코딩 테스트 웹사이트

소프트웨어 디자인 명세서

2022.11.13.

소프트웨어공학개론 41

TEAM 8

팀장 **박건우**

팀원 **김동희**

서의태

윤재한

정찬

목차

Diagram 목차

Figure 목차

1. Preface

1.1. Readership

본 문서는 개발팀이 열람하는 것을 상정하여 작성하였다. 사용자가 따르게 될 시스템의 구성 요소를 자연어 중심으로 표현하여, 개발팀이 이 시스템의 요구사항과 구성 요소를 쉽게 이해할 수 있도록 작성한다

1.2. Version history

2022-11-13, Version 1.0, 팀원 회의 후 최초 버전 발행, 박건우 외 4명.

1.3. Document structure

읽기에 앞서, 추후 본 웹서비스를 개발하기 위해 추가적인 오브젝트 및 컴포넌트의 생성과 수정, 삭제 등이 발생할 수 있다.

(1) Introduction

이 장에서는 본 서비스의 설계에 사용된 다이어그램과 도구를 소개하고, 시스템의 개발 배경이 된 문제 상황과 이 시스템이 해당 문제를 어떻게 해결할 수 있는지 기술한다.

(2) Overall System Architecture

이 장에서는 시스템과 각 서브 시스템의 구조를 개략적으로 기술하고 서브 시스템 간의 연결과 전체적인 기능을 보여주는 것에 집중되어 있다.

(3) System Architecture – Frontend

이 장에서는 본 서비스에서 사용되는 언어에 대하여 기술한다. React를 통해 Frontend 개발이 진행되며, Frontend의 전체적인 구조와 해당 구조를 구현하기 위해 클래스가 필요한지 기술한다.

(4) System Architecture – Backend

이 장에서는 Frontend의 요청을 처리하는 Backend의 설계의 관한 내용을 기술한다.

(5) Protocol Design

이 장에서는 Frontend와 Backend의 상호작용이 어떤 방식으로 진행되는지 기술한다.

(6) Database Design

이 장에서는 관계형 데이터베이스 다이어그램을 기술한다.

2. Introduction

2.1. Applied diagram

state diagram, entity-relation diagram, class diagram을 중점으로 사용하여 시스템을 기술하였다.

State Diagram: State Diagram은 오브젝트 간 상호작용을 시간순으로 나타낸 도표이다. 사전에 작성한 시나리오를 바탕으로 다이어그램을 작성해 필요한 오브젝트와 기능을 식별하였다. 작성한 Sequence Diagram은 이후 Relation Diagram과 Class Diagram을 작성할 때에도 사용되며, 다른 다이어그램에서 보여주기 힘든 상호작용의 시간 순 진행을 보여주고, 추상적인 하나의 기능을 여러 단계로 나누어 생

각할 수 있게 돕는다.

Entity-Relation Diagram: Entity-Relation Diagram은 여러 Entity 간의 관계와 각 Entity의 간단한 세 부사항을 보여주는 다이어그램으로, 시스템에 어떤 Entity가 필요하며, 여러 Entity들이 어떤 관계를 가질지를 식별하기 위해 작성한다. Entity는 이전에 작성한 시나리오와 다른 다이어그램을 통해 식별해 나가며 다이어그램에 기술하며, 식별된 Entity는 필요한만큼 세분화하여 작성한다.

Class Diagram: Class Diagram은 시스템의 class, attribute, method와 object 간의 포함관계를 보여 주는 다이어그램으로 시스템의 전체적인 구조를 서술하기 위해 작성한다. 가장 상위 Entity인 시스템을 위에 기술하며, 아래로 갈수록 직접적으로 실행되는 각각의 operation들을 기술한다

2.2. System overview

본 서비스 Python 코딩 테스트를 연습할 수 있는 웹 어플리케이션이다. 코딩 테스트는 주어진 문제를 제한 시간 안에 프로그래밍을 통해 해결하는 것이다. 기존에는 프로그래밍 경진대회를 준비하는 사람들이 주로 연습했지만 최근에는 기업의 채용 과정에도 코딩 테스트가 있을 만큼 해당 역량을 기르는 것은 꼭 필요하다. 따라서 본 서비스의 목적은 Python 코딩 테스트의 연습으로 설정했다.

본 서비스는 웹 어플리케이션을 기반으로 사용자가 Python 언어로 코딩 테스트 문제를 풀이할 수 있는 환경을 제공한다. 서비스 안에서 코드를 작성하여 제출하고, 문제에 따른 예제의 채점 결과와 코드 작성의 가독성, 효율성 등을 점수화하고 문제 풀이에 도움이 되는 추천 콘텐츠를 제공받는다. 이를 위해 웹 서버, 코드 채점 시스템 및 추천 시스템 등이 개발된다.

본 서비스는 Frontend와 Backend로 구성된다. Frontend는 React로 구현되며 Backend에 필요한 정보를 요청한다. Backend는 웹 서버와 데이터베이스로 구성된다. 웹 서버는 Django를 통해 구현되고 데이터베이스는 sqlite3 플러그인을 활용한다. Backend에 요청이 들어오면 그 요청에 해당하는 값을 반환한다. 반환 값의 key, value를 가지는 JSON 형태이다.

3. System Architecture - Overall

3.1. System Organization

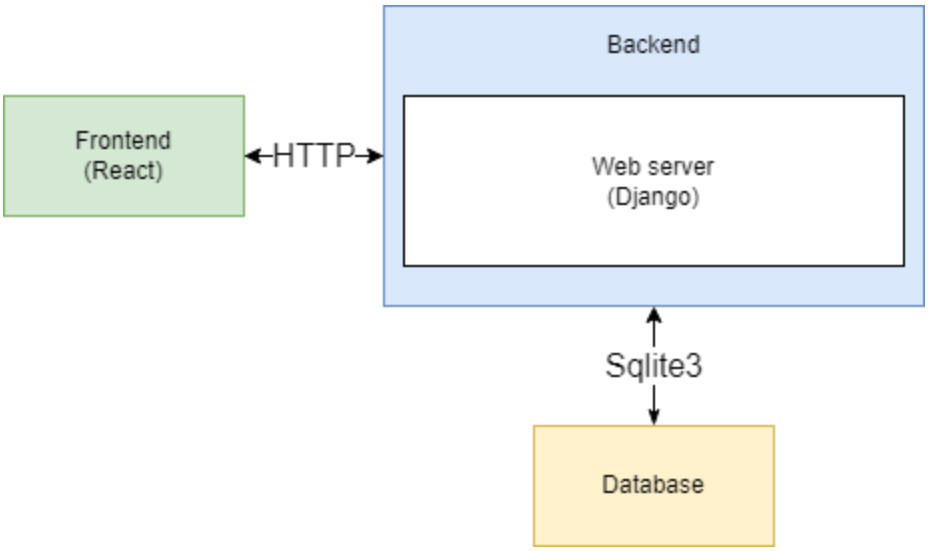

Diagram 1 System Organization

본 서비스의 시스템은 Frontend와 Backend로 구성된다. Frontend는 사용자의 입력을 처리하고 Backend에 필요한 정보를 요청하고 Backend로부터 반환된 정보를 이용하여 사용자에게 적절한 정보를 표시하는 역할을 한다. Backend는 웹 서버와 데이터베이스로 구성되며 Frontend의 요청에 따른 적절한 값을 반환한다.

