Design Specification

소프트웨어 공학개론 **9**조

제출일	22.11.13	그룹	9조
과목	소프트웨어공학개론	담당교수	이은석 교수님
이름	위성은	학번	2016314668
이름	김수겸	학번	2017311745
이름	구자현	학번	2017314006
이름	강민구	학번	2017314806
이름	연민석	학번	2018312322
이름	이원규	학번	2019315505

목차

1. Preface

1.1 Readership

본 문서는 시스템의 디자인을 설명하고 시스템 내부의 동작과 상호작용을 자연어와 다이어그램으로 서술하였다. 주 독자는 해당 시스템의 개발팀이 되며 본 문서에 내용에 따라 개발이 진행될 예정이다.

1.2 Version history

2022.11.05, v 0.1.0 회의 후 최초버전 제작

2022.11.07 v 0.2.0 보충 자료 추가

2022.11.08 v 0.3.0 맞춤법 검수

2022.11.11 v 0.4.0 추가 자료 삽입

2022.11.12 v 1.0.0 최종 검수 후 최종 버전 완성

1.3 Document structure

Introduction

이 장에서는 본 시스템에 대해 설명하고 시스템의 개발 아키텍처와 전체적인 design specification에 대해 소개한다.

System Architecture - Overall

시스템의 전체적인 아키텍처를 설명한다. 아키텍처 다이어그램으로 보기 쉽게 시스템을 설명하며 프론트엔드와 백엔드 그리고 데이터베이스를 아우르는 전체적인 시스템의 모습을 파악한다.

System Architecture – Frontend

시스템 아키텍처 중에서도 특히 프론트엔드 파트의 아키텍처에 대해 살펴본다. 프론트엔드 시스템을 서브 시스템으로 나누어 속성과 기능을 서술하고 각 서브시스템의 class 다이어그램과 sequence 다이어그램을 살펴본다.

System Architecture - Backend

시스템 아키텍처 중에서도 특히 백엔드 파트의 아키텍처의 대해 살펴본다. 앱간의 상호작용과 기능들을 다양한 다이어그램들을 통해 살펴본다.

Protocol

프론트엔드와 백엔드 사이에서 일어나는 API통신 프로토콜들을 살펴본다. Rest API형식을 따르는 해당 시스템에서 어떠한 프로토콜들이 있는지 서술한다.

Database Design

관계형 데이터베이스 다이어그램과 이를 표현한 SQL을 서술한다. 또한, 각 데이터베이스 값의 속성과 자료형을 나타낸다.

2. Introduction

이 장에서는 본 시스템에 대해 설명하고 시스템의 개발 아키텍처와 전체적인 design specification에 대해 소개한다.

2.1 Applied 다이어그램

시스템을 설명하기 위해 Sequence 다이어그램, Entity-relationship 다이어그램, Class 다이어그램을 사용하였다.

Sequence 다이어그램

특정 행동이 어떠한 순서로 어떤 객체와 어떻게 상호작용을 하는 지 시간순으로 나타내며 시스템이 어떠한 시나리오로 움직이는 지를 보여주는 데 유용하다. Sequence 다이어그램을 사용함으로써 API 등으 use-case를 상세하게 알 수 있으며 method call, 데이터베이스 조회, API 호출 등 로직을 모델링할 수 있다.

Entity-relationship 다이어그램

시스템을 구성하는 Entity에 대한 세부사항과 각 Entity간의 상호 작용과 그 관계를 보여주는 다이어그램으로 이를 통해 프로젝트에서 사용되는 데이터베이스의 구조를 이해하는 데 도움을 주며 필요한 API를 효율적으로 생성할 수 있다.

Class 다이어그램

시스템에서 사용되는 객체 타입인 Class를 정의하고 Class 간의 정적인 상호 작용 관계를 표현한 다이어그램으로 시스템의 전체적인 구조를 설명하기에 유용하다. 시스템의 분석 단계와 설계 단계에서 여러 번 작성되며 여러 번에 걸쳐 작성되면서 상세화된다.

2.2 System overview

본 시스템은 성균관대학교 학생들의 알고리즘 문제 풀이를 위한 코딩 플랫폼 사이트이다. 최근 많은 기업에서 코딩테스트를 진행하고 있으며 현업에서 문제를 효율적으로 해결하기 위해 알고리즘 역량은 선택이 아닌 필수가 되었다. 따라서 학생들이 자신이 수강하는 과목과 과제를 선택한 후, 문제를 해결하고 관련 자료를 학습하는 과정을 통해서 알고리즘 역량을 향상시키기 위해 기획하였다.

본 시스템은 Virtual DOM의 특징을 가진 React를 사용한 웹사이트를 구축하여 사용자에게 변화하는 환경을 더 빨리 제공할 수 있으며 컴포넌트화를 통해 재사용이 가능하다. 또한, Django을 통해 백엔드를 구현하였고 sqlite를 사용하여 사용자의 정보를 저장하고 조회한다.

사용자의 만족도를 높이기 위해 3단계의 저장 기능을 제공하며, 이에 따라 사용자는 다양한 알고리즘을 구현하여 결과를 얻을 수 있다. 또한, 자동저장 기능을 제공함으로써, 사용자가 의도되지 않은 행동으로 브라우저를 종료했을 때도 자신의 코드를 복원하고 확인할 수 있다.

3. System Architecture – Overall

시스템의 전체적인 아키텍처를 설명한다. 아키텍처 다이어그램으로 보기 쉽게 시스템을 설명하며 프론트엔드와 백엔드 그리고 데이터베이스를 아우르는 전체적인 시스템의 모습을 파악한다.

3.1 System Organization

3.1.1 Class 다이어그램

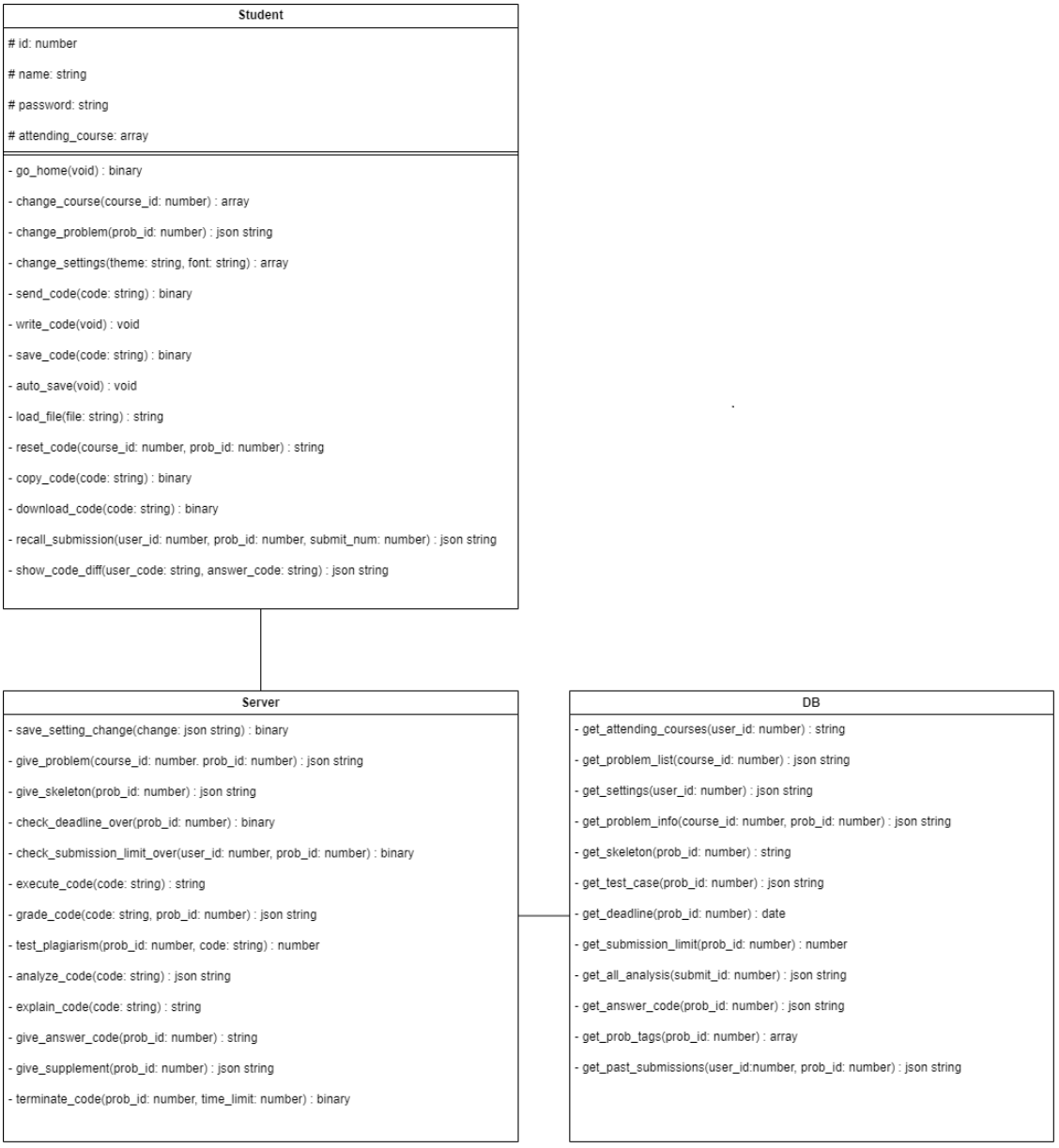


Diagram 1 Overall Class Diagram

Student

서버에 코드를 전송하는 객체이며 코딩테스트 사이트의 실사용자이다.

