Requirement Specification

소프트웨어 공학개론 9조

제출일	22.10.30	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. Objective

Preface는 문서를 읽을 독자를 정의하고, 문서의 전체적인 구조와 각 챕터에서 서술할 내용에 대해 소개한다.

1.2. Readership

본 시스템은 성균관대학교 학생들이 수강한 과목의 코딩 과제를 제출하는 것을 목표로 하는 코딩테스트 플랫폼이다. 교수는 사이트에서 문제를 등록할 수 있고 학생은 수강하는 과목의 과제를 선택하고 그에 대한 문제를 풀고 제출한다. 제출 결과에 대해 기능 점수, 효율 점수, 가독성 점수 확인을 통해서 피드백을 받고 관련 자료를 학습하는 과정을 통해서 학생들의 코딩 실력이 향상될 수 있도록 한다.

1.3. Document Structure

1.3.1. Preface

Preface는 문서를 읽을 독자를 정의하고, 문서의 전체적인 구조와 각 챕터에서 서술할 내용에 대해 소개한다.

1.3.2. Introduction

Introduction는 해당 시스템의 필요성을 확인하고 본 시스템이 제공하는 기능들에 대해 서술한다. 또한, 기존에 존재하는 코딩테스트 플랫폼의 특징에 대해 설명한다.

1.3.3. Glossary

Glossary는 본문의 이해하기 어려운 용어를 정리하고 설명하여 배경지식이 없는 독자도 내용을 이해할 수 있도록 한다.

1.3.4. User Requirement Definition

User Requirement Definition는 본 시스템에 대한 Functional Requirement와 Nonfunctional Requirement를 서술하고 사용자의 requirements를 파악하고 정의함으로써 본 시스템의 기능들을 사용자의 관점에서 서술한다.

1.3.5. System Architecture

System Architecture는 본 시스템의 아키텍처에 대해 설명한다. 시스템을 구성하고 있는 컴포넌트들을 정의하고 역할과 컴포넌트들끼리 상호작용하는 과정을 시각적인 자료를 사용하여 설명함으로써 독자가 시스템의 전체적인 구조를 이해하는 것에 목적을 둔다.

1.3.6. System Requirements Specification

System Requirements Specification는 User Requirement Definition의 내용을 바탕으로 본 시스템을 구성하는 각 기능들에 대해서 Functional Requirement와 Nonfunctional Requirement를 상세히 서술함으로써 요구 사항을 구현한다.

1.3.7. System Model

System Model는 본 시스템의 기능과 해당 기능을 구성하는 컴포넌트와 컴포넌트 간의 관계와 상호작용하는 과정을 다양한 모델을 사용하여 설명함으로써 독자가 본 시스템에 대한 이해도를 높이는 것을 목적으로 한다.

1.3.8. System Evolution

System Evolution는 본 시스템에 사용된 가정과 한계에 대해 서술하고 변화하는 고객의 요구사항과 환경적 변화에 대해 예상되는 시스템 변경에 대해 서술한다. 시스템의 미래에 대해 예측함으로써 설계 과정에서의 한계점을 보완한다.

1.3.9. Index

Index는 본 문서에 사용된 표와 그림에 대한 인덱스에 대한 설명을 제공한다.

2. Introduction

2.1 Objective

Introduction 은 본 시스템의 필요성과 제공하는 기능들에 대하여 서술한다. 또한 추가로 기존에 존재하는 코딩 교육시스템 및 Online judge 시스템의 특징에 대해 서술한다.

2.2 Needs

학교 자체 코딩교육시스템, OJ 시스템의 부재로 인하여 성균관대학교에서는 OJ 시스템이 필요한 과목의 경우 구름과의 계약을 통해 구름 EDU 를 사용해 강의를 진행해 오고 있다. 외부 업체와의 계약을 학교에서 관리하는 OJ 시스템 기능으로 대체할 경우 비용 절감을 통해 다른 분야에 교육비를 더 투자할 수 있다.

학교는 외부의 OJ 시스템에 의존하는 현 상황을 개선하기 위해 NPC 를 통해 자체 PS 플랫폼 제작을 시도하고, 산학협력과제로 코딩 플랫폼 제작을 하는 등 다양한 시도를 해왔다. 이 프로젝트 또한 교내 자체 코딩 플랫폼 제작 시도의 일환이다.

기존 코딩 플랫폼에서는 문제에 대한 분류만을 제공하고 추가적인 학습자료나 학습자가 작성한 코드에 대한 피드백은 하지 못하고 있다. 우리는 학습자에게 추가적인 피드백을 자체적으로 제공하여 학습에 대한 능률을 끌어올릴 것이다. 추가 학습자료는 크롤링을 통하여 적절한 학습 자료를 제공한다. 사용자가 제출한 코드는 open API 를 사용하여 AI 기반으로 평가될 것이며 사용자에게 적절한 피드백을 제공할 수 있을 것이다.

2.3 Exist System Analysis

코딩 플랫폼, Online Judge 서비스들의 특징을 비교한다. 우리가 제작하려는 코딩플랫폼은 기존에 존재하는 웹 서비스를 바탕으로 제작하는 것이기에 이미 시장에 출시되어 있는 코딩플랫폼의 특징에 대해 소개할 것이다.

2.3.1 백준 Online Judge

Figure 1. 백준

백준 OJ 는 국내 최대 규모 알고리즘 사이트이다. 특징으로는 표준 입출력 기반 풀이를 작성하며 50 가지 프로그래밍 언어를 지원한다. 단계별, 카테고리 별 분류된 문제를 제공하고 있으며 문제 풀이 성공과 실패 외에도 소요 시간과 메모리 사용량 등 다양한 평가 결과를 제공한다. 국내 알고리즘 대회 플랫폼으로도 많이 쓰이고 있다.

2.3.2 Programmers

Figure 2. 프로그래머스

네이버, 카카오 등 여러 국내 기업의 코딩테스트 플랫폼으로 사용되고 있다. 입출력 기반의 백준과는 다르게 솔루션 함수 작성 기반 문제 풀이다. 12 가지 프로그래밍 언어를 제공하고 있으며 테스트 케이스 채점 결과를 사용자에게 알려주어 가시성을 높이기도 하였다.

2.3.3. CodeForce

Figure 3. 코드포스

백준과 프로그래머스가 일반적인 문제 풀이 기능에 중점을 두었다면 코드포스는 대회 준비에 최적화된 플랫폼이다. 매주 3 개 정도의 온라인 대회가 개최되며 알고리즘 대회를 대비하려는 사람들이 주로 참여한다.

2.3.4. LeetCode

Figure 4. 리트코드

구글, 페이스북, 아마존 등 세계적인 IT 기업의 코딩 테스트 및 면접 기출문제를 보유하고 있다. Weekly Contest 가 있으며 문제 난이도 분류로는 Easy, Normal, Hard 의 세가지 분류를 가지고 있다. 문제 풀이에 성공할 경우 전체 풀이 제출에 대비하여 내 풀이의 성능을 퍼센테이지로 평가해준다.

리트코드의 가장 큰 특징은 각 문제에 대한 커뮤니티가 활발하다는 점이다. 문제를 푼 후 discuss 탭을 통해 토의를 하거나 타인이 작성한 글을 참고하여 추가적인 학습을 할 수 있다.

2.4 Overview

기존 OJ 의 틀에서 벗어나 학생의 코드에 대한 피드백과 적합한 추가적인 학습자료를 제공하는 것이 가장 중요한 기능이다. 기본적인 OJ 기능 외에도 두 기능을 원활히 구현하는 것이 본 글의 목표이다.

Overview 에서는 본 프로젝트 결과물에 대한 사용법에 대해 설명한다. 웹에 접속을 한 뒤 로그인하게 되면 문제 리스트를 확인할 수 있다. 문제 리스트를 통하여 내가 원하는 문제풀이로 이동할 수 있다.

문제 풀이 페이지로 이동하면 학습자는 에디터를 사용한 코드 작성이 가능하다. 실행 버튼을 통해 작성 코드의 실행 결과를 얻을 수 있으며 채점을 통해 교수자가 등록한 테스트 케이스에 대한 정답을 확인할 수 있다. 마지막으로 제출을 통해 최종적인 정답을 확인할 수 있으며 이 때 다양한 metric 을 기반으로 하는 채점 결과를 시스템이 제공한다.

3. Glossary

3.1 Objective

Glossary는 본문의 이해하기 어려운 용어를 정리하고 설명하여 배경지식이 없는 독자도 내용을 이해할 수 있도록 한다.

3.2 Term Definition

01. 컨테이너

가상화 기술의 일종인 컨테이너는 개별 SW 의 실행에 필요한 실행환경을 독립적으로 제공해주는 기술이다. 컨테이너 안에서 각 SW 는 서로 다른 환경에서 실행되는 것처럼 작동한다. 컨테이너는 가볍고, 빠를 뿐 아니라 호스트 OS 안에서 여러 개를 만들 수 있다는 장점이 있다.

02. 프로그래밍 언어

컴퓨터에 명령이나 연산을 시킬 목적으로 설계된 언어이다. 즉 인간이 컴퓨터와 소통하기 위한 언어이다.

03. 파이썬

컴퓨터 프로그래밍 언어의 일종으로 영어 문법과 비슷해 읽고 쓰기 쉽다. 게다가 제공하는 기능이 풍부하기 때문에 세계적으로 널리 쓰이고 있다.

04. 테스트케이스

유저가 제출한 코드의 입력값이다.

05. 컴파일

프로그래밍 언어는 기계가 이해하지 못한다. 컴파일은 프로그래밍 언어를 컴퓨터가 이해 가능하고 실행 가능한 언어로 바꿔주는 과정이다.

06. Malicious user

시스템에 나쁜 영향을 끼치기 위해 의도적으로 시스템에 접근하는 유저이다.

07. 응답 시간

유저가 시스템에 입력을 준 후 그에 대한 반응을 얻기까지 걸리는 시간이다.

08. XP(extreme programming)

소프트웨어 개발 방법론의 일종이다.

09. Pair programming

두 사람이 짝을 이뤄 프로그래밍을 하는 방식이다. 한 명이 코드를 작성하고 한 명이 그에 대한 피드백을 하며, 주기적으로 역할을 바꿔준다.

10. BFS(Breadth-First Search)

코딩 테스트 문제의 종류 중 하나이다. 인접한 노드를 먼저 탐색하여 그래프의 모든 노드를 방문한다.

11. 데이터베이스

데이터베이스는 구조화된 정보 또는 데이터의 조직화된 모음이다. 데이터베이스 안에 시스템에 필요한 정보들을 모아둠으로써 관리한다.

12. 코드 에디터

코드 에디터(및 소스 코드 편집기)는 프로그래머에 의해 컴퓨터 프로그램의 소스 코드를 편집하기 위해 설계된 프로그램의 하나이다.

13. 라이트 모드

밝은 화면과 검은 글자로 이루어진 기본적인 사용자 인터페이스 테마.

14. 다크 모드

사용자 인터페이스에서 밝은 화면에 검은 글자 대신 어두운 화면에 흰 글자를 나타내는 테마.

15. 파일 확장자

컴퓨터 파일의 이름에서 파일의 종류와 그 역할을 표시하기 위해 사용하는 부분이다. 예를 들어 .py 는 파이썬 파일을 의미하며 .txt 는 텍스트 파일을 의미한다.

