 출력한다. 사용자가 제출한 세 개의 풀이 코드를 정답 코드와 비교한 Diff 결과와 풀이 코드의 채점 결과, 코드 설명, 관련 자료를 제공한다. 채점 결과는 기능 점수, 효율 점수, 가독성 점수와 이를 종합한 총점으로 나누어 제공되고, 관련 자료는 문제와 관련된 추천 영상 및 학습 자료를 제공한다. 제출 결과 창의 속성과 메소드는 다음과 같다.

4.5.1. Attributes

Diff: 사용자가 제출한 세 개의 풀이 코드를 정답 코드와 비교한 Diff 결과에 해당한다.

AnalysisResult: 풀이 코드의 채점 결과로, 기능 점수, 효율 점수, 가독성 점수와 이를 종합한 총점으로 구성되어 있다.

CodeDescription: 풀이 코드의 설명에 해당한다.

Reference: 풀이 코드의 관련 자료에 해당한다.

ErrorMessage: 사용자가 작성한 코드를 제출하기 위해 실행하는 도중 에러가 발생한 경우 저장되는 에러 메시지에 해당한다.

4.5.2. Methods

showDiff(): Diff 결과를 코드 에디터 대신 표시한다.

showAnalysisResult(): 풀이 코드의 채점 결과를 제출 결과 창에 표시한다.

showCodeDescription(): 풀이 코드의 설명을 제출 결과 창에 표시한다.

showReference(): 풀이 코드의 관련 자료를 제출 결과 창에 표시한다.

showError(): 에러가 발생했을 때 에러 메시지를 표시한다.

4.5.3. State Diagram

다음은 사용자가 코드를 작성하고 제출 버튼을 눌렀을 때의 메인 페이지 상태를 그린 state diagram 이다.

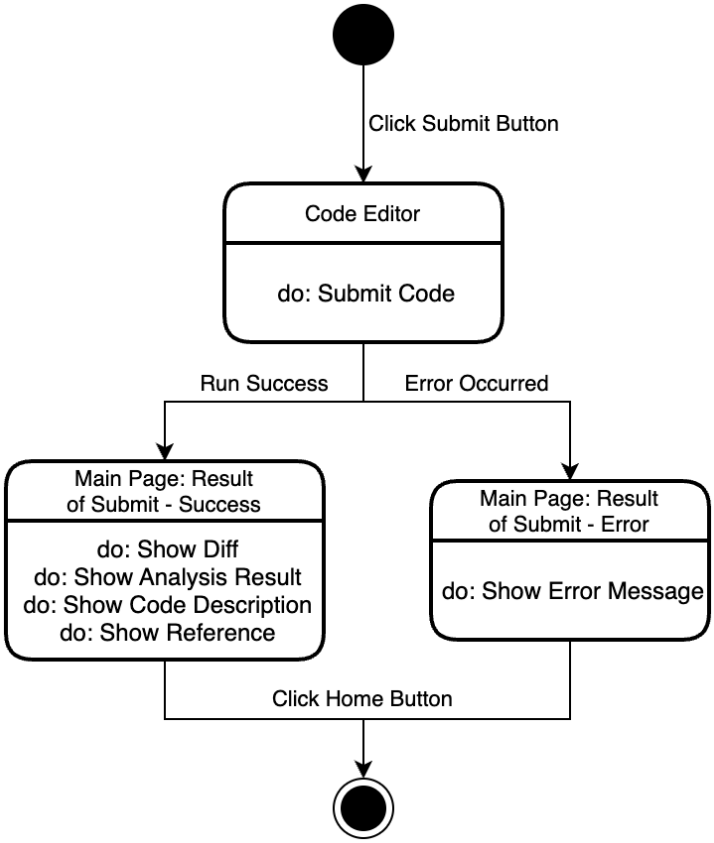

Figure 2. State Diagram of Main Page – Result of Submit

사용자가 제출 버튼을 누르면 작성한 코드를 실행 및 저장하고, 사용자가 제출한 세 개의 풀이 코드를 정답 코드와 비교한 Diff 결과와 풀이 코드의 채점 결과, 코드 설명, 관련 자료를 생성한다. 제출에 성공한 경우 그 결과를, 에러가 발생한 경우 error message 를 제공한다. 제출 결과를 확인한 후 홈 버튼을 누르면 다시 코드 작성 상태로 복귀한다.

5. System Architecture – Backend

5.1. Objectives

Back-end 의 전반적인 구조와 세부 시스템 구조를 설명한다. 각각의 sub-system 은 class diagram 을 통해 객체 간의 관계를 확인하고 sequence diagram 을 통해 데이터의 흐름을 파악한다. 각 DB 에 대한 자세한 설명은 7 장에서 확인할 수 있다.

