Software Design Specification

BePro

소프트웨어공학개론 10조

2019313485 김지윤

2019312493 최윤채

2020312562 심윤보

2017312967 김주엽

2018313398 서정엽

2019314980 부윤종

2018311304 성보경

Contents

1. Preface 10

1.1. Objective 10

1.2. readership 10

1.3. Document structure 10

1.3.1. Introduction 10

1.3.2. Overall System Architecture 10

1.3.3. System Architecture – Frontend 10

1.3.4. System Architecture – Backend 10

1.3.5. Protocol Design 10

1.3.6. Database Design 11

2. Introduction 11

2.1. Applied Diagram 11

2.1.1. State Diagram 11

2.1.2. Class Diagram 11

2.1.3. Sequence Diagram 11

2.2. System Overview 11

3. System Architecture - Overall 12

3.1. System Organization 12

3.2. System Architecture - Frontend Application 12

3.3. System Architecture – Backend 13

4. System Architecture - Frontend 14

4.1. Objectives 14

4.2. Main Page 14

4.2.1. Attributes 14

4.2.2. Methods 15

4.2.3. State Diagram 15

4.3. Home Page 16
4.3.1. Attributes 16
4.3.2. Methods 16
4.3.3. State Diagram 16
4.4. Login Page 17
4.4.1. Attributes 17
4.4.2. Methods 18
4.4.3. State Diagram 18
4.5. Questions Page - Contents List 19
4.5.1. Attributes 19
4.5.2. Methods 19
4.5.3. State Diagram 20
4.6. Questions Page - Problem Solving 21
4.6.1. Attributes 21
4.6.2. Methods 22
4.6.3. State Diagram 22
4.7. Result Page 23
4.7.1. Attributes 23
4.7.2. Methods 24
4.7.3. State Diagram 24
4.8. My Page 25
4.8.1. Attributes 25
4.8.2. Methods 26
4.8.3. State Diagram 26
4.9. Notice Page 27
4.9.1. Attributes 27
4.9.2. Methods 28

4.9.3. State Diagram 28
4.10. Problem Examiner Page 29
4.10.1. Attributes 29
4.10.2. Methods 30
4.10.3. State Diagram 31
5. System Architecture – Backend 31
5.1. Objectives 31
5.2. Overall Architecture 31
5.3. Subcomponents 32
5.3.1. Code Management System 32
5.3.2. Code Execution System 34
5.3.3. Grade System 35
5.3.4. Refactoring System 36
5.3.6. Profile Management System 39
6. Protocol 40
6.1. HTTP 40
6.2. JSON 41
6.3. Protocol Description 41
6.3.1. Problems List Protocol 41
6.3.2. Problem Detail Protocol 41
6.3.3. Problem Code Execution Protocol 42
6.3.4. Problem Code Save Protocol 42
6.3.5. Problem Code Submission Protocol 43
6.3.6. Problem Code Load Protocol 43
6.3.7. Problem Creation Protocol 43
6.3.8. Code Commit Protocol 44
6.3.9. Code Review Request Protocol 44

6.3.10. Code Comment Request Protocol 45
6.3.11. Dead Code Detection Request Protocol 45
6.3.12. Code Refactoring Request Protocol 46
6.3.13. GitHub OAuth Login Protocol 46
6.3.14. Lecture Guideline Recommendation Protocol 47
6.3.15. Lecture History Protocol 47
6.3.16. User Problem History Protocol 47
6.3.17. Notice Protocol 48
7. Database Design 48
7.1. Objectives 48
7.2. Entity Tables 48
7.2.1. Problem App Tables 49
7.2.2. Code App Tables 50
7.2.3. Lecture App Tables 52
7.2.4. Boards App Tables 53
7.2.5. Accounts App Tables 53
7.3. Relational Schema 54
8. Supporting Information 55
8.1. Software Design Specification 55
8.2. Document History 56

그림 목차

Figure 1 System Organization 12
Figure 2 System Architecture Frontend Application 13
Figure 4 Overall Backend Architecture 13
Figure 5 Main Page State Diagram 15
Figure 6 Home Page Diagram 17
Figure 7 Login Page State Diagram 18
Figure 8 Questions Page – Contents List State Diagram 21
Figure 9 Questions Page – Problem Solving State Diagram 23
Figure 10 Result Page State Diagram 25
Figure 11 My Page State Diagram 27
Figure 12 Notice Page State Diagram 29
Figure 13 Problem Examiner Page State Diagram 31
Figure 14 Overall Backend Architecture 32
Figure 15 Code Management System Class Diagram 33
Figure 16 Code Management System Sequence Diagram 34
Figure 17 Code Execution System Class Diagram 34
Figure 18 Grade System Class Diagram 35
Figure 19 Grade System Sequence Diagram 36
Figure 20 Refactoring System Class Diagram 36
Figure 21 Refactoring System Sequence Diagram 37
Figure 22 Problem & Lecture Management System Class Diagram 38
Figure 23 Problem & Lecture Management System Sequence Diagram 39
Figure 24. Profile Management System Class Diagram 39
Figure 25. Profile Management System Sequence Diagram 40

Figure 26 Problems와 Boards의 Relational Schema 54

Figure 27 Codes와 Lectures의 Relational Schema 55

표 목차

Table 1 Problems List API 41
Table 2 Problem Detail API 41
Table 3 Problem Code Execution API 42
Table 4 Problem Code Save API 42
Table 5 Problem Code Submission API 43
Table 6 Problem Code Load API 43
Table 7 Problem Creation API 44
Table 8 Code Commit API 44
Table 9 Code Review Request API 45
Table 10 Code Comment Request API 45
Table 11 Dead Code Detection Request API 45
Table 12 Code Refactoring Request API 46
Table 13 GitHub OAuth Login API 46
Table 14 Lecture Guideline Recommendation API 47
Table 15 Lecture History API 47
Table 16 User Problem History API 47
Table 17 Notice Protocol 48
Table 18 Problem Table 49
Table 19 AlgorithmField Table 49
Table 20 ProblemFieldRelation Table 49
Table 21 Testcase Table 49
Table 22 Execution Table 49
Table 23 Submission Table 50
Table 24 UserCodeHistory Table 50

Table 25 Commit Table 50
Table 26 Review Table 51
Table 27 Review Table 51
Table 28 Comment Table 51
Table 29 Deadcode Table 51
Table 30 Refactor Table 52
Table 31 Lecture Table 52
Table 32 SoftwareField Table 52
Table 33 LectureFieldRelation 52
Table 34 LectureHistory Table 52
Table 35 Board Table 53
Table 36 Article 53
Table 37 User Table 53

1. Preface

1.1. Objective

본 문서의 목적과 대상, 문서 구성요소를 기술한다.

1.2. readership

본 문서는 프로그래밍 학습자를 위한 "BePro" 서비스를 제공하기 위한 소프트웨어 디자인 명세서이다. 본 서비스는 2023년 1학기 성균관대학교 소프트웨어공학개론 강의 10조(이하 '개발팀')에 의해 고안되고 개발된다. 본 디자인 명세서에는 시스템 디자인과 디자인 설명을 기술하였다. 개발팀이 구성요소를 구체적으로 이해할 수 있도록 작성하였으며, 이에 근거하여 본 서비스의 시스템을 개발한다.

본 문서는 개발팀 7명과 강의 조교님, 강의 교수님이 열람하는 것을 상정하여 작성되었다. 예외적으로, 소프트웨어공학개론 수강자 또한 학습 및 교육의 용도로 본 문서를 열람할 수 있다. 본 문서를 재 배포 및 수정하는 것은 자유이지만, 상업적 용도로 활용 시 개발팀의 허가를 얻어야 한다.

1.3. Document structure

1.3.1. Introduction

2장 Introduction에서는 전반적인 시스템 구조에 대해서 기술한다. 시스템을 설명하기 위한 다이어그램과 시스템의 목적과 목적에 따른 디자인 개요를 설명한다.

1.3.2. Overall System Architecture

3장 Overall System Architecture에서는 전체 시스템 조직 구조와 전체적인 프론트엔드와 백엔드 아키텍처 구조를 개괄적으로 설명한다.

1.3.3. System Architecture – Frontend

4장 System Architecture – Frontend에서는 프론트엔드의 아키텍처, 속성, 기능, 동작 방식을 페이지 별로 기술한다.

1.3.4. System Architecture – Backend

5장 System Architecture – Backend에서는 백엔드의 아키텍처, 컴포넌트, 상호작용을 기술한다.

1.3.5. Protocol Design

6장 Protocol Design에서는 프론트엔드와 백엔드가 상호작용하는데 필요한 프로토콜을 설명하고 request와 response에 대해 기술한다.

1.3.6. Database Design

데이터베이스의 테이블을 디자인하고 각 테이블 필드의 속성, 자료형, 설명을 기술한다.

2. Introduction

2.1. Applied Diagram

본 문서에서 시스템을 표현하기 위한 diagram은 다음과 같다.

2.1.1. State Diagram

객체는 특정 상태를 지니며 이 상태는 이벤트와 같은 액션에 의하여 상태는 변경될 수 있는데 이러한 객체의 상태와 상태의 변화를 도식화한 다이어그램이다.

2.1.2. Class Diagram

시스템의 class 내부의 정적인 내용이나 class 사이의 관계를 표기하는 다이어그램으로 시스템의 일부 또는 전체의 구조를 나타낼 수 있다. 하나의 클래스는 이름(Name), 속성 (Attribute), 기능이나 함수(Method)을 가질 수 있다.

2.1.3. Sequence Diagram

특정 action이 어떤 순서로, 어떤 class와, 어떻게 상호작용하는 지를 표현하는 행위 다이어그램이다. 시스템의 시나리오를 나타낼 수 있다.

2.2. System Overview

본 서비스 "BePro"는 프로그래밍에 대한 수요가 빠르게 늘어남에 따라, 혼자 프로그래밍을 학습하고자 하는 사람들에게 학습의 장을 제공한다. 본 서비스는 각각의 사람들에게 적합한 난이도의 코딩 문제를 제공하여 프로그래밍 입문자부터 전공자까지 컴퓨팅 역량을 키울 수 있도록 돕는다. 사용 가능한 언어로는 Python, C, C++, JAVA가 있고, 학습자가 원하는 언어로 문제를 해결하기 위한 코드를 작성하여 제출하면 본 서비스가 해당 코드를 채점한다.

프로그래밍을 학습자가 겪을 수 있는 어려움은 여러가지가 있다. 코드 작성법과 해당 언어의 문법에 맞게 알고리즘 코드로 구현하는 것, 디버깅 등 어려움을 겪는 경우가 흔하다. 따라서 학습자의 원활한 학습을 돕기 위해 AI 기술을 활용하여 학습자가 단순히 코드의 정답 유무를 확인하는 것에서 그치지 않고 코드 중에서 문제가 있는 부분을 시각적으로 쉽게

파악할 수 있도록 하여 학습자에게 효과적인 프로그래밍 학습 경험을 제공한다. 문제풀이 외에 강의 수강을 원하는 학습자는 본인의 관심사나 실력에 적절한 강의를 추천받아 수강할 수 있다.

본 서비스는 기능과 목적에 맞게 학습자가 편리하고 쉽게 사용하여 학습에만 집중할 수 있는 환경을 제공하기 위한 UI/UX를 디자인하고 구현한다. 또한 현재 설계된 기능 외의 추가 기능의 확장 가능성을 염두에 두고 API 작성 및 DB 설계를 진행한다.

3. System Architecture - Overall

3.1. System Organization

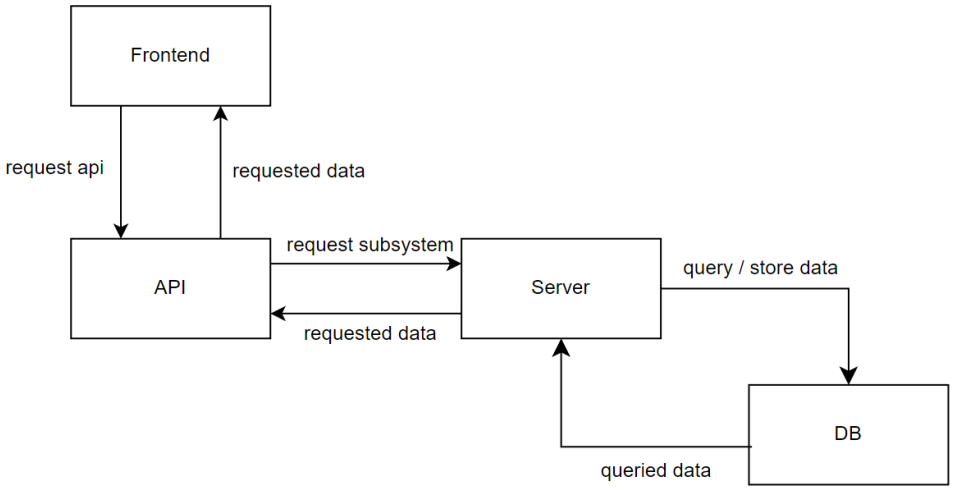

Figure 1 System Organization

BEPRO의 시스템은 크게 frontend와 backend로 이루어진다. Frontend는 사용자와 상호작용하여 데이터를 요청받고, 요청받은 데이터를 확인할 수 있도록 한다. Backend는 web server와 DB로 구성된다. Server는 여러 subsystem들로 구성되어 있으며, frontend는 REST API를 통해 subsystem을 호출할 수 있다. Server는 DB와 연결되어 필요한 데이터를 query를 통해 가져오거나, 새로운 데이터를 데이터베이스에 저장하는 등의 동작을 할 수 있다.