- go_home(): 학생이 홈버튼을 눌러 홈화면으로 되돌아간다
- change_course() : 학생이 수강 중인 과목 중 현재 진행 중인 과목과 다른 과목을 선택한다. 서버로부터 변경된 과목의 문제들을 array 형태로 받는다.
- change_problem() : 학생이 풀 문제를 변경한다. 서버로부터 변경된 문제의 정보들을 json string 형태로 받는다.
- change_settings() : 학생이 코딩테스트 페이지의 설정을 변경한다. 서버로부터 변경된 설정을 array 형태로 받으면 설정이 적용된다.
- send_code() : 학생이 검증, 실행, 채점, 제출 버튼 중 하나를 눌렀을 때 작성한 코드를 서버로 전송한다.
- write_code() : 학생이 코드에디터에 코드를 입력한다.
- save_code() : 학생이 코드를 임시 저장한다. 코드는 학생의 로컬 스토리지에 저장된다.
- auto_save() : 코드를 주기적으로 로컬 스토리지에 자동 저장한다.
- load_file() : 학생이 로컬 디렉토리에서 파일의 내용을 가져온다.
- reset_code() : 학생이 작성 중인 코드를 초기화한다. 서버로부터 해당 문제의 스켈레톤 코드를 받아 코드 에디터에 붙여넣는다.
- copy_code() : 학생이 작성 중인 문제의 코드를 클립보드에 복사한다.
- download_code() : 학생이 작성 중인 코드를 로컬 디렉토리에 저장한다.
- recall_submission() : 학생이 특정 문제에 대해 자신의 이전 제출 결과를 확인한다. 제출 횟수 제한은 총 3회이며 서버로부터 이전 제출한 결과에 대한 정보를 json string으로 받는다. 이전 제출 결과에는 정답 코드와의 diff, 표절률, 분석 결과, 기능 채점 결과, 코드 설명, 보충 자료가 포함된다.
- show_code_diff() : 서버로부터 정답코드를 받아 학생의 코드와 다른 점을 보여준다.

Server

학생이 제출한 코드를 다루는 객체이다.

- save_setting_change() : 학생이 코딩테스트 사이트의 설정을 변경하면 이후 로그인 할 때에도 같은 환경을 제공하기 위해 DB에 변경된 설정을 저장한다.
- give_problem() : DB로부터 문제의 정보를 받아 json string 형태로 학생에게 전송한다.
- give_skeleton() : 학생이 작성 중인 코드를 초기화하면 DB로부터 문제에 대한 스켈레톤 코드를 받아 학생에게 전송한다.
- check_deadline_over() : 과제 마감일이 지났는지 확인한다. 과제의 마감일이 지났다면 코딩테스트 사이트의 제출 버튼이 비활성화 된다.
- check_submission_limit_over() : 문제의 최대 전송 횟수를 넘겼는지 확인한다. 최대 전송 횟수를 초과했다면 코딩테스트 사이트의 제출 버튼이 비활성화 된다.
- execute_code() : 학생이 서버로 전송한 코드를 실행한다. 정상적으로 실행되면 출력값을 학생에게 전송한다. 만약 오류가 발생하면 오류 위치, 오류 타입을 전송한다.

- grade_code() : 학생이 전송한 코드의 결과를 정답과 비교한다. 정상적으로 실행되면 출력값과 함께 정답 여부를 json string 형태로 학생에게 전송한다. 만약 오류가 발생하면 오류 위치, 오류 타입을 반환한다.
- test_plagiarism() : 학생이 전송한 코드를 이미 다른 유저들이 전송한 동일한 문제 코드와 비교하여 표절율을 구한다. 표절율은 백분율 값으로 표현되어 학생에게 전송된다.
- analyze_code() : 학생이 제출한 코드의 가독성, 효율성을 체크하여 json string으로 학생에게 전송한다.
- explain_code() : openAI codex API를 이용해 얻은 코드 설명을 학생에게 string 형태로 전송한다.
- give_answer_code() : 문제의 정답 코드를 string 형태로 학생에게 전송한다.
- give_supplement() : 문제 태그를 이용해 프로그래밍 사이트를 크롤링한다. 이를 통해 문제와 관련된 자료를 json string 형태로 학생에게 전송한다.
- terminate_code() : 실행한 코드가 일정시간 이상 실행되면 강제로 종료한다. 코드를 강제로 종료하는 이유는 서버의 security, safety, performance를 유지하기 위함이다.

DB

서버에 필요한 정보들을 저장하고 불러올 수 있는 저장소이다. 본 프로젝트는 Django framework를 기반으로 하기 때문에 serializer 형태로 함수를 정의한다. DB의 세부 구성은 본 문서의 7.1에 명시되어 있다.

- get_attending_courses() : 학생이 수강중인 과목들을 반환한다.
- get_problem_list(): 학생이 과제를 진행하고 있는 과목의 문제들을 반환한다.
- get_settings() : 학생의 설정을 반환한다.
- get_problem_info(): 문제의 정보를 반환한다.
- get_skeleton() : 문제의 스켈레톤 코드를 반환한다.
- get_test_case() : 문제의 테스트케이스를 반환한다.
- get_deadlines() : 문제의 마감일을 반환한다.
- get_submission_limit() : 문제의 최대 제출 횟수를 반환한다.
- get_all_analysis() : 문제에 대해 유저가 제출한 코드, 표절율, 분석 결과, 기능 채점 결과, 코드 설명, 보충 자료를 반환한다.
- get_answer_code(): 문제의 정답 코드를 반환한다.
- get_prob_tags() : 문제의 태그를 반환한다.
- get_past_submissions() : 유저가 이전에 제출한 코드, 표절율, 분석 결과, 기능 채점 결과, 코드설명, 보충 자료를 반환한다.

3.2 System Architecture – Front-end Application

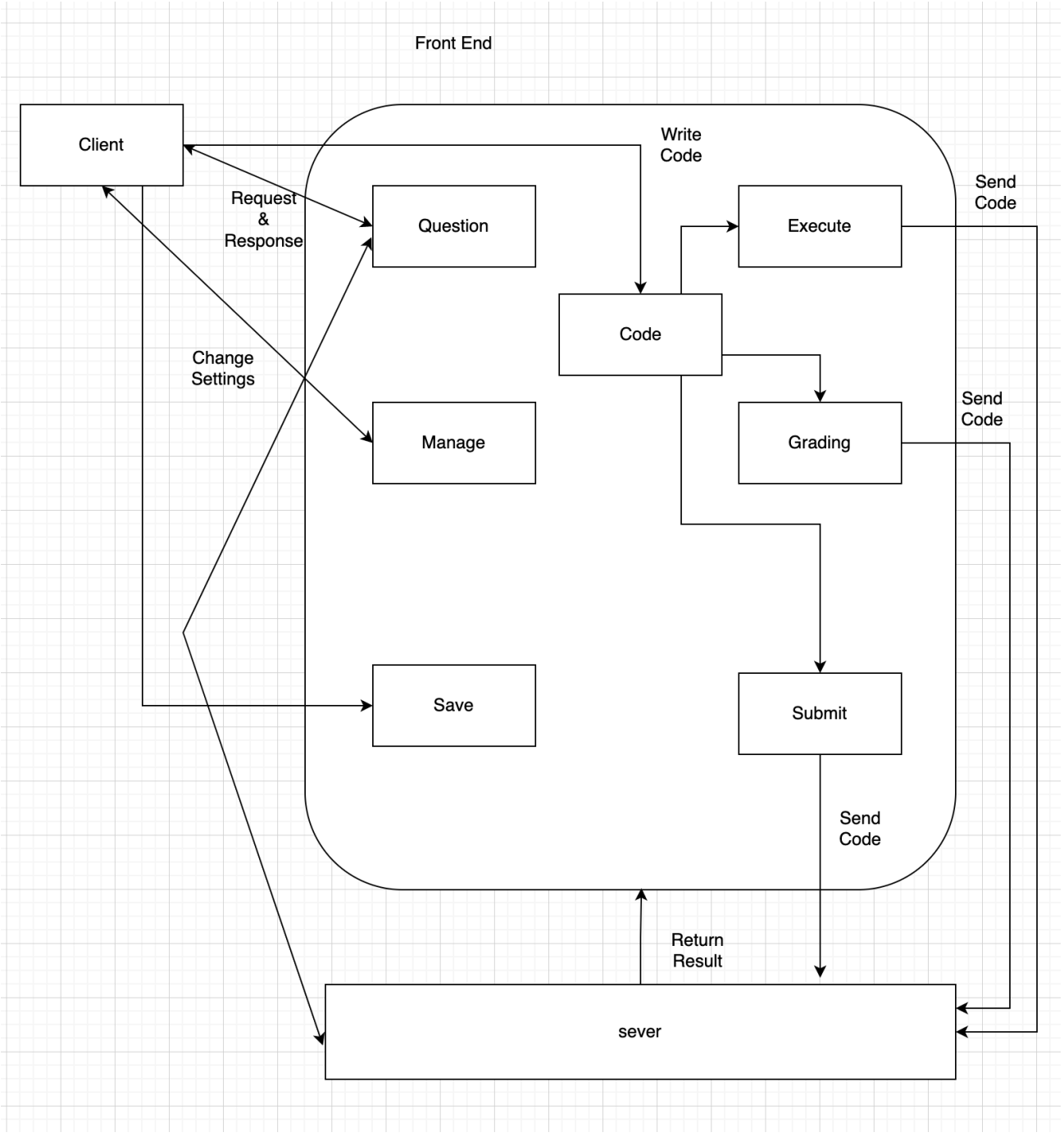


Diagram 2 Frontend Architecture

프론트 엔드는 사용자 및 서버와 상호작용한다. 사용자의 요청이 들어오면 이를 서버로 전달하여 알맞은 결과를 받아서 사용자에게 적절한 응답을 보여준다. 프론트엔드 시스템은 여러개의 서브시스템으로 구성되어 있으며 Question, Manage, Save, Execute, Grading, Submit이 해당 서브시스템들이다.