3.2. System Architecture – Frontend

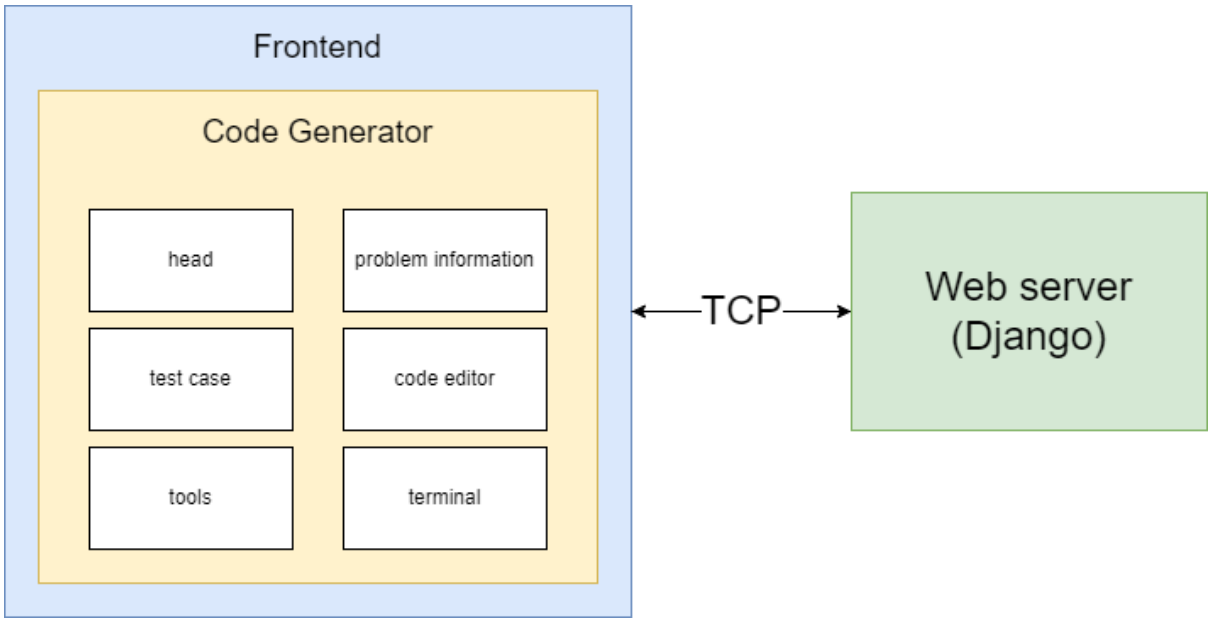

Diagram 2 System Architecture - Frontend

3.3. System Architecture – Backend

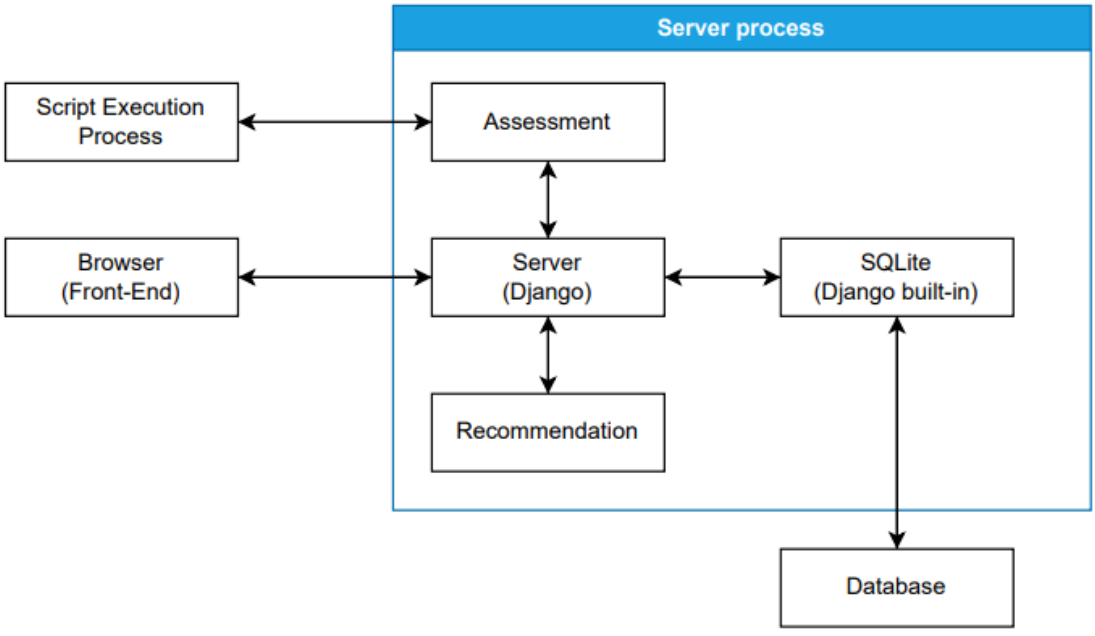

Diagram 3 System Architecture - Backend

본 시스템의 백 엔드에서는 프론트 엔드의 요청에 따라 함께 보낸 정보를 처리하고 그 결과를 반환하는 방식으로 동작한다. 로그인, 문제 선택, 코드 실행, 코드

저장, 채점, 제출과 같은 기능을 수행하고 이와 같은 기능들은 Django 프레임워크의 view와 url mapping을 이용하여 구현한다. 또한 사용자의 제출 코드가 오답일 경우, 문제 풀이에 도움이 되는 컨텐츠를 추천한다.

데이터베이스의 경우 사용자 정보, 사용자의 문제풀이, 문제 정보와 같은 정보를 담는다. 해당 서비스의 데이터베이스는 Django 프레임워크에 내장된 SQLite를 이용하여 5.3에 기술된 기능들을 수행하기 위한 정보 처리작업을 진행하고 정보를 저장, 관리한다. 결과는 server를 통해 프론트 엔드로 전달된다.

3.3.1. Administrator

본 서비스의 관리자 페이지의 경우 Django에서 제공되는 admin 기능을 활용한다. 데이터 베이스의 관리역할을 수행하게 된다.

4. System Architecture – Frontend

사용자의 요청과 서버의 반응에 따른 state diagram과 각 component들의 class diagram으로 Frontend Architecture를 설명한다.

4.1. Overall Frontend Architecture

Frontend는 backend에 필요한 정보를 요청하고 요청받은 정보를 이용하여 사용자에게 웹페이지를 표시한다. 그리고 사용자에게 필요한 정보를 입력 받는다.