3.2. System Architecture - Frontend Application

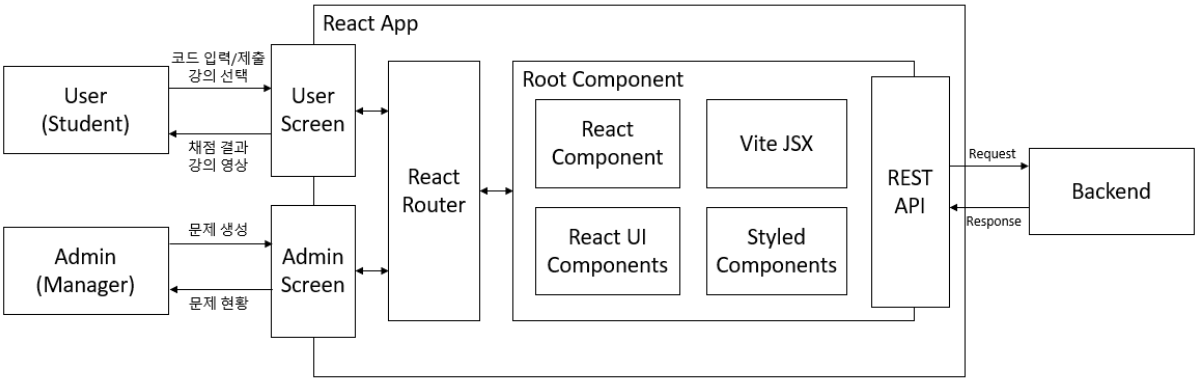

Figure 2 System Architecture Frontend Application

Frontend는 사용자로부터 데이터를 요청받아, 백엔드와 상호작용하며 사용자에게 요청받은 데이터를 확인할 수 있도록 작동한다. 일반 Student User와 추가 권한이 있는 Admin Manager는 각자의 웹 브라우저를 통해 React App에 접근할 수 있으며, Router를 통해 각자 요청한 페이지의 Component가 Root Component 아래에 Composition 된다. 백엔드로부터 필요한 데이터는 REST API를 통해 request를 보내고 response를 받아오는 역할을 한다.

3.3. System Architecture – Backend

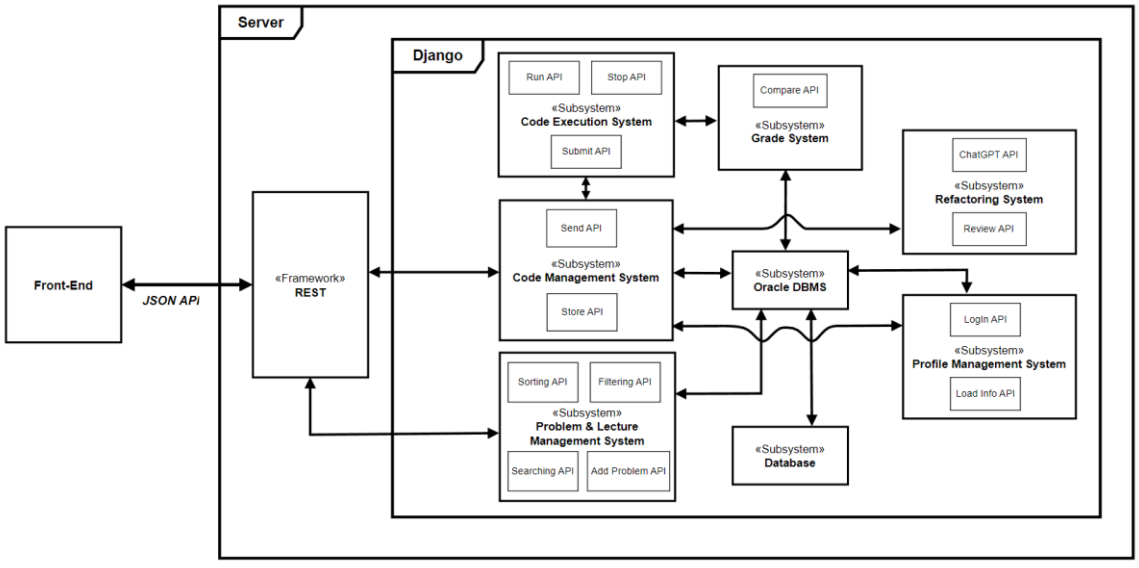

Figure 3 Overall Backend Architecture

Backend의 Django는 크게 여덟 개의 Sub-System으로 구성되어 있다. Sub-System중 Oracle DMBS와 Database를 제외한 Code Management System, Code Execution System,

Problem & Lecture Management System, Grade System, Refactoring System, Profile Management System까지 여섯 개의 Sub-System은 다음과 같은 부분들을 담당한다.

- Code Management System: Code 저장 및 필요한 Sub-System으로의 전송을 담당
- Code Execution System: Code 실행, 중지, 제출을 담당
- Problem & Lecture Management System: 제공하는 컨텐츠들의 검색, 정렬 등을 담당
- Grade System: 실행한 Code 결과를 정답과 일치하는지 확인을 담당
- Refactoring System: ChatGPT API를 활용하여 코드 수정 및 리뷰를 담당
- Profile Management System: Login과 사용자 정보 관리를 담당

위와 같이 구성된 서버는 REST Framework를 통해 Frontend와 소통할 수 있다. Frontend에서 JSON의 형태로 요청한 정보에 따라 Django에서 알맞은 Sub-System에 해당하는 API에 따라 요청을 처리한다.

4. System Architecture - Frontend

4.1. Objectives

시스템의 진행상태에 따른 기능을 설명하고 각 페이지의 state diagram을 통해 architecture를 표현하고자 한다.

4.2. Main Page

4.2.1. Attributes

Main Page가 가진 속성은 다음과 같다.

- Hero Section: 웹페이지에 대한 간단한 소개이다.
  i. Background - 배경 이미지이다.
  ii. Title - 서비스에 대한 요약이다.
  iii. Description - 서비스에 대한 조금 더 자세한 설명이다.
  iv. Sign Up - Sign Up 페이지로 연결되는 버튼이다.
- Introduction Section

i. Section Title - 주요 학습 서비스를 표시한다.

ii. Posts Square - 플랫폼의 주요 서비스들에 대해 표시한다.

iii. More - 서비스들을 자세히 소개하는 페이지로 연결되는 버튼이다.

4.2.2. Methods

각 하위 파트가 가지고 있는 methods는 다음과 같다.

i. clickSignup(): user가 Sign Up 버튼을 클릭하였을 때 Sign Up 페이지로 연결시킨다.

ii. clickMore(): More을 클릭하였을 때 자세한 플랫폼의 주요 서비스들을 소개하는 페이지로 연결시킨다.

iii. isLogined(): 현재 유저가 로그인 되어 있는 상태인지, 아닌지를 확인한다.

iv. setDescription(): 홈페이지에 대한 정보를 카드 형태로 나타내어 표시한다.

4.2.3. State Diagram

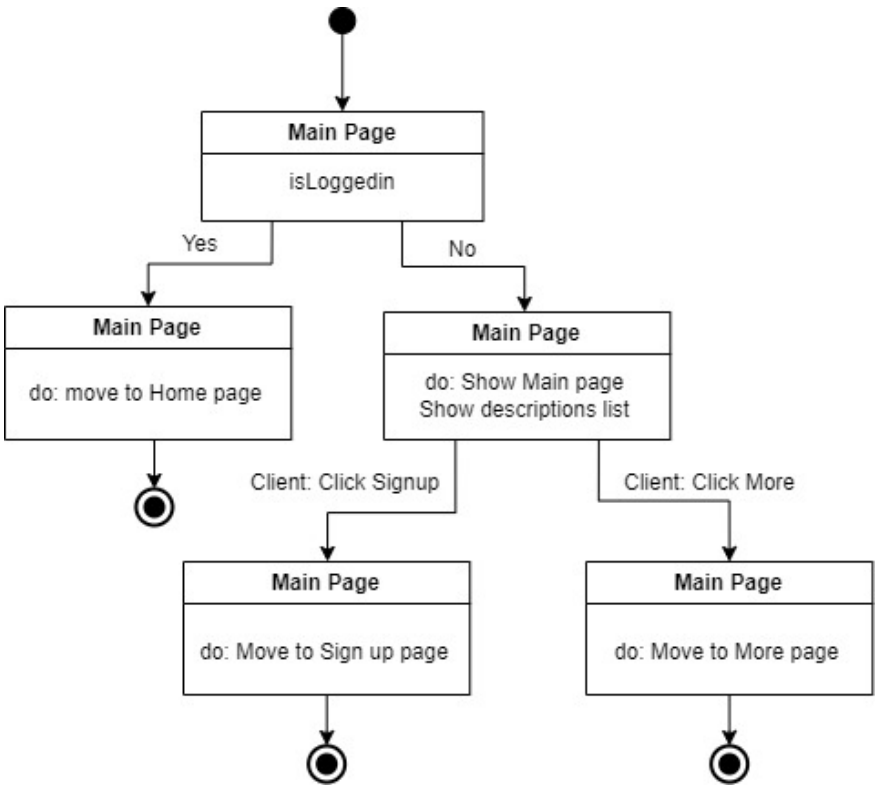

Figure 4 Main Page State Diagram

4.3. Home Page

4.3.1. Attributes

Home Page가 가진 속성은 다음과 같다.

- Ongoing Lectures - 현재 수강하고 있는 강의 내역이다. 강의명과 강사 이름이 같이 표시된다.
- Ongoing Questions - 사용자가 아직 완료하지 못한 문제의 리스트이다.
- Recommendation - 사용자에게 추천하는 강의들이다.

4.3.2. Methods

각 하위 파트가 가지고 있는 methods는 다음과 같다.

- setOnLectureList(): 사용자가 현재 수강 중인 강의 내역 데이터 리스트를 가져와 강의 이름, 강사명, 수강 상태를 표시한다.
- setOnQuestionList(): 사용자가 현재 풀이 중인 문제 내역 데이터 리스트를 가져와 문제 번호, 이름, 종류, 난이도를 표시한다.
- clickOnQuestion(): 문제를 클릭하면 해당 문제 풀이 페이지로 이동한다.
- setRecommendation(): 사용자의 문제풀이 내역을 바탕으로 추천 강의 리스트를 가져와 강의 이름, 강사명, 수강차수를 표시한다.

4.3.3. State Diagram

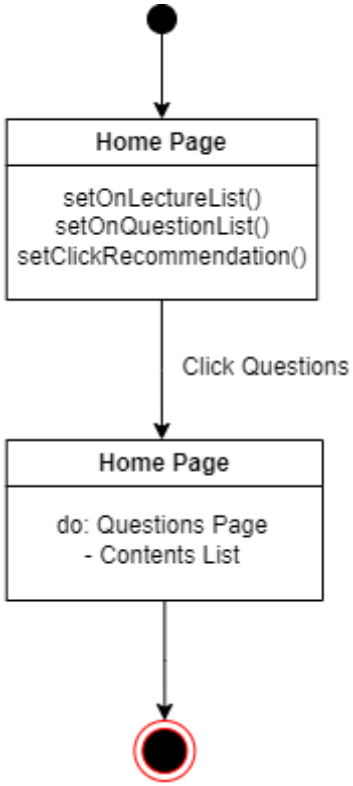

Figure 5 Home Page Diagram

4.4. Login Page

4.4.1. Attributes

Login Page가 가진 속성은 다음과 같다.

- Logo: 서비스 로고다.
- Github Account: GitHub 계정으로 로그인할 수 있게 연결되는 버튼이다.
- Notice: FAQ를 볼 수 있는 notice 페이지로 연결되는 고객센터 버튼이다.

(1) Logo는 아래의 속성을 가지고 있다.

- Logo: 서비스 로고 이미지다.

(2) Github Account는 아래의 속성을 가지고 있다.

- GithubLoginButton: GitHub 계정으로 로그인하는 버튼이다.

(3) Notice는 아래의 속성을 가지고 있다.

- NoticeButton: 공지사항 및 고객센터로 가는 버튼이다.

4.4.2. Methods

Login 페이지가 가지고 있는 methods를 파트에 따라 나누면 다음과 같다.

(1) Logo가 가지고 있는 method는 아래와 같다.

- clickLogo(): 로고 버튼을 누르면 Main page로 연결된다.

(2) Github Account가 가지고 있는 method는 아래와 같다.