3.3 System Architecture – Back-end

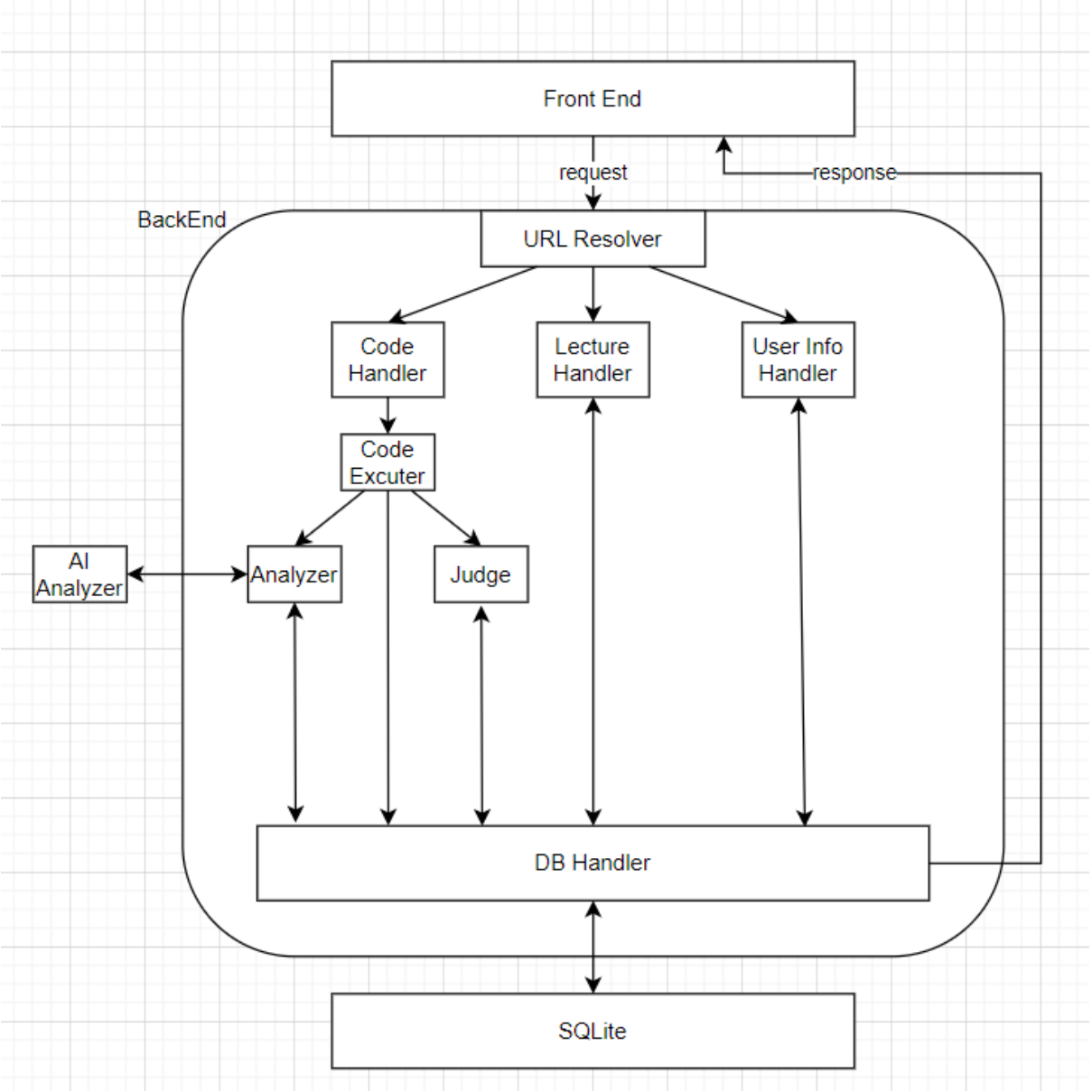


Diagram 3 Backend Architecture

백엔드 시스템은 프론트와 HTTP로 통신을 한다. 각 request에 대한 URL은 장고 프로젝트에 존재하는 urls.py를 사용해 resolve된다. 장고 프로젝트에서의 controller는 views.py가 controller의 역할을 한다.

앱 단위로 분할된 urls.py와 view.py는 기능별로 component를 나누는 데 용이하며 논리적 분할로 인해 분할 개발이 가능하게한다.

3.4 Use state 다이어그램

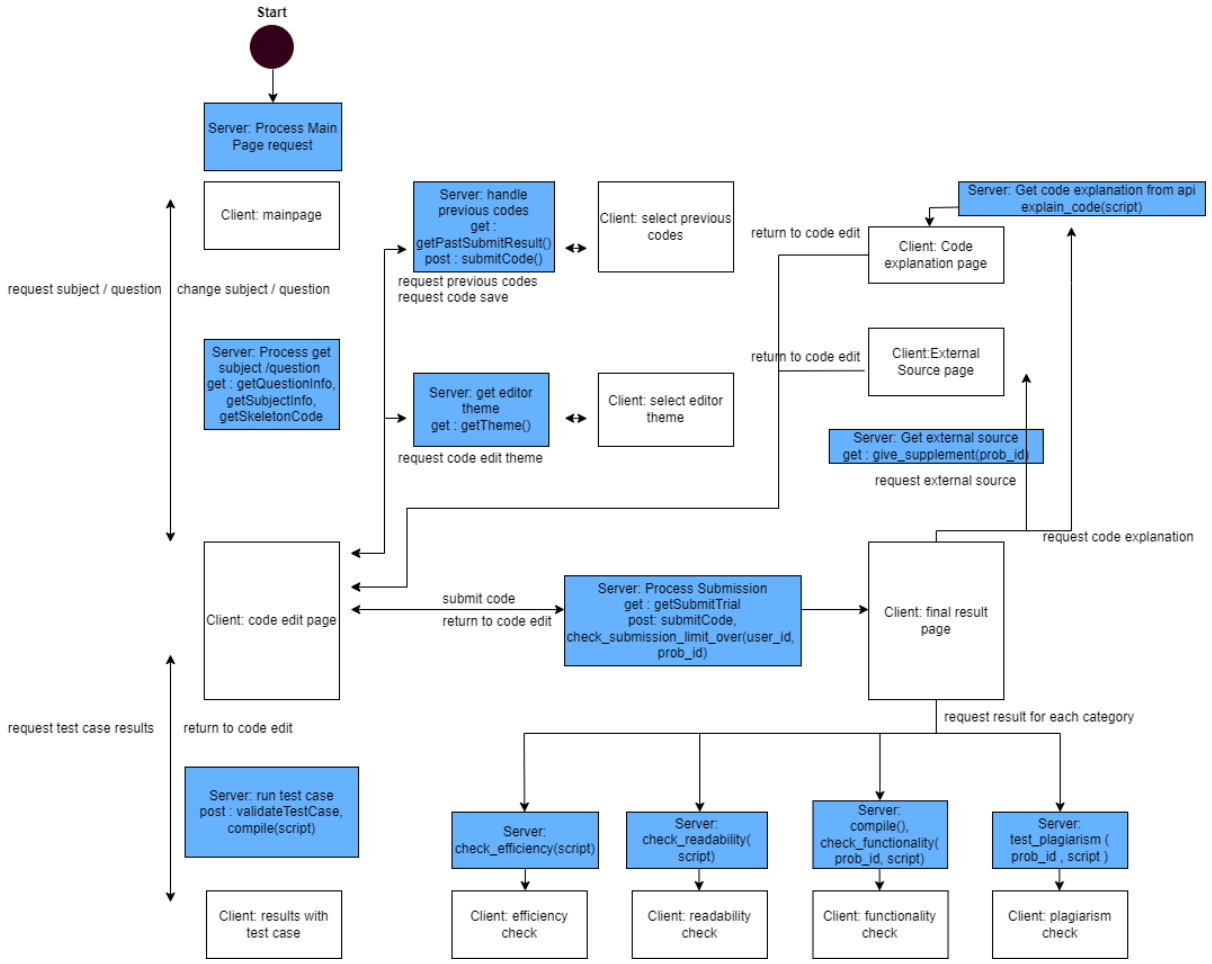


Diagram 4 Use State Diagram

사용자는 main page에 들어간 후, subject 와 question 을 선택한다. 선택 후 사용자는 code edit page 로 이동하게 된다. 사용자는 코드 저장 및 불러오기를 할 수 있으며, 에디터 테마 변경을 할 수 있다. 또한 사용자는 코드를 제출하거나 테스트 케이스를 실행해볼 수 있다. 코드 제출 시 서버로부터 제공된 efficiency, readability, functionality, plagiarism 에 대한 채점 결과를 확인해볼 수 있다. 또한 외부 자료를 요청하거나 코드에 대한 설명을 요청할 수 있다.

4. System Architecture – Frontend

전체적인 프론트엔드의 시스템 아키텍처를 소개하고, 전체 시스템을 서브시스템으로 세분화하여 각 서브시스템의 기능과 속성을 다이어그램으로 표현한다.

4.1 Sub-system

4.1.1 Question

Question은 특정문제의 정보를 관리하고 저장하는 서브 시스템이다. 문제의 정보로는 문제가 속한 과목, 문제 번호, 문제 기한, 문제 내용, 문제 참조 및 제약사항, 문제 예시 테스트 케이스정보가 있다. 문제의 정보는 변경이 불가능하며 사용자에게 보여지게 된다.

Attributes

Question 시스템의 속성은 다음과 같다.

- class<string>: 해당문제가 속한 과목
- number<number>: 해당 문제의 번호
- question<string>: 해당문제의 문제 지문
- restriction<string>: 해당문제의 참조 및 제약사항
- open_testcase_one<object>: 해당문제의 첫번째 예시 테스트 케이스 정보
- open_testcase_two<object>: 해당문제의 두번째 예시 테스트 케이스 정보

Methods

Question 시스템의 기능은 다음과 같다.