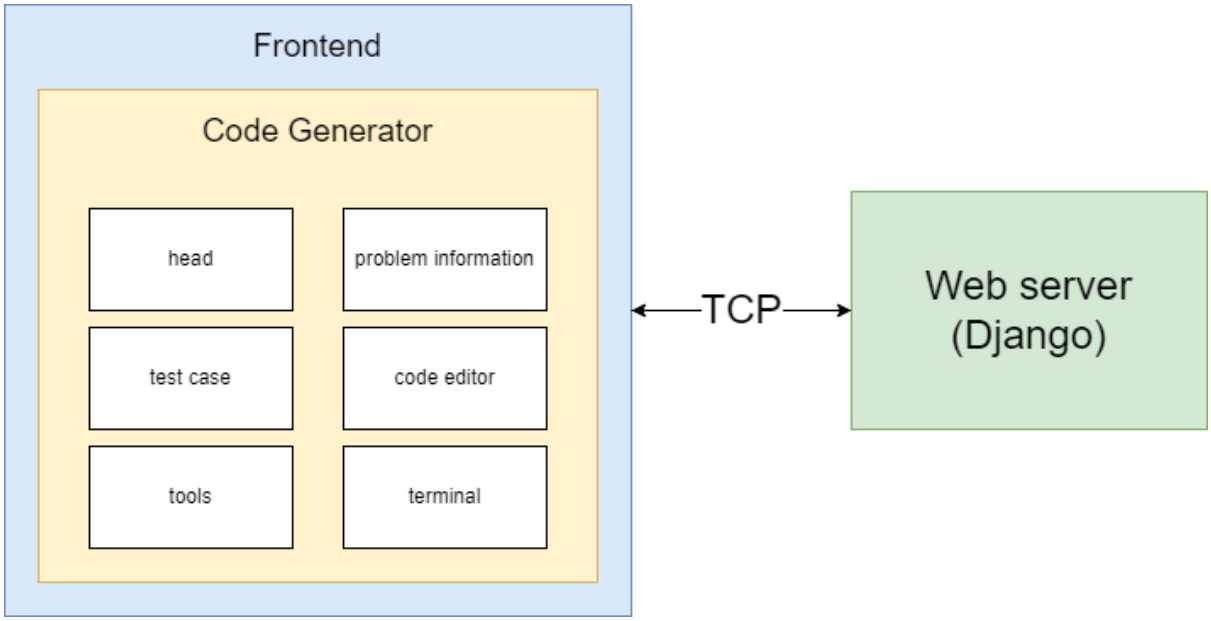

Diagram 4 Overall Frontend Architecture

4.2. Class diagram

코딩 테스트를 진행하는 웹페이지는 총 6가지 section으로 구성되며 각 section은 클래스로 관리된다.

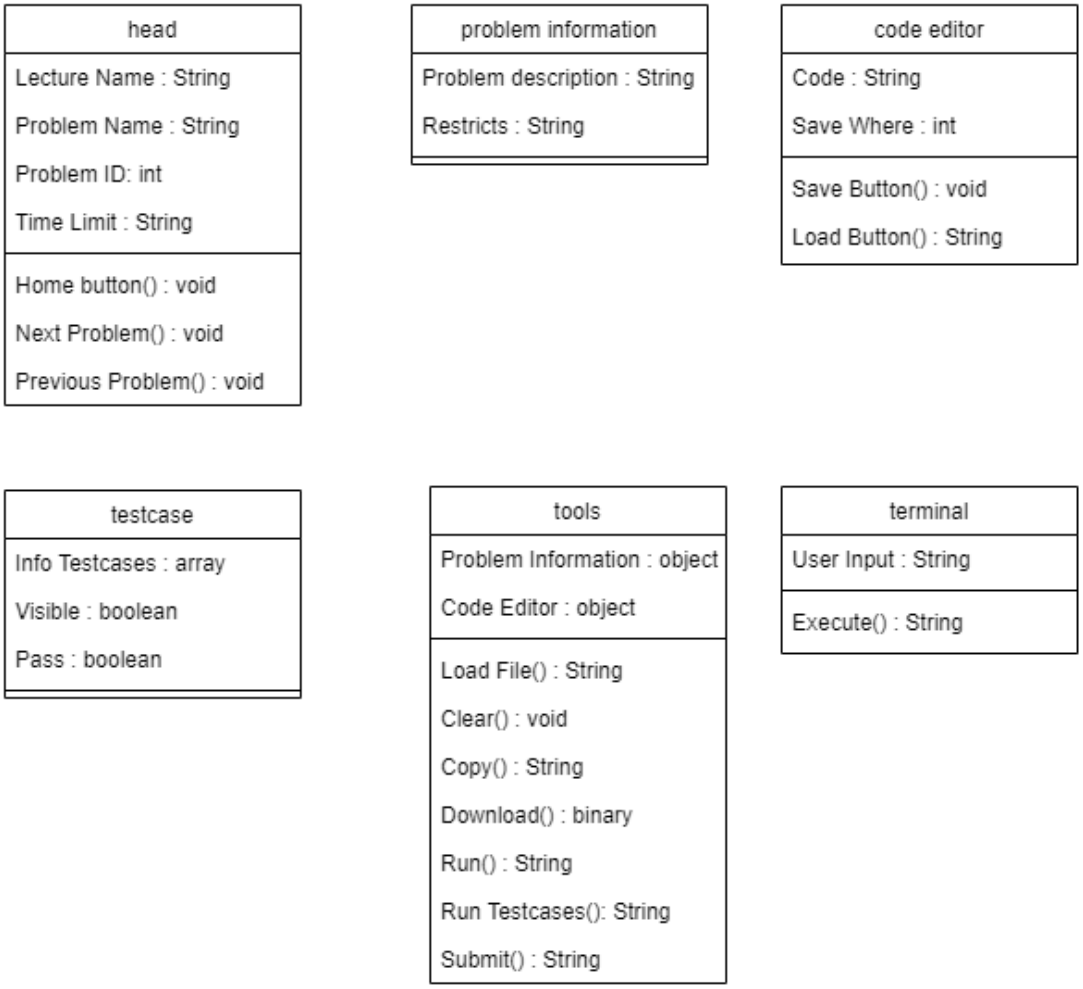

Diagram 5 Frontend class diagram

4.2.1. Head

4.2.1.1. Member variables

- Lecture Name: 수강중인 과목의 이름
- Problem Name: 수강중인 문제의 이름
- Time Limit: 수강중인 과목의 시간 제한

4.2.1.2. Member function

- Home button: Home 화면으로 돌아가는 버튼

- Next Problem: 다음 문제로 이동하는 버튼
- Previous Problem: 이전 문제로 이동하는 버튼

4.2.2. Problem information

4.2.2.1. Member variables

- Problem description: 수강중인 문제의 내용
- Restricts: 문제의 참조 및 제약사항

4.2.3. Code editor

4.2.3.1. Member variables

- Code: 사용자가 입력한 코드
- Save Where: 1,2,3 중 저장 위치

4.2.3.2. Member function

- Save Button: Save Where에 저장하는 버튼
- Load Button: 저장된 코드 불러오는 버튼

4.2.4. Testcase

4.2.4.1. Member variables

- Info Testcases: 테스트 케이스의 정보를 담는 배열, 배열의 각 원소는 문제의 입력과 출력을 저장하고 있음
- Visible: Pass/Fail을 표시할지 여부
- Pass: Pass 여부

4.2.5. Tools

4.2.5.1. Member variables

- Problem Information: problem information class의 참조 변수
- Code Editor: code editor class의 참조 변수

4.2.5.2. Member function

- Load File: 파일에서 코드 불러오는 버튼
- Clear: 입력된 사용자 코드 초기화 버튼
- Copy: 사용자 코드 클립보드에 복사
- Download: 사용자 코드 다운로드
- Run: 사용자 코드 실행
- Run Testcases: open testcase 채점
- Submit: 제출 버튼