5.2. Overall Architecture

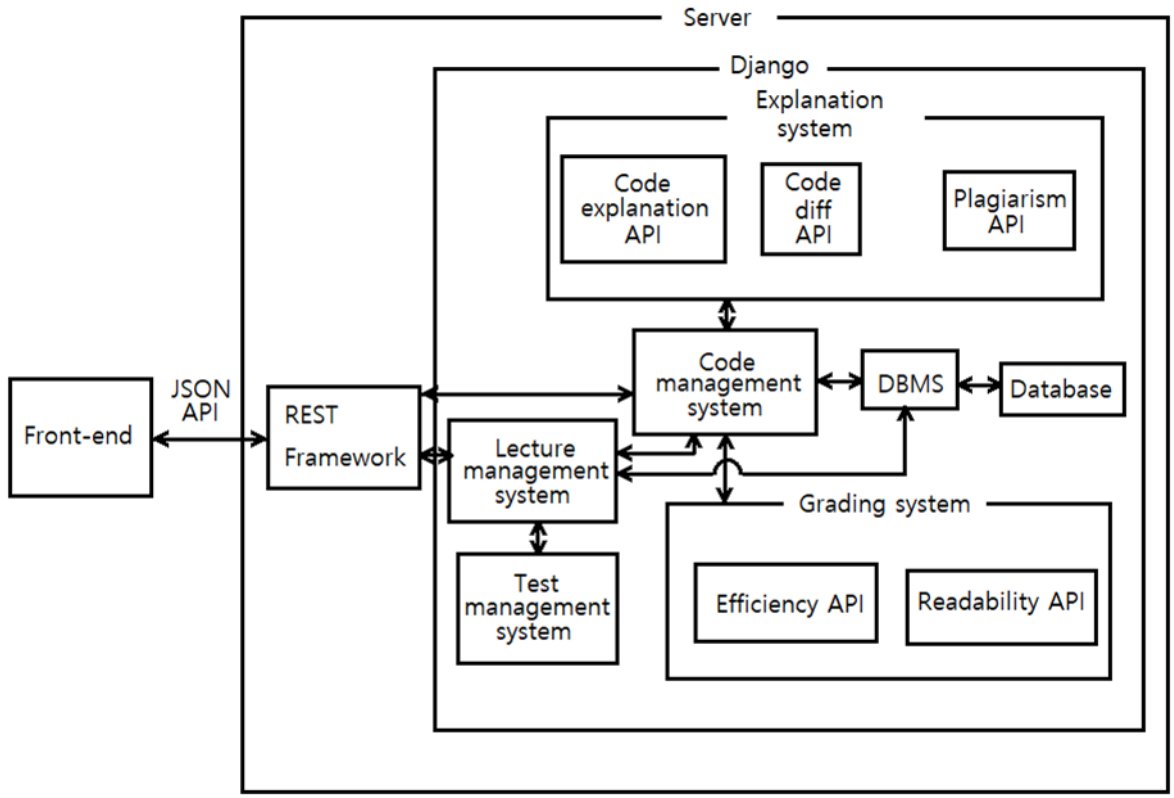

Figure 3. Overall Backend Architecture

Back-end 의 전체 구조는 다음과 같다. 먼저 Front-end 에서 서버에 API 를 호출하면 Django 에서 작성한 REST Framework 에 따라 sub-system 이 호출된다. Django 는 총 5 개의 sub-system 으로 구성되어 있다. lecture 와 question 을 보여주는 Lecture Management System, code 저장 및 제출을 담당하는 Code Management System, testcase 를 통해 code 를 검증하는 Test Management System, code 의 효율성 및 가독성을 채점하는 Grading System, 기타 code 에 대한 분석 및 설명을 하는 Explanation System 으로 구성되어 있다.

5.3. Subcomponents

5.3.1. Lecture Management System

Lecture 와 Question 을 선택하고 보여주는 작업을 하는 시스템이다.

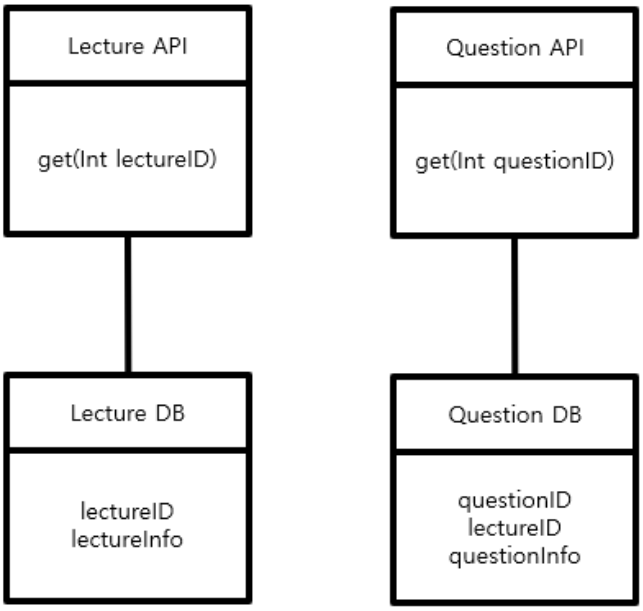

Figure 2. Lecture Management System Class Diagram

- Lecture API Class

Front-end 로부터 REST 를 통해 GET 요청을 받으면 Lecture DB 의 정보들을 가져와서 보여주는 역할을 한다. 해당 클래스에는 한 개의 메소드가 존재한다. 이는 parameter 로 lectureID 를 받아서 원하는 lecture 정보를 Lecture DB 에서 가져오는 get() 이다.