- ShowInfo(): 시스템이 가지고 있는 정보를 사용자에게 보내주는 기능
- ChangeQuestion(): 사용자가 선택한 문제로 바꾸는 기능
- TestcaseValidate(): 사용자의 인풋을 가지고 예시 테스트 케이스의 검증을 수행하는 기능

Class 다이어그램

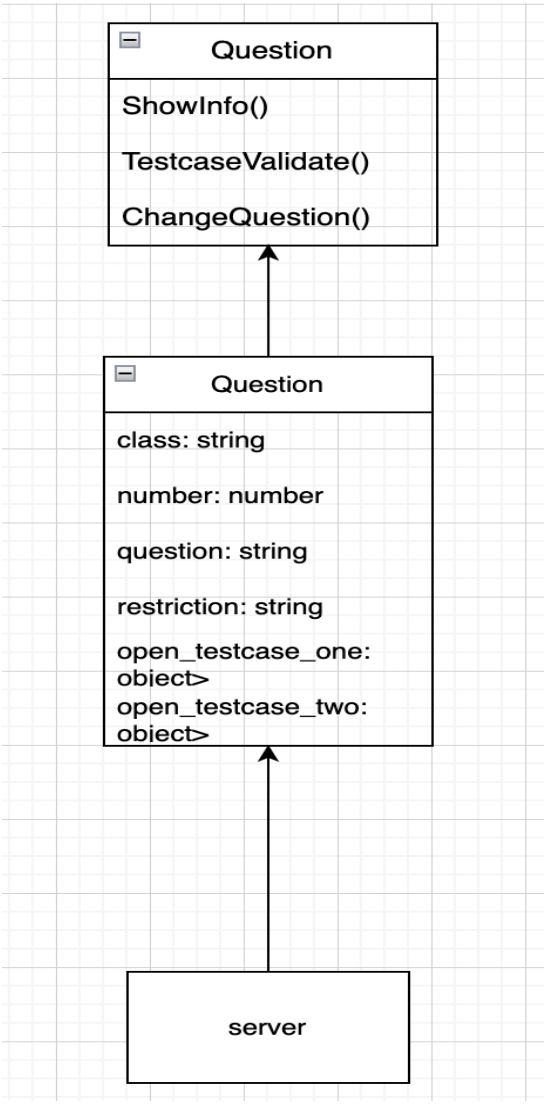


Diagram 5 Question Class Diagram

Sequence 다이어그램

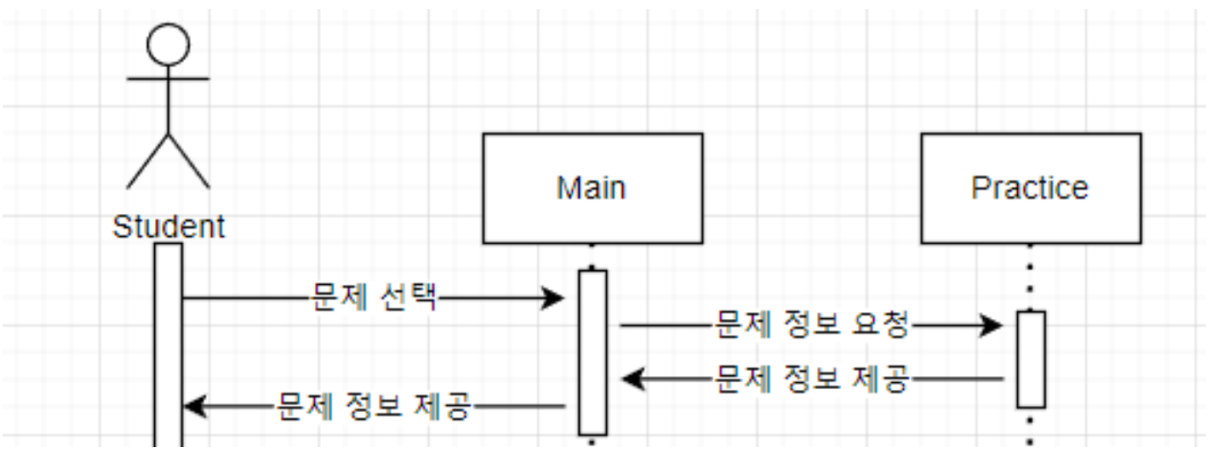


Diagram 6 Question Sequence Diagram

4.1.2 Manage

Manage 서브 시스템은 프로그램의 설정을 관리한다. 설정 항목중에는 웹의 배경색 모드(라이트 모드, 다크모드 설정), 전체 웹사이트 폰트 크기 설정, 코드 에디터 폰트 크기 설정이 있다.

Attributes

Manage 시스템의 속성은 다음과 같다

- is_dark_mode<boolean> : 프로그램이 다크모드인지 아닌지(라이트 모드인지)
- program_font_size<number> : 에디터 외의 프로그램 폰트 크기
- editor_font_size<number> : 에디터의 폰트 크기

Methods

Manage 시스템의 기능은 다음과 같다

- changeMode(): 프로그램의 모드를 바꿔주는 기능
- changeFontSize(): 프로그램(에디터 외)의 폰트 크기를 바꿔주는 기능
- changeEditorFontSize(): 에디터의 폰트 크기를 바꿔주는 기능

Class 다이어그램

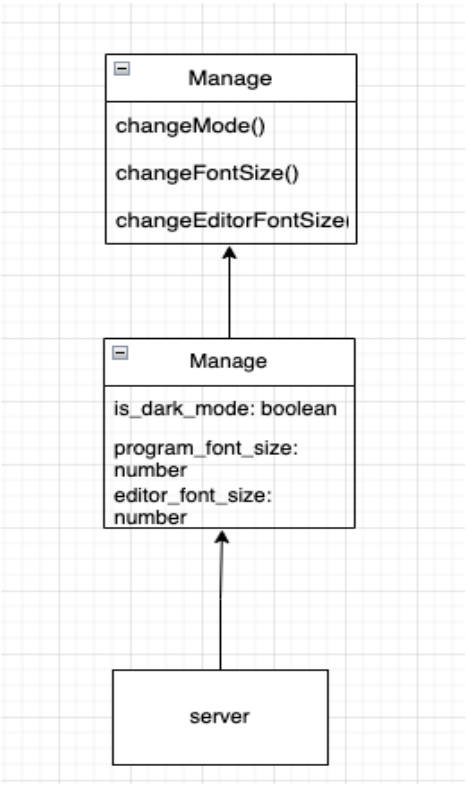


Diagram 7 Manage Class Diagram

Sequence 다이어그램

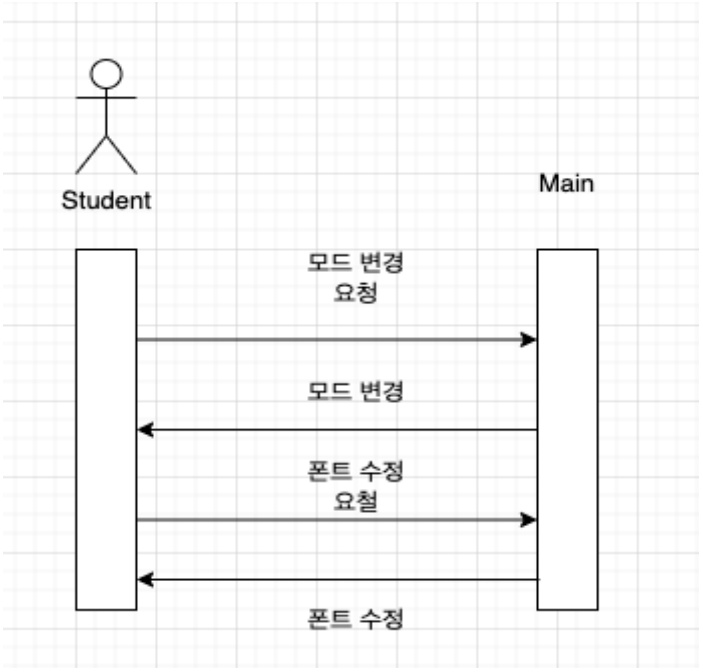


Diagram 8 Manage Sequence Diagram

4.1.3 Save

Save은 사용자가 쓴 코드를 3단계에 걸쳐서 데이터베이스에 저장하는 시스템으로 사용자의 의도와 다르게 브라우저가 종료되었을 때, 이를 복원할 수 있다. 또한, 다양한 알고리즘으로 코드를 구현함으로써 사용자에게 편의성을 제공한다.

Attributes

Save 시스템의 속성은 다음과 같다.

- code_first<string> : 사용자가 첫 번째 구역에 저장한 code
- code_second<string> : 사용자가 두 번째 구역에 저장한 code
- code_third<string> : 사용자가 세 번째 구역에 저장한 code
- temp_code<string> : 사용자가 저장 버튼을 누른 것이 아닌 코드를 입력할 때 일정 시간마다 자동으로 저장되는 code
- user_state<number> : 현재 사용자가 작업 중인 구역
- save_result<boolean> : 코드 저장 상태 결과

Methods

Save 시스템의 기능은 다음과 같다.

- save() : 사용자가 코드를 저장한다
- save_temp() : 일정 시간마다 사용자가 현재 작업 중인 구역의 code를 저장한다.

Class 다이어그램

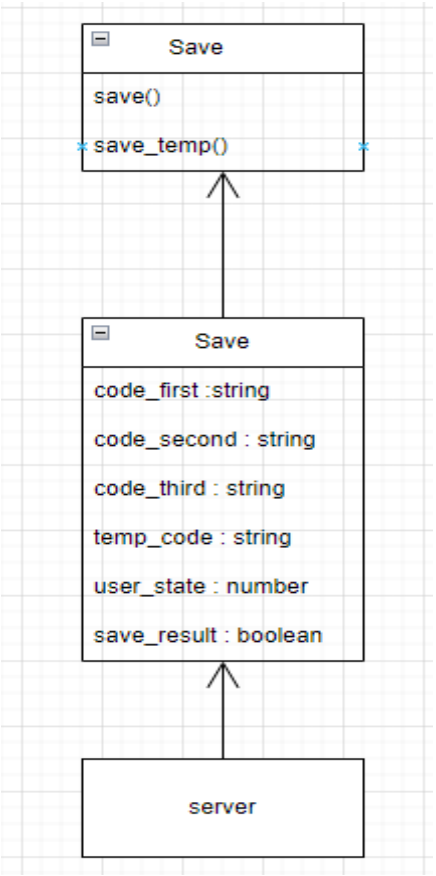


Diagram 9 Save Class Diagram

Sequence 다이어그램

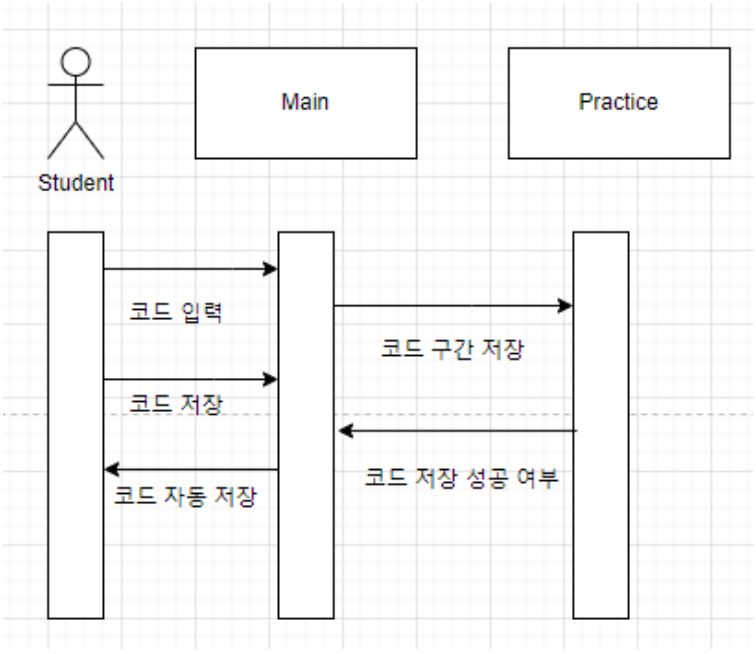


Diagram 10 Save Sequence Diagram

4.1.4 Execute

Execute은 사용자가 쓴 코드를 터미너를 통해 실행하며 실행은 여러번 가능하다. 실행 결과가 에러가 아닐 경우 적절한 출력값을 제공한다. 실행 결과가 에러일 경우 에러 메시지를 출력하며 에러 메시지에 따른 위치를 사용자가 코드를 입력한 editor에 하이라이트를 보여줌으로써 제공한다. 사용자는 Execute를 통해 자신의 코드를 디버깅할 수 있다.