16. 스켈레톤 코드

하나의 프로그램이 동작하는 전체 과정을 한눈에 알아볼 수 있도록 그 구조를 나타낸 틀만 구현한 코드. 본 시스템에서는 사용자가 코드를 어떻게 작성할 지에 대한 방향을 제시해준다.

17. 라이브러리

라이브러리는 주로 소프트웨어를 개발할 때 컴퓨터 프로그램이 사용하는 비휘발성 자원의 모임이다. 여기에는 구성 데이터, 문서, 도움말 자료, 메시지 틀, 미리 작성할 코드, 서브루틴(함수), 클래스, 값, 자료형 사양을 포함할 수 있다.

18. 클립보드

파일을 복사하거나 잘라내기할 때 붙여넣기하기 전까지 파일을 임시로 저장해주는 수단

19. 문제

프로그래밍 역량을 시험하기 위한 알고리즘 구현 문제를 의미한다.

20. 채점

출제된 문제에 대한 점수를 제공하고, Pass/Fail 여부를 제공하는 것을 말한다.

21. 외부 자료

출제된 문제와 관련된 영상, 학습 자료 등을 말한다.

22. 학습자

웹사이트를 사용하여 주어진 문제를 풀고, 작성한 코드를 제출하고, 제출한 코드의 분석을 열람할 수 있는 사용자이다.

23. 교수자

웹사이트에 문제를 제출하고, 테스트 케이스를 등록할 수 있는 사용자이다.

24. 채점자

웹사이트를 사용하여 학습자가 제출한 코드와 채점 결과를 확인할 수 있는 사용자이다. 교수자와 동일 인물일 수 있다.

25. OJ, Online Judge

프로그래밍 대회에서 프로그램들을 시험할 목적으로 만들어진 온라인 시스템이다. 대회 연습용으로 사용되기도 한다. 시스템은 코드를 컴파일하여 실행하고, 미리 작성된 데이터로 테스트 하게 된다. 제출된 코드는 시간, 메모리, 보안 등에 관련된 제한을 받을 수 있다. 시스템은 출력 데이터를 받아 표준 출력과 비교한 뒤, 결과를 내어 준다.

26. 크롤링

웹 페이지를 그대로 가져와서 거기서 데이터를 추출해 내는 행위다. **Open API:API 는** 응용프로그램이나 서비스를 개발하는데 필요한 운영체제(OS)나 라이브러리 등의 특정 기능을 추상화하여 사용하기 쉽도록 만든 인터페이스.

27. System model

요구사항을 분석 중인 시스템에 대한 **추상화된** 표현

28. Sequence Diagram

시계열로 정렬된 객체 상호작용을 보여준다. 시나리오 기능을 수행하는데 필수적인 객체들 간에 교환되는 일련의 메시지들과 시나리오에 수반되는 객체와 클래스를 표현한다.

29. Tuple

데이터베이스 내에 저장된 각 엔트리를 의미한다.

30. Syntax highlighter module

학습자가 작성한 문자열을 파이썬 코드로서 인식하는 모듈이다. 파이썬의 특정 키워드를 인식하여 색상 등으로 강조처리를 해준다.

4. User Requirement Definition

4.1 Objective

User requirement definition 에서는 사용자의 requirements 를 파악하고 정의함으로써 시스템이 제공해야 할 기능들을 사용자의 관점에서 서술한다. 서술 형식은 form-based 를 따른다. 본 장에서 정리한 user requirement 는 추후 소개할 system requirement 를 비롯한 시스템의 전체적인 기능과 구조를 확립하는 데 활용한다.

4.2 Functional Requirement

4.2.1 회원가입

사용자는 프로그램의 회원으로 등록되어 있지 않다면 자신의 이메일, 아이디, 비밀번호, 이름 등을 사용하여 회원가입을 하여야 한다. 앞에서 사용한 정보들은 이후 사용자들을 식별하는 데 사용되며 모두 데이터베이스에 저장된다. 입력한 아이디와 비밀번호는 로그인 시 사용된다.

4.2.2 로그인

사용자가 프로그램의 회원으로 등록되어 있다면 회원가입 시 기재한 아이디와 비밀번호를 통해 로그인을 할 수 있어야 한다. 로그인하게 된다면 프로그램의 메인페이지로 진입할 수 있어야 하며 이후 프로그램이 제공하는 기능들은 정상적으로 이용할 수 있어야 한다.

4.2.3 로그아웃

사용자가 로그인 상태에서 벗어나고 싶다면 로그아웃을 하여 페이지에 로그인된 회원 정보가 남아있지 않을 수 있어야 한다. 로그아웃한다면 프로그램의 초기 페이지로 진입하게 되며 이후 다시 로그인해야 프로그램의 기능들을 이용할 수 있어야 한다.

4.2.4 문제 선택

사용자는 자신이 풀 문제를 선택할 수 있어야 한다. 문제에 따라 보여지는 화면이 달라야 하며 클릭과 같은 간단한 조작만으로 문제를 선택할 수 있어야 한다.

4.2.5 선택한 문제, 참조, 그리고 제약사항 확인

사용자는 자신이 선택한 문제의 문제 내용과 참조내용 그리고 제약사항을 확인할 수 있어야 한다. 문제와 참조, 제약사항은 알아보기 쉽도록 구분되어야 하며 이해할 수 있는 언어로 서술되어야 한다.

4.2.6 문제 유의사항 확인

4.2.5 에서 충분히 서술되지 않은, 사용자가 헷갈릴 수 있는 유의 사항이 존재한다면 이 또한 사용자가 확인할 수 있어야 한다. 해당 내용은 알아보기 쉬운 색상, 위치 등의 속성을 가지고 보여져야 하며 마찬가지로 이해할 수 있는 언어로 서술되어야 한다.

4.2.7 코드 작성

사용자는 프로그램의 코드 에디터 구역에서 코드를 작성할 수 있어야 한다. 언어는 파이썬을 고정으로 하며 키보드를 통한 입력이 이루어지면 화면을 통해 입력된 코드가 출력되어야 한다. 코드의 각 줄은 선으로 구분되어야 하며 작성한 코드는 알아보기 쉬워야 한다.

4.2.8 예시 테스트 케이스 확인

해당 문제에 대한 예시 테스트 케이스를 사용자는 확인할 수 있어야 한다. 테스트 케이스의 input 값과 기대되는 output 값, 예시 테스트케이스의 번호를 확인할 수 있어야 한다.

4.2.9 예시 테스트 케이스를 통한 검증

사용자는 작성한 코드를 예시(오픈) 테스트 케이스를 통해 검증할 수 있어야 한다. 이때 '검증'이란 선택한 예시 테스트 케이스의 Input 값을 작성한 코드에 input 으로 주었을 때 나온 output 이 예시 테스트 케이스의 기대되는 output 값과 일치하는지 확인하는 행위이다. 사용자는 검증의 결과를 확인할 수 있어야 하며 횟수 제한 없이 검증이 가능하여야 한다.

4.2.10 문제 마감 기한 확인

사용자는 문제의 마감 기한을 확인할 수 있어야 한다. 사용자의 현재 시각부터 해당 문제의 마감 기한까지 얼마나 남았는지 확인할 수 있어야 하며 만약 마감 기한을 넘겼다면 그 여부 또한 알 수 있어야 한다.

4.2.11 프로그램의 색 변경

사용자는 프로그램의 색을 자신의 선호에 맞게 변경할 수 있어야 한다. 프로그램은 두 개의 색상 모드(light mode, dark mode)를 제공하고 있으며 사용자가 선택한 모드에 따라 프로그램의 배경색을 비롯한 프로그램 구성요소의 색이 바뀌어야 한다.

4.2.12 작성중인 코드의 글자 크기 변경

사용자는 코드 에디터의 작성 중인 코드의 글자 크기를 변경할 수 있어야 한다. 사용자가 설정한 크기에 따라 글자의 크기가 바뀌어야 한다.

4.2.13 코드 다운로드

사용자는 코드 에디터에서 작성 중인 코드를 자신의 로컬 컴퓨터에 파일 형태로 다운로드 받을 수 있어야 한다. 다운받은 파일은 .py 혹은 .txt 확장자 중 하나이며 글자의 손상 없이 안전하게 파일을 다운로드 받을 수 있어야 한다. 다운로드 받은 파일은 로컬 컴퓨터에서 문제없이 열릴 수 있어야 한다.

4.2.14 코드 초기화

사용자는 코드 에디터에서 작성 중인 코드를 초기 skeleton 코드로 초기화할 수 있어야 한다. 기존에 작성 중이던 코드의 정보는 삭제되고 데이터베이스에 저장된 해당 문제의 skeleton 코드가 코드 에디터에 나타나야 한다.

4.2.15 코드 초기화 경고 창

사용자는 코드 초기화를 하기 전 정말로 코드 초기화 작업을 진행할 것인지 경고 문구를 볼 수 있어야 한다. 이때 '네'라는 신호를 보내면 코드 초기화가 정상적으로 진행되며 '아니오'라는 신호를 보내면 코드 초기화 작업은 취소되고 기존에 작성 중이던 코드로 돌아올 수 있어야 한다.

4.2.16 코드 복사

사용자는 코드 에디터에서 작성 중인 코드를 클립보드로 복사할 수 있어야 한다. 복사한 코드는 나중에 클립보드에서 불러오기 작업을 하였을 시 무사히 손상 없이 불러올 수 있어야 한다.

4.2.17 코드 불러오기

사용자는 자신의 로컬 컴퓨터에 있는 코드를 프로그램의 코드 에디터로 불러올 수 있어야 한다. 파일에 있는 코드는 손상 없이 코드 에디터에 나타나야 한다.

4.2.18 실시간 코드 저장

사용자가 코드 에디터에 작성된 자신의 코드를 따로 저장하지 않아도 프로그램이 자동으로 저장할 수 있어야 한다. 프로그램이 예기치 못하게 종료되거나 페이지가 재렌더링되어도 사용자가 수동으로 저장한 코드가 아닌 프로그램이 자동으로 데이터베이스에 저장한 코드가 보여야 한다.

4.2.19 코드 실행

사용자는 코드 에디터에 작성된 자신의 코드를 실행할 수 있어야 한다. 실행한 결과 또한 확인할 수 있어야 하며 코드에는 반드시 input 값을 포함해야 한다. 만약 input 값을

포함하지 않았거나 코드에 오류가 있을 경우 에러 메시지 또한 확인할 수 있어야 한다. 또한 코드의 어느 부분에서 에러가 나왔는지 코드 라인의 색의 구분을 통해 명확히 확인 가능하여야 한다.

4.2.20 코드 채점

사용자는 코드 에디터에 작성된 자신의 코드를 채점할 수 있어야 한다. 이때 '채점'이란 해당 문제의 예시 테스트 케이스를 포함한 문제의 모든 테스트 케이스를 작성된 코드를 통해 검증하는 것이며 이에 따른 결과를 사용자는 확인할 수 있어야 한다. 채점 결과는 테스트케이스 각각의 검증 결과를 말하며 만약 예시(오픈) 테스트 케이스에서 fail 했을 경우 테스트 케이스의 정보를 제공하고 하든 테스트케이스에서 fail 했을 경우에는 테스트 케이스의 정보를 제공하지 않는다. 채점 결과에 따른 총점도 확인할 수 있어야 한다.

4.2.21 코드 제출

사용자는 코드 에디터에 작성된 코드를 제출할 수 있어야 한다. 이때 '제출'이란 해당 코드를 채점 받고 코드와 문제에 관련된 다양한 점수와 자료를 받는 것이다. 또한 정답 코드와 제출한 코드 간의 차이를 확인할 수 있어야 한다.