4.2.6. Terminal

4.2.6.1. Member variables

- User Input: 사용자의 입력

4.2.6.2. Member function

- Execute: 사용자 입력을 실행하는 함수

4.3. State diagram

4.3.1. Problem information request

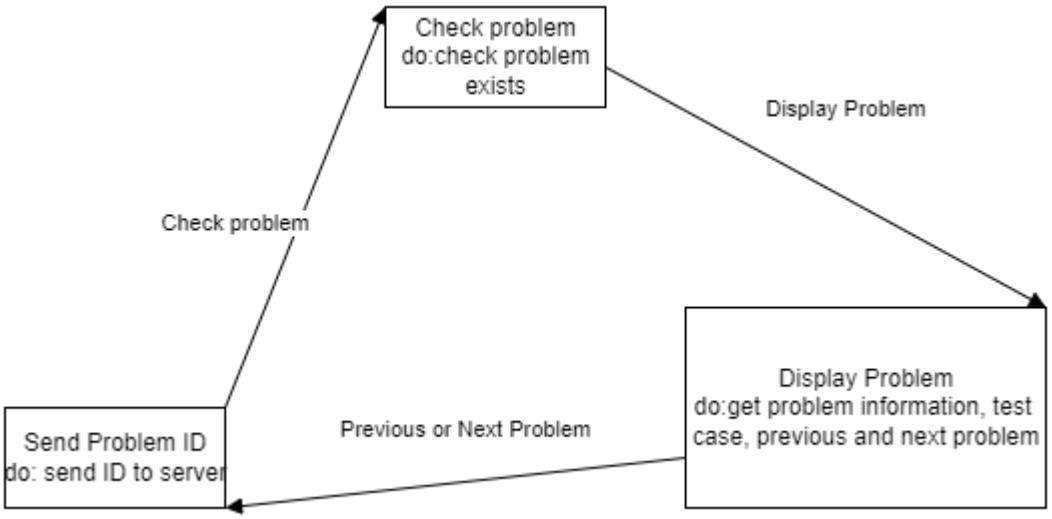

Diagram 6 State diagram - Problem information request

4.3.2. Run request

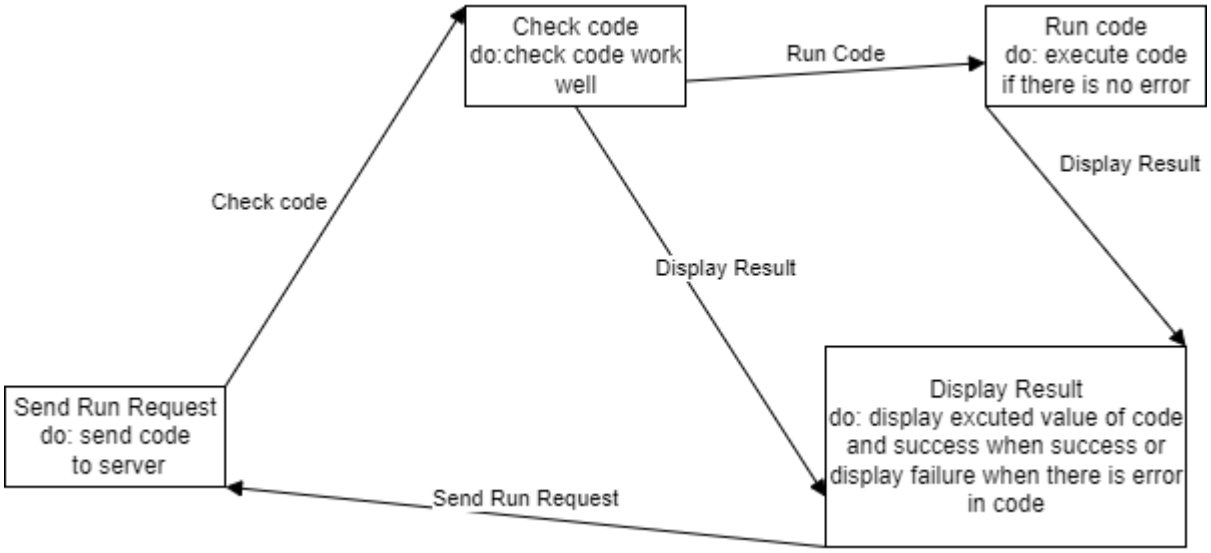

Diagram 7 State diagram – Run request

4.3.3. Run testcases request

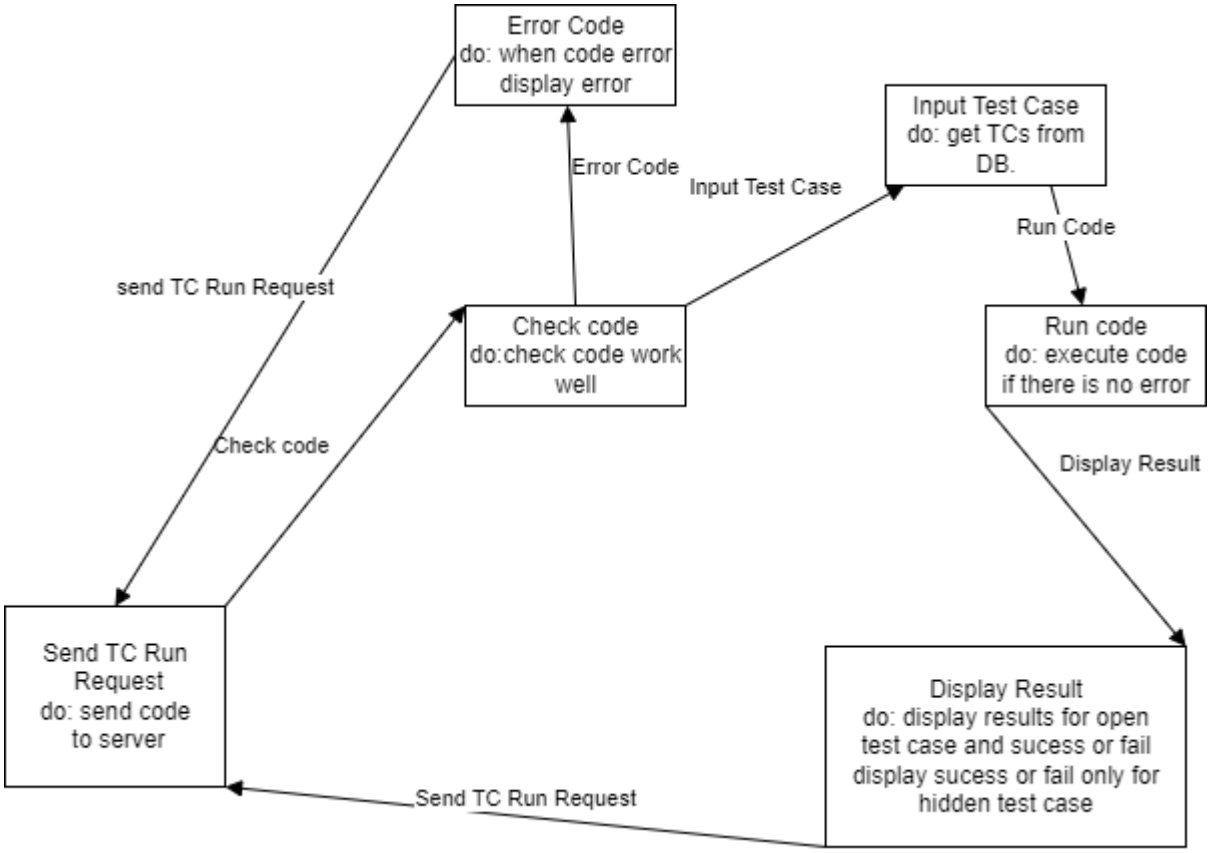

Diagram 8 State diagram – Run testcases request

4.3.4. Submit request

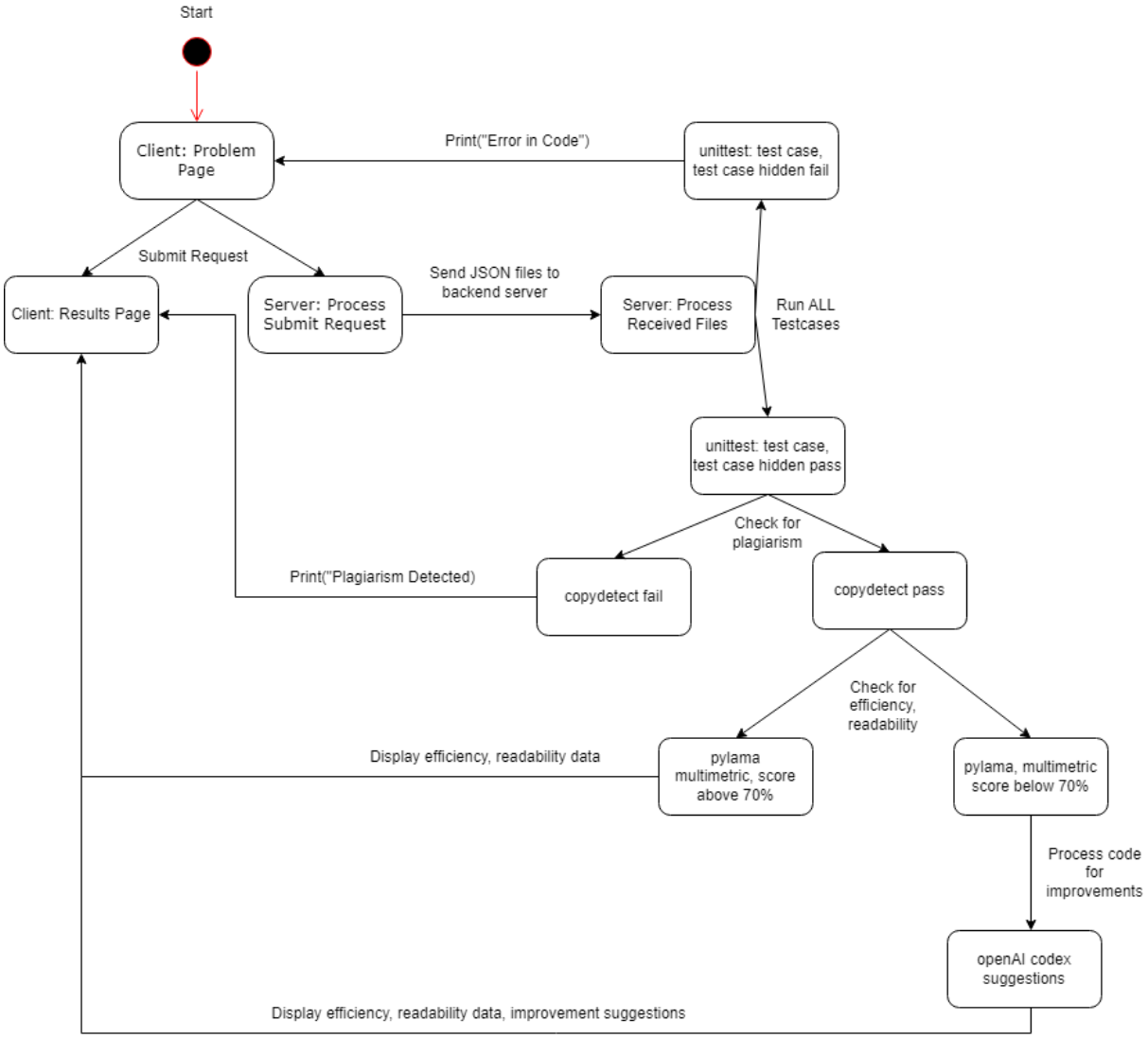

Diagram 9 State diagram – Submit request