Attributes

Execute 시스템의 속성은 다음과 같다.

- code<string> : 사용자가 현재 작업 중인 구역에서의 code
- execute_result_state<boolean> : 코드 실행 결과
- error_position<string> : 에러 위치
- error_message<string> : 에러 메세지
- user_input<string> : 사용자의 입력 값

Methods

Execute 시스템의 기능은 다음과 같다.

- execute() : 사용자가 코드를 실행한다.
- showExecuteResult() : 실행 결과를 사용자에게 보여준다.
- showErrorMessage() : 사용자에게 에러 메세지를 보여준다.
- showErrorPosition() : 사용자에게 에러 위치를 보여준다.
- getUserInput() : 사용자의 입력 값을 저장한다.

Class 다이어그램

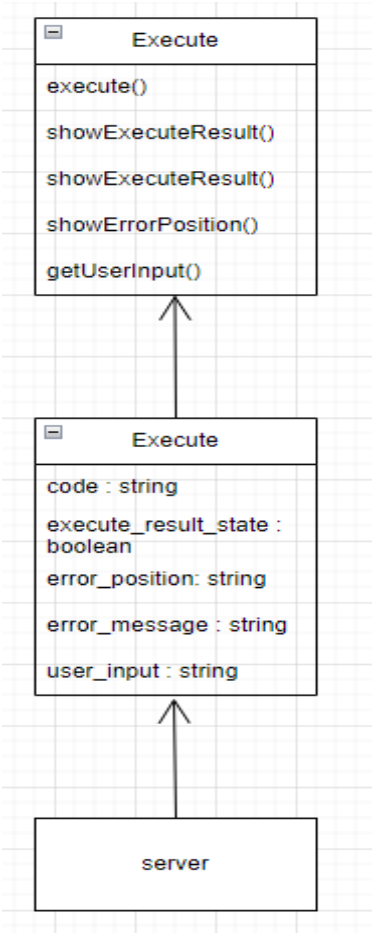


Diagram 11 Excecute Class Diagram

Sequence 다이어그램

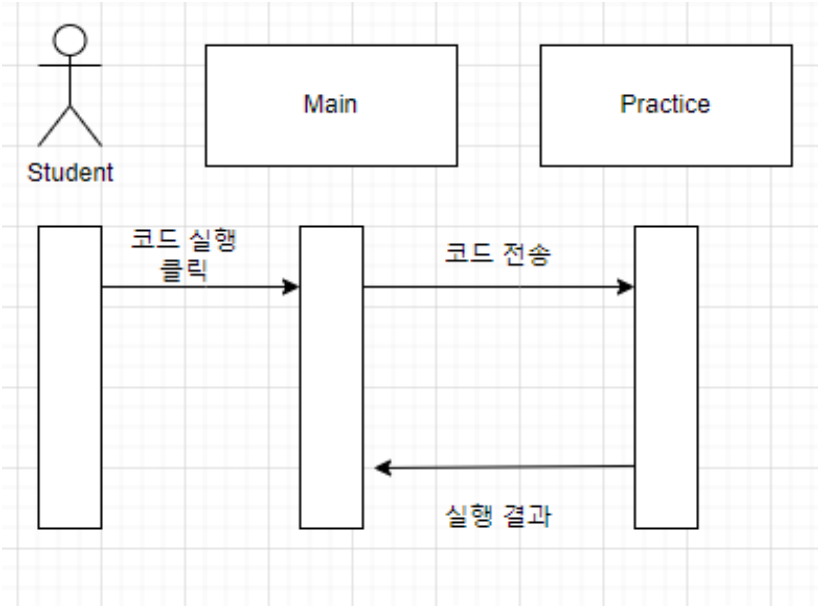


Diagram 12 Execute Sequence Diagram

4.1.5 Grading

Grading은 사용자가 쓴 코드에 대한 점수로 제공한다. Grading을 위해 오픈 테스트케이스와 히든 테스트케이스를 모두 테스트하며 각각의 테스트케이스에 따라 통과나 실패에 대한 결과를 터미널 창을 통해 사용자에게 제공한다. 오픈 테스트케이스가 실패했을 경우 테스트케이스 정보를 제공하며 히든 테스트케이스가 실패했을 경우 테스트케이스 정보를 제공하지 않는다.

Attributes

Grading 시스템의 속성은 다음과 같다.

- code<string> : 사용자가 현재 작업 중인 구역에서의 code
- open_testcase<list> : 오픈 테스트케이스 실행 결과
- hidden_testcase<list> : 히든 테스트케이스 실행 결과
- score<number> : 코드 점수

Methods

Grading 시스템의 기능은 다음과 같다.

- grade() : 사용자의 코드를 전송하고 채점한다.
- showGradeResult() : 채점 결과를 사용자에게 보여준다.

Class 다이어그램

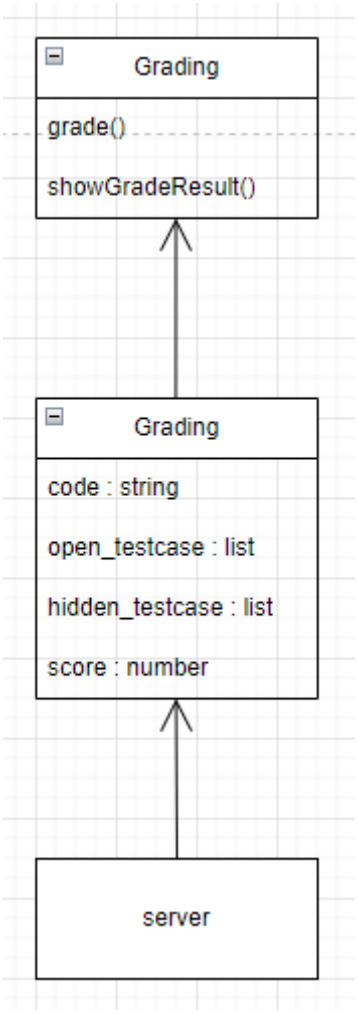


Diagram 13 Grading Class Diagram

Sequence 다이어그램

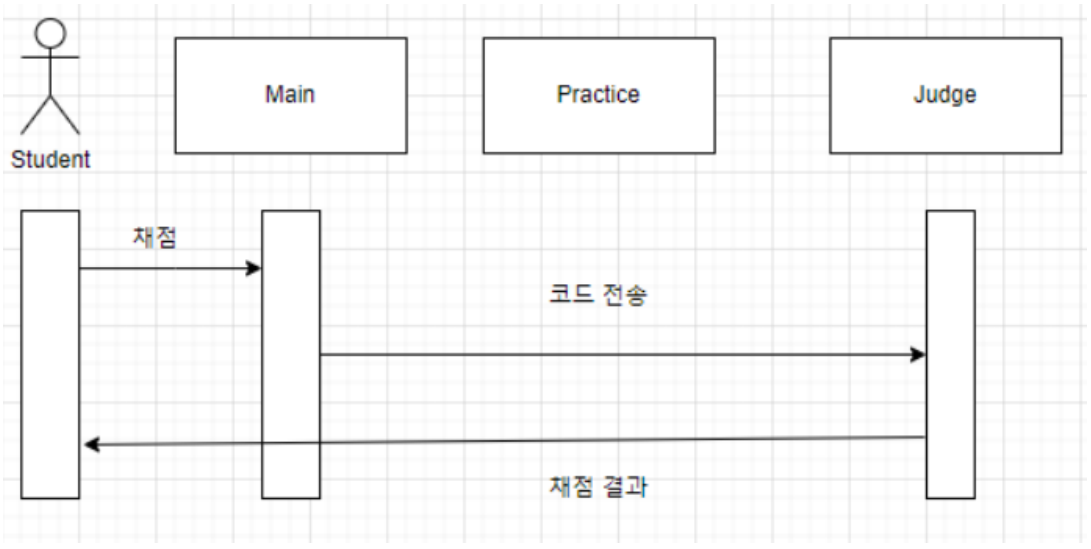


Diagram 14 Grading Sequence Diagram

4.1.6 Submit

Submit 서브시스템은 유저가 제출한 코드에 대한 제출 결과를 저장하고 관리한다. 제출결과에 대한 정보로는, 표절 검사 결과, 기능성 검사 결과, 효율성 검사 결과, 가독성 검사 결과, 코드 설명, 관련 자료, 제출 코드와 정답 코드간의 diff결과가 있다.

Attributes

Submit 시스템의 속성은 다음과 같다

- functional_result<object> : 제출 코드에 대한 기능성 검사 결과. 각 테스트 케이스에 대한 채점 결과가 들어있다.
- readability_result<object> :제출 코드에 대한 가독성 검사 결과. Python pylama로 검사한 각 항목들의 전체 결과가 들어 있다.
- efficiency_result<object> : 제출 코드에 대한 효율성 검사 결과. Python multimetric으로 검사한 각 항목들의 전체 결과가 들어 있다.
- code_explain<object> :제출 코드를 Open_AI_Codex로 분석한 결과(코드 설명)이 들어있다.
- related_data<object> : 해당 문제와 관련된 관련 자료.
- plagirism_rate<number> : 제출 코드와 정답 코드를 비교한 표절률이 들어 있다.
- code_diff<object> : 제출 코드와 정답 코드간의 diff를 수행한 결과.
- code<string>: 사용자가 제출할 코드

Methods

Submit 시스템의 기능은 다음과 같다.