- Question API Class

Front-end 로부터 REST 를 통해 GET 요청을 받으면 Question DB 의 정보들을 가져와서 보여주는 역할을 한다. 해당 클래스에는 한 개의 메소드가 존재한다. 이는 parameter 로 questionID 를 받아서 원하는 question 정보를 Question DB 에서 가져오는 get() 이다.

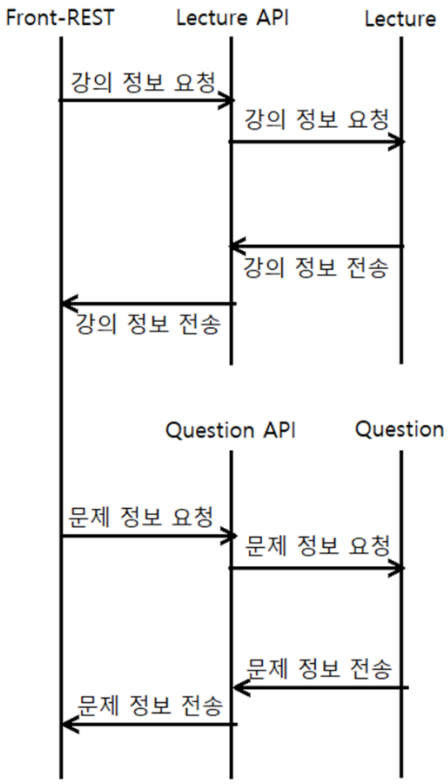

Figure 2. Lecture Management System Sequence Diagram

5.3.2. Code Management System

코드의 저장과 제출 및 코드를 활용하는 다른 시스템들에게 코드를 제공하는, 전반적인 코드의 관리를 담당하는 시스템이다.

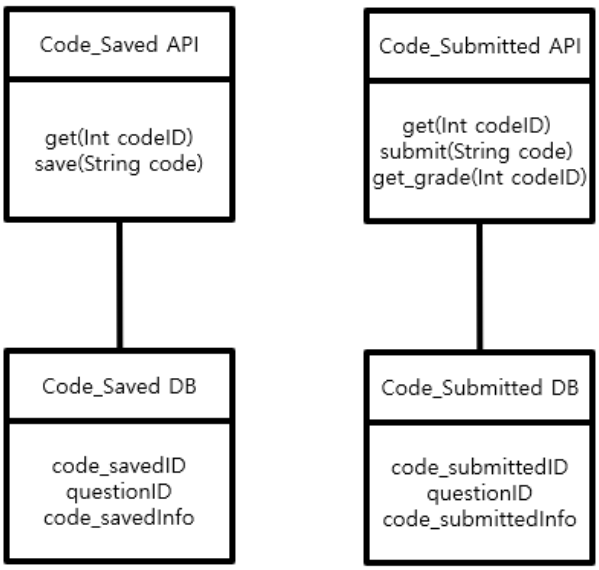

Figure 2. Code Management System Class Diagram

- Code_Saved API

저장할 코드를 받아 Code_Saved DB 에 저장하거나 저장된 코드를 불러와 보여주는 역할을 한다. 해당 클래스에는 두 개의 메소드가 존재한다. 첫 번째는 parameter 로 code_savedID 를 받아서 원하는 저장된 코드 정보를 Code_Saved DB 에서 가져오는 get() 이다. 두 번째는 parameter 로 code 를 받아서 Code_Saved DB 에 저장하는 save() 이다.

- Code_Submitted API

제출한 코드를 Code_Submitted DB 에서 가져와 보여주거나 제출할 코드를 받아 제출 횟수를 판단 후 저장하거나 반려하는 역할을 한다. 또한, 제출된 코드를 바탕으로 Grading Management System 에게 채점을 요청할 수 있다. 해당 클래스에는 세 개의 메소드가 존재한다. 첫 번째는 parameter 로 code_submittedID 를 받아서 원하는 제출된 코드 정보를 Code_Submitted DB 에서 가져오는 get() 이다. 두 번째는 parameter 로 code 를 받아서 제출 횟수 고려 후 Code_Submitted DB 에 저장하는 submit() 이다. 세 번째는 parameter 로 code_submittedID 를 받아서 해당 코드를 Grading Management System 에게 채점을 요청하는 get_grade() 이다.