4.3.5. Code save request

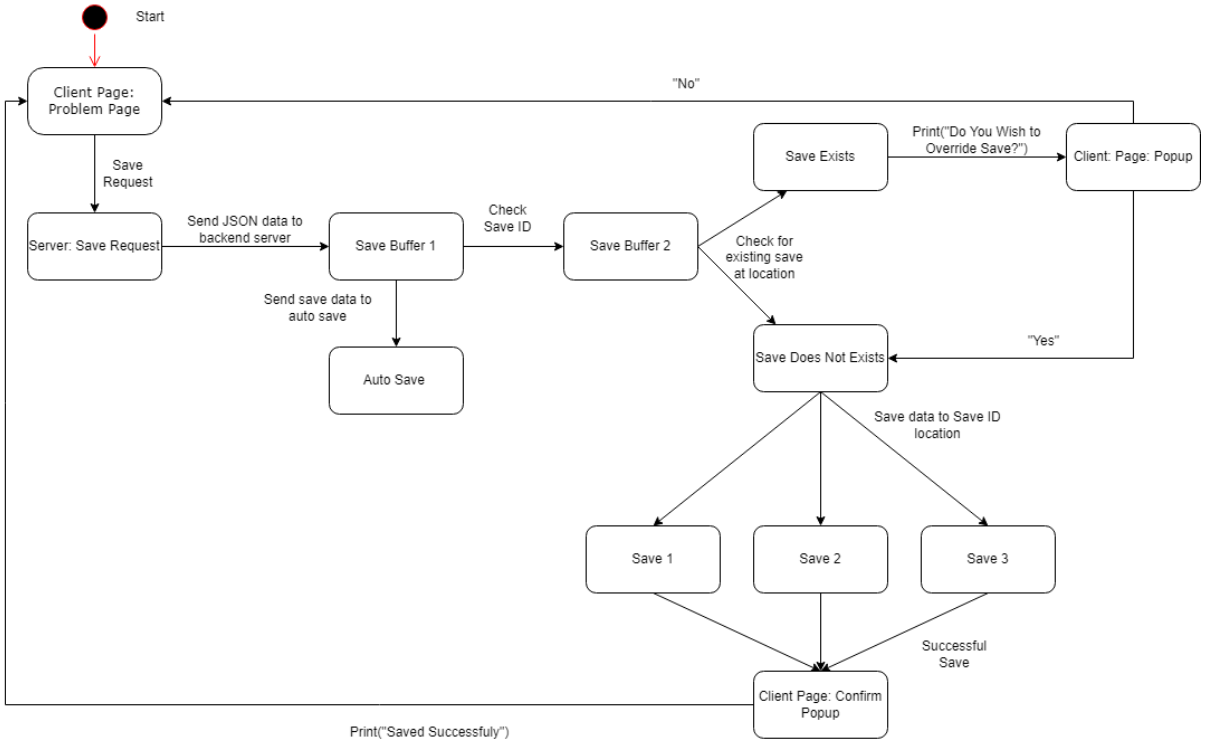

Diagram 10 State diagram – Code save request

4.3.6. Terminal Execute

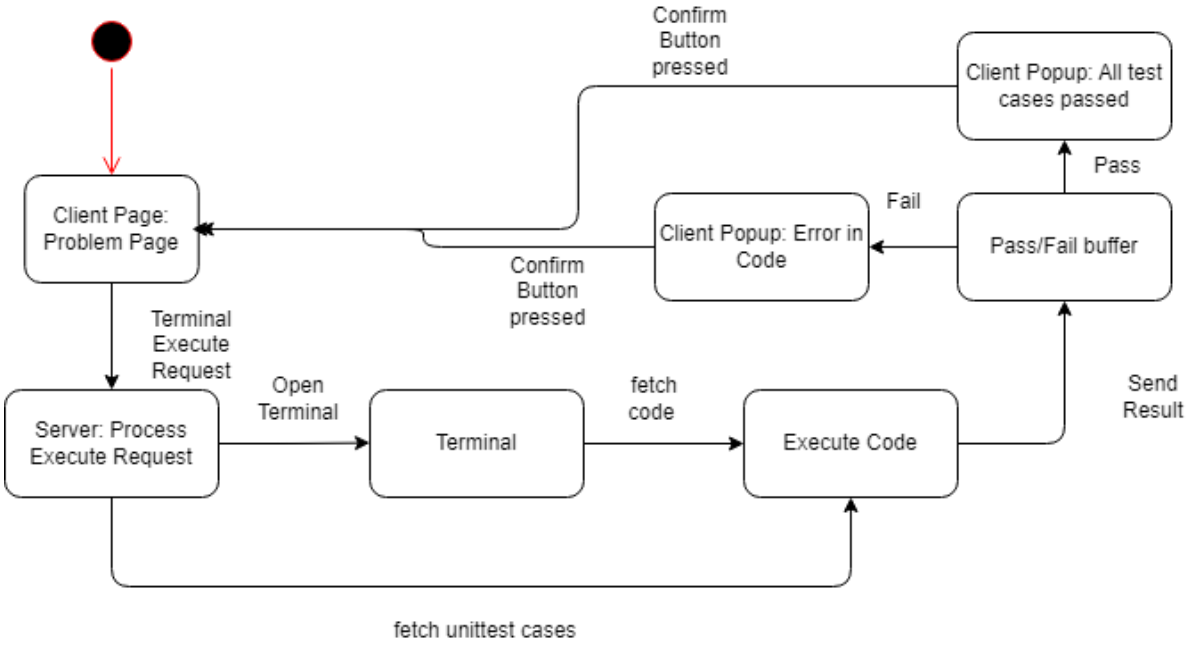

Diagram 11 State diagram – Terminal Execute request

5. System Architecture – Backend

5.1. objectives

데이터베이스, 파이썬 스크립트 실행, 사용자 페이지 렌더링 등 Backend의 전반적인 구조와 데이터 베이스, 스크립트 실행 등의 세부 시스템 구조를 설명한다.

5.2. Overall Architecture

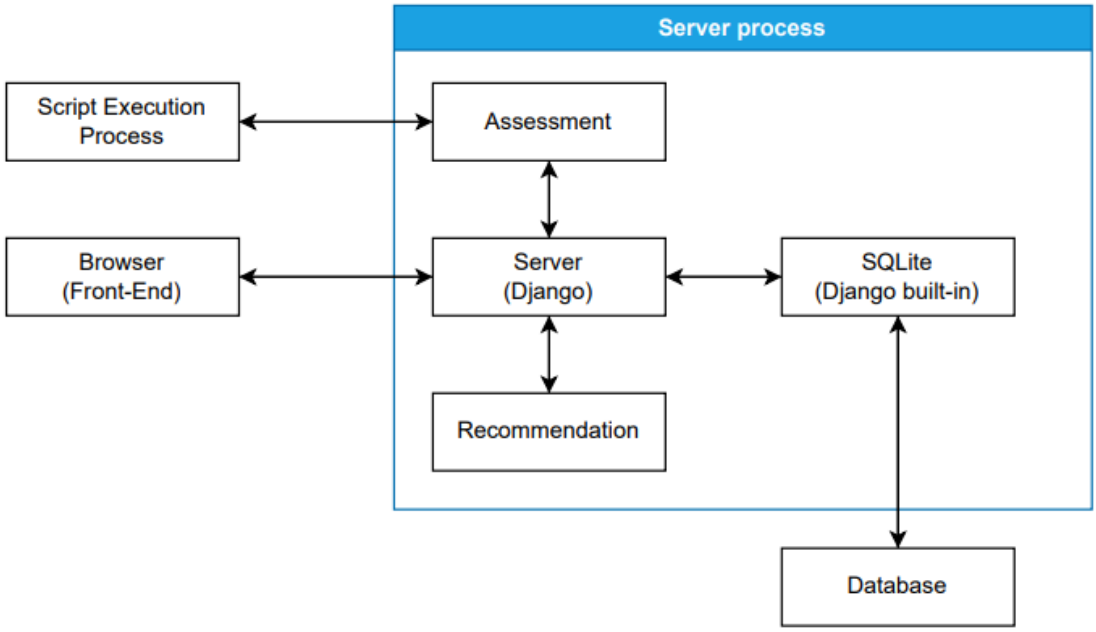

Diagram 12 Overall Backend Architecture

5.3. Overall procedure

데이터베이스는 Django에서 기본으로 제공하는 SQLite를 활용하여 SQL문을 이용하여 처리한다. 각 과정들은 상황에 맞게 아래에 있는 ER diagram에서 필요한 정보를 SQL문을 이용하여 front-end로 보내주거나 받은 정보를 이용하여 database를 수정하거나 새로운 출력 값을 만들어 front-end로 보내준다.