- submit() : 코드를 제출한다.
- showResult() : 제출 결과를 사용자에게 보여준다.

Class 다이어그램

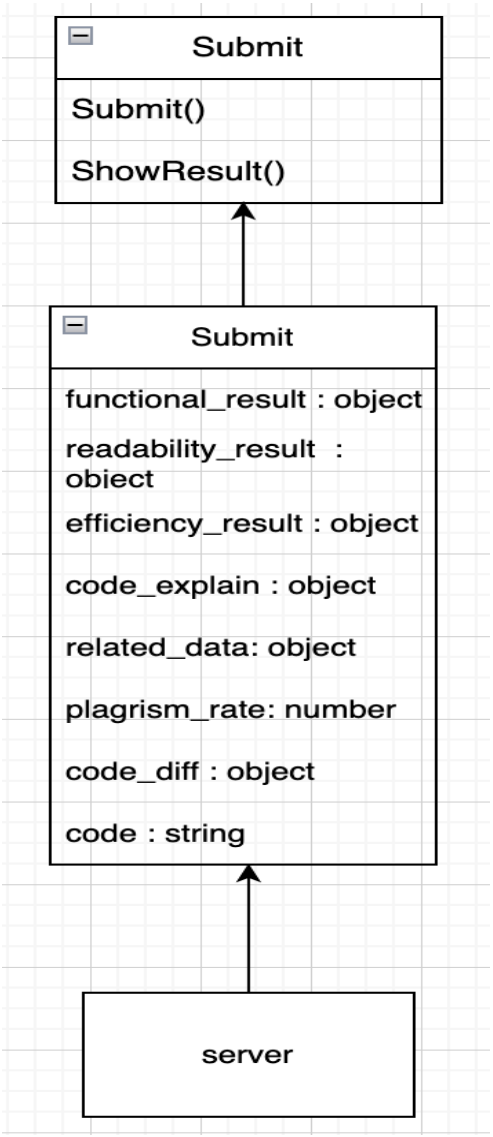


Diagram 15 Submit Class Diagram

Sequence 다이어그램

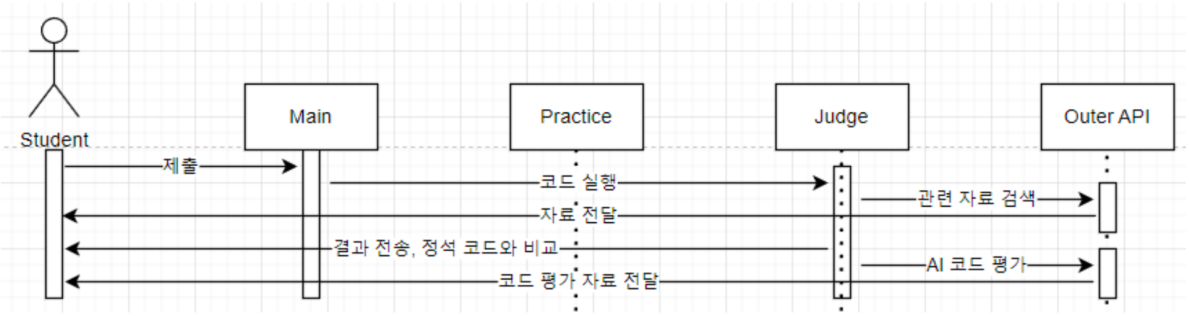


Diagram 16 Submit Sequence Diagram

5. System Architecture - Backend

백엔드 시스템의 전반적인 구조를 설명한다. rest api를 사용한 request가 어떻게 resolve되어 처리되는지 소개할 것이며 본 프로젝트의 DRF(Django Rest Framework) 구조 또한 설명한다.

5.1 Overall Architecture

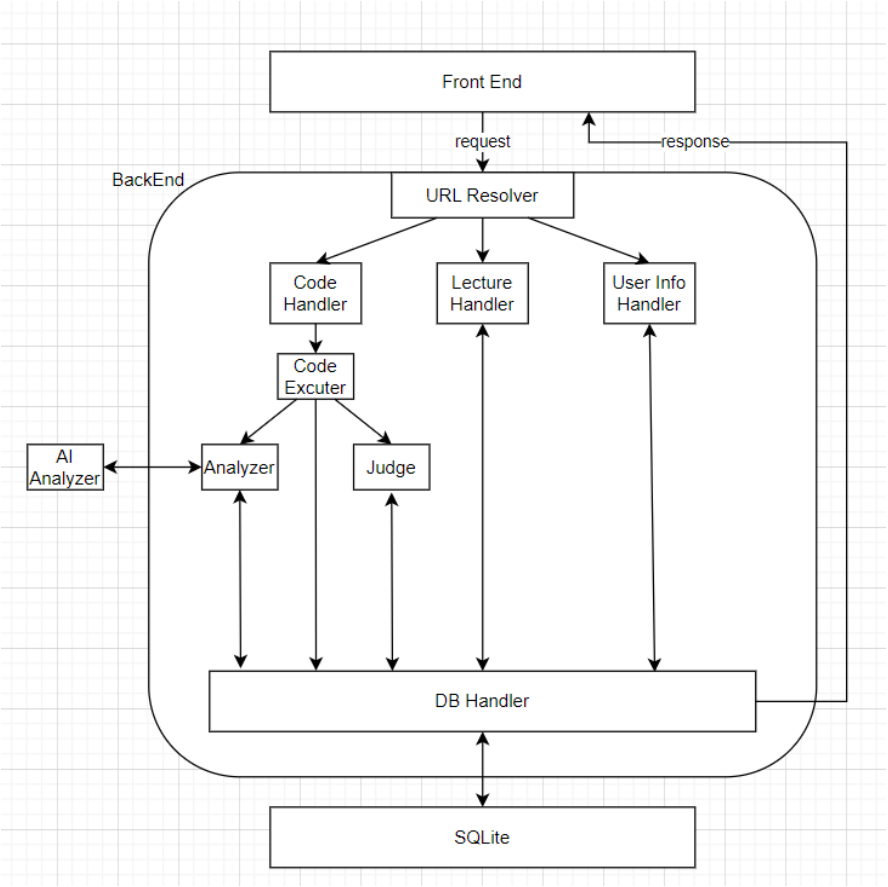


Diagram 17 Overall architecture

백엔드는 DRF(Django REST Framework)를 기반으로 구현된다. MVC 패턴을 사용해 구현하였다. 백엔드의 역할은 프론트엔드에서 REST api를 통해 들어온 request에 맞는 작업을 수행하는 것이다.

백엔드 아키텍처를 구성하는 요소로는 Code handler와 User Interaction handler, Lecture Handler, User Info Handler 그리고 DB handler가 있다. 각각 handler는 django app단위로 구현된다.

Code handler는 프론트엔드에서 Code관련 처리를 담당하게 된다. Code handler의 기능적 분류는 Code submission, Code Execution, Code Analyze, Code Judge가 있는데 각 기능은 app단위로 구현되며 Code handler가 request의 URL을 resolve해 필요한 기능을 실행시킨다.

Code handler의 기능이 마무리 되면 해당 데이터들을 DB에 저장하고 response를 보내게 된다.
Lecture handler는 유저의 강의 데이터와 문제 정보 등을 response로 return한다.
User Info handler는 유저의 UI/UX와 같은 유저의 개인 설정을 handle하는 역할을 맡는다.
그 외의 특이사항은 백엔드와 DB의 interaction을 전담하는 layer를 새롭게 만든 것이다. 구현 방법은 Model, Serializer를 하나의 app 안에서만 구현을 하였다. 다른 app이 DB의 데이터가 필요하다면 DB 관리 app에 존재하는 model과 serializer를 import하여 사용하게 된다. DB 전담 레이어를 새롭게 만듦으로써 얻을 수 있는 장점으로는 쿼리문을 BE 프로그래머가 관리하지 않고 DB 전담 프로그래머가 관리함으로써 BE 개발자들은 abstraction하게 DB를 관리할 수 있다.

5.2 Internal Functions

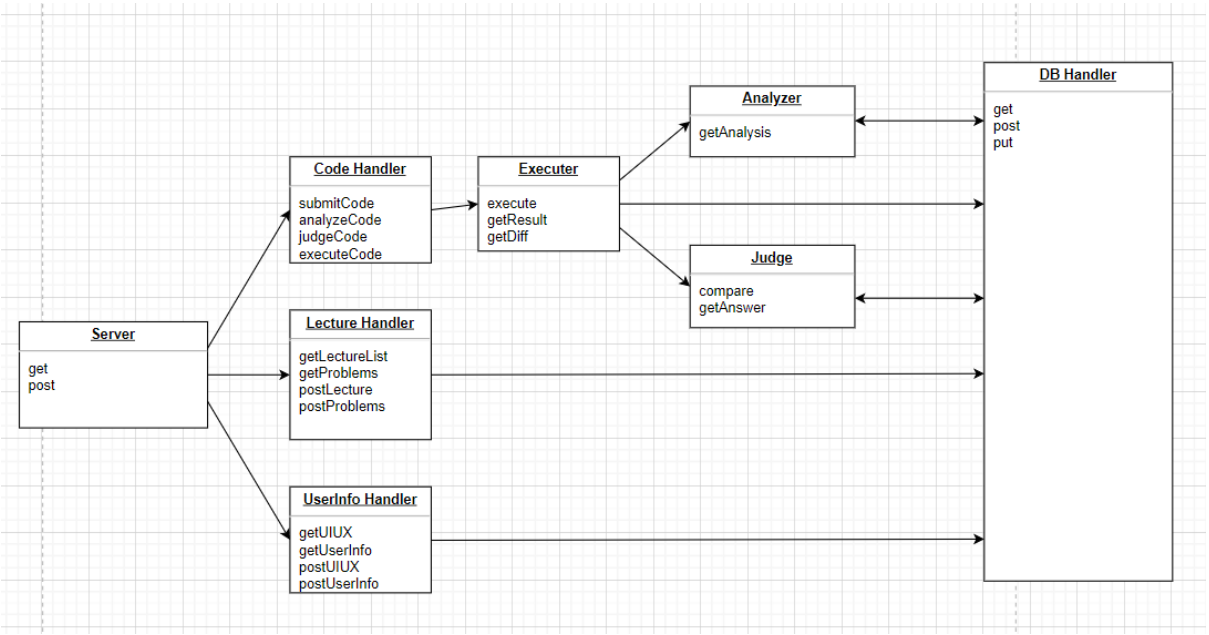


Diagram 18 Internal Functions

Server

get/ post: restAPI를 기반으로 한 req, res를 url resolver가 라우팅하여 정보를 전달한다.