- isLogined(): 현재 유저가 로그인 되어 있는 상태인지, 아닌지를 확인한다.
- clickLogin(): 깃허브 로그인 버튼을 누르면 백엔드로 연결되어 github api를 통해 처음 이용할 경우 계정 생성, 이전에 이용한 적이 있을 경우 로그인이 된 후, 홈페이지로 돌아간다.

(3) Notice가 가지고 있는 method는 아래와 같다.

- clickNotice(): 고객센터 버튼을 누르면 공지사항 페이지로 연결된다.

4.4.3. State Diagram

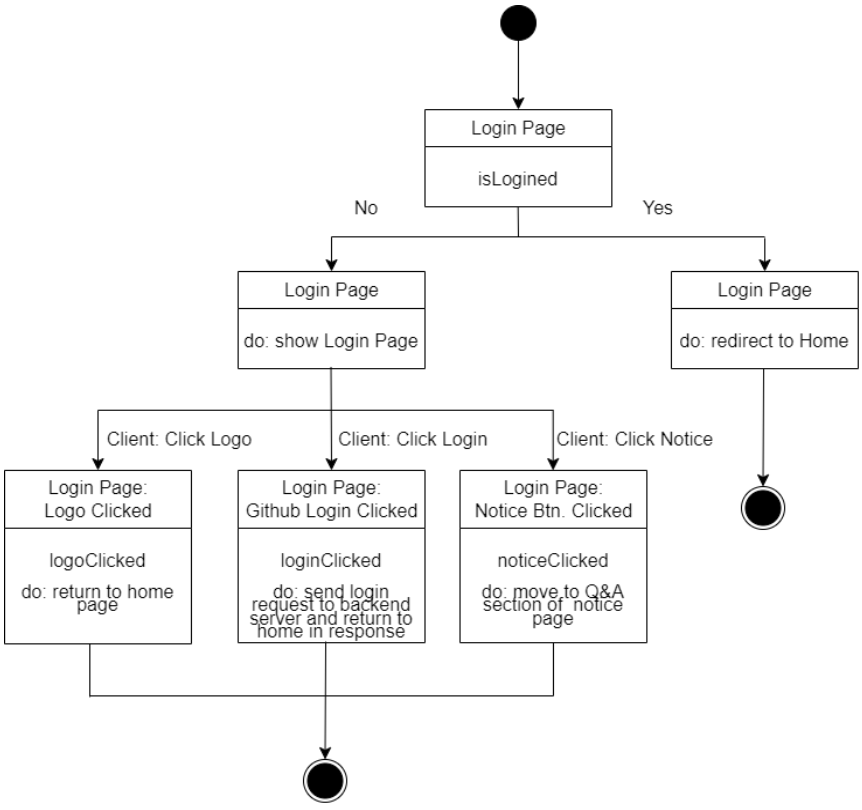

Figure 6 Login Page State Diagram

사용자가 Login Page에 입장하면, isLogined method가 바로 실행되어 현재 사용자가 로그인 되어있는 상태인지 확인한 후, 이미 로그인이 되어있다면 홈페이지로 다시 돌아간다. 로그인 되어 있지 않은 상태라면, 로그인 페이지를 보여준 후, Logo, GitHub Login, Notice Button 중 하나를 클릭하면 해당되는 페이지로 이동한다.

4.5. Questions Page - Contents List

4.5.1. Attributes

Questions Page - Contents List가 가진 속성은 다음과 같다.

- Filtering Section: 어떤 문제들을 Problems Section에 불러올지에 대한 조건을 설정하는 섹션이다.
- Problems Section: Filtering Section에서 지정된 조건에 해당하는 문제들을 List 형태로 나열해주는 섹션이다.

(1) Filtering Section은 아래의 속성을 가지고 있다.

- Filter Buttons: 완료, 진행중, 미완료, AI 추천 이 네가지 버튼의 묶음이다.
- Additional Filters: 세부 필터링을 적용하기 위한 버튼이다.
- Search: 문자열 입력 가능한 TextBox와 그 우측의 돋보기 이미지가 그려진 버튼을 Search라고 부른다.

(2) Problems Section은 아래의 속성을 가지고 있다.

- List: Filtering Section에서의 필터 조건에 맞는 문제들을 ListBox 형태로 보여준다. ListBox의 Element는 문제 번호, 문제 이름, 문제 카테고리, 난이도를 Bar 형태이며 클릭 가능하다 (단, 초기화면에서는 모든 문제를 보여준다).
- Pagination: List의 문제들을 페이지로 나눈 것으로, (현재 페이지 #) / (전체 페이지 #) 형태로 표시된다.

4.5.2. Methods

Questions Page - Contents List 가 가지고 있는 methods를 파트에 따라 나누면 다음과 같다.

(1) Filtering Section의 methods

- setFilter(): Filter Buttons 중에서 클릭된 버튼의 조건으로 Filter 조건을 설정한다.
- setAdditionalFilter(): 문제의 카테고리 혹은 레벨 중 선택된 것의 조건으로 Additional Filter 조건을 설정한다.
- search(): TextBox에 문자열을 입력한 후 돋보기 버튼을 누르면 유저가 입력한 문자열로 Search Filter 조건을 설정한다.

setFilter(), setAdditionalFilter(), search() 이 세 methods는 코드 마지막 단계에서 Problems Section의 method인 getProblems()를 호출한다.

(2) Problems Section의 methods

- getProblems(): Filtering Section에서 설정된 Filter, Additional Filter, Search Filter 조건과 부합하는 문제들만 골라준다. 추가로, 초기화면에 진입 시 기본적으로 호출되는 메서드이며, 그 때는 필터 조건이 비어 있는 상황이기에 모든 문제들을 가져온다.
- getTotalPages(): getProblems()에서 얻은 문제의 수를 가지고 총 페이지 수를 계산해준 뒤 showProblems()를 호출한다.
- showProblems(): getProblems()에서 얻은 문제들 중 Parameter로 넘겨받은 페이지 번호에 해당하는 문제들을 List에 출력해준다. 그 후 Pagination에 (현재 페이지 #) / (전체 페이지 #) 형태로 표시한다 (Parameter에 넘겨받는 페이지 번호가 비었으면 디폴트로 첫번째 페이지에 해당하는 문제들을 List에 출력한다).
- showPrevPage() / showNextPage(): Pagination 칸에서 이전/다음 페이지로 가는 요청을 할 시에 (현재 페이지)을 페이지 번호로 하여 showProblems()에 Argument로 넘기며 해당 method를 호출한다.
- clickProblem(): List에서 클릭된 문제 페이지로 이동한다 (Questions Page - Problem Solving으로 이동)

4.5.3. State Diagram

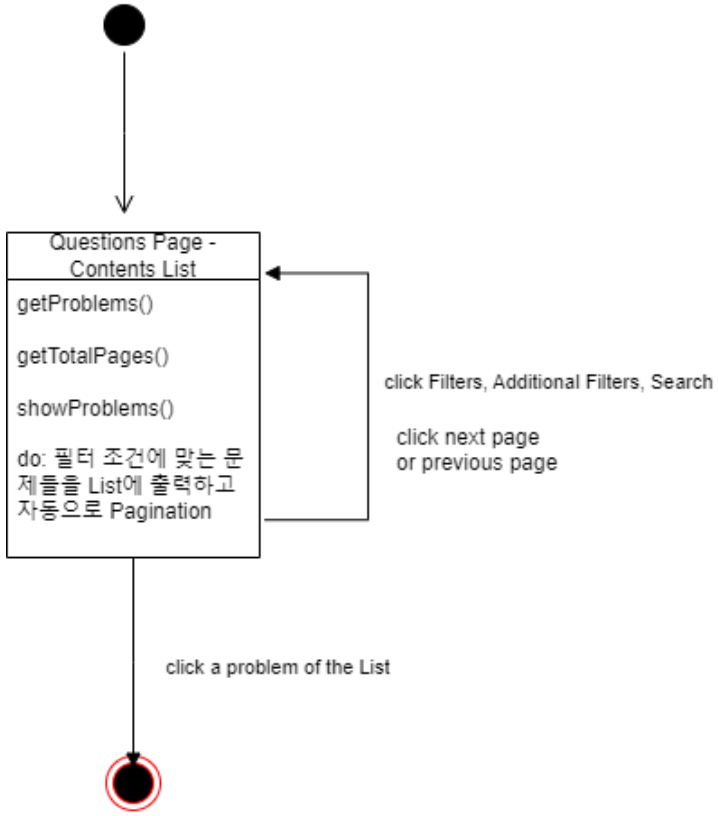

Figure 7 Questions Page – Contents List State Diagram

4.6. Questions Page - Problem Solving

4.6.1. Attributes

Questions Page - Problem Solving가 가진 속성은 다음과 같다.

- Description/Settings Section
  i. Question Info - 현재 풀고 있는 문제의 번호, 이름, 분류, 난이도이다.
  ii. Question Description - 해당 문제 내용이다.
  iii. Language Selection - 프로그래밍 언어를 선택하기 위한 드롭박스다.
- Coding Section
  i. Coding Interface - 문제에 대한 코드를 입력할 수 있는 입력창이다.
  ii. Input - 테스트해보고 싶은 input값을 입력하는 창이 있다.

iii. Output - testcase를 실행한 결과값이다.

- Button Section

i. Back - 문제풀이 리스트 페이지로 돌아가는 버튼이다다.

ii. Run - 작성한 코드를 실행하는 버튼이다.

iii. Stop - 코드의 실행을 중단하는 버튼이다.

iv. Submit - 해당 코드 채점을 요청하는 버튼이다.

4.6.2. Methods

각 하위 파트가 가지고 있는 methods는 다음과 같다.

- Description/Settings Section

i. showQuestionInfo()

ii. showQuestionDescription()

iii. selectLanguage()

- Button Section

i. clickBack() - 문제풀이 리스트 페이지로 돌아가는 버튼이다다.

ii. runCode() - 작성한 코드를 실행하는 버튼이다.

iii. stopCode() - 코드의 실행을 중단하는 버튼이다.

iv. submitCode() - 해당 코드 채점을 요청하는 버튼이다.

4.6.3. State Diagram

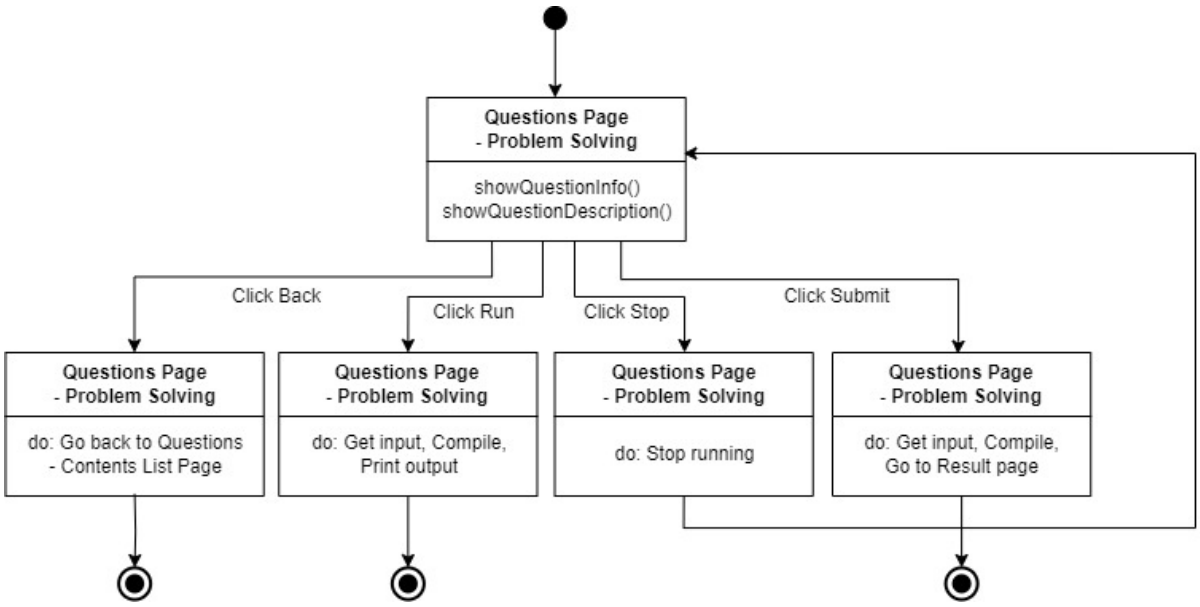

Figure 8 Questions Page – Problem Solving State Diagram

4.7. Result Page

4.7.1. Attributes

Result Page가 가진 속성은 다음과 같다.

- Information Section: 문제에 대한 정보들을 표시하는 섹션이다.
- Code Comparing Section: 유저의 코드와 AI에 의해 보완된 코드가 나란하게 표시되는 섹션이다.
- Review Section: ChatGPT API가 유저의 코드에 대해 평가한 내용을 보여주는 섹션이다.

(1) Information Section 은 아래의 속성을 가지고 있다.

- Question Info: 현재 문제의 번호, 이름, 분류, 난이도를 나타내는 Bar이다.
- Programming Language: 유저가 제출할 때 사용한 언어를 표시한다.

(2) Code Comparing Section 은 아래의 속성을 가지고 있다.