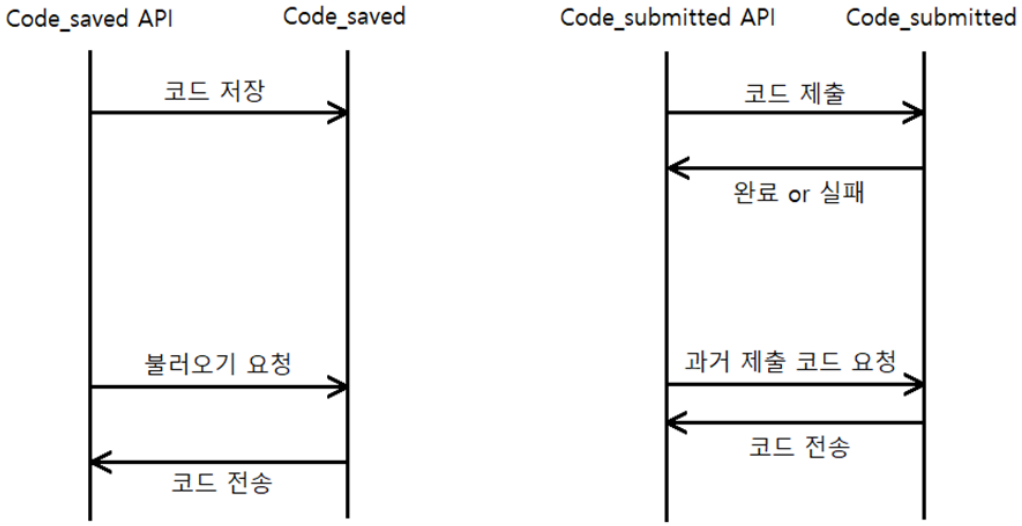

Figure 2. Code Management System Sequence Diagram

5.3.2. Grade Management System

코드의 효율성과 가독성을 채점해 주는 시스템이다.

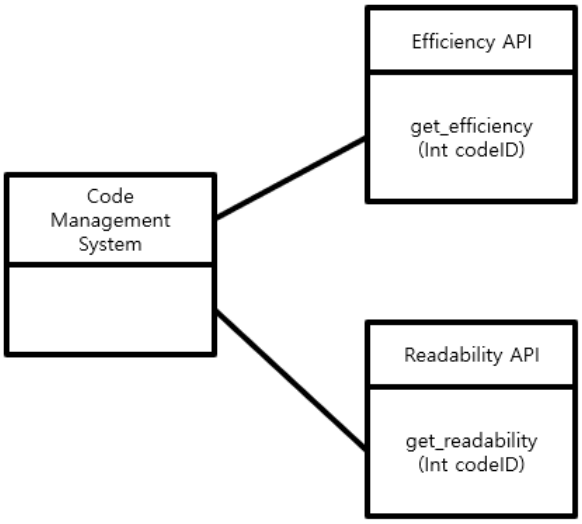

Figure 2. Grade Management System Class Diagram

- Efficiency API

Code Management System 으로부터 코드와 효율성 검증 요청을 받아 효율성 채점 결과를 보여주는 역할을 한다. 해당 클래스에는 한 개의 메소드가 존재한다. 이는 parameter 로 codeID 를 받아서 효율성 채점 및 결과를 보여주는 get_efficiency() 이다.

- Readability API

Code Management System 으로부터 코드와 가독성 검증 요청을 받아 가독성 채점 결과를 보여주는 역할을 한다. 해당 클래스에는 한 개의 메소드가 존재한다. 이는 parameter 로 codeID 를 받아서 가독성 채점 및 결과를 보여주는 get_readability() 이다.

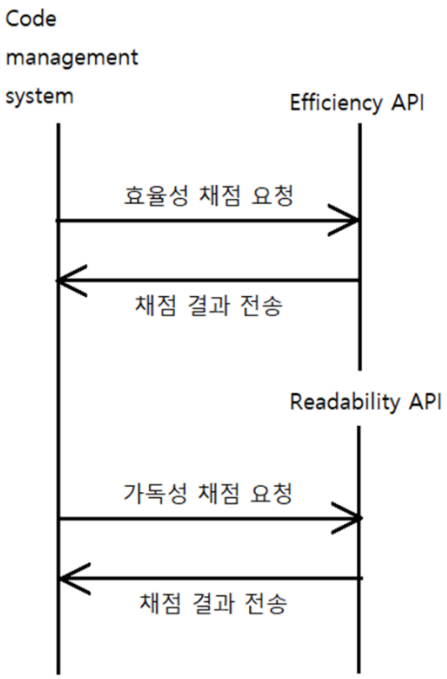

Figure 2. Grade Management System Sequence Diagram

5.3.4. Test Management System

Testcase 에 대한 정보를 출력해주거나 Code management system 으로부터 code 를 받아 테스트케이스로 검증을 해 주는 시스템이다.