Code Handler

submitCode: execution후 결과와 함께 DB에 저장한다.
analyzeCode: Analyzer를 호출하는 함수
executeCode: code를 execution하는 함수
judge: Judge를 호출하는 함수

Executer

execute: junit을 사용하여 전달된 코드를 실행한다.
getResult: execute에서 나온 결과값이 저장된 데이터를 불러온다.
getDiff: DB에서 정답 코드를 불러와 diff한다.

Analyzer

getAnalysis: 외부 api를 사용하여 코드를 분석한 뒤 DB에 저장한다. 그 후 해당 정보를 불러온다.

Judge

compare: execution후 결과값과 테스트케이스의 정답을 비교해본다.
getAnswer: compare 함수 실행 전 DB에서 테스트케이스의 정답을 불러온다.

Lecture Handler
getLectureList: 강의 목록을 불러온다
postLecture: 강의 정보를 저장한다.
getProblems: 문제 정보들을 불러온다.
postProblems: 문제 정보를 post하여 서버에 저장한다.

UserInfoHandler
getUIUX: 유저가 저장한 UI정보를 DB에서 불러온다
postUIUX: 유저의 UIUX정보를 DB에 저장한다.
getUserInfo: 유저의 정보를 불러온다
postUserInfo: 유저의 정보를 DB에 저장한다.

DB Handler
get: DB에서 요청한 정보를 꺼내온다 (select)
post: DB에 정보를 저장한다.(insert)
put: DB의 정보를 수정한다 (update)

5.3 Grading Process

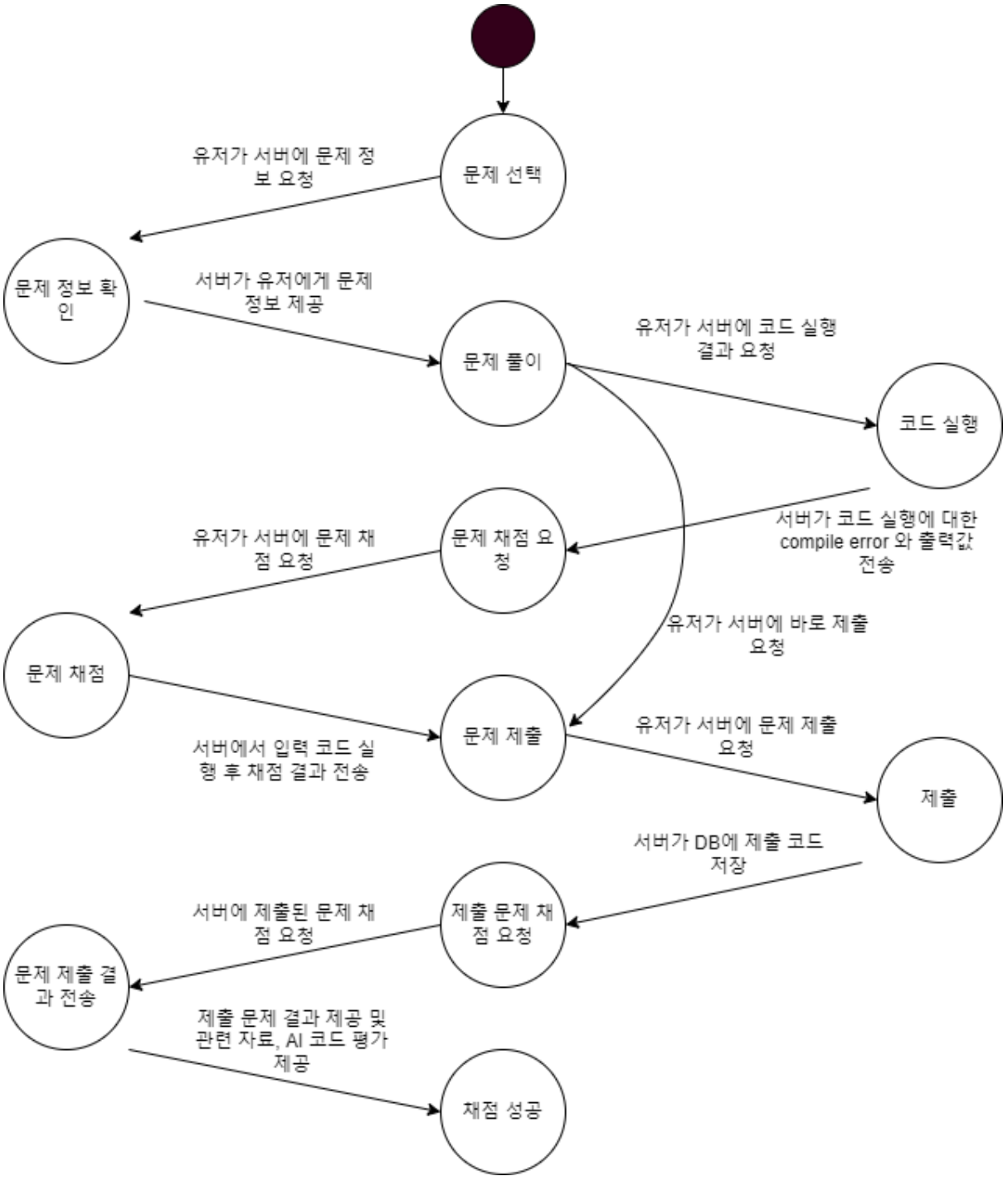


Diagram 19 Grading Process

사용자는 문제 선택, 문제 채점을 위해 서버에 요청을 보낸다. 요청을 받은 후 서버는 DB를 접근하여 필요한 정보를 얻은 후 결과를 도출하여 사용자에게 결과를 전해준다. 서버는 코드 실행 결과에 compile error 발생 시 해당 error 또한 전송한다. 제출 시 서버는 사용자가 요청 시 관련 자료, AI 코드 평가를 사용자에게 제공한다.

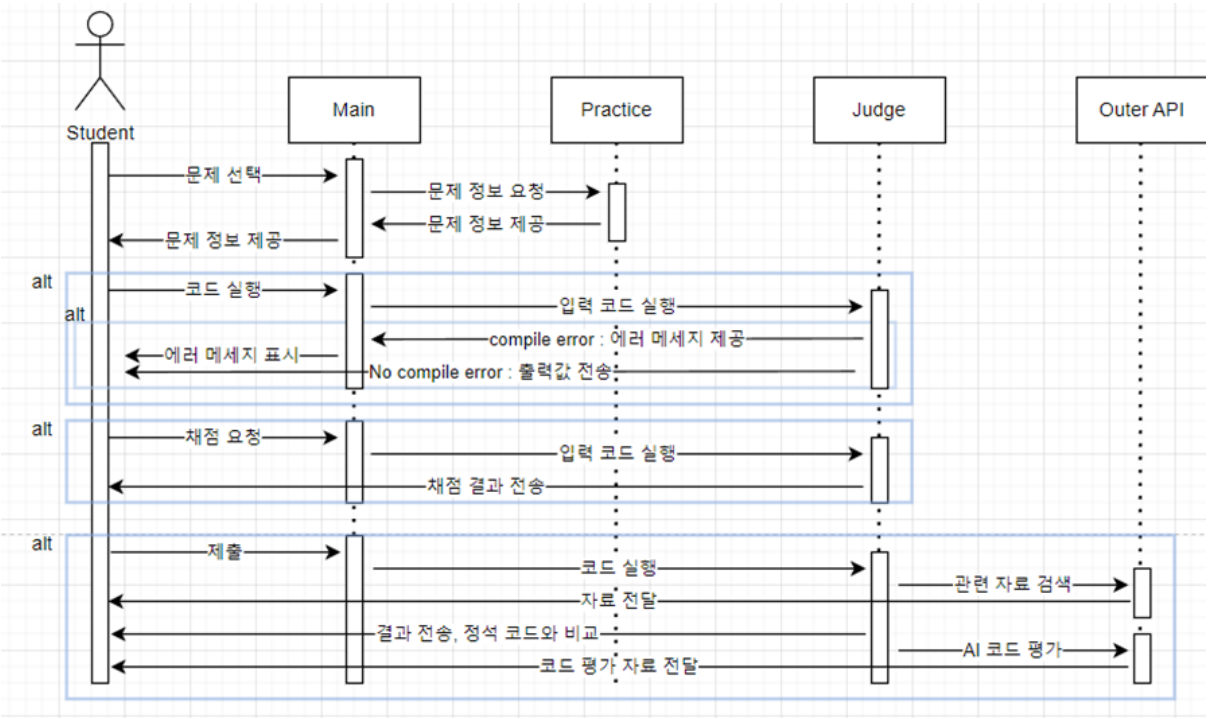


Diagram 20 Code Handler Sequence

Code handler의 상세 기능에 따른 sequence 다이어그램이다. DB layer를 abstraction처리하여 Component단위로만 나타내었다.

6. Protocol

프론트엔드와 백엔드는 HTTP 통신을 통해서 정보를 주고 받는다. 본 시스템을 제공하기 위한 환경은 Django 프레임워크를 사용해서 제공된다. 사용자의 코드와 같은 사용자 정보는 sqlite 안에 저장되며 통신할 때 사용되는 데이터의 형태는 JSON을 사용한다.

6.1 HTTP

HTTP는 W3상에서 정보를 주고받을 수 있는 프로토콜이다. 본 시스템에서는 클라이언트와 서버 사이에 이루어지는 요청과 응답을 다루는 프로토콜이라고 할 수 있다. 클라이언트가 서버에 HTML을 요청하면 요청은 HTTP프로토콜 을 통해 이루어지며 요청에 따른 HTML파일은 서버가 다시 HTTP프로토콜을 통해 응답을 보낸다.