- My Code: 유저가 제출한 코드를 보여주는 TextBox이다.

- Refactored Code: AI에 의해 보완된 코드를 보여주는 TextBox이다.

(3) Review Section 은 아래의 속성을 가지고 있다.

- Review: 제출된 코드에 대한 ChatGPT API의 리뷰를 출력하는 TextBox이다.

4.7.2. Methods

Result Page 가 가지고 있는 methods를 파트에 따라 나누면 다음과 같다.

(1) Information Section의 Methods

- showQuestionInfo(): Question Info에 문제 번호, 이름, 분류, 난이도를 표시해준다.
- showLanguageInfo(): Programming Language에 어떤 언어로 작성되었는지를 표시해준다.

(2) Code Comparing Section의 Methods

- showUserCode(): My Code에 유저가 작성하고 제출한 코드를 출력한다.
- showRefactoredCode(): Refactored Code에 AI에 의해 보완되어 작성된 코드를 Refactored Code에 출력한다.

(3) Review Section의 Methods

- showReview(): ChatGPT API의 코드 리뷰 내용을 Review에 출력한다.

4.7.3. State Diagram

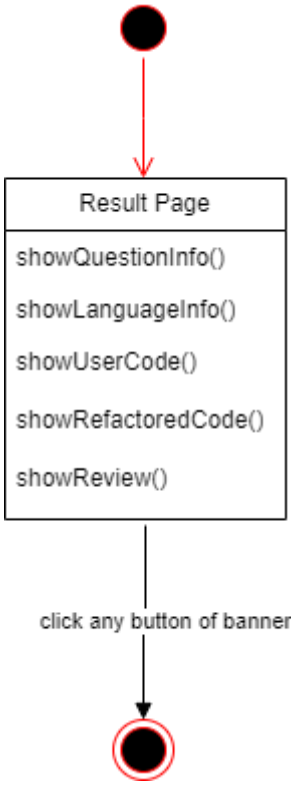

Figure 9 Result Page State Diagram

Result Page의 Body 단에는 유저와 상호작용할 수 있는 수단(버튼 등)이 존재하지 않기 때문에 Navigation Bar 혹은 Footer의 버튼 중 하나를 눌러야지 Result Page에서 빠져나갈 수 있다.

4.8. My Page

4.8.1. Attributes

My Page가 가진 속성은 다음과 같다.

- Account Settings Section: 유저가 본인의 계정과 관해 설정할 수 있는 도구들을 모아놓은 섹션이다.
- History Section: 유저가 최근 푼 문제와 최근 본 강의들을 모아놓은 섹션이다.

(1) Account Settings Section는 아래의 속성을 가지고 있다.

- Logout Button: 현재 로그인 된 계정에서 로그아웃하는 버튼이다.
- Nickname: 유저의 현재 닉네임을 표시하며, 클릭하여 닉네임을 변경할 수 있다.

(2) History Section는 아래의 속성을 가지고 있다.

- Recent Problems: 유저가 최근 푼 문제들의 목록이 표시되며, 문제를 클릭할 시 해당 문제 페이지로 연결된다.
- Recent Lectures: 유저가 최근 본 강의들의 목록이 표시되며, 문제를 클릭할 시 해당 강의 페이지로 연결된다.

4.8.2. Methods

My Page 가 가지고 있는 methods를 파트에 따라 나누면 다음과 같다.

(1) Account Settings Section 이 가지고 있는 method 는 아래와 같다.

- clickLogout(): 로그아웃 버튼을 누르면 현재 계정에서 로그아웃하고 홈페이지로 이동한다.
- getNickname(): 현재 계정의 닉네임을 가져온다.
- changeNickname(): 현재 계정의 닉네임을 변경한다.

(2) GitHub Account 가 가지고 있는 method 는 아래와 같다.

- getRecentProblems(): 유저가 최근 푼 문제들의 목록을 가져온다.
- clickProblem(): 문제 목록 중 하나를 클릭했을 때 해당 문제 페이지로 연결된다.
- getRecentLectures(): 유저가 최근 본 강의들의 목록을 가져온다.
- clickLecture(): 강의 목록 중 하나를 클릭했을 때 해당 강의 페이지로 연결된다.

4.8.3. State Diagram

사용자가 My Page에 입장하면, getNickname, getRecentProblems, getRecentLectures method가 실행되며 My Page를 구성하는 데 필요한 닉네임, 최근 푼 문제, 최근 본 강의 목록을 가져온다. 그 후 유저가 로그아웃 버튼을 클릭하면 계정에서 로그아웃하고 홈페이지로 이동하며, problem이나 lecture 중 하나를 클릭할 경우 해당 페이지로 연결된다. 닉네임을 클릭하여 새로운 이름을 치고 엔터를 누를 경우, changeNickname이 실행되며 서버에서 계정의 닉네임을 변경하고, 페이지를 리로드한다.

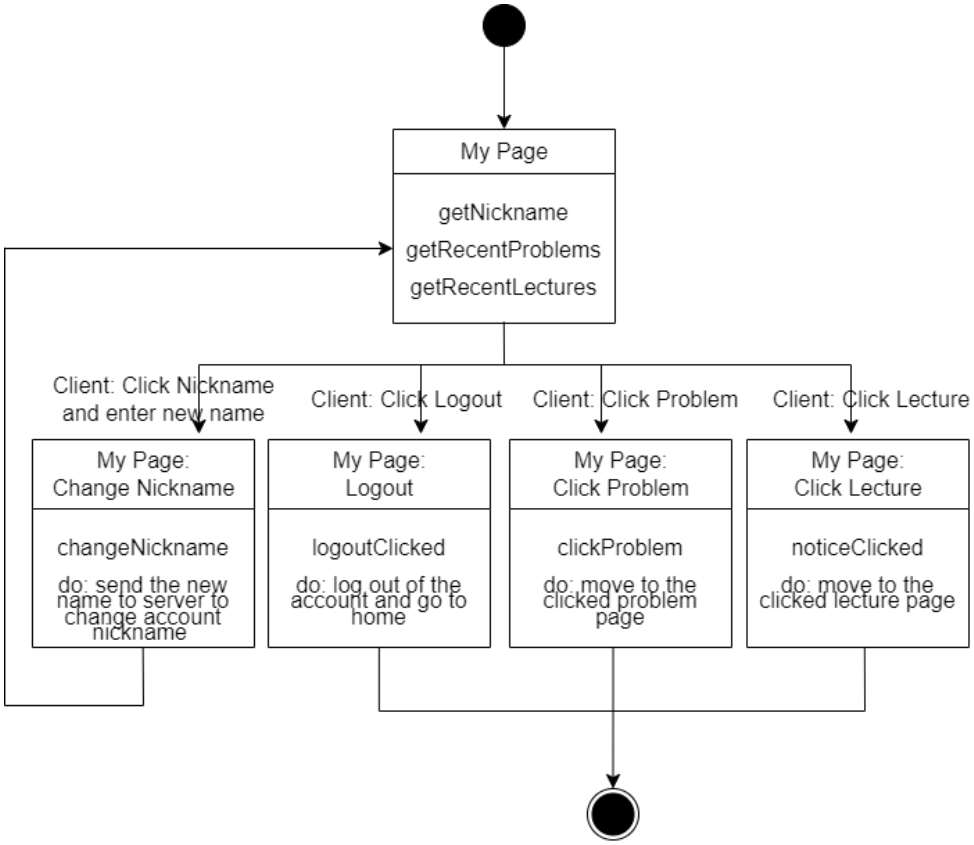

Figure 10 My Page State Diagram

4.9. Notice Page

4.9.1. Attributes

Notice Page가 가진 속성은 다음과 같다.

- Selection Buttons: 공지사항 버튼과 FAQ 버튼 두개가 나란히 표시된 형태이다 (기본적으로 공지사항 버튼이 눌려 있다).
- Information Section: 공지사항 혹은 FAQ 항목들의 제목과 내용을 보여주는 섹션이다.

(1) Selection Buttons는 아래의 속성을 가지고 있다.

- Notice Button: 공지사항 버튼이다.
- FAQ Button: FAQ 버튼이다.

(2) Information Section 은 아래의 속성을 가지고 있다.

- List: 공지사항 혹은 FAQ의 제목들을 Bar형태의 Element로 나열하는 리스트이다.

4.9.2. Methods

Notice Page 가 가지고 있는 methods를 파트에 따라 나누면 다음과 같다.

(1) Selection Buttons의 Methods

- clickNoticeBtn(): Notice Button(공지사항 버튼)을 클릭하면 Information Section에 공지사항들을 표시하게 한다.
- clickFAQBtn(): FAQ Button(FAQ 버튼)을 클릭하면 Information Section에 FAQ들을 표시하게 한다.

(2) Information Section의 Methods

- showNotices(): 공지사항 버튼이 눌렸을 경우 List에 공지사항 제목들을 Bar 형태로 나열한다.
- showFAQs(): FAQ 버튼이 눌렸을 경우 List에 FAQ 제목들을 Bar 형태로 나열한다.
- clickNotice(): List가 공지사항들을 보여주고 있을 때, 클릭된 공지사항 Bar에 해당하는 공지사항 내용을 드롭박스 형태로 보여준다. 이미 내용을 보여주고 있던 공지사항이었으면 내용을 접는다.
- clickFAQ(): List가 FAQ들을 보여주고 있을 때, 클릭된 FAQ Bar에 해당하는 FAQ의 내용을 드롭박스 형태로 보여준다. 이미 내용을 보여주고 있던 FAQ였으면 내용을 접는다.

4.9.3. State Diagram

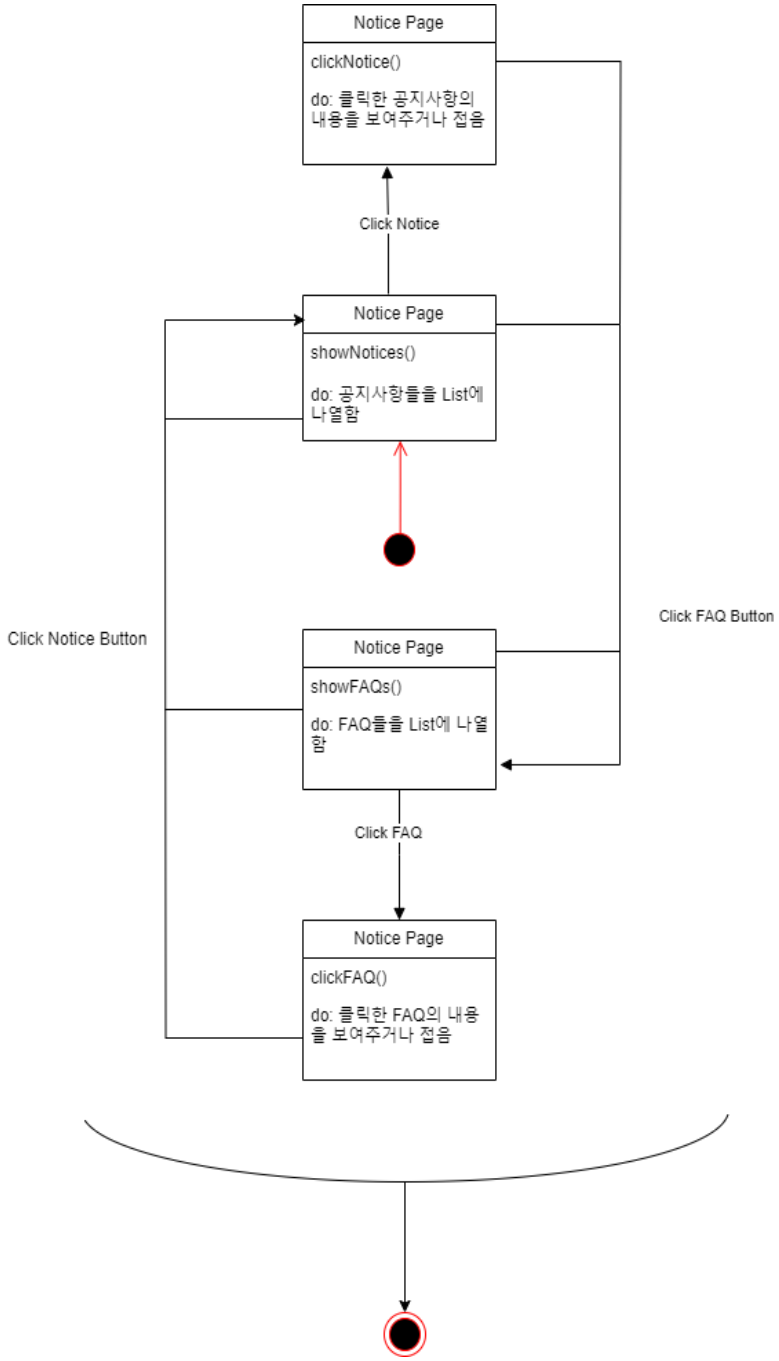

Figure 11 Notice Page State Diagram

Notice Page에서 다른 페이지로 이동하기 위해서는 Navigator나 Footer에 있는 버튼을 클릭하면 된다.