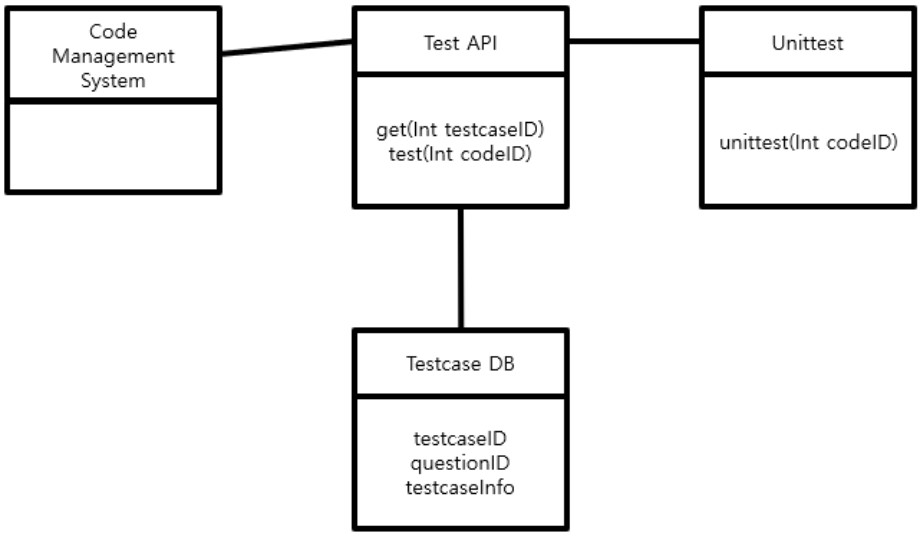

Figure 2. Test Management System Class Diagram

- Test API Class

Lecture Management System 으로부터 testcase 에 대한 GET 요청이 들어오면 원하는 testcase 를 Testcase DB 에서 가져와 보내주는 역할을 한다. 또한, Code Management System 으로부터 검증 요청이 들어오면 받은 코드 정보를 바탕으로 unittest 에 검증 요청을 보내고 그 결과를 되돌려 보내준다. 해당 클래스에는 두 개의 메소드가 존재한다. 첫 번째는 parameter 로 testcaseID 를 받아서 원하는 testcase 정보를 Testcase DB 에서 가져오는 get() 이다. 두 번째는 parameter 로 codeID 를 받아서 해당 코드를 Unittest 에게 채점을 요청하는 test() 이다.

- Unittest Class

해당 클래스에는 한 개의 메소드가 존재한다. Test API 로부터 codeID 를 받고 이를 통해 해당 question 에 속하는 testcase 를 가져와 테스트케이스 검증을 한 후 결과를 TestAPI 로 보내주는 unittest() 이다.

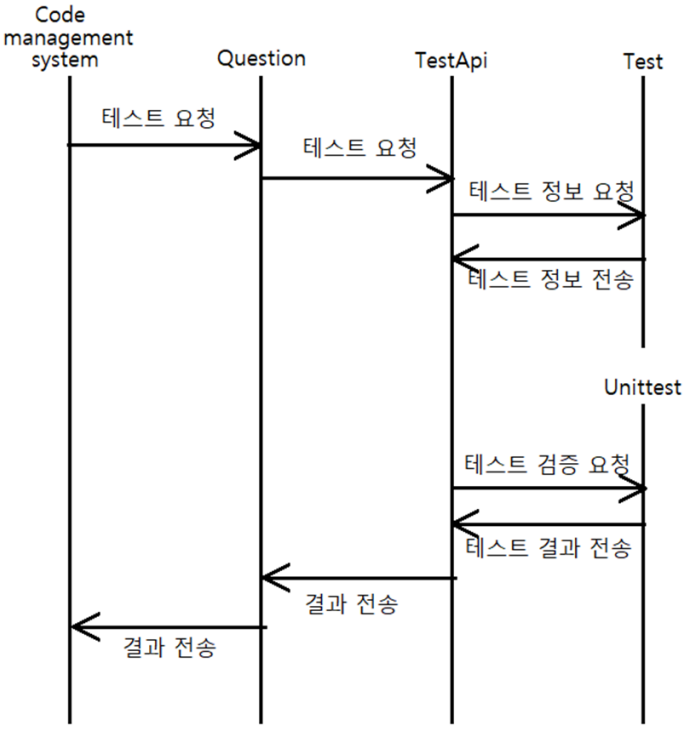

Figure 2. Test Management System Sequence Diagram

5.3.5. Explanation System

Code management system 과 코드를 주고받으며 제출한 코드에 대한 설명과, 정답 코드와의 차이, 표절 검사를 하는 시스템이다.