4.2.22 code diff 확인

사용자가 제출하게 된다면 제출한 코드와 데이터베이스에 저장된 해당 문제의 정답 코드를 비교할 수 있어야 한다. 이때 비교는 코드의 라인을 단위로 이루어지게 되며 차이가 있는 부분의 코드만 반복해서 보여지며 나머지 동일한 코드는 중복되어서 나타나지 않는다. 상반되는 두 색깔을 통해 사용자는 차이점을 맨눈으로 쉽게 확인할 수 있어야 한다. 만약 사용자가 코드 입력창을 확대한다면 정답 코드와 제출한 코드가 각각 따로 다른 구역에서 보여야 한다.

4.2.23 과거 제출 코드 불러오기

사용자는 과거에 자신이 해당 문제에 제출한 코드가 있다면 그 코드를 다시 불러올 수 있어야 한다. 제출한 코드와 연관된 제출 결과 또한 같이 불러올 수 있어야 하며 코드는 코드 에디터 섹션에서 보여야 한다.

4.2.24 제출 코드 기능 점수 확인

사용자는 자신이 제출한 코드에 대한 제출 결과 중 하나로 코드의 기능 점수를 확인할 수 있어야 한다. 코드의 기능 점수는 정해진 여러 가지 기준(라이브러리)에 따라 채점되며 이에 따른 점수는 모두 보여져야 한다.

4.2.25 제출 코드 효율 점수 확인

사용자는 자신이 제출한 코드에 대한 제출 결과 중 하나로 코드의 효율 점수를 확인할 수 있어야 한다. 코드의 효율 점수는 정해진 여러 가지 기준(라이브러리)에 따라 채점되며 이에 따른 점수는 모두 보여져야 한다.

4.2.26 제출 코드 가독성 점수 확인

사용자는 자신이 제출한 코드에 대한 제출 결과 중 하나로 코드의 가독성 점수를 확인할 수 있어야 한다. 코드의 가독성 점수는 정해진 여러 가지 기준(라이브러리)에 따라 채점되며 이에 따른 점수는 모두 보여져야 한다.

4.2.27 제출 코드 설명 확인

사용자는 자신이 제출한 코드를 Open AI Codex 를 사용해 분석한 설명을 제출 결과 중 하나로 확인할 수 있어야 한다. 사용자는 이 설명에 대해 좋고 나쁨을 표시할 수 있어야 한다.

4.2.28 해당 문제 관련 자료 확인

사용자는 자신이 해당 문제와 관련된 자료를 제출 결과 중 하나로 확인할 수 있어야 한다. 관련 자료는 연관 문제, 영상, 학습자료 등이며 하이퍼 링크 형태로 다른 외부 페이지와 연결될 수도 있다.

4.2.29 제출 코드 표절률 확인

사용자는 자신이 제출한 코드와 해당 문제의 정답 표본 코드를 비교한 표절률을 제출 결과 중 하나로 확인할 수 있어야 한다.

4.2.30 제출 횟수 제한

사용자의 코드 제출 횟수는 3 번으로 제한되어야 한다.

4.3 Nonfunctional Requirement

4.3.1 Product Requirements

A. Usability Requirements

시스템은 사용자가 접근하기 쉬워야 한다. 또한 사용자가 사용하기 쉬운 디자인을 가져야 하며 직관적인 화면 구성으로 처음 사용하는 사용자도 2 시간이면 시스템에 익숙해질 수 있을 정도의 usability 를 가져야 한다.

B. Efficiency Requirements

시스템은 효율적으로 작동해야 한다. 사용자의 입력이 들어온 후 입력에 따른 결과를 사용자가 인식하기까지 15 초를 넘겨서는 안 되며 사용자의 액션에 따라 생성되는 데이터는 메모리에 효율적으로 저장되어야 한다.

C. Dependability Requirements

시스템은 높은 dependability 를 가져야 한다. 사용자가 프로그램을 사용하는 시간당 프로그램이 결함을 일으키는 횟수는 2 번을 넘지 말아야 하며 예상치 못하게 프로그램이 종료되더라도 사용자의 정보는 결함 없이 저장되어야 한다.

D. Security Requirements

사용자의 정보는 안전히 보호되어야 한다. 사용자가 회원가입과 로그인을 할 때 사용하는 정보, 시스템을 사용하면서 생성하는 정보(코드, 결과)들은 안전히 보호되어야 하며 사용자 외의 다른 사용자가 함부로 열람할 수 없어야 한다.

4.3.2 Organizational Requirements

A. Environmental Requirements

사용자는 시스템을 통해 코드를 작성하고 이를 통한 결과를 열람할 수 있어야 하므로 시스템은 PC 환경에서 작동되는 것을 전제로 한다. 또한 웹에서 작동해야 하며 만약 웹브라우저의 사용이 불가능하다면 사용자는 시스템을 사용할 수 없다.

B. Operational Requirements

시스템은 웹에서 동작해야 한다. 또한 사용자가 언제든지 시스템을 사용할 수 있도록 24 시간 작동해야 한다.

C. Development Requirements

시스템의 프론트엔드는 HTML, CSS, JavaScript 를 사용하여 개발되어야 한다. 웹 개발라이브러리로는 React 가 사용되어야 하며 이 외에 개발에 도움이 되는 각종 라이브러리 또한 사용하여야 한다. 시스템은 파이썬을 사용하여 개발되어야 한다. 웹

프레임워크인 Django 를 이용하여 개발되어야 하며 사용자의 코드를 분석할 수 있는 Open AI Codex 를 비롯한 각종 라이브러리 또한 사용되어야 한다

4.3.3 External Requirements

A. Regulatory Requirements

시스템은 외부 라이브러리를 사용할 경우 각 라이브러리에 해당하는 라이선스 규칙을 지켜야 한다. 또한 사용자가 로그인할 경우 브라우저의 프로토콜 규약을 지켜야 한다.

B. Ethical Requirements

시스템은 사용자의 정보를 왜곡해선 안 되며 사용자의 정보를 무단으로 사용해서도 안 된다. 또한 사용자가 로그아웃했을 경우 사용자의 정보를 성공적으로 회수할 수 있어야 한다.

C. Legislative Requirements

시스템은 사용자의 로그인/아웃 시 사용자의 정보를 쿠키를 통해 전달 및 전송한다. 이때 브라우저의 쿠키는 외부로 유출되어서는 안 되며 각 쿠키는 잘 식별되어야 한다.
사용자가 작성한 코드는 모두 서버에서 실행되며 이 과정은 격리된 환경에서 사용자가 개입할 수 없어야 한다. 사용자의 개인정보는 유출되거나 오용되지 않아야 한다.

5. System Architecture

5.1 Objective

System Architecture에서는 본 시스템의 아키텍처에 대해 상세하게 설명한다. 시스템을 구성하고 있는 각 컴포넌트들과 이들의 역할과 상호작용 등에 대해 기술한다. 추가적으로 시각적인 자료를 이용하여 표현한다. 이 목차는 시스템의 전체적인 구조를 이해하는데 목적을 두고 있다.

5.2 System Architecture

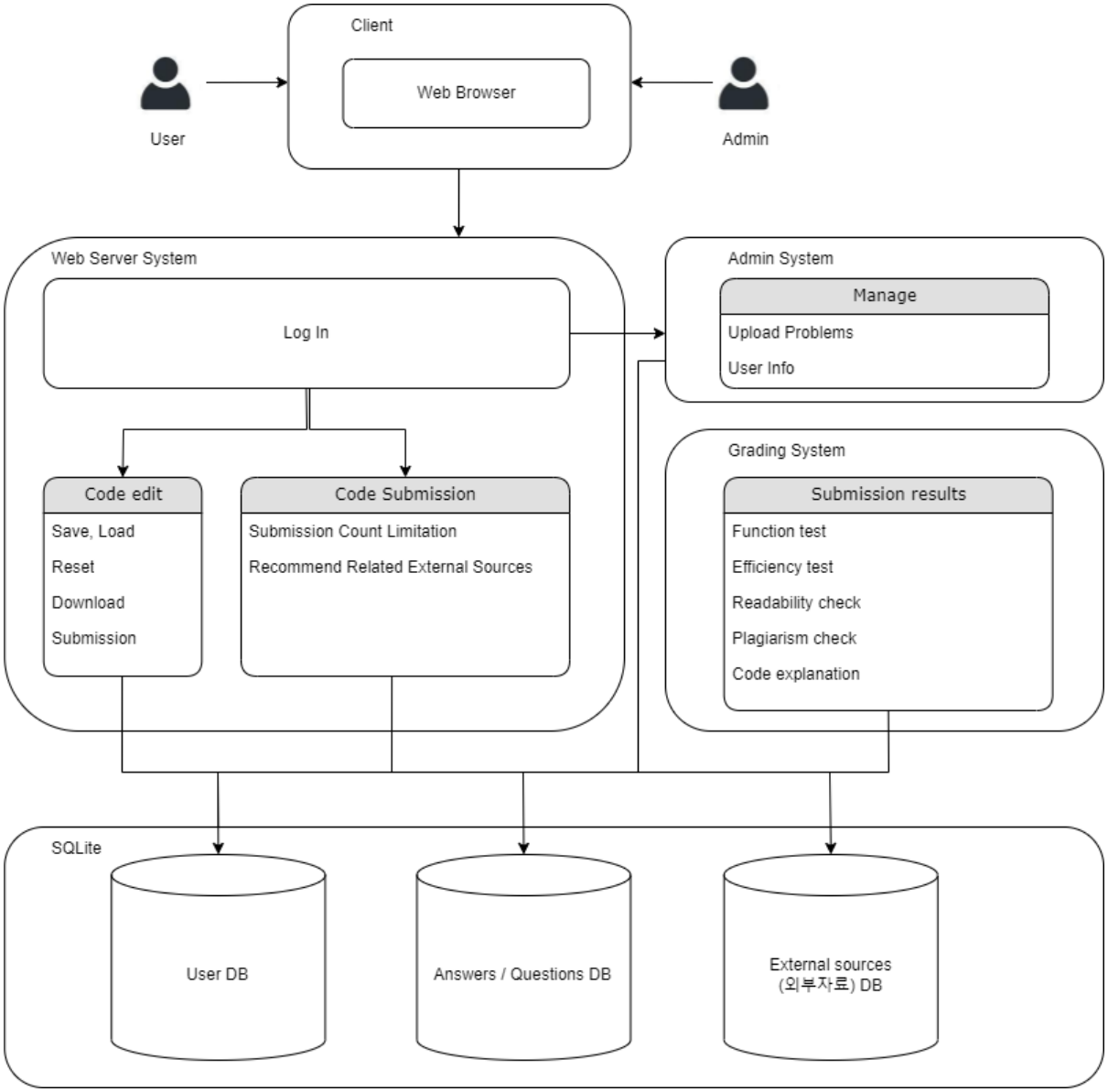

Diagram 1. Overall System Architecture : 전체적인 시스템의 구조

본 시스템은 Web-based 시스템으로 Web Browser을 이용하여 접속한다. Web Server System, Admin System, Grading System으로 구성된다. Web Server System은 Web의 모든 interface를 총괄한다. 이는 코드 수정과 제출을 포함한다. Admin Server System은 코드 컨텐츠와 유저 정보에 대한 management를 담당한다. Grading Server System은 제출된 코드에 대한 채점을 관리한다.