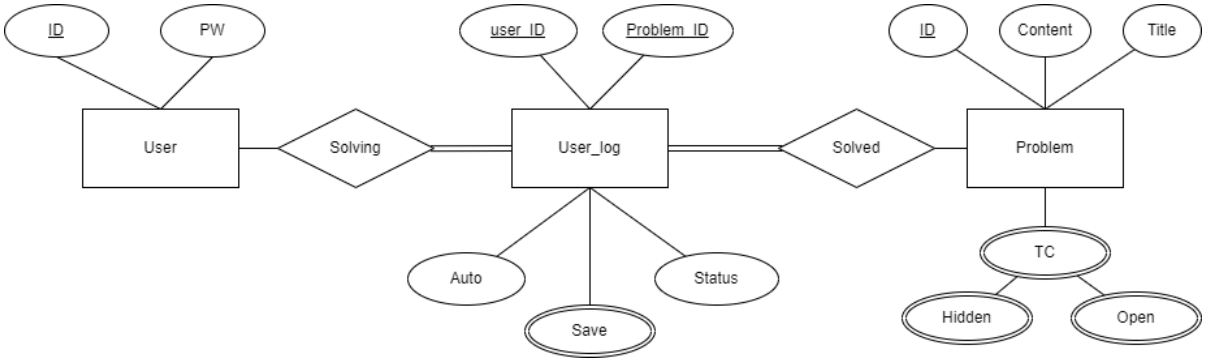

Diagram 13 Database ER diagram

5.3.1. Login process

로그인 성공후에 서버는 사용자에게 쿠키를 발급한다. 해당 쿠키를 이용하여 이후 사용자의 로그 인 정보를 저장한다. 서버는 로그인정보를 활용하여 페이지를 렌더링하며 발급한 쿠키가 유효기 간이 지나면 새로운 쿠키를 발급받기 위해서는 재로그인이 필요하다. 보안을 위해 DB에 저장되고 불러오는 비밀번호 역시 해싱을 진행하여 활용한다. 고로 사용자가 입력한 ID PW를 이용하여 생 성한 해싱된 값을 DB와 대조하여 로그인 성공여부를 결정한다.

5.3.2. Problem list

사용자가 로그인을 한 이후 가장 처음 보는 화면에 problem list들이 제공이 된다. backend에서는 특정 문제들의 id에 해당하는 problem[title]을 제공하고 사용자가 여기서 특정 문제를 선택하게 되면 main page로 이동하여 5.5~5.8의 기능을 수행하게 된다.