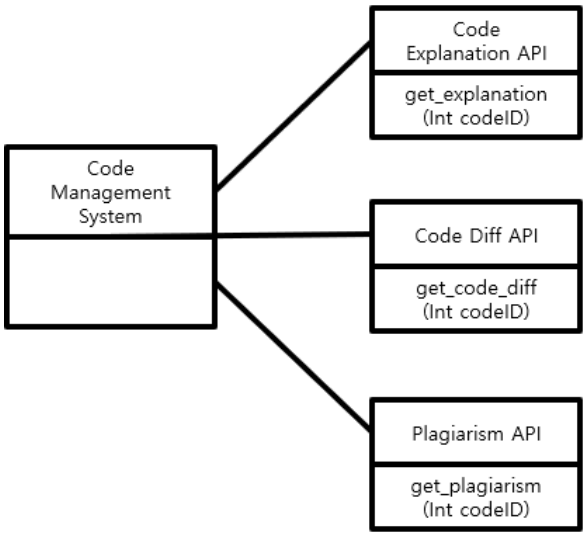

Figure 2. Explanation System Class Diagram

- Code Explanation API

Code Management System 으로부터 코드와 코드 설명 요청을 받아 코드에 대한 설명을 보여주는 역할을 한다. 해당 클래스에는 한 개의 메소드가 존재한다. 이는 parameter 로 codeID 를 받아서 코드에 대한 설명을 보여주는 get_explanation() 이다.

- Code Diff API

Code Management System 으로부터 코드와 코드 비교 요청을 받아 코드와 정답 코드 사이의 차이를 보여주는 역할을 한다. 해당 클래스에는 한 개의 메소드가 존재한다. 이는 parameter 로 codeID 를 받아서 코드와 정답 코드 사이의 차이를 보여주는 get_code_diff() 이다.

- Plagiarism API

Code Management System 으로부터 코드와 표절 검사 요청을 받아 코드와 정답 코드 사이의 표절률을 보여주는 역할을 한다. 해당 클래스에는 한 개의 메소드가 존재한다. 이는 parameter 로 codeID 를 받아서 코드와 정답 코드 사이의 표절률를 보여주는 get_plagiarism() 이다.

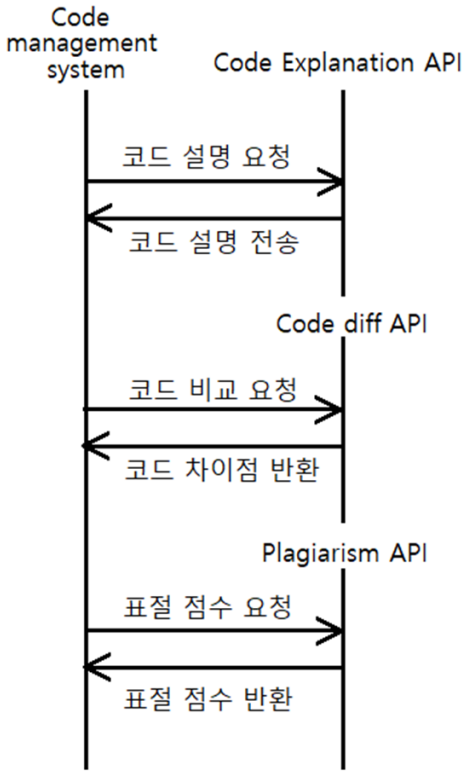

Figure 2. Explanation System Sequence Diagram

6. Protocol

6.1. HTTP

클라이언트-서버 프로토콜이며, 각각의 개별적인 요청은 서버로 보내지고, 서버는 요청을 처리한 후에 response 라고 불리는 응답을 제공한다. 웹에서 이루어지는 모든 데이터 교환의 기초로, HTML 문서와 같은 리소스들을 가져올 수 있도록 하는 프로토콜이다.

6.2. JSON

객체 문법으로 구조화된 데이터를 표현하기 위한 문자 기반의 표준 포맷이다. 속성-값 쌍, 배열 자료형 등 시리얼화 가능한 값 또는 "키-값 쌍"으로 이루어진 데이터 오브젝트를 전달하기 위해 인간이 읽을 수 있는 텍스트를 사용한다. 특히, 인터넷에서 자료를 주고받을 때 그 자료를 표현하는 방법으로 알려져 있다.

6.3. Protocol Description

6.3.1. 강의 조회 프로토콜

GET	lecture/<id>	
사용자가 강의를 조회할 때 요청하는 주소		
분류	항목명	설명 및 제약사항
Parameter	id	강의 id 값
Response	lectureId	강의 id 값
	lectureName	강의 제목
Status Code	200	불러오기 성공
	400	존재하지 않음

Table 1. 강의 조회 프로토콜 표

6.3.2. 문제 조회 프로토콜

GET	question/<id>	
사용자가 문제를 조회할 때 요청하는 주소		
분류	항목명	설명 및 제약사항
Parameter	id	문제 id 값
Response	questionId	문제 id 값
	lectureId	문제 제목
	skeletonCode	스켈레톤 코드
	questionName	문제 이름
	deadline	제출 기한
	Requirements	요구사항