5.3 Subsystem

5.3.1 Web Server System

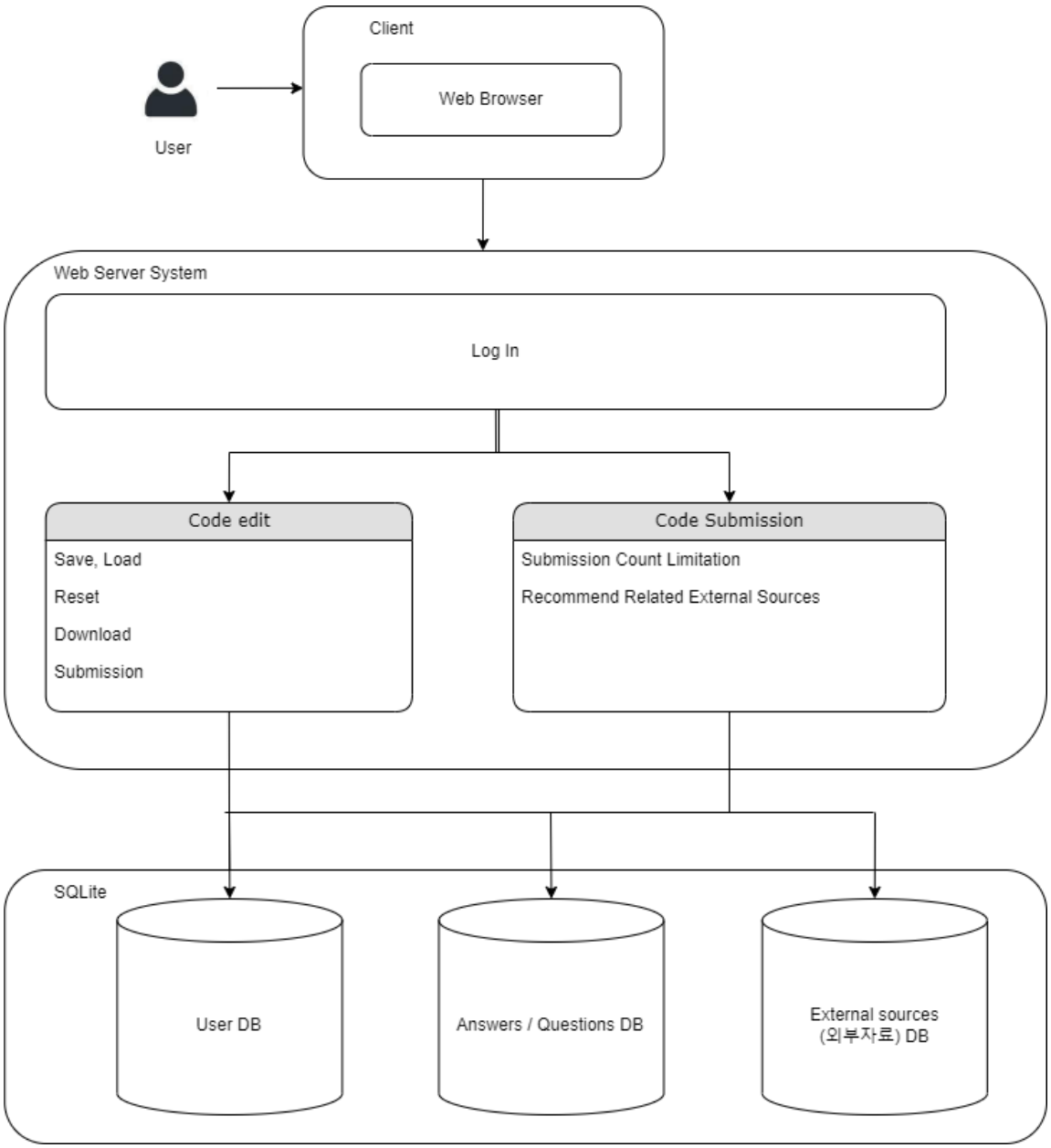

Diagram 2. Web Server System Architecture : 웹 서버 시스템의 구조

Web Server는 User에게 User Interface를 제공하며 Log-In interface, Code edit interface, Code submission interface를 가진다. Interface들은 로그인, 문제 코드 확인, 문제 코드 수정, 문제 코드 다운로드, 문제 코드 제출, 외부 자료 확인을 담당하며, 각 페이지에서 발생한 event의 데이터는 해당 database에 저장한다.

5.3.2 Admin Server System

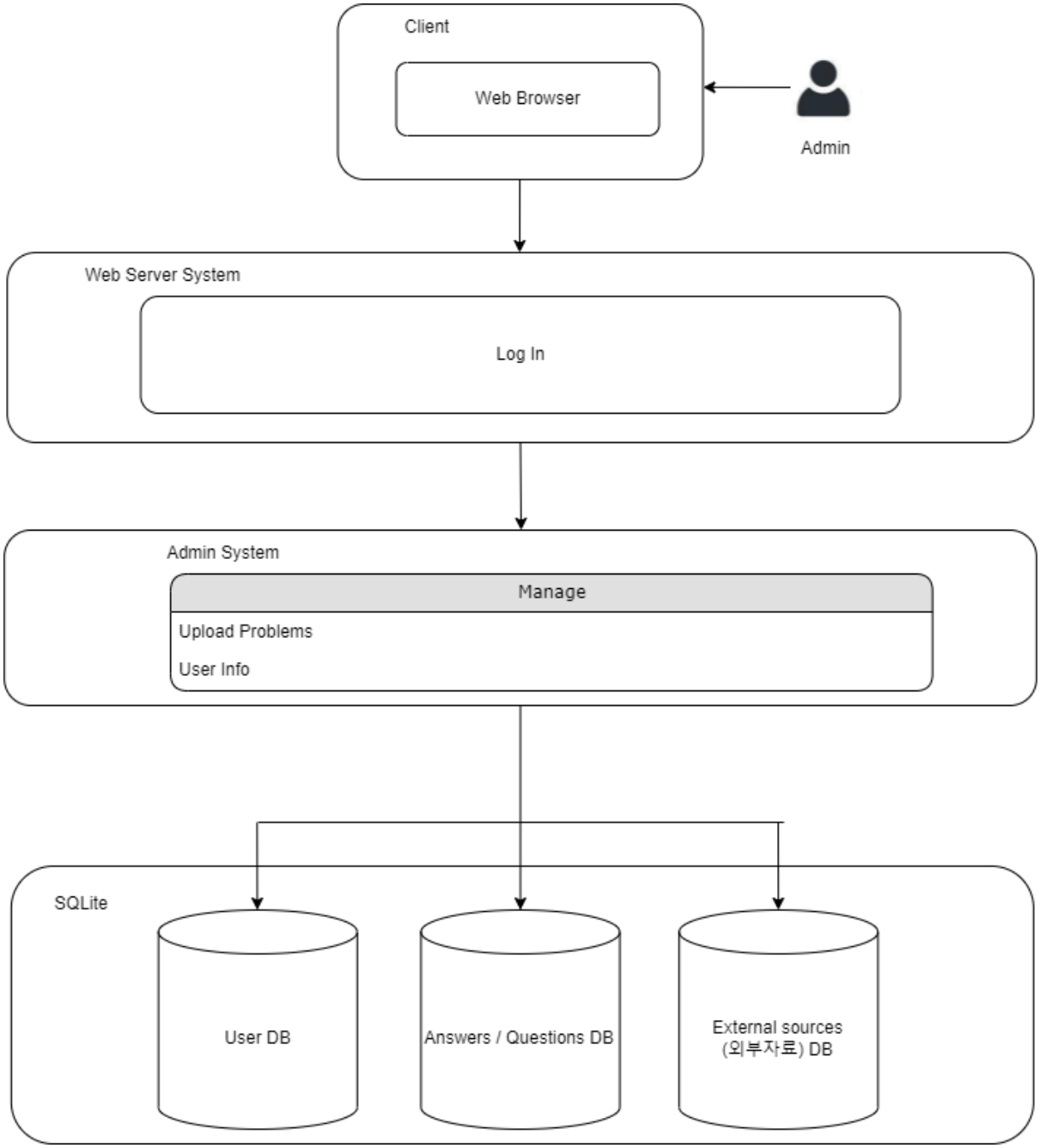

Diagram 3. Admin Server System Architecture : 관리자 서버 시스템의 구조

Admin Server system은 Web server와 상호운용하며, 문제 코드에 대한 추가/삭제/업데이트 및

수정을 담당한다. 관리자의 권한으로 접속하여 동작하며, 업데이트 된 컨텐츠의 내용에 대해 database에 접속하여 전체적인 문제 코드를 관리한다. Web Server에서 추가할 문제 코드가 존재하는 경우 관리자 정보로 Log In 하여 Admin Server에 접속한 뒤 문제 코드를 추가한다. 그 후 Answers/Questions DB에 수정한 데이터를 저장하고, User DB에서 사용자 정보를 수정한다.