5.3.3. Problem information request

사용자가 5.4의 과정을 끝낸 뒤 main page로 이동하게 되면 사용자에게 문제에 대한 정보와 test case들을 제공해야한다. 여기서 문제에 대한 정보는 문제의 이름과 문제의 설명을 비롯해 참조/제약사항을 포함한다. 5.4에서 사용자가 문제를 선택하게 되면 해당 문제의 id를 포함한 problem에서 SQL문을 이용하여 desc, restrict, tc[open][input]과 tc[open][output]을 출력해준다. 이 모든 값들은 정적인 값이고 5.4를 통해 5.5로 넘어온 이후에는 다시 read되거나 write되지 않는다.

5.3.4. Save

Saved에 코드가 저장된다. 각 problem별로 saved에 3개의 슬롯을 포함하여 saved는 s1 s2 s3를 포함한다. 현재의 코드가 몇 번째 슬롯에 저장되는지는 front-end쪽에서 저장을 할 때 알려준다. Auto와 saved 모두 코드를 string의 형태로 저장하며, auto의 경우는 일정 시간마다 현재 코드 입력 창에 있는 코드를 저장하는 공간이다.

5.3.5. Run code / test examples

실행과 채점, 제출 기능이 backend에서 가장 중요한 기능이다. 모든 버튼을 누를 시 공통적으로 현재 코드 입력 칸에 있는 소스코드를 문자열의 형태로 front-end에서 제공받고, backend는 이 코드를 실행한다. 그 방법의 예로는 exec함수가 있다.

```
seouitae@NVLINK-V100:~/assignmnet/sgg$ cat test.py

str="helloworld"
print(str)

seouitae@NVLINK-V100:~/assignmnet/sgg$ cat execute.py
file_name = "test.py"
content = open(file_name).read()
exec(content)
seouitae@NVLINK-V100:~/assignmnet/sgg$ python3 execute.py
helloworld
seouitae@NVLINK-V100:~/assignmnet/sgg$
```

Figure 1 exec function example

위는 간단한 예제인데, exec함수로 쉽게 파이썬 파일의 결과를 출력하는 것을 볼 수 있다. 이런 식으로 출력 값을 뽑은 뒤 출력 값을 front-end에 다시 보내주면 된다. 파이썬 코드의 경우는 스켈레톤 코드가 제공되어 사용자는 특정함수의 기능을 함수의 argument와 return value에 맞게 구현할 것이므로 실행 버튼의 경우는 첫 번째 open test case를 input으로 하여 출력 값을 보내주고, 채점버튼의 경우는 제공된 모든 open test case와 hidden test case에 대해 pass or fail을 출력하여 보내준다. 에러가 발생하였을 경우 에러 메시지를 front-end로 보내준다.

5.3.6. Submit

제출버튼을 누르면 정답코드와 diff된 결과와 함께 제출결과를 보여준다. 제출결과에는 표절검사, 기능 채점, 효율 채점, 가독성 채점이 포함되며 OpenAI Codex API를 이용하여 코드에 대한 설명을 출력한다. 또한 관련 자료를 추천한다. 이 경우는 각 problem의 title에 “python” 혹은 “백준”, “알고리즘” 등의 키워드를 추가하여 만든 문자열을 google에 검색하고 그 링크들을 가져와 보내주는 방

법이 있다.

6. Protocol

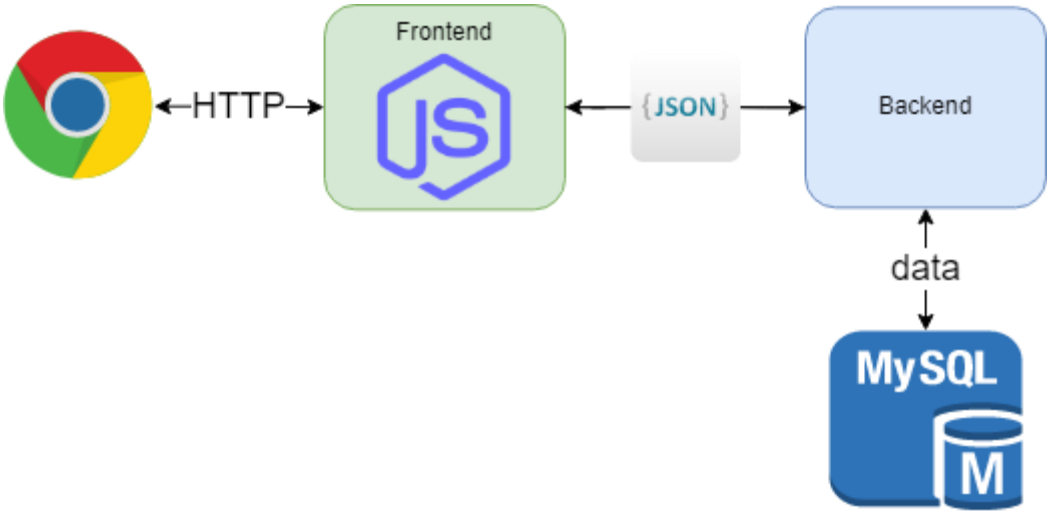

Figure 2 Overall Protocol Structure

6.1. HTTP

HTTP(HyperText Transfer Protocol)은 Hypertext 형식의 데이터를 빠르게 교환하기 위한 프로토콜의 일종으로 클라이언트가 요청을 하면 서버가 응답하는 구조로 되어 있다. 요청(request)과 응답(response)으로 구성되어 있으며, 서버와 클라이언트 사이의 이루어지는 연결과 통신이 HTTP를 통해서 이루어진다. 이 때 주고받는 문서는 HTML(Hyper Text Markup Language)의 형식을 따르며 해당 시스템에서는 EJS를 통해서 HTML 형식을 구현해 내고 있다.

6.2. NodeJS

NodeJS는 네트워크 애플리케이션 개발에 사용되는 소프트웨어 플랫폼으로, 자바스크립트를 작성 언어로 사용하며 Non-blocking I/O와 단일 스레드 이벤트 루프를 통한 높은 처리 성능을 가지고 있다. 내장 HTTP 서버 라이브러리를 포함하고 있어서 웹 서버를 동작하기에 알맞으며 이벤트 기반으로 시스템이 작동한다.

6.3. EJS

EJS(Embedded JavaScript)는 자바스크립트를 HTML내에서 구현할 수 있도록 도와주는 템플릿 엔진으로, 클라이언트의 요청에 따라서 달라지는 동적인 결과를 파일에 담을 수 있도록 도와준다. 자바스크립트의 연산 결과를 손쉽게 HTML 파일로 넣을 수 있다는 장점이 있다.

6.4. JSON

JSON은 JavaScript Object Notation의 약자로 Name/Value 쌍으로 이루어진 경량의 data 교환 형식이다. 사람이 읽을 수 있는 텍스트라서 간단하고, 자료의 종류에 큰 제한이 없으며, 플랫폼이나, 프로그래밍 언어에 독립적이므로 다양한 언어에서 쉽게 사용할 수 있다. 따라서, 서버와 웹 애플리케이션이 data를 교환할 때 사용된다. 해당 시스템에서는 Frontend와 Backend의 interaction에 사용된다.

7. Database Design

7.1. Overall procedure

요구사항 명세서에서 작성된 내용을 바탕으로 database design을 기술한다. 변경사항으로, 불필요하게 분산되었다고 판단되는 Status, Save Status, Problem log를 통합하고, 각 테이블에서 불필요하다고 판단되는 요소를 제거하였다. User, Problem, User_log로 이루어진다. ‘User_log’ 테이블은 User의 ID와 Problem의 ID를 foreign key로 가지고 User가 저장한 코드, 현재 상태 등을 저장한다.