6.2 Django

Django는 Python으로 작동하는 웹 프레임워크로 보안에 뛰어나며 웹 사이트 개발 속도를 높이는데 도움을 준다. Django의 디자인 패턴은 Model, View, Controller로 이루어진 MVC 패턴을 이루며 관리자 페이지를 기본적으로 제공하기 때문에 데이터베이스의 구조를 쉽게 파악하는 등 개발 과정에서 도움을 준다.

6.3 JSON

JSON은 Javascript Object Notation의 약자이다. 모든 시리얼화 가능한 값 또는 “키-값" 쌍으로 이루어진 데이터 오브젝트를 전달하기 위해 인간이 읽을 수 있는 텍스트를 사용하는 데이터 포맷이다. 클라이언트는 백엔드로부터 데이터를 받을 때 JSON의 형태로 받게 되며 받은 JSON형태의 데이터를 적절한 포맷으로 바꾸어 사용자에게 보여주게 된다.

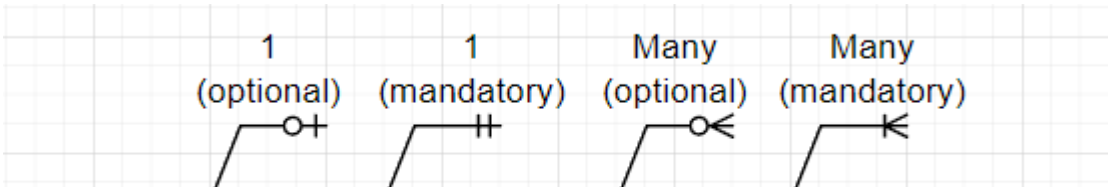
7. Database Design

시스템의 데이터베이스의 디자인에 대해 서술한다.

7.1 ER 다이어그램

시스템에 존재하는 entity를 사각형으로, 해당 entity가 가지는 attribute를 사각형 내에 표시하였다. 각 attribute에는 이름, 타입, 기본 키/외래 키 여부를 표시하였다.

Entity 간 관계는 선으로 표시된다. 표기법은 아래와 같다.



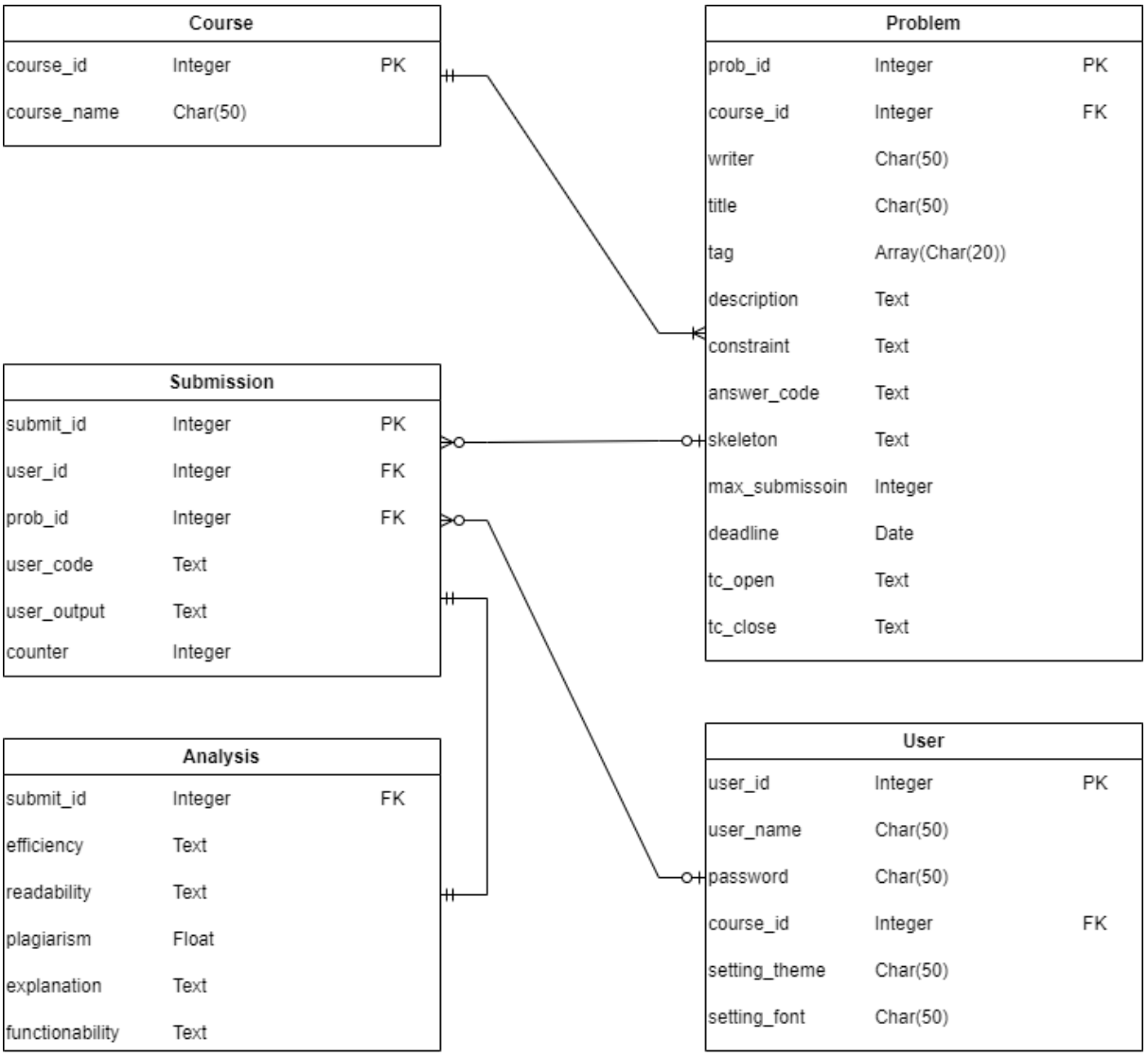


Diagram 21 ER Diagram

7.2 Relational Schema

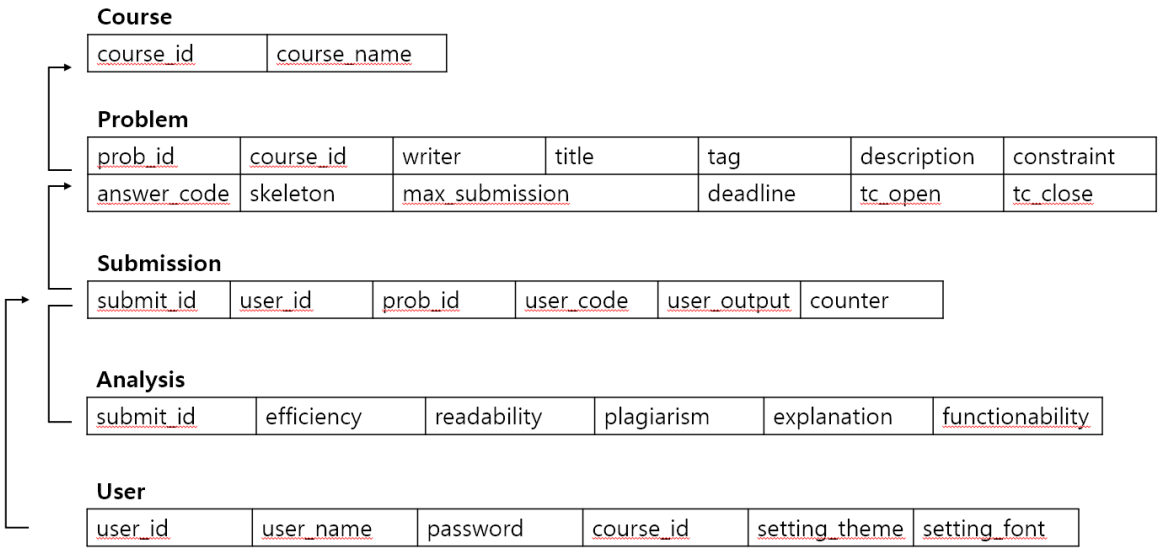


Diagram 22 Relational Schema

7.3 SQL DDL

본 프로젝트는 Django REST 프레임워크를 기반으로 하기 때문에 일반적인 SQL DDL과는 다소 차이가 있을 수 있다. 아래에 사용한 models.py를 첨부하였다.

```
class Course(models.Model):
    course_id = models.IntegerField(primary_key=True)
    course_name = models.CharField(max_length=50)

class User(models.Model):
    user_id = models.IntegerField(primary_key=True)
    username = models.CharField(max_length=50)
    password = models.CharField(max_length=50)
    course_id = models.ForeignKey(
        Course, on_delete=models.SET_NULL, null=True, blank=True
    )
    setting_theme = models.CharField(choices=THEME,
```

```
default="Light", max_length=50)
    setting_font = models.CharField(choices=FONT, default="C",
max_length=50)

class Problem(models.Model):
    prob_id = models.IntegerField(primary_key=True)
    course_id = models.ForeignKey(Course,
on_delete=models.CASCADE)
    writer = models.CharField(max_length=50)
    title = models.CharField(max_length=50)
    tag = ArrayField(ArrayField(models.CharField(max_length=20)))
    description = models.TextField()
    constraint = models.TextField(blank=True)
    answer_code = models.TextField()
    skeleton = models.TextField(blank=True)
    max_submission = models.IntegerField()
    deadline = models.DateField()
    tc_open = models.TextField()
    tc_close = models.TextField()

class Submission(models.Model):
    submit_id = models.IntegerField(primary_key=True)
    user_id = models.ForeignKey(User, on_delete=models.CASCADE)
    prob_id = models.ForeignKey(Problem, on_delete=models.CASCADE)
    user_code = models.TextField(blank=True)
    user_output = models.TextField(blank=True)
    counter = models.IntegerField()

class Analysis(models.Model):
    submit_id = models.ForeignKey(Submission,
on_delete=models.CASCADE)
    efficiency = models.TextField(blank=True)
    readability = models.TextField(blank=True)
    plagiarism = models.FloatField()
    explanation = models.TextField(blank=True)
    functionability = models.TextField(blank=True)
```

Diagram 23 SQL DDL