5.3.3 Grading Server System

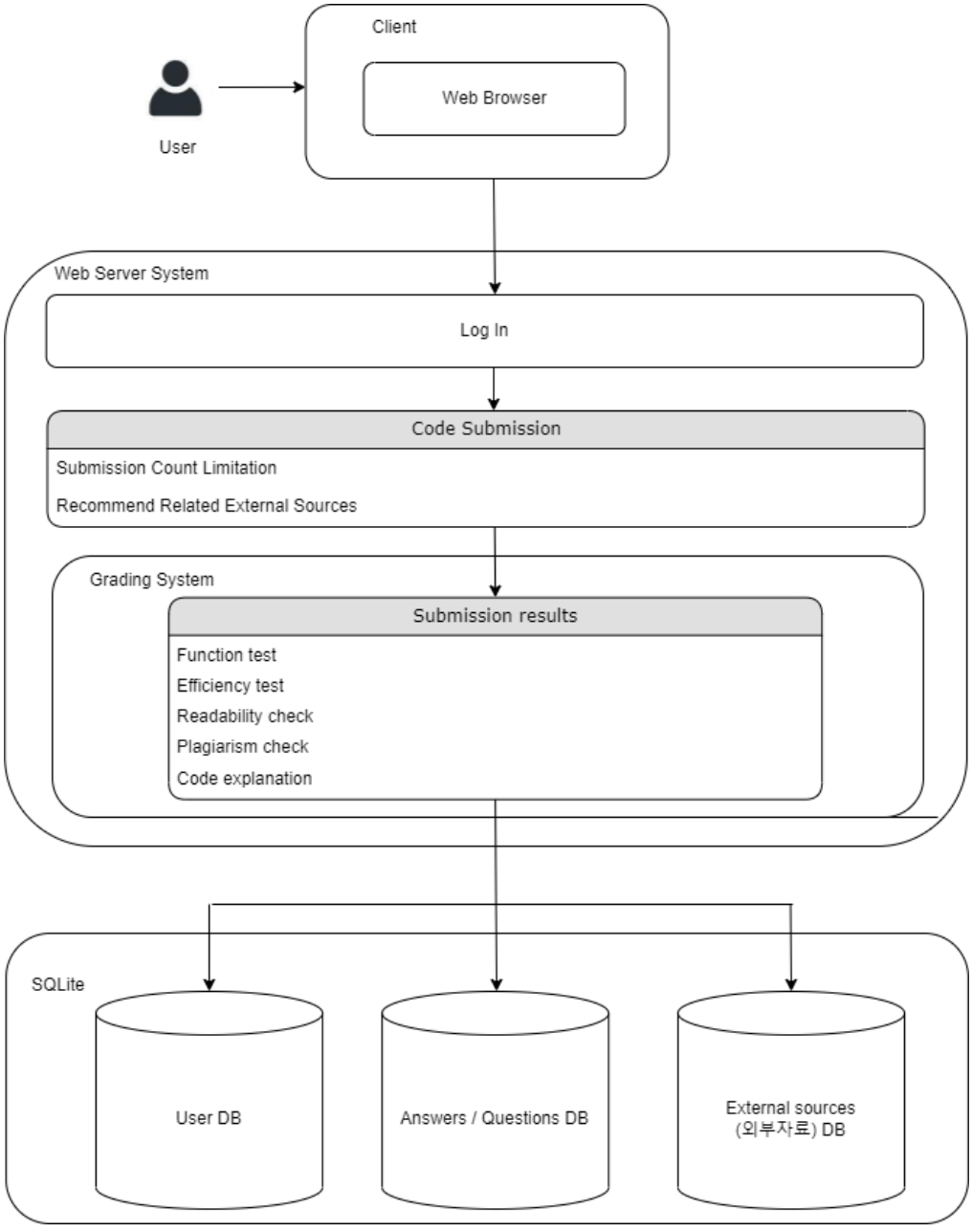

Diagram 4. Grading Server System Architecture : 채점 서버 시스템의 구조

Grading Server System은 Web Server에서 제출한 사용자의 문제 코드에 대해 Compile을 진행

하며 해당 코드를 채점한다. Answers / Questions DB 로부터 해당 문제 코드에 대한 정보를 가져와 사용자가 제출한 문제 코드에 대한 Function test, Efficiency test, Readability check, Plagiarism check, Code explanation을 진행한다 . 이후 User DB에 제출 log, 제출한 코드, 채점 결과를 저장한다.

6. System Requirements Specification

6.1 Objective

User Requirements Definition, System Architecture 장을 바탕으로 functional, non-functional requirements 에 대하여 상세하게 기술한다. 기능적 요구사항의 경우 각각의 기능, 설명, 입력, 출력, 처리, 조건을 표 기반으로 작성한다. 비기능적 요구사항의 경우 각각의 목적에 맞게 요구되는 사항들을 자연어 기반으로 작성한다. 사용자의 종류는 학습자, 교수자, 채점자이다.

6.2 Functional Requirements Specification

기능	설명	입력	출력	처리	조건
회원가입	사용자가 본인의 계정을 등록한다. 기존에 동일한 이메일이 존재하거나, 비밀번호 확인이 다를 경우엔 거부한다.	사용자의 이메일, 비밀번호, 비밀번호 확인, 이름, 역할 정보	계정 생성 성공 여부	User DB 에 {이메일, 비밀번호, 이름, 역할}을 저장한다.	교수자와 채점자로 등록이 가능한 이메일을 미리 알고 있어야 한다.
로그인	사용자가 메인 페이지 접근을 위해 등록한 계정으로 로그인한다. 존재하지 않는 계정이면 거부한다.	사용자의 이메일, 비밀번호	로그인 성공 여부	User DB 에 등록된 {이메일, 비밀번호} 정보와 비교하여 일치하는 tuple 이 있으면 허가한다.	
로그아웃	사용자가 저장되지 않은 데이터를 모두 삭제하고 로그아웃한다.	로그아웃 버튼의 클릭, 또는 타임아웃		작성 코드, 출력 결과를 삭제하고 로그인	

				화면으로 복귀한다.	
문제 선택	학습자가 UI 를 통해 문제를 선택하면 문제 정보를 가져온다. 임시 저장된 내용이 있다면 가져온다.	UI 를 통한 문제 번호 선택	문제 정보를 화면에 정돈하여 출력한다. 임시 저장 파일이 존재한다면 에디터에 내용을 쓴다.	QA DB 에서 {문제 제목, 문제 설명, 남은 시간, 스켈레톤 코드}을 가져와 출력한다. 존재한다면 {유저, 문제번호, 임시저장코드} 를 가져와 에디터에 쓴다.	로그인 된 상태
코드 작성	학습자가 코드 에디터 섹션에 입력한다.	줄바꿈을 포함한 가변 길이 문자열	입력한 문자열을 화면에 정돈하여 출력한다.	Syntax highlighter module 을 사용하여 학습자가 입력한 코드를 시각적으로 나타낸다.	로그인 된 상태
코드 검증	학습자가 공개 테스트 케이스를 사용하여 작성한 코드를 검증한다. 각 테스트 케이스에 대해 PASS / FAIL 여부와 input, output 값을 확인할 수 있다.	검증할 테스트 케이스 번호, 작성한 코드, 검증 버튼의 클릭	PASS/FAIL 여부, 입력과 출력값을 화면에 정돈하여 출력한다.	테스트 케이스 실행 코드를 QA DB 에서 가져온다.	로그인 된 상태. 공개 테스트 케이스 개수에 맞춰 검증 버튼을 표시해야 한다.

테마 변경	사용자가 코드 에디터 및 페이지의 테마 및 폰트 크기를 변경할 수 있다.	선택한 테마 값	변경된 CSS 적용	프론트엔드에서 CSS 값을 변경하여 적용한다.	
코드 다운로드	학습자가 에디터에 작성한 코드를 파일의 형태로 다운로드한다.	작성한 코드, 다운로드 버튼의 클릭	작성된 py 파일	에디터의 모든 내용을 가져와 사용자 로컬 PC 의 파일에 쓴다.	로그인 된 상태.
코드 초기화	학습자가 작성 중이던 코드를 스켈레톤 코드로 되돌린다.	문제 번호, 초기화 버튼의 클릭	해당 문제의 스켈레톤 코드를 에디터에 쓴다.	문제 선택 시 불러온 스켈레톤 코드의 내용을 에디터에 쓴다. 기존에 작성 중이던 내용은 지워진다.	로그인 된 상태.
코드 복사	학습자가 작성 중이던 코드를 로컬 클립보드로 복사한다.	작성한 코드, 복사 버튼의 클릭	에디터의 내용을 로컬 클립보드에 쓴다.		로그인 된 상태.
코드 불러오기	학습자가 로컬 컴퓨터에 작성된 파일을 불러와 코드 에디터에 쓴다.	파일 업로드, 불러오기 버튼의 클릭	파일의 내용	기존에 작성 중이던 내용은 지워진다.	로그인 된 상태.
실시간 코드 저장	주기적으로 에디터에 작성된 내용을 보관하여 예기치 못한 종료 또는	문제 번호, 작성한 코드, 유저 아이디	문제 번호에 대응하는 임시 저장 파일	일정 시간마다 QA DB 에 접근하여, {유저, 문제 번호, 작성한 코드}를 삽입한다.	로그인 된 상태. 타임아웃으로 인한 로그아웃 전에 임시저장이 될 수 있도록

	페이지 재렌더딩 시에 작성한 내용을 다시 가져올 수 있게 한다.				시간을 설정한다. (타임아웃 > 임시 저장 주기)
코드 실행	학습자가 작성한 코드를 실행한다. 임의로 작성한 표준 입출력 값을 확인할 수 있다.	문제 번호, 작성한 코드	stdout 출력값 및 stderr 출력값을 화면에 정돈하여 표시한다.	QA DB 에 접근하여 테스트 케이스 코드를 가져온다. TC 를 실행한 stdout, stderr 출력을 모두 표시한다. unittest 라이브러리 활용	로그인 된 상태.
코드 채점	학습자가 작성한 코드로 공개/비공개 테스트케이스를 시험한다.	문제 번호, 작성한 코드	정답/오답 테스트 케이스의 정보를 화면에 정돈하여 표시한다.	QA DB 에 접근하여 모든 테스트 케이스 코드를 가져온다. TC 를 실행하고, 정답/오답 여부를 표시한다. unittest 라이브러리 활용	로그인 된 상태.

코드 제출	학습자가 작성 코드를 최종 제출한다. 3 회까지 제출할 수 있다. 제출 코드는 기록되며 이후 교수자/채점자가 확인할 수 있다.	문제 번호, 유저 아이디, 작성 코드, 제출 횟수	기능성 모듈, 효율성 모듈, 가독성 모듈, 해설 결과를 화면에 정돈하여 표시한다.	제출 횟수를 초과하였으면 거부한다. QA DB 에 {유저 아이디, 문제 번호, 제출 번호, 제출 코드, 결과}를 삽입한다. 분석 모듈을 실행한다.	로그인 된 상태.
코드 비교	학습자가 제출한 코드를 diff 를 통해 교수자의 정답 코드와 비교한다.	문제 번호, 제출한 코드, 정답 코드	diff 결과를 화면에 정돈하여 표시한다.	QA DB 에서 {문제 번호, 정답 코드}를 가져온다. diff2html 라이브러리 활용.	로그인 된 상태.
과거 코드 불러오기	학습자가 지금까지 제출한 모든 코드를 확인할 수 있다.	문제 번호, 제출 번호, 불러오기 버튼의 클릭	제출 번호별 코드와 정답 코드와의 diff 결과를 화면에 정돈하여 표시한다.	QA DB 에서 {유저 아이디, 문제 번호, 제출 번호, 코드, 결과}를 가져온다.	로그인 된 상태. 존재하는 제출 번호 수에 맞도록 불러오기 버튼을 표시해야 한다.
기능 점수 확인	학습자가 최종 제출한 코드를 공개/비공개 테스트케이스에 적용한 결과를 보여준다.	문제 번호, 제출한 코드, 기능 점수 확인 버튼의 클릭	테스트케이스별 결과를 화면에 정돈하여 표시한다.	필요한 경우 QA DB 에 접근하여 가져온다.	로그인 된 상태. 문제별 테스트케이스 개수에 맞도록 UI 에 표시해야 한다.
효율 점수 확인	학습자가 최종 제출한 코드를 multimetric 라이브러리를	제출한 코드			로그인 된 상태.

	통해 효율성을 보여준다.				
가독성 점수 확인	학습자가 최종 제출한 코드를 pylama 라이브러리를 통해 가독성을 보여준다.	제출한 코드			로그인 된 상태.
해설 확인	학습자가 최종 제출한 코드를 codex 라이브러리를 통해 분석한다.	제출한 코드.			로그인 된 상태.
관련 자료 확인	학습자가 마감 시간이 지난 문제의 관련 학습 자료를 확인할 수 있다.	문제 번호.	관련 자료의 내용과 외부 링크 정보를 정돈하여 표시한다.	QA DB 에 접근하여 {문제 번호, 마감 기한} 정보를 가져온다. 마감 기한이 지나지 않았다면 거부한다. External source DB 에 접근하여 {문제 번호, 자료 리스트}를 가져온다.	로그인 된 상태.

Table 1. Functional Requirements Specification

6.3 Nonfunctional Requirements Specification.

6.3.1 Product Requirements.

A. Usability

직관적인 디자인 템플릿을 참고한다. 필요한 경우 첫 사용자를 위한 도움말 툴팁을 삽입할 수 있다.

B. Efficiency

반복문을 다중으로 사용하여 시간 복잡도가 커지는 일을 피한다. 남은 시간의 계산 등을 프론트 단에서 진행하지 않는다. DB IO를 최대한 줄일 수 있도록 한다. 유사한 정보를 계속해서 필요로 할 경우, 변수에 미리 캐싱해두고 DB 접근을 피한다.