4.10. Problem Examiner Page

4.10.1.Attributes

Problem Examiner Page가 가진 속성은 다음과 같다.

- Question Info: Problem에 대한 메타데이터를 입력받는다.
- Problem Details: 문제를 이루는 요소들(허용된 언어, 문제 설명과 테스트 케이스)을 입력받는다.
- Submission: 문제 구성 요소가 모두 채워지면 문제로 등록한다.

(1) Question Info는 아래의 속성을 가지고 있다.

- Problem Number: 문제의 고유 번호를 입력받는다.
- Problem Name: 문제의 이름을 입력받는다.
- Problem Category: 문제의 분류를 선택한다.
- Problem Level: 문제의 난이도를 설정한다.

(2) Problem Details는 아래의 속성을 가지고 있다.

- Allowed Languages: 문제를 푸는 데 허용된 언어 목록을 선택한다.
- Problem Description: 유저들이 읽을 문제에 대한 상세 설명을 입력한다.
- Test Cases: 문제 채점을 위한 테스트 케이스들을 추가한다.

(3) Submission는 아래의 속성을 가지고 있다.

- Submit Button: 위 항목들을 모두 기입한 뒤 데이터베이스에 새 문제로 저장한다.

4.10.2. Methods

Problem Examiner Page가 가지고 있는 methods를 파트에 따라 나누면 다음과 같다.

(1) Question Info가 가지고 있는 method는 아래와 같다.

- checkQuestionInfo(): Questions Info에 들어갈 내용이 모두 정상적으로 기입되었는지 확인한다.

(2) Problem Details가 가지고 있는 method는 아래와 같다.

- checkProblemDetails(): Problem Details에 들어갈 내용이 모두 정상적으로 기입되었는지 확인한다.

(3) Submission가 가지고 있는 method는 아래와 같다.

- submitProblem(): 위 항목들이 모두 정상적으로 기입이 되었는지 확인한 뒤, 정상적

이라면 데이터베이스에 문제를 새로 넣기 위해 요청을 보내고, 응답이 확인되면 화면을 리로드한다.

4.10.3. State Diagram

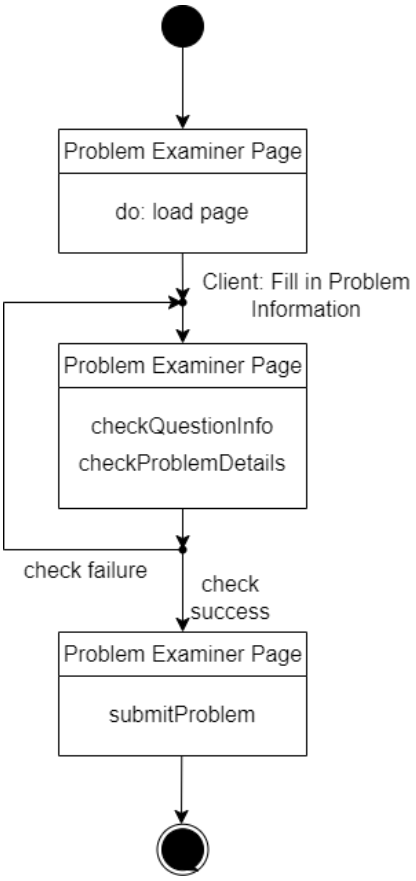

Figure 12 Problem Examiner Page State Diagram

5. System Architecture – Backend

5.1. Objectives

Backend의 전반적인 구조와 세부 시스템 구조를 설명한다. 각각의 Sub-System은 Class Diagram을 통해 객체 간의 관계를 확인하고 Sequence Diagram을 통해 데이터의 흐름을 파악한다. 각 DB에 대한 자세한 설명은 7장에서 확인할 수 있다.

5.2. Overall Architecture

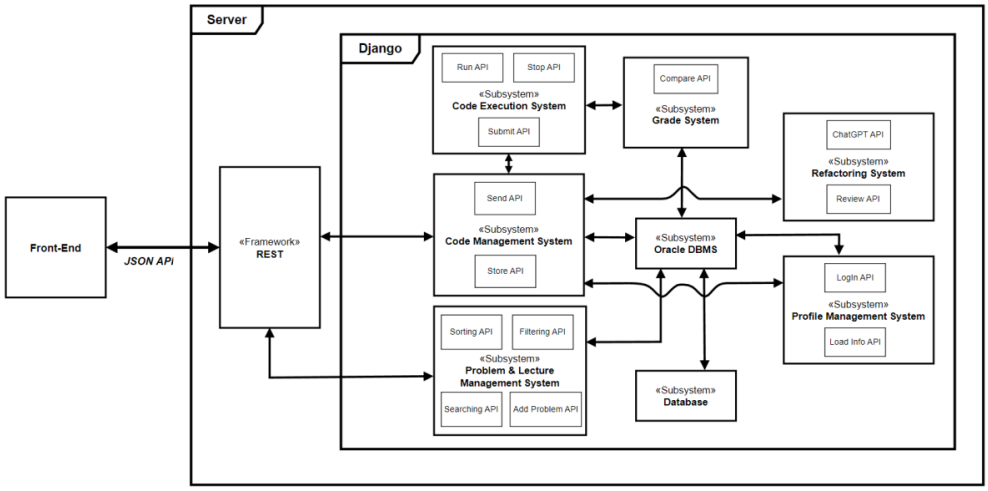

Figure 13 Overall Backend Architecture

Backend의 전체 구조는 위의 그림과 같다. 먼저, Frontend에서 서버로 요청을 보내면 Django에서 작성한 REST Framework에 알맞도록 Sub-System이 호출된다. Django는 총 여덟 개의 Sub-System으로 구성되어 있지만 이 중 Oracle DBMS와 Database를 제외한 나머지 6개의 Sub-System은 다음과 같은 부분을 담당한다.

- Code Management System: Code 저장 및 필요한 Sub-System으로의 전송을 담당
- Code Execution System: Code 실행, 중지, 제출을 담당
- Problem & Lecture Management System: 제공하는 컨텐츠들의 검색, 정렬 등을 담당
- Grade System: 실행한 Code 결과를 정답과 일치하는지 확인을 담당
- Refactoring System: ChatGPT API를 활용하여 코드 수정 및 리뷰를 담당
- Profile Management System: Login과 사용자 정보 관리를 담당

5.3. Subcomponents

5.3.1. Code Management System

Code Management System은 사용자가 제출한 코드의 저장과, 다른 subsystem이 해당 코드를 필요로 할 때 데이터베이스로부터 코드를 가져와 건네 주는 역할을 한다.

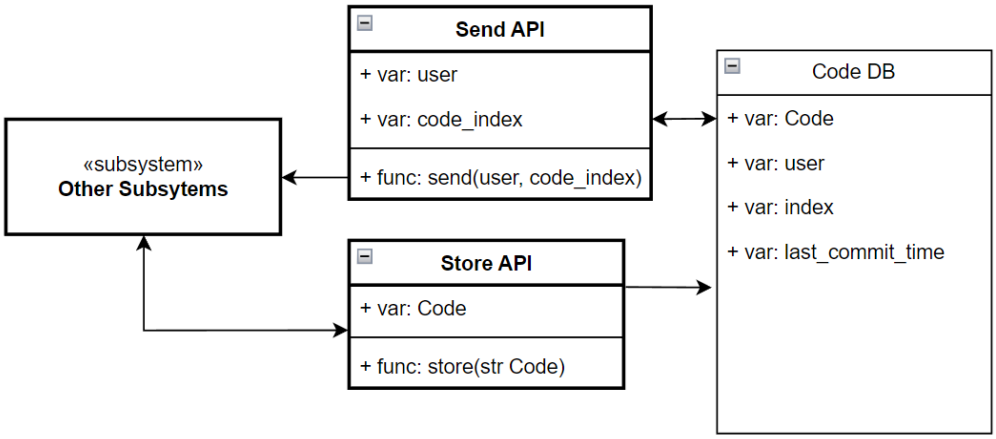

Figure 14 Code Management System Class Diagram

(1) Store API Class

사용자가 제출한 코드를 데이터베이스에 저장하는 역할을 한다. 사용자가 frontend를 통해 코드를 제출하면, Store API class는 이를 받아 Oracle DBMS를 통해 이 코드를 데이터베이스에 추가한다. 이는 store() 함수를 통해 이루어진다.

(2) Send API Class

Send API class는 다른 subsystem이 데이터베이스에 저장되어 있는 사용자의 코드를 필요할 때 사용된다. 외부 subsystem은 user와 code_index를 통해 특정 코드를 요청하며, 이를 parameter로 가지는 send 함수를 호출한다. 데이터베이스에 접근해 성공적으로 코드가 반환되면 해당 코드를 요청한 subsystem에게 전달한다.

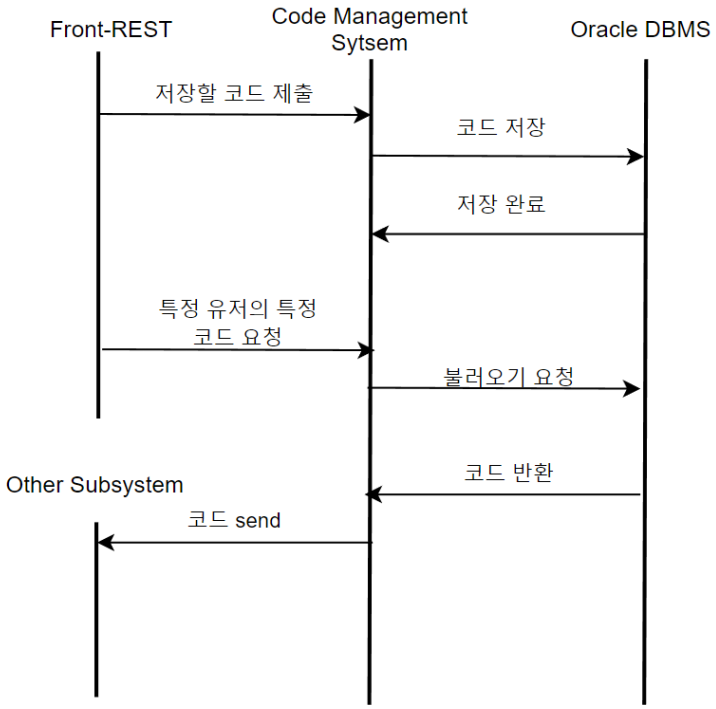

Figure 15 Code Management System Sequence Diagram

5.3.2. Code Execution System

Code Execution System은 문제를 풀기 위해 작성한 코드를 실행하고, 실행을 중단하거나, 문제를 해결했는지 확인하기 위해 코드를 제출하는 동작들을 담당하는 subsytem이다.

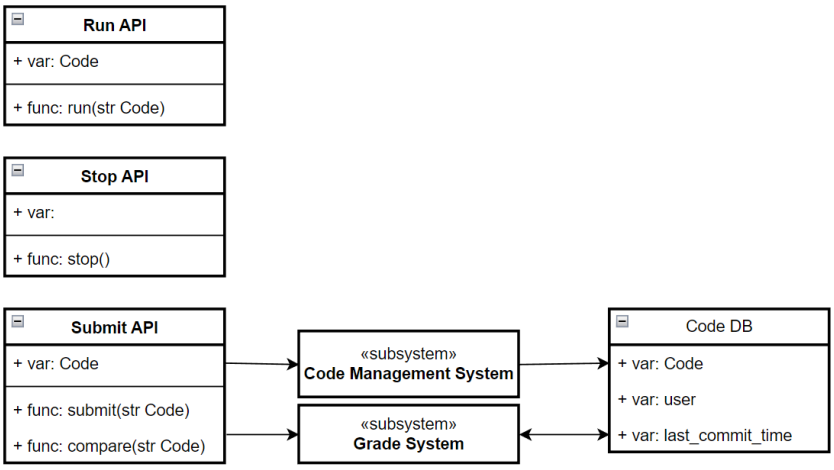

Figure 16 Code Execution System Class Diagram

(1) Run API Class

이 클래스는 frontend로부터 사용자가 작성한 코드를 받아, run() 함수를 통해 해당 코드를 실행한다. 실행 후 그 output을 다시 frontend로 반환해 사용자가 그 결과를 볼 수 있게 한다.

(2) Stop API Class

Stop API Class는 실행 중인 코드를 중단시키는 역할을 한다. Stop() 함수가 호출되면 실행 중인 코드가 중단되고, 종료 메시지가 frontend로 반환된다.

(3) Submit API Class

Submit API class는 submit() 함수를 사용해 사용자가 제출한 코드를 code management system에 넘겨 코드가 데이터베이스에 저장될 수 있게 하고, compare() 함수를 통해 코드를 grade system에 넘겨 여러 테스트케이스를 실행한 결과와 비교해 모든 테스트케이스를 통과했다면 정답, 아니라면 오답임을 사용자에게 알려준다.

5.3.3. Grade System

사용자가 작성한 Code를 실행한 결과를 받아 정답과 일치하는지 확인을 담당하는 시스템이다.