Status Code	200	불러오기 성공
	400	존재하지 않음

Table 2. 문제 조회 프로토콜 표

6.3.3. 테스트케이스 실행 프로토콜

GET	testcase/<id>	
사용자가 테스트케이스를 실행할 때 요청하는 주소		
분류	항목명	설명 및 제약사항
Parameter	testcaseId	테스트케이스 id 값
	QuestionId	문제 id 값
Response	Input	테스트케이스 인풋값
	output	테스트케이스 아웃풋값
Status Code	200	불러오기 성공
	400	존재하지 않음

Table 3. 테스트케이스 실행 프로토콜 표

6.3.4. 코드 저장 프로토콜

POST	code_saved/<question_id>	
사용자가 코드를 저장할 때 요청하는 주소		
분류	항목명	설명 및 제약사항
Parameter	Code_savedId	저장 코드 id 값
	questionId	문제 id 값
Response	message	“코드 저장 실패”
Status Code	201	저장 성공
	400	저장 실패

Table 4. 코드 저장 프로토콜 표

6.3.5. 저장 코드 불러오기 프로토콜

GET	code_saved/<question_id>/<id>	
사용자가 저장된 코드를 조회할 때 요청하는 주소		
분류	항목명	설명 및 제약사항
Parameter	Code_savedId	저장 코드 id 값
	questionId	문제 id 값

Response	Code	저장된 코드 내용
Status Code	200	불러오기 성공
	400	불러오기 실패

Table 5. 저장 코드 불러오기 프로토콜 표

6.3.6. 코드 제출 프로토콜

POST	Code_submitted/<question_id>	
사용자가 코드를 제출할 때 요청하는 주소		
분류	항목명	설명 및 제약사항
Parameter	Code_savedId	제출 코드 id 값
	QuestionId	문제 id 값
Response	message	“코드 제출 실패”
Status Code	201	코드 제출 성공
	400	코드 제출 실패

Table 6. 코드 제출 프로토콜 표

7. database Design

7.1. Objectives

Database design 의 전반적인 부분에 대해 설명한다. ER diagram 을 통해 Entity 와 Relationship 을 설명하고, Relational Schema 를 통해 서로의 관계를 설명한다. 또, django 의 models.py 의 코드를 직접 보여줌으로써 어떻게 구현이 되어 있는지 설명한다.

7.2. ER Diagram

해당 시스템에는 Lecture, Question, Testcase, Code_Saved, Code_Submitted 총 5 개의 Entity 가 존재한다. ER diagram 에서 Entity 는 초록색 직사각형, Entity 간의 관계는 파란색 마름모와 숫자로 표현했다. Entity 가 가지는 Attribute 는 주황색 타원형으로 표현했으며, Primary key 는 빨간색 타원형에 밑줄을 그어 표현했다.