C. Dependability

시스템의 Fail 을 막는다. 유저 코드의 잘못된 접근으로 인한 런타임 오류를 미리 감지하고 코드를 중단시킨다. Exception handling 코드를 미리 작성한다.

D. Security

로그인 과정을 통해 다른 사용자의 정보에 접근할 수 없도록 막는다. 회원가입 시 다른 사람, 다른 역할의 이메일을 도용할 수 없도록 인증 과정을 추가한다. 유저 코드가 시스템 파일에 접근할 수 없도록 권한을 설정한다.

6.3.2 Organizational Requirements

A. Environment

유저 에이전트가 모바일로 인식될 경우 사이트에 접근할 수 없도록 막는다.

B. Operation

별도의 상주 서버를 사용하여 시스템 서비스가 중단되는 일이 없도록 한다.

C. Development

프론트엔드 단에는 HTML, CSS, React.js 계열을 사용한다. 웹 프레임워크는 Django 를 사용한다. 코드 분석 기능 구현에는 Multimetric, pylama, codex 라이브러리를 사용한다. 그 외 Python3 를 사용한다.

6.3.3 External Requirements

A. Regulations

Multimetric: BSD2 라이선스
Pylama: MIT 라이선스
OpenAI Codex: MIT 라이선스

6.4 Use Cases

USE CASE	회원가입
ACTOR	User, System, DB
DESCRIPTION	학습자가 자신의 정보를 사용하여 시스템에 회원으로 가입한다.
STIMULUS	정보를 기입을 한 상태로 회원가입 버튼을 클릭
RESPONSE	회원가입이 된 사실을 출력한다.
COMMENTS	같은 정보로 회원가입을 한 이력이 없어야 한다.

Table 2. 회원가입

USE CASE	로그인
ACTOR	User, System, DB
DESCRIPTION	유저가 입력한 정보를 토대로 로그인을 진행한다.
STIMULUS	로그인 페이지에서 정보를 입력한 뒤 로그인 버튼을 누른다.
RESPONSE	입력한 정보를 DB 와 대조한 뒤 맞을 경우 로그인 허가한다.
COMMENTS	

Table 3. 로그인

USE CASE	로그아웃
ACTOR	User, System
DESCRIPTION	로그인 되어있는 상태에서 벗어난다
STIMULUS	로그아웃 버튼 클릭
RESPONSE	로그아웃된 사실, expire 된 토큰
COMMENTS	로그인된 상태여야 한다.

Table 4. 로그아웃

USE CASE	문제 선택
ACTOR	user, System, DB
DESCRIPTION	유저가 문제를 선택한다.
STIMULUS	UI 를 통해 유저가 원하는 문제를 선택한다.
RESPONSE	QA DB 에서 문제 제목, 문제 설명을 출력한다. 유저 정보가 존재한다면, 임시저장코드를 가져와 에디터에 쓴다.
COMMENTS	

Table 5. 문제 선택

USE CASE	코드 작성
ACTOR	User, System
DESCRIPTION	유저가 에디터에 코드를 작성하고 확인한다
STIMULUS	유저가 에디터로 코드 내용을 수정한다. 줄바꿈을 포함한 가변길이 문자열
RESPONSE	입력한 문자열이 화면에 정돈되어 출력된다.
COMMENTS	

Table 6. 코드 작성

USE CASE	예시 테스트 케이스 검증
ACTOR	User, System
DESCRIPTION	학습자가 공개 테스트 케이스를 사용하여 작성한 코드를 검증한다. 각 테스트 케이스에 대해 PASS / FAIL 여부와 input, output 값을 확인할 수 있다.
STIMULUS	검증할 테스트 케이스 번호, 작성한 코드, 검증 버튼의 클릭
RESPONSE	PASS/FAIL 여부, 입력과 출력값을 화면에 정돈하여 출력한다.
COMMENTS	

Table 7. 예시 테스트 케이스 검증

USE CASE	코드 다운로드
ACTOR	User, System
DESCRIPTION	User 가 자신의 작성 코드를 컴퓨터로 다운로드 할 수 있다.
STIMULUS	코드 다운로드 버튼을 클릭한다
RESPONSE	버튼을 클릭하면 에디터에 쓰여진 코드를 .py 파일 형태로 바꾸어 로컬에 저장한다.
COMMENTS	

Table 8. 코드 다운로드

USE CASE	테마 변경
ACTOR	User, System, DB
DESCRIPTION	사용자 인터페이스의 테마를 변경한다.
STIMULUS	설정에 있는 테마 변경 버튼 클릭
RESPONSE	바뀐 테마로 변경된 사용자 인터페이스
COMMENTS	

Table 9. 테마 변경

USE CASE	코드 초기화
ACTOR	User, System, DB
DESCRIPTION	학습자가 작성 중이던 코드를 스켈레톤 코드로 되돌린다.
STIMULUS	문제 번호, 초기화 버튼의 클릭
RESPONSE	해당 문제의 스켈레톤 코드를 에디터에 쓴다.
COMMENTS	문제 선택 시 불러온 스켈레톤 코드의 내용을 에디터에 쓴다. 기존에 작성 중이던 내용은 지워진다.

Table 10. 코드 초기화

USE CASE	코드 복사
ACTOR	User, System
DESCRIPTION	학습자가 작성 중이던 코드를 로컬 클립보드로 복사한다.
STIMULUS	작성한 코드, 복사 버튼의 클릭
RESPONSE	에디터의 내용을 로컬 클립보드에 쓴다.
COMMENTS	

Table 11. 코드 복사

USE CASE	코드 불러오기
ACTOR	User, System
DESCRIPTION	학습자가 로컬 컴퓨터에 작성된 파일을 불러와 코드 에디터에 쓴다.
STIMULUS	파일 업로드, 불러오기 버튼의 클릭
RESPONSE	에디터의 내용을 로컬 클립보드에 쓴다.
COMMENTS	

Table 12. 코드 불러오기

USE CASE	코드 실행
ACTOR	User, System
DESCRIPTION	학습자가 작성한 코드를 실행한다. 임의로 작성한 표준 입출력 값을 확인할 수 있다.
STIMULUS	코드 실행 버튼을 누른다.
RESPONSE	stdout 출력값 및 stderr 출력값을 화면에 정돈하여 표시한다.
COMMENTS	

Table 13. 코드 실행

USE CASE	코드 제출
ACTOR	User, System, DB
DESCRIPTION	학습자가 작성 코드를 최종 제출한다. 코드는 DB 에 저장되며 N 회까지 제출할 수 있다. 제출 코드는 기록되며 이후 교수자/채점자가 확인할 수 있다
STIMULUS	제출 버튼을 클릭한다.
RESPONSE	제출된 DB 에 저장한다. 제출된 코드를 채점하며 코드를 AI 기반으로 평가하여 해당 자료를 유저에게 전달한다.
COMMENTS	

Table 14. 코드 제출

USE CASE	실시간 코드 저장
ACTOR	System
DESCRIPTION	주기적으로 에디터에 작성된 내용을 보관하여 예기치 못한 종료 또는 페이지 재렌더딩 시에 작성한 내용을 다시 가져올 수 있게 한다.
STIMULUS	일정이 시간이 지날 때마다 자동저장을 한다.
RESPONSE	문제 번호에 대응하는 임시 저장 파일
COMMENTS	

Table 15. 실시간 코드 저장

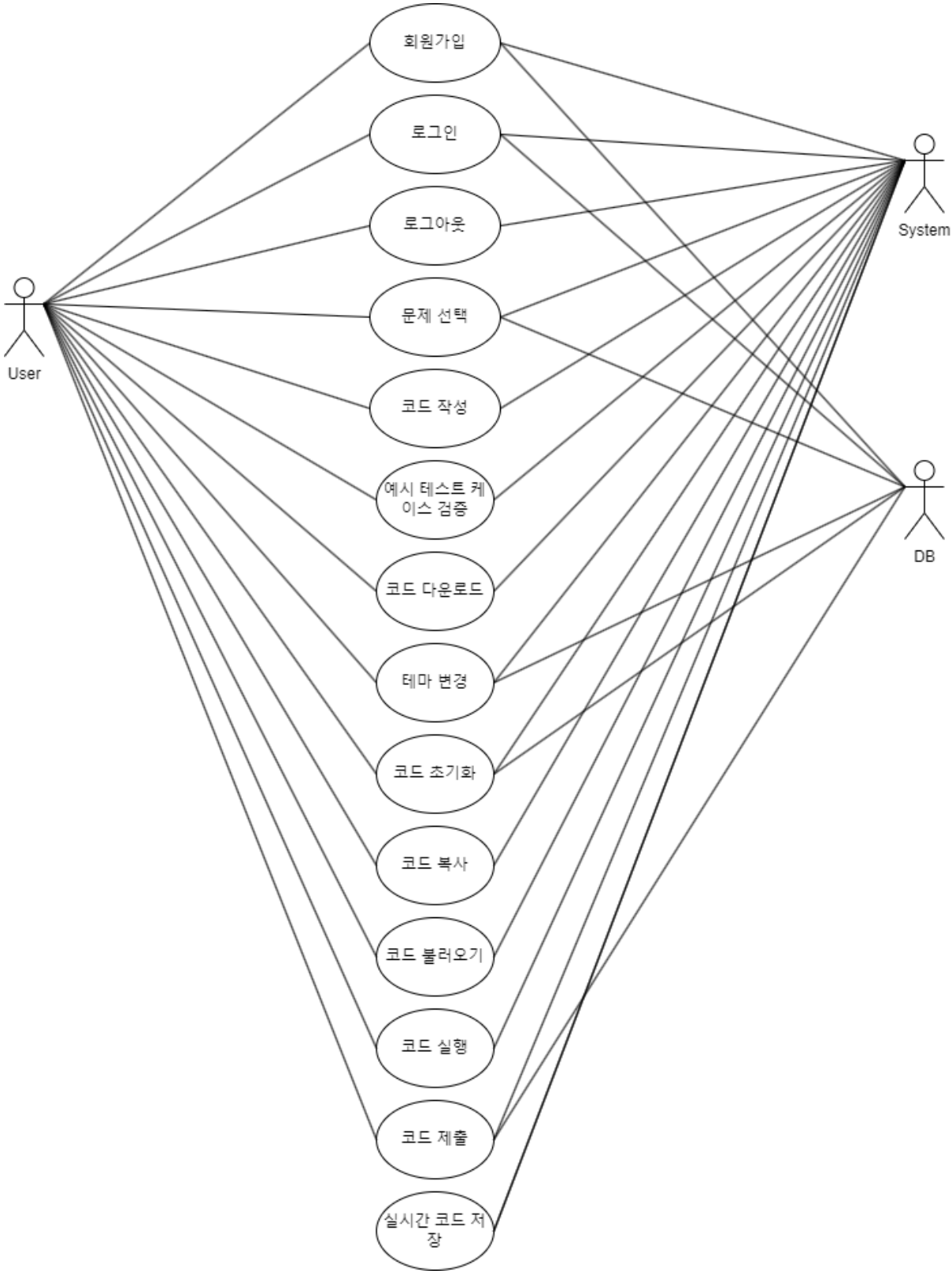

Diagram 5. Use Case 다이어그램

7. System models

7.1 Objective

System Requirements Model 에서는 Sequence Diagram, Structural Models, Behavioral Models 같은 다양한 모델을 사용하여 시스템의 기능과 컴포넌트 사이의 관계와 커뮤니케이션에 대해 설명한다. System Requirement Model 을 사용하여 시스템에 대한 가시성을 높이고 이해를 도울 수 있다.