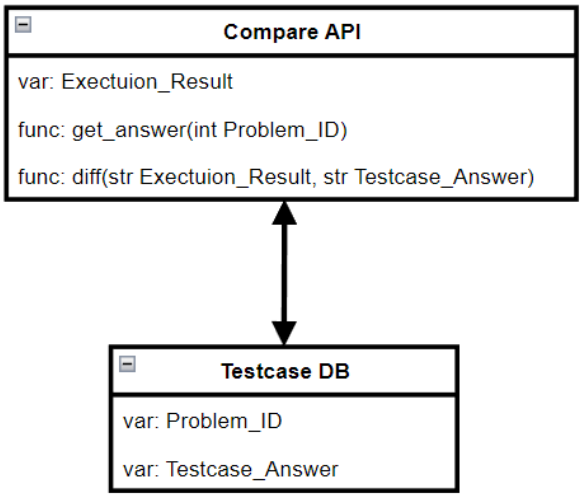

Figure 17 Grade System Class Diagram

(1) Compare API Class

실행 결과를 Execution_Result 변수에 할당한 후 get_answer(int Problem_ID)를 통해 해당 문제의 ID를 인자로 전달하고 정답을 DB로부터 받아온다. 받아온 Testcase_Answer와 Execution_Result와 일치하는지 확인을 하기 위해 diff()를 실행한다.

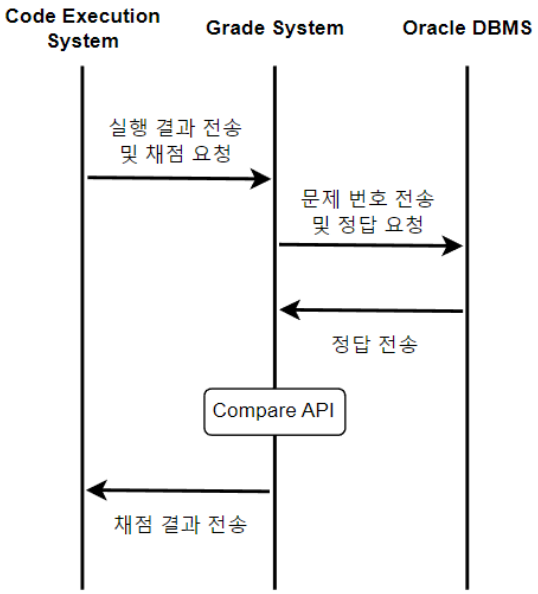

Figure 18 Grade System Sequence Diagram

5.3.4. Refactoring System

학습자가 제출한 코드를 개선시킬 수 있는 AI의 피드백을 제공하는 시스템이다.

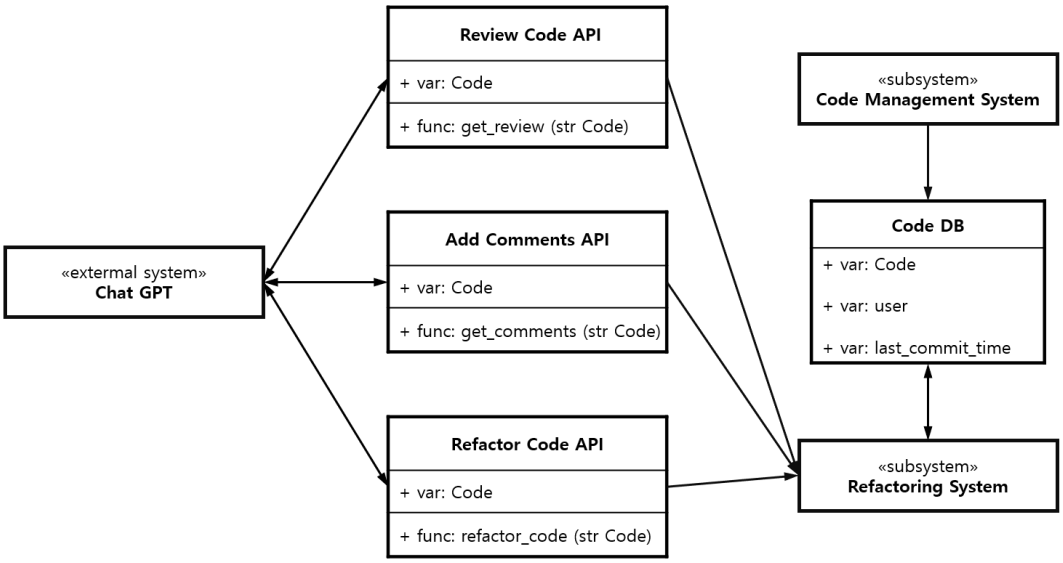

Figure 19 Refactoring System Class Diagram

(1) Review Code API

전반적인 코드에 대한 AI의 리뷰를 텍스트로 받아서 학습자에게 제공하는 역할을 한다. 해당 클래스에는 학습자가 작성한 코드를 parameter로 받아와 저장할 변수 Code와 chat

GPT로부터 코드에 대한 리뷰를 받아 제공하는 함수 get_review()가 있다.

(2) Add Comments API

학습자가 작성한 코드를 설명하는 주석을 달아주는 기능이다. 해당 클래스에는 학습자가 작성한 코드를 parameter로 받아와 저장할 변수 Code와 chat GPT를 통해 주석을 작성한 코드를 받아오는 함수 get_comments()가 있다.

(3) Refactor Code API

학습자가 작성한 코드를 수정하여 개선된 코드를 제공하는 기능이다. 해당 클래스에는 학습자가 작성한 코드를 parameter로 받아와 저장할 변수 Code와 chat GPT가 해당 코드를 개선하기 위해 수정한 코드를 받아오는 함수 refactor_code()가 있다.

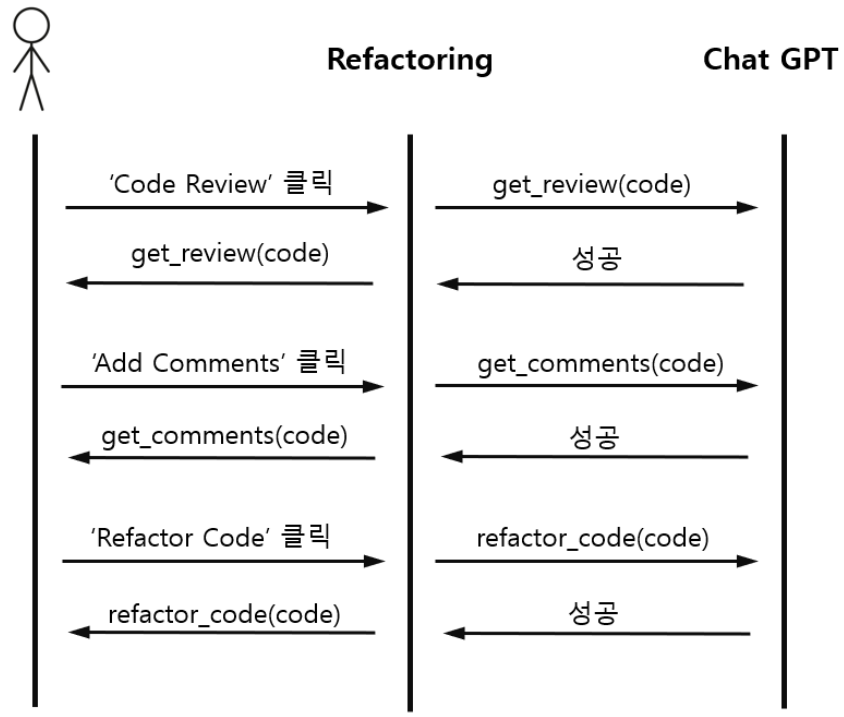

Figure 20 Refactoring System Sequence Diagram

5.3.5. Problem & Lecture Management System

문제와 강의와 같이 제공하는 컨텐츠들에 관한 전반적인 관리와 편의기능 제공을 담당하며 추가적으로 관리자 계정으로 문제를 추가할 수 있는 기능을 담당하는 시스템이다.

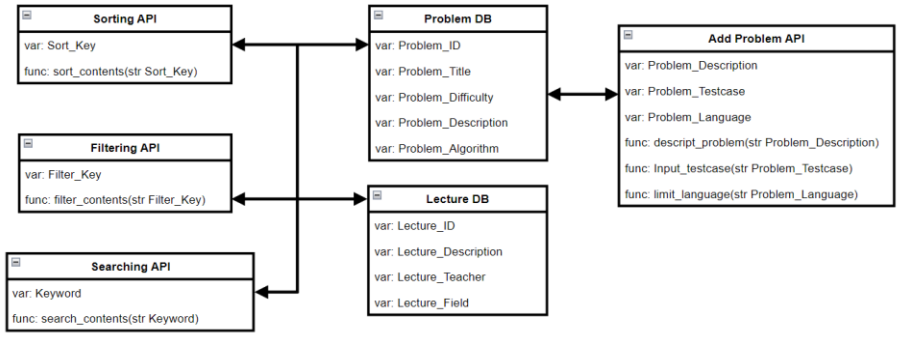

Figure 21 Problem & Lecture Management System Class Diagram

(1) Sorting API Class

사용자로부터 정렬을 위한 기준을 Sort_Key에 할당 받은 후 해당 Sort_Key를 인자로 가지는 sort_contents() 함수를 통해서 DB에 존재하는 컨텐츠들을 입력 받은 Sort_Key를 기준으로 정렬한다.

(2) Filtering API Class

사용자로부터 필터링을 위한 기준을 Filter_Key에 할당 받은 후 해당 Filter_Key를 인자로 가지는 filter_contents() 함수를 통해서 DB에 존재하는 컨텐츠들을 입력 받은 Filter_Key를 기준으로 여과한다.

(3) Searching API Class

사용자로부터 검색을 위한 문자열을 Keyword에 할당 받은 후 해당 Keyword를 인자로 가지는 search_contents() 함수를 통해서 DB에 존재하는 컨텐츠들에서 Keyword를 포함하고 있는 컨텐츠들을 선별한다.

(4) Add Problem API Class

관리자가 문제를 추가하기 위한 API Class이며 Problem DB에 새로운 문제를 추가하기 위해서 필요한 정보들을 Problem_Description, Problem_Testcase, Problem_Language에 각각 할당한 후 각 변수를 저장해주는 descript_problem(), input_testcase(), input_language() 함수를 통해서 Problem DB에 저장한다.

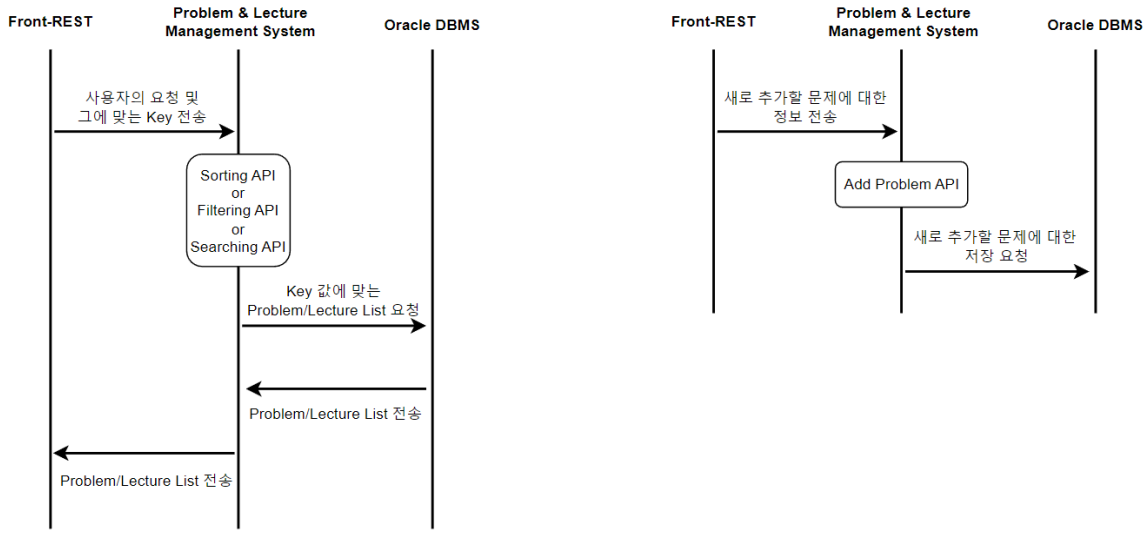

Figure 22 Problem & Lecture Management System Sequence Diagram

Figure 8의 왼쪽은 Sorting/Filtering/Searching API에 대한 Sequence Diagram이며 오른쪽은 Add Problem API에 대한 Sequence Diagram이다.

5.3.6. Profile Management System

Profile Management System은 사용자의 회원가입, 로그인, 그리고 사용자와 관련된 여러 정보들을 보여주는 subsystem이다.