7.2. ER Diagram

해당 시스템에서는 User, Problem, User_log 총 세 개의 entity가 존재한다. Entity는 직사각형, Entity간의 관계는 마름모, 전체/부분 참여 여부를 이중선, 단일선으로 표현했다. Entity가 가지는 Attribute들은 타원형으로 표현되었고 multi attribute의 경우 이중 타원형으로 표현하였다. Primary Key는 밑줄을 그어 표시했으며, 각 Entity의 상세한 정보는 7.3에서 서술한다.

Solving relation의 경우 1명의 User에 대해 여러 문제에 대한 User_log가 존재할 수

있고, Solved relation 역시 하나의 Problem에 대한 User_log가 User별로 존재할 수 있다. 또한 User ID와 Problem ID를 각각 foreign key로 하여 composite key로 이용한다.

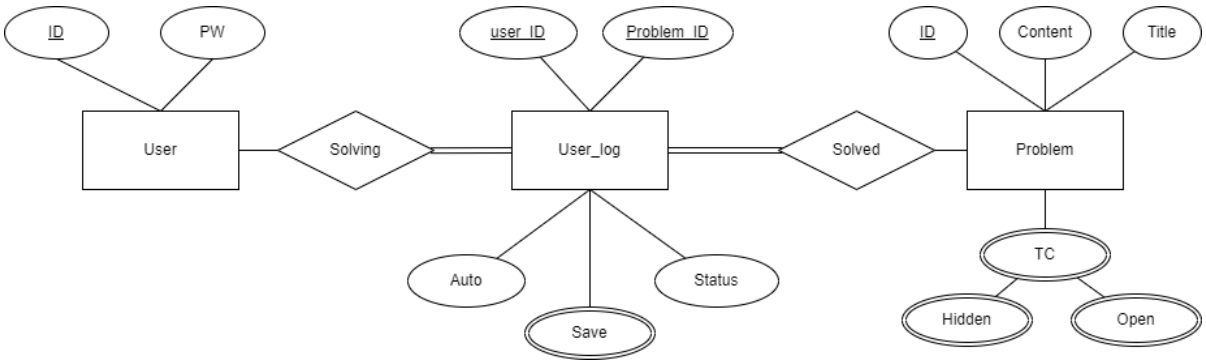

Diagram 14 Database ER Diagram

7.3. Entity Description

7.3.1. User

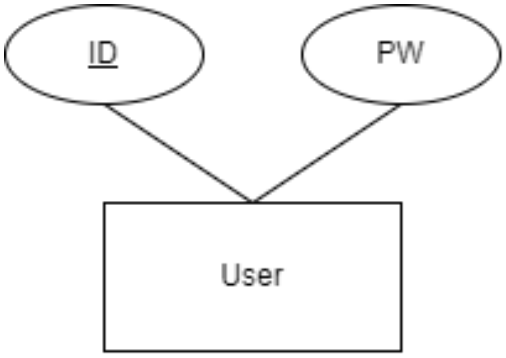

Diagram 15 Entity Diagram - User

사용자의 ID, Password를 저장한다. ID를 primary key로 사용한다.

7.3.2. Problem

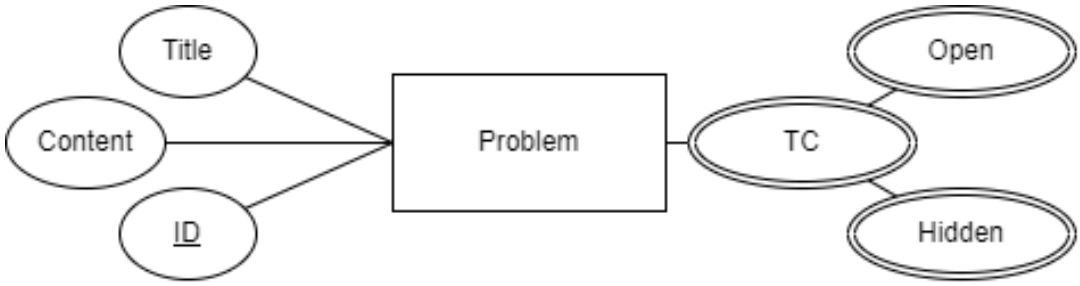

Diagram 16 Entity Diagram - Problem

ID, Contents, Title, Testcase를 저장한다. ID를 primary key로 사용하고, content에는 문제, 제약사항 등을 포함한 문제 내용을 저장하고, Title에는 사용자에게 문제 리스트로 제공하는 제목을 저장한다. Testcase의 경우 입력과 출력으로 이루어져 있고 여러 개의 정보를 저장하는데, 사용자에게 입력, 출력이 공개되는 Open case와 사용자에게 공개되지 않는 hidden case로 나누어진다.

7.3.3. User_log

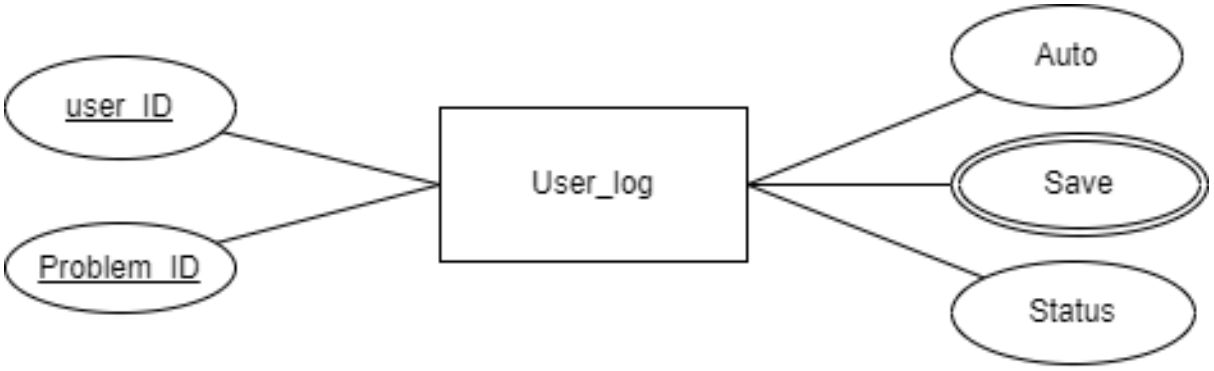

Diagram 17 Entity Diagram - User_log

User_ID, Problem_ID, Auto, Save, Status를 저장한다. User_ID와 Problem_ID는 해당 User_log 데이터가 어떤 사용자가 어떤 문제를 푼 기록인지 나타내고 User Entity와 Problem Entity의 ID attribute를 foreign key로 사용한다. 각 사용자는 하나의 문제에 대한 기록은 하나만 있으면 되므로 두 foreign key를 결합하여 composite primary key로 사용한다. Auto의 경우, 사용자와의 연결이 비정상적으로 끊어질 때를 대비하여 주기적으로 자동 저장되는 코드를 저장한다. Save의 경우, 사용자가 저장 요

청한 코드를 저장한다. 여러 개의 코드를 저장할 수 있으며 해당 서비스에는 최대 3개의 코드를 저장하도록 제한한다. Status의 경우 사용자가 해당문제를 해결하였는지, 아닌지 여부를 저장한다.

8. Index