7.2 Sequence Diagram

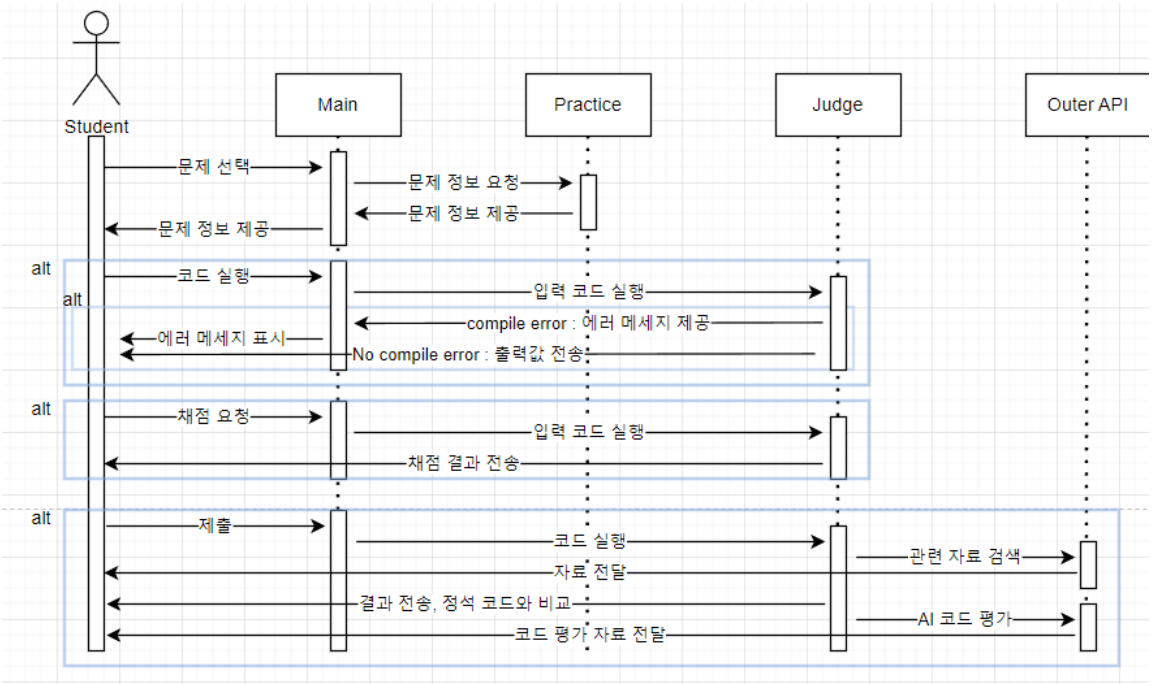

Diagram 6. 문제 풀이 다이어그램

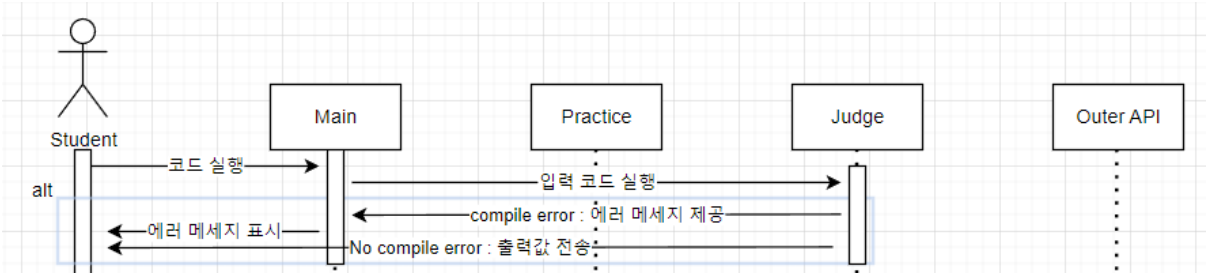

Diagram 7. 코드 실행

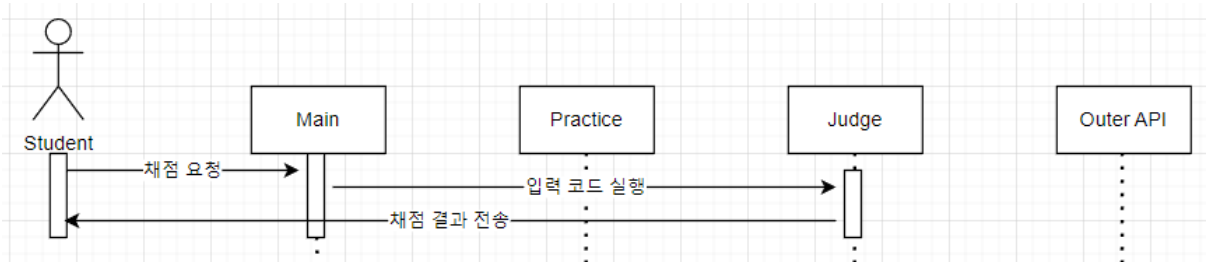

Diagram 8. 코드 채점

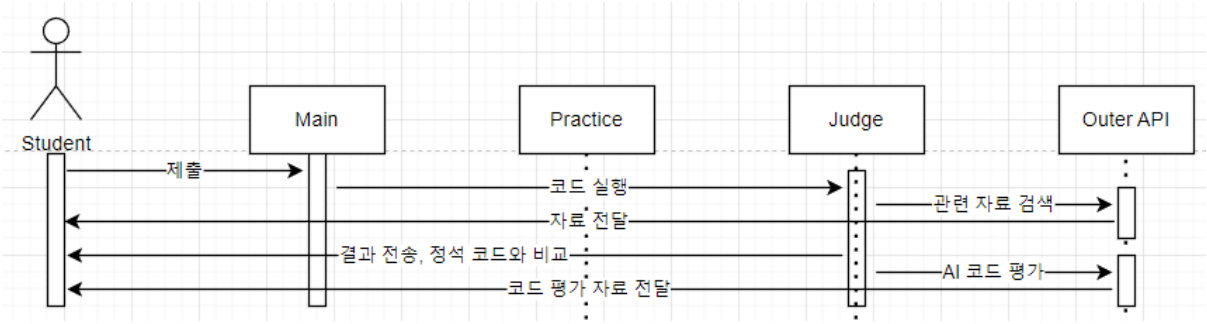

Diagram 9. 코드 제출

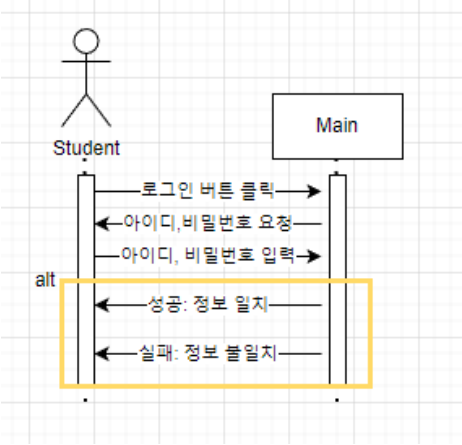

Diagram 10. 유저 로그인

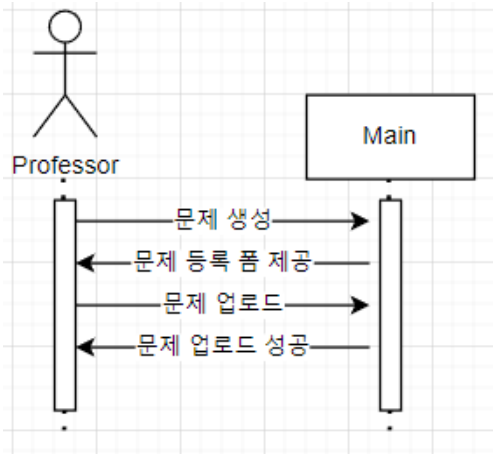

Diagram 11. Professor 문제 등록

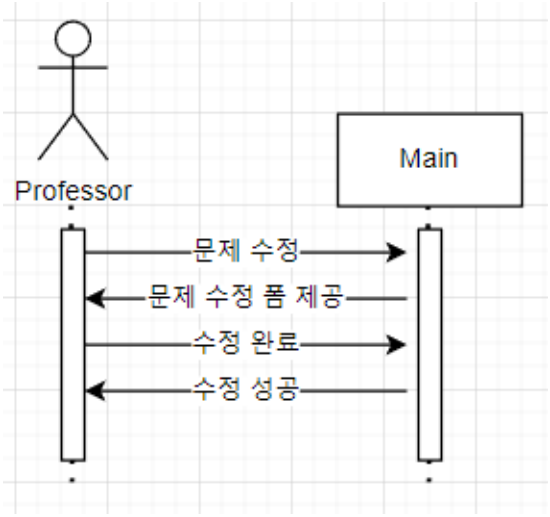

Diagram 12. 문제 수정

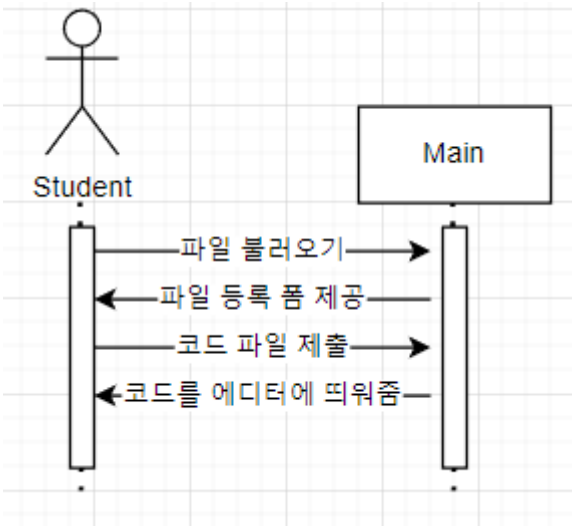

Diagram 13. 파일 불러오기

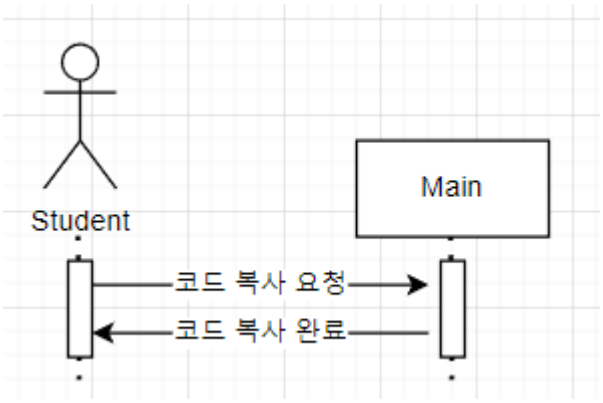

Diagram 14. 코드 복사

7.3 Structural Models

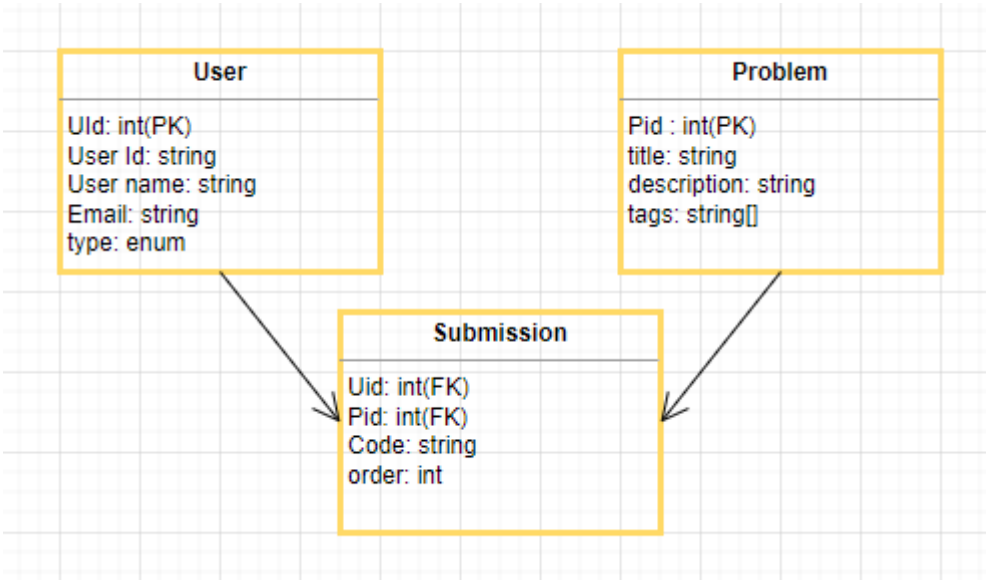

Diagram 15. Class Diagram

7.4 Behavioral Models

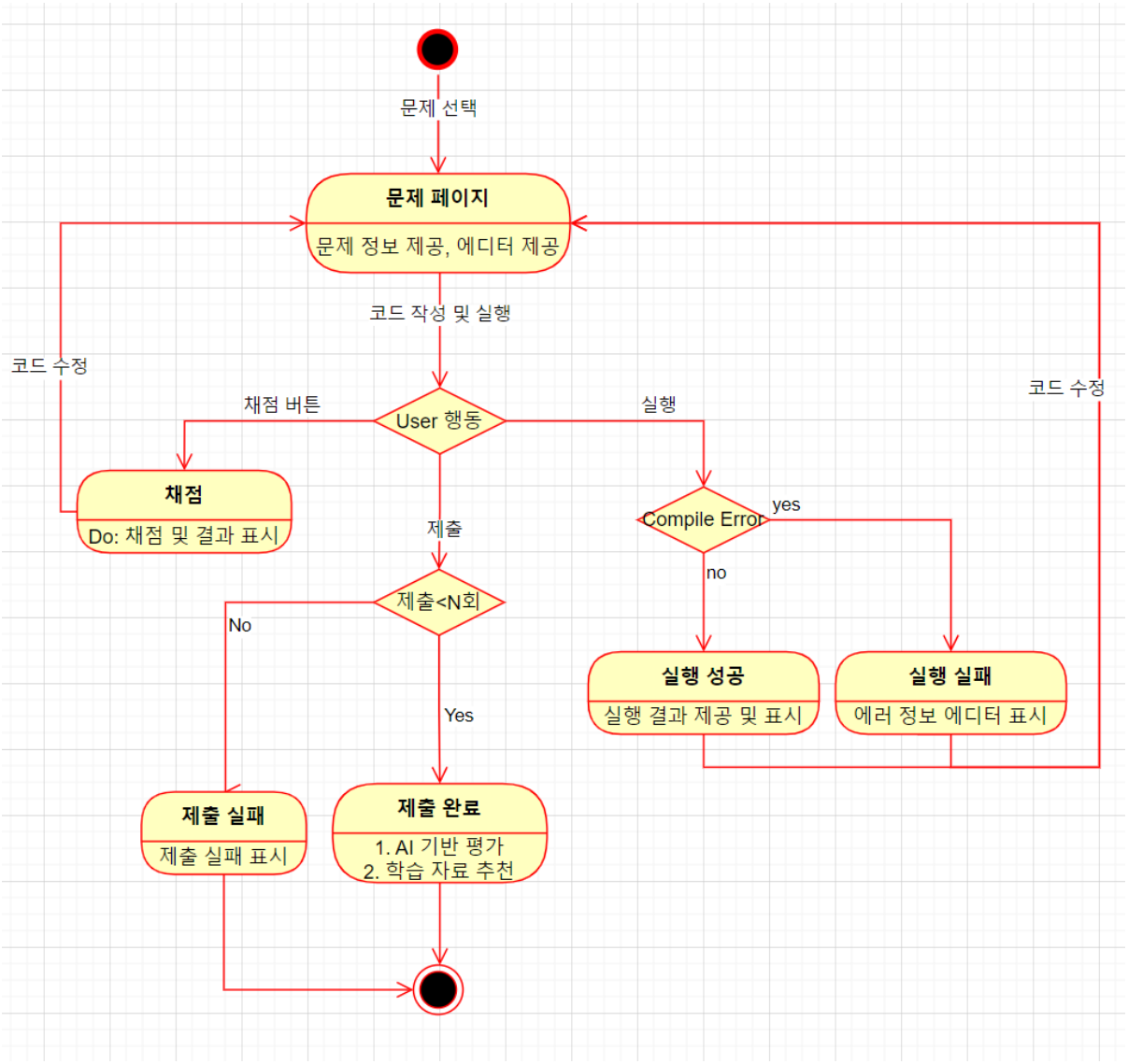

Diagram 16. Event Driven Diagram: 코드 실행, 채점, 제출

8. System Requirement Evolution

8.1 Objective

이번 장에서는 본 시스템의 기본 가정과 한계에 대해 서술한다. 그리고 HW 개선, 고객 요구사항 변화 등에 따른 예상되는 시스템 변경에 대해 서술한다. System evolution 장을 통해 미래의 시스템 변경에 대해 서술함으로써 변경을 제약하는 설계 결정을 피할 수 있다.

8.2 Assumptions and Limitation

8.2.1 지원가능한 프로그래밍 언어의 부족

본 시스템은 파이썬 언어를 지원하는 코딩 테스트 사이트이다. 프로그래밍 언어는 각각의 장단점이 명확하게 존재한다. 예를 들어, 파이썬 언어는 편의성 등의 측면에서 장점이 있지만 실행속도 등의 측면에서 단점이 있는 언어이다. 프로그래밍 언어로 파이썬 언어만을 지원하는 것은 유저의 usability를 떨어트린다.

8.2.2 유저 테스트케이스 추가의 불편함

현재 시스템에서 유저 테스트케이스를 추가하는 방법은 코드 안에 테스트케이스를 직접 코딩한 후 실행 버튼을 누르는 것이다. 이것은 유저가 정답 코드 외에 추가적인 코딩을 하게 만들기 때문에 usability를 떨어트릴 수 있다.

8.2.3. Malicious user

본 시스템은 유저가 제출한 코드를 정해진 컴파일 옵션에 따라 그대로 실행해준다. 코드에 대한 검사를 하지 않고 그대로 실행하기 때문에 malicious user를 필터링 하는 것이 매우 어려우며 security에 큰 위협이 된다.

8.2.4. 동시 접속 가능한 유저 수의 제한

현재 본 시스템은 서버로 전송된 코드를 직접 컴파일 한 후 실행하고 있다. 이러한 순차적 실행 방식은 유저 수가 많아질수록 응답시간을 크게 높이는 단점이 있다. 응답시간이 길어질수록 유저의 usability가 떨어지게 된다.

8.2.5. 유저에 대한 분석 결과의 부재

현재 본 시스템은 각 문제에 대한 분석 결과를 제공하고 있다. 하지만 각 문제에 대한 분석 결과만으로는 유저에게 적절한 가이드라인을 제공하는 것이 어렵다. 즉, 유저 개인에 대한 분석 결과가 있어야만, 유저가 향후 자신의 단점을 개선하기 위해 어떤 것을 공부해야 할지에 대한 가이드라인을 제공할 수 있다.