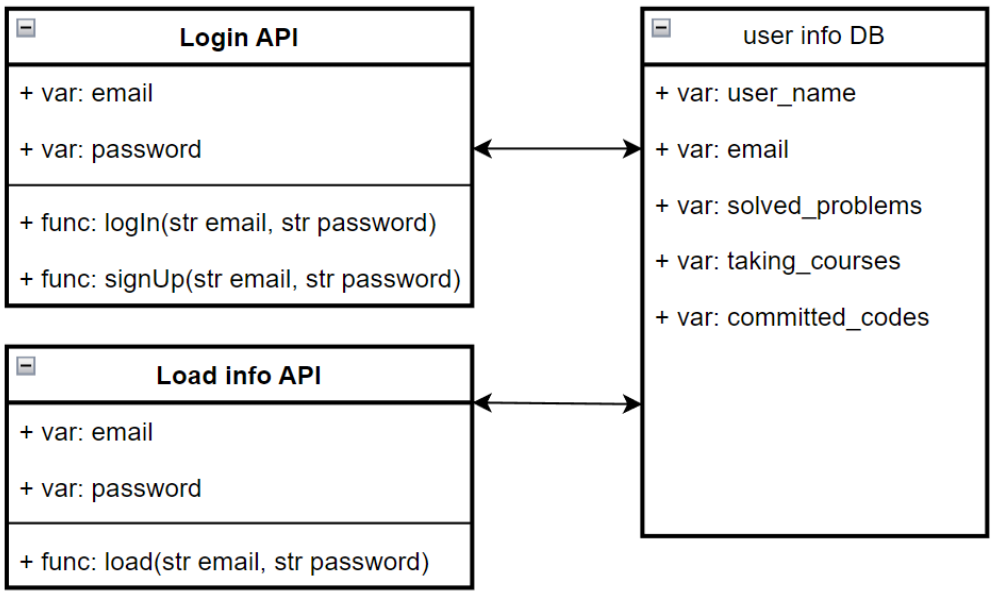

Figure 23. Profile Management System Class Diagram

(1) Login API Class

Login API class는 사용자의 로그인을 담당한다. frontend에서 email과 password를 받아 이것을 argument로 login 함수를 호출해, 데이터베이스에서 해당 유저가 존재하는 지를 확인한다. 존재한다면 로그인이 완료되고, 아니라면 실패 메시지가 반환된다. 이후 signUp 함수를 통해 회원가입을 진행할 수 있다. 회원가입을 하면 데이터베이스에 새로운 사용자 정보가 추가된다.

(2) Load info API Class

Load info API class는 사용자가 My Page를 요청했을 때 이에 필요한 정보를 가져와 보여주는 class이다. Login()과 동일하게 email과 password를 argument로 가지는 load() 함수를 호출해 데이터베이스에 접근한다. 이후 My page에 필요한 정보들을 가져와 이를 frontend에 넘겨주어 사용자가 자신과 관련된 정보를 확인할 수 있게 한다.

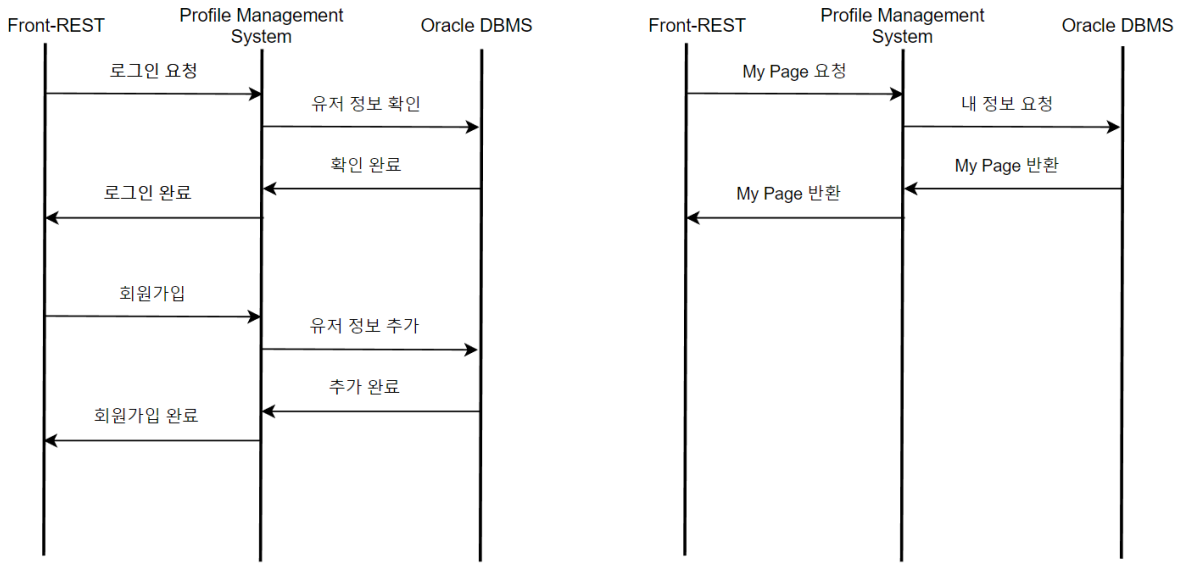

Figure 24. Profile Management System Sequence Diagram

6. Protocol

6.1. HTTP

HyperText Transfer Protocol, HTTP 프로토콜은 클라이언트와 서버가 통신하는 프로토콜이다. 웹에서 데이터를 교환할 때 사용하며 response 내에 status와 data를 포함하여 전달하며 JSON, HTML 등의 형식을 전달할 수 있다.

6.2. JSON

JavaScript Object Notation, JSON은 key-value 쌍, 배열 등 serialization 가능한 값을 전달하는 데이터 객체로 open standard format이다.

6.3. Protocol Description

서버와 클라이언트가 communication 하기 위해 필요한 프로토콜 디자인을 설명한다. REST API 형식을 따르며 버전은 v0와 v1으로 나누었다. v0는 샘플데이터를 반환하고 v1은 실제 데이터베이스에 저장된 데이터를 반환할 것이기 때문에 구조는 같으며 모든 설명은 v1 기준으로 작성하였다.

6.3.1. Problems List Protocol

Table 1 Problems List API

Attribute	Description	
Name	문제 리스트 조회	
Method	GET	
URL	problems/v1/list	
Parameters	-	
Response(List)	id	문제의 id 값
	title	문제 제목
	field	알고리즘 분류 리스트
	level	문제 난이도
Status	200	성공
	400	존재하지 않음

6.3.2. Problem Detail Protocol

Table 2 Problem Detail API

Attribute	Description	
Name	문제 상세 조회	
Method	GET	
URL	problems/v1/:id	
Parameters	id	문제의 id 값
Response(Object)	id	문제의 id 값
	title	문제 제목

	field	알고리즘 분류 리스트
	level	문제 난이도
	description	문제 설명
	tc_sample	샘플 테스트 케이스
Status	200	성공
	400	존재하지 않음

6.3.3. Problem Code Execution Protocol

Table 3 Problem Code Execution API

Attribute	Description	
Name	문제 코드 실행	
Method	POST	
URL	problems/v1/:id/exec	
Parameters	id	문제의 id 값
Request	code	입력한 코드
	language	선택한 프로그래밍 언어
	tc_user	유저가 입력한 테스트케이스
Response(Object)	id	문제의 id 값
	result	실행 결과
	exec_time	실행시간
Status	200	성공
	400	실패

6.3.4. Problem Code Save Protocol

Table 4 Problem Code Save API

Attribute	Description	
Name	문제 코드 저장	
Method	POST	
URL	problems/v1/:id/save	
Parameters	id	문제의 id 값
Request	code	입력한 코드
	language	선택한 프로그래밍 언어
	tc_user	유저가 입력한 테스트케이스
Response(Object)	id	문제의 id 값

	message	저장 결과 문구
Status	200	성공
	400	실패

6.3.5. Problem Code Submission Protocol

Table 5 Problem Code Submission API

Attribute	Description	
Name	문제 코드 제출	
Method	POST	
URL	problems/v1/:id/submit	
Parameters	id	문제의 id 값
Request	code	입력한 코드
	language	선택한 프로그래밍 언어
Response(Object)	id	문제의 id 값
	num_tc	테스트케이스 개수
	num_pass	통과한 테스트케이스 개수
	exec_time	실행시간
	result	제출 결과
Status	200	성공
	400	실패

6.3.6. Problem Code Load Protocol

Table 6 Problem Code Load API

Attribute	Description	
Name	저장 코드 불러오기	
Method	POST	
URL	problems/v1/:id/load	
Parameters	id	문제의 id 값
Response(Object)	id	문제의 id 값
	code	저장했던 코드
Status	200	성공
	400	실패

6.3.7. Problem Creation Protocol

Table 7 Problem Creation API

Attribute	Description	
Name	문제 생성	
Method	POST	
URL	problems/v1/create	
Parameters	-	
Request	title	문제 제목
	field	알고리즘 분류 리스트
	level	문제 난이도
	description	문제 설명
	tc_sample	샘플 테스트 케이스
Response(Object)	id	생성된 문제의 id
	message	생성 결과 메시지
Status	200	성공
	400	실패

6.3.8. Code Commit Protocol

Table 8 Code Commit API

Attribute	Description	
Name	코드 커밋	
Method	POST	
URL	code/v1/commit	
Parameters	-	
Request	code	입력한 코드
	message	커밋 메시지
	repository	리포지토리 이름
Response(Object)	result	커밋 결과
Status	200	성공
	400	실패

6.3.9. Code Review Request Protocol

Table 9 Code Review Request API

Attribute	Description	
Name	코드 리뷰 요청	
Method	POST	
URL	code/v1/review	
Parameters	-	
Request	code	지정한 코드
Response(Object)	id	문제의 id 값
	code	변경된 코드
	message	부연 설명
Status	200	성공
	400	실패

6.3.10. Code Comment Request Protocol

Table 10 Code Comment Request API

Attribute	Description	
Name	코드 주석 요청	
Method	POST	
URL	code/v1/comment	
Parameters	-	
Request	code	지정한 코드
Response(Object)	id	문제의 id 값
	code	변경된 코드
	message	부연설명
Status	200	성공
	400	실패

6.3.11. Dead Code Detection Request Protocol

Table 11 Dead Code Detection Request API

Attribute	Description	
Name	코드 불필요 코드 탐색 요청	
Method	POST	
URL	code/v1/dead	
Parameters	-	

Request	code	지정한 코드
Response(Object)	id	문제의 id 값
	code	변경된 코드
	message	부연설명
Status	200	성공
	400	실패

6.3.12. Code Refactoring Request Protocol

Table 12 Code Refactoring Request API

Attribute	Description	
Name	코드 불필요 코드 탐색 요청	
Method	POST	
URL	code/v1/refactoring	
Parameters	-	
Request	code	지정한 코드
Response(Object)	id	문제의 id 값
	code	변경된 코드
	message	부연설명
Status	200	성공
	400	실패

6.3.13. GitHub OAuth Login Protocol

Table 13 GitHub OAuth Login API

Attribute	Description	
Name	GiHub OAuth 로그인	
Method	POST	
URL	login/github/callback	
Parameters	-	
Request	-	
Response(Object)	user_id	유저 id 값
	access_token	GitHub Access Token
	refresh_token	GitHub Refresh Token
Status	200	성공
	400	실패

6.3.14. Lecture Guideline Recommendation Protocol

Table 14 Lecture Guideline Recommendation API

Attribute	Description	
Name	강의 가이드라인 조회	
Method	POST	
URL	user/v1/lectures/guideline	
Parameters	-	
Request	user_id	유저 id 값
Response(List)	user_id	유저 id 값
	lecture_link	강의 링크
	lecture_title	강의 제목
Status	200	성공
	400	실패

6.3.15. Lecture History Protocol

Table 15 Lecture History API

Attribute	Description	
Name	강의 기록 조회	
Method	POST	
URL	user/v1/lectures/history	
Parameters	-	
Request	user_id	유저 id 값
Response(List)	user_id	유저 id 값
	lecture_link	강의 링크
	lecture_title	강의 제목
Status	200	성공
	400	존재하지 않음

6.3.16. User Problem History Protocol

Table 16 User Problem History API

Attribute	Description
Name	유저 문제풀이 기록 조회

Method	POST	
URL	user/v1/problems	
Parameters	-	
Request	user_id	유저 id 값
Response	user_id	유저 id 값
	title	문제 제목
	problem_id	문제 id
	submit_at	제출날짜
	result	제출 결과
Status	200	성공
	400	존재하지 않음

6.3.17. Notice Protocol

Table 17 Notice Protocol

Attribute	Description	
Name	Notice 조회	
Method	Get	
URL	boards/v1/notice/:id	
Parameters	id	게시글 id
Response	title	게시글 제목
	description	게시글 내용
	create_at	생성날짜
Status	200	성공
	400	존재하지 않음

7. Database Design

7.1. Objectives

데이터베이스의 디자인을 설명한다. 데이터베이스의 테이블 구조를 설명하고 Relational Schema를 다이어그램으로 제시하고자 한다.