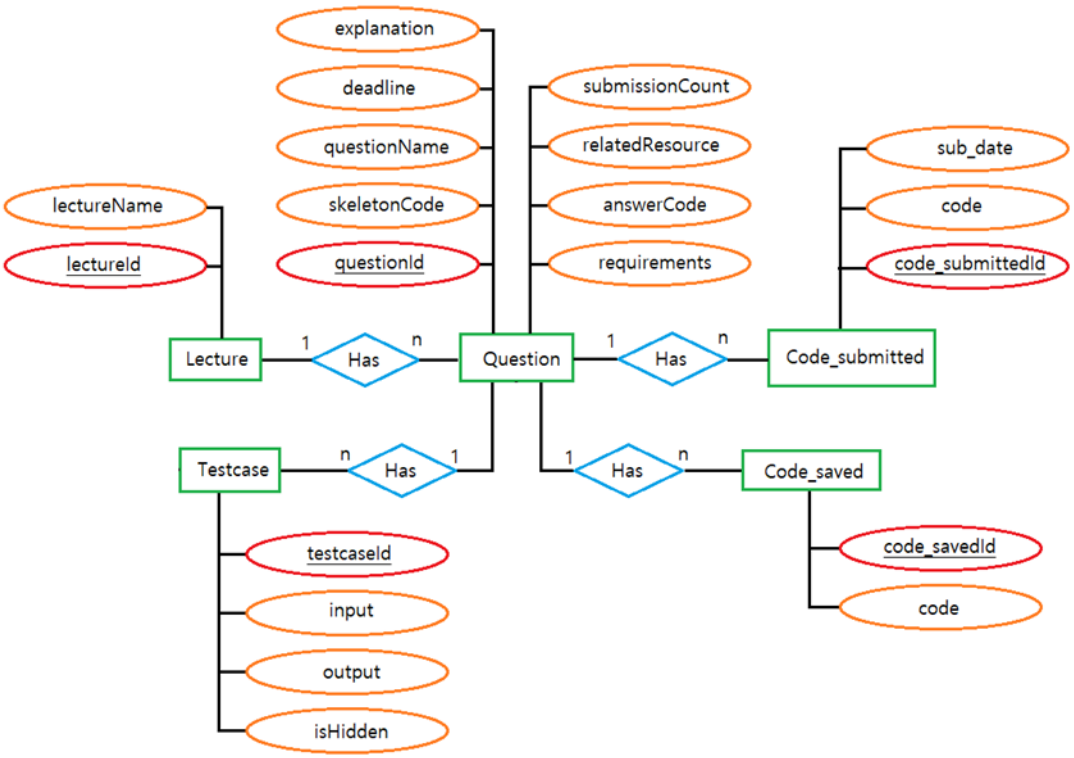

Figure 2. ER Diagram

7.3. Relational Schema

DB 의 Relational Schema diagram 이다. 각 class 와 서로의 관계에 대해 볼 수 있다. Primary key 는 붉은색, Foreign key 는 푸른색으로 표시되어 있다. 또한 Foreign key 가 참조하는 Primary key 는 화살표를 통해 표현되어 있다.

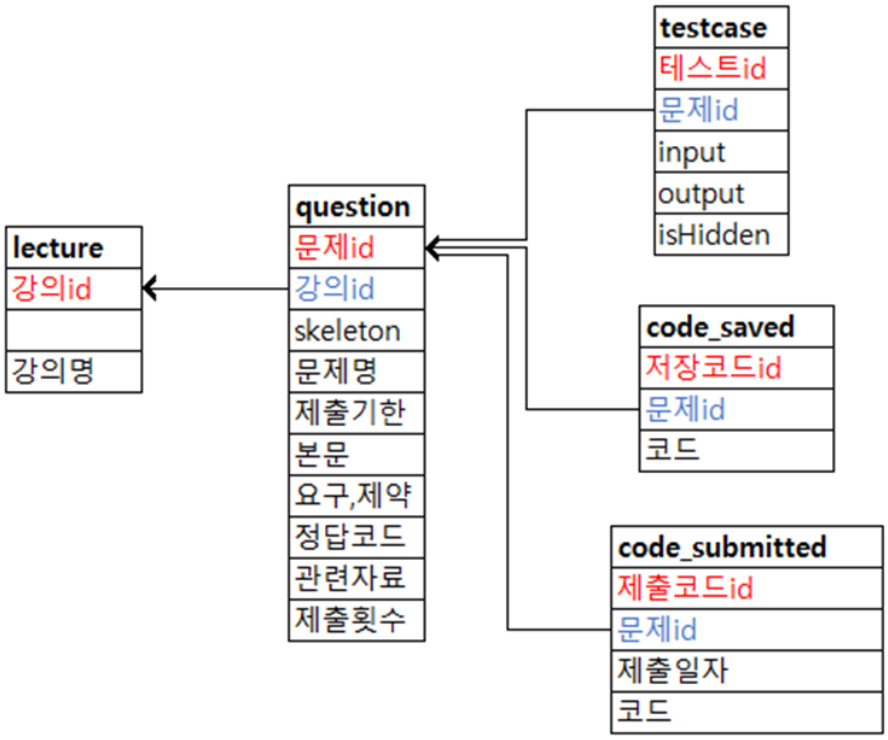

Figure 2. Relational Schema

7.4. Models.py

Django 내의 models.py 에 저장된 코드이다. 이는 자동으로 sqlite DB 에 저장된다.

7.4.1. Lecture

```
class Lecture(models.Model):
    lectureId = models.IntegerField(primary_key = True)
    lectureName = models.TextField(default='')
```

7.4.2. Question

```
class Question(models.Model):
    questionId = models.IntegerField(primary_key = True)
    lectureId = models.ForeignKey(Lecture, on_delete = models.CASCADE)
    skeletonCode = models.TextField(default = '')
    questionName = models.CharField(max_length = 500, default = '')
    deadline = models.DateField(null = True)
    problemExplain = models.TextField(default = '')
    requirements = models.TextField(default = '')
    answerCode = models.TextField(default = '')
    relatedResource = models.TextField(default = '')
```

```
submissionCount = models.IntegerField(default = 0)
```

7.4.3. Testcase

```
class Testcase(models.Model):
    testcaseId = models.IntegerField(primary_key = True)
    questionId = models.ForeignKey(Question, on_delete = models.CASCADE)
    input = models.TextField(default = '')
    output = models.TextField(default = '')
    isHidden = models.BooleanField(default = False)
```

7.4.4. Code_Saved

```
class Code_Saved(models.Model):
    code_savedId = models.IntegerField(primary_key = True)
    questionId = models.ForeignKey(Question, on_delete = models.CASCADE)
    code = models.TextField(default = '')
```

7.4.5. Code_Submitted

```
class Code_Submitted(models.Model):
    code_submittedId = models.IntegerField(primary_key = True)
    questionId = models.ForeignKey(Question, on_delete = models.CASCADE)
    sub_date = models.DateField(null = True)
    code = models.TextField(default = '')
```