8.2.6. 코드 및 문제 해결 방법 공유의 부재

사람들은 다른 사람들과 소통하면서 자신의 문제점을 찾아내고, 보다 더 높은 수준의 성취를 이룰 수 있다. Extreme programming의 pair programming이 소통의 좋은 예시라 볼 수 있다. 그러나, 현재 본 시스템에서 유저는 다른 유저들과 연결되어 있지 않다. 그저 자신의 문제 해결 방법에 대한 결과만을 볼 수 있기 때문에 유저의 학습 효율성이 떨어질 수 있다.

8.3 System Evolution

8.3.1 컨테이너 방식의 도입

각 유저에게 컨테이너를 제공한다. 각 유저의 코드가 자신의 컨테이너 안에서 실행되기 때문에 이전의 순차적 실행 방식과 비교해 유저 수가 많아지더라도 짧은 응답시간을 확보할 수 있다. 뿐만 아니라 컨테이너 방식을 도입함으로써 다양한 프로그래밍 언어 및 버전을 제공하는 것이 편해진다.

8.3.2 유저 테스트케이스 박스 추가

유저가 박스 안에 자신의 테스트케이스를 작성하여 코드와 함께 제출한다. 유저는 자신의 테스트케이스에 대한 실행 결과를 서버로부터 돌려받는다. 유저 테스트케이스 박스를 통해 유저는 테스트케이스에 대한 추가적인 코드를 작성할 필요가 없게 된다.

8.3.3 컴파일 전 코드 분석

유저가 서버로 전송한 코드를 컴파일하기 전에 분석하고, 전송된 코드가 악성 코드인지 판별한다. 컴파일하기 전에 코드를 분석함으로써 악성 코드가 실행되는 것을 막을 수 있을 뿐 아니라 malicious user를 필터링 할 수 있다. 뿐만 아니라, 악성 코드 분석 결과는 다른 malicious user를 필터링 하는데 활용할 수도 있다.

8.3.4 유저 분석 결과 제공

유저에 대한 분석 결과를 제공한다. 예를 들어, 유저가 BFS 문제에 대한 문제들의 정답율이 낮다면 BFS를 먼저 공부하는 것을 추천한다. 이를 통해 유저는 자신의 강약점을 정확히 파악하고, 자신이 공부해야 할 것들에 대한 우선순위를 정할 수 있다.

8.3.5 질문 페이지 개설

문제에 대한 질문 페이지를 만든다. 질문 페이지에는 여러 장점이 있다. 첫째, 질문 페이지에서 유저는 자신의 코드 및 해결 방법을 공유하고 피드백 받을 수 있다. 둘째, 유저는 문제에 대한 자신보다 더 나은 혹은 새로운 접근 방법을 공부할 수 있다. 셋째, 이전 질문들을 통해 에러를 해결함으로써 불필요한 시간 낭비를 줄여준다.

9. Index

9.1 Objectives

Index에서는 본 문서에 사용된 표와 그림에 대한 인덱스를 나타낸다.

9.2 Table Index

9.3 Diagram Index

9.4 Figure Index

10. Reference

10.1 아키텍처 설계

https://www.lucidchart.com/blog/how-to-draw-architectural-diagrams

https://www.edrawsoft.com/kr/program-review/architecture-drawing-program.html

https://product.kyobobook.co.kr/detail/S000001033082

10.2 시퀀스 다이어그램

https://ko.wikipedia.org/wiki/시퀀스_다이어그램

10.3 Behavior models

https://www.conradbock.org/processcategory.html

10.4 백준

http://www.ekoreanews.co.kr/news/articleView.html?idxno=45510

https://www.acmicpc.net/

10.5 리트코드

https://leetcode.com/

10.6 코드포스

https://codeforces.com/

10.7 프로그래머스

https://programmers.co.kr/