7.2. Entity Tables

7.2.1. Problem App Tables

Table 18 Problem Table

Field	Key	Type	Description
id	PK	Number	문제 id
title		NVARCHAR	문제 제목
level		Number	문제 난이도
description		NClop	문제 설명
create_at		Timestamp	생성시간

Table 19 AlgorithmField Table

Field	Key	Type	Description
id	PK	Number	알고리즘 분야 id
field		NVARCHAR	알고리즘 분야

Table 20 ProblemFieldRelation Table

Field	Key	Type	Description
id	PK	Number	문제와 분야 relation id
field	FK	Number	알고리즘 분야 id
problem	FK	Number	문제 id

Table 21 Testcase Table

Field	Key	Type	Description
id	PK	Number	Testcase id
problem_id	FK	Number	문제 id
testcase		NCLOB	테스트케이스
result		NCLOB	테스트케이스 결과
is_sample		Number	샘플케이스 여부

Table 22 Execution Table

Field	Key	Type	Description
id	PK	Number	실행 요청 id
problem_id	FK	Number	문제 id
lang		NCLOB	사용 프로그래밍 언어

create_at		Timestamp	실행 요청일
status		Number	실행 상태
exec_time		Float	코드 실행시간
user		Number	유저 id
result		NCLOB	실행 결과

Table 23 Submission Table

Field	Key	Type	Description
id	PK	Number	제출 id
problem_id	FK	Number	문제 id
lang		NVARCHAR	사용 프로그래밍 언어
create_at		Timestamp	제출 요청일
status		NVARCHAR	실행 상태
exec_time		Float	코드 실행시간
user_id		Number	유저 id
num_pass		Number	패스한 테스트케이스 개수

Table 24 UserCodeHistory Table

Field	Key	Type	Description
id	PK	Number	코드 저장 id
problem_id	FK	Number	문제 id
user_id		Number	유저 id
create_at		Timestamp	저장 요청일
version		Number	버전 넘버
code		NCLOB	저장된 코드
memo		Number	메모

7.2.2. Code App Tables

Table 25 Commit Table

Field	Key	Type	Description
id	PK	Number	코드 저장 id
commit_message		NCLOB	커밋 메시지
user_id		Number	유저 id
repo_name		NVARCHAR	리포지토리 이름

create_at		Timestamp	커밋 생성일

Table 26 Review Table

Field	Key	Type	Description
id	PK	Number	코드 저장 id
code		NCLOB	바뀐 코드
message		NCLOB	리뷰 메시지
history_id	FK	Number	저장 코드 id
create_at		Timestamp	요청일

Table 27 Review Table

Field	Key	Type	Description
id	PK	Number	코드 저장 id
code		NCLOB	개선 코드 제안
message		NCLOB	리뷰 메시지
history_id	FK	Number	저장 코드 id
create_at		Timestamp	요청일

Table 28 Comment Table

Field	Key	Type	Description
id	PK	Number	코드 저장 id
code		NCLOB	개선 코드 제안
message		NCLOB	주석 관련 메시지
history_id	FK	Number	저장 코드 id
create_at		Timestamp	요청일

Table 29 Deadcode Table

Field	Key	Type	Description
id	PK	Number	코드 저장 id
code		NCLOB	개선 코드 제안
message		NCLOB	안 쓰는 코드 탐지 메시지
history_id	FK	Number	저장 코드 id
create_at		Timestamp	요청일

Table 30 Refactor Table

Field	Key	Type	Description
id	PK	Number	코드 저장 id
code		NCLOB	개선 코드 제안
message		NCLOB	리팩토링 메시지
history_id	FK	Number	저장 코드 id
create_at		Timestamp	요청일

7.2.3. Lecture App Tables

Table 31 Lecture Table

Field	Key	Type	Description
id	PK	Number	강의 id
title		NVARCHAR	강의 제목
video_link		NVARCHAR	강의 유튜브 링크
author_id		Number	업로드 유저 id
create_at		Timestamp	업로드 일시

Table 32 SoftwareField Table

Field	Key	Type	Description
id	PK	Number	소프트웨어 분야 id
field		NVARCHAR	소프트웨어 분야

Table 33 LectureFieldRelation

Field	Key	Type	Description
id	PK	Number	강의 분야 관계 id
field_id	FK	Number	소프트웨어 분야 id
lecture_id	FK	Number	강의 id

Table 34 LectureHistory Table

Field	Key	Type	Description
id	PK	Number	강의 기록 id
lecture_id	FK	Number	강의 id
user_id		Number	유저 id

create_at		Timestamp	업로드 일시

7.2.4. Boards App Tables

Table 35 Board Table

Field	Key	Type	Description
id	PK	Number	게시판 id
title		NVARCHAR	게시판 이름
create_at		Timestamp	생성 일시

Table 36 Article

Field	Key	Type	Description
id	PK	Number	게시판 id
title		NVARCHAR	게시글 제목
body		NCLOB	게시글 내용
author_id		Number	작성자 유저 id
board_id	FK	Number	게시판 id
create_at		Timestamp	생성 일시
update_at		Timestamp	업데이트 일시

7.2.5. Accounts App Tables

Table 37 User Table

Field	Key	Type	Description
id	PK	Number	사용자 id
name		NVARCHAR	사용자 이름
github_username		NVARCHAR	GitHub username
email		NVARCHAR	이메일
create_at	FK	Timestamp	생성 일시
last_login		Timestamp	마지막 로그인 일시
profile_image		NVARCHAR	프로필 이미지 경로
is_superuser		Number	Admin 여부
is_lecturer		Number	강의 업로드 권한 여부

7.3. Relational Schema

7.2장을 기반으로 Django의 models.py를 작성한 이후 migration한 결과를 스키마로 나타내었다.

(1) Problems와 Boards의 Relational Schema

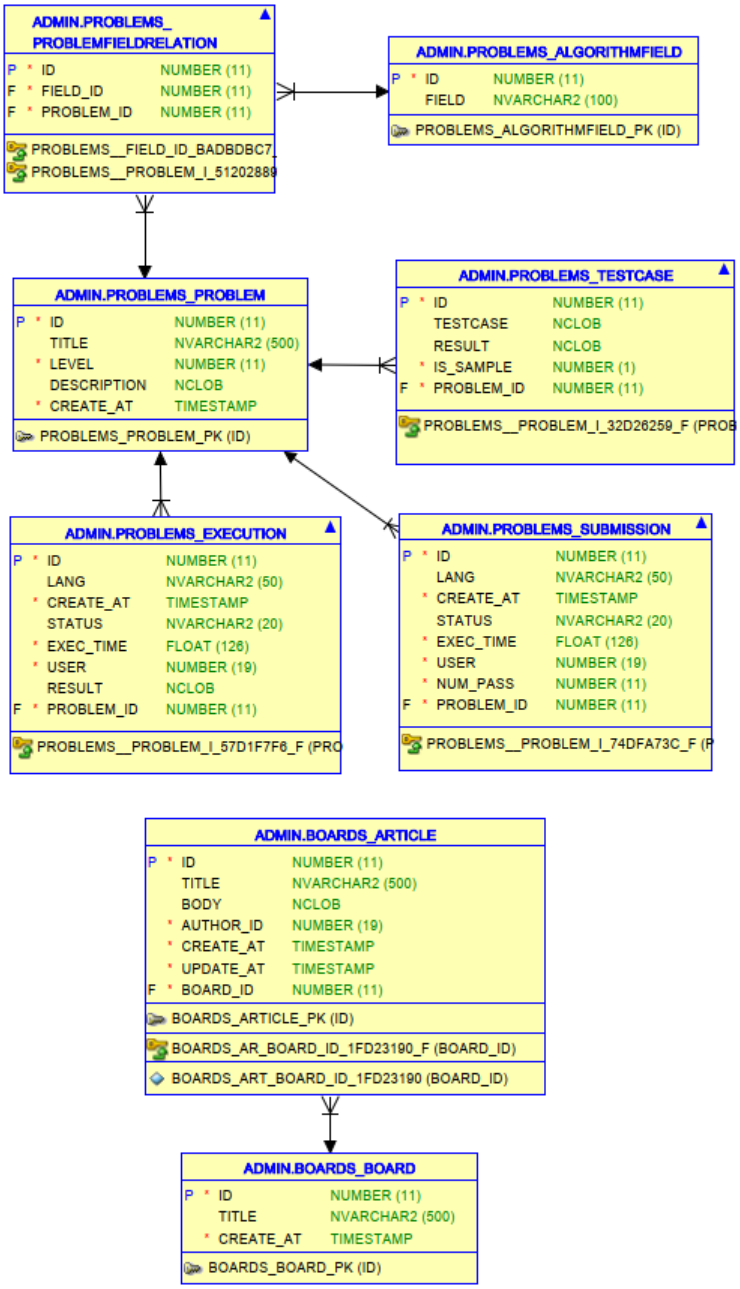

Figure 25 Problems와 Boards의 Relational Schema

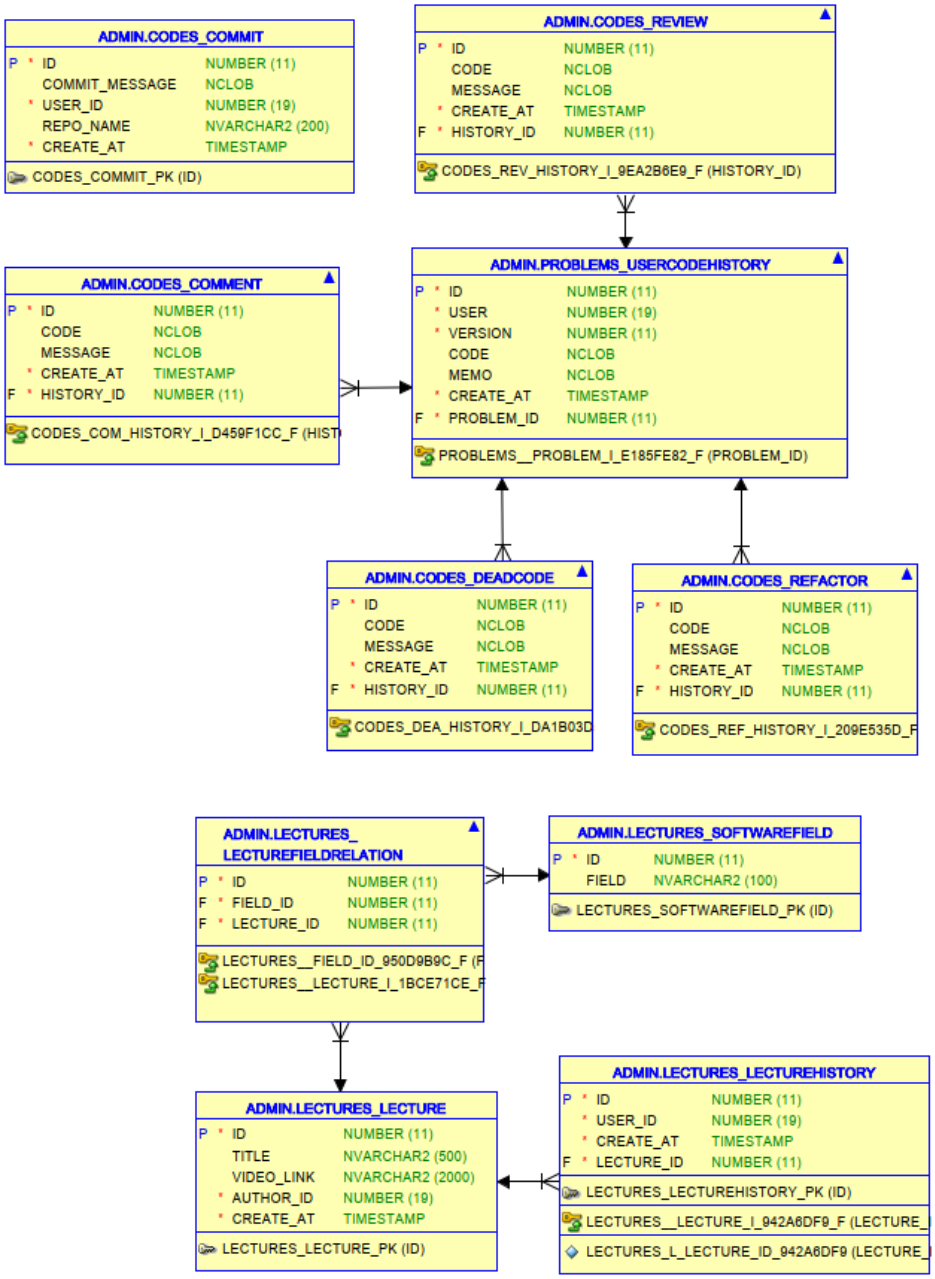

Figure 26 Codes와 Lectures의 Relational Schema

8. Supporting Information

8.1. Software Design Specification

본 소프트웨어 디자인 명세서는 IEEE 권장 사항에 맞추어 작성되었다 (IEEE Standard for Information Technology Systems Design Software Design Descriptions, IEEE-Std-1016). 다만

본 시스템의 요구사항에 맞게 Structure 관점, Interaction 관점, State Dynamics 관점, Information 관점 위주로 작성되었다.

8.2. Document History

Date	Version	Description	Writer
2023.05.19	0.1	초안 작성	김지윤 외 6명
2023.05.20	1.0	통합	김지윤 외 6명