

# **Software Design Specification**

**for**

**StoryTelling-based  
Personalized Learning Platform**



2018314952 박성민  
2018312672 김민석  
2017310277 서보현  
2017314533 유현택  
2018314980 이재필  
2018312863 조성원  
2019313156 채서영

소프트웨어공학개론 6 조

2023. 05. 21

# 목차

<b>1. PREFACE .....</b>	<b>7</b>
<b>1.1. OBJECTIVE.....</b>	<b>7</b>
<b>1.2. READERSHIP.....</b>	<b>7</b>
<b>1.3. DOCUMENT STRUCTURE.....</b>	<b>7</b>
1.3.1 INTRODUCTION .....	7
1.3.2 OVERALL SYSTEM ARCHITECTURE .....	7
1.3.3 SYSTEM ARCHITECTURE - FRONTEND .....	7
1.3.4 SYSTEM ARCHITECTURE - BACKEND.....	7
1.3.5 SYSTEM ARCHITECTURE - AI.....	7
1.3.6 PROTOCOL DESIGN.....	7
1.3.7 DATABASE / SQL .....	7
<b>2 INTRODUCTION.....</b>	<b>8</b>
<b>2.1. APPLIED DIAGRAM.....</b>	<b>8</b>
<b>2.2 SYSTEM OVERVIEW.....</b>	<b>8</b>
<b>3 SYSTEM ARCHITECTURE – OVERALL.....</b>	<b>9</b>
<b>3.1 SYSTEM ORGANIZATION.....</b>	<b>9</b>
<b>3.2. SYSTEM ARCHITECTURE – FRONTEND APPLICATION .....</b>	<b>9</b>
<b>3.3. SYSTEM ARCHITECTURE – AI .....</b>	<b>10</b>
<b>3.4. SYSTEM ARCHITECTURE – BACKEND.....</b>	<b>10</b>
<b>4. SYSTEM ARCHITECTURE – FRONTEND.....</b>	<b>11</b>
<b>4.1. OBJECTIVES.....</b>	<b>11</b>
<b>4.2. MAIN PAGE .....</b>	<b>11</b>
4.2.1. ATTRIBUTES.....	12
4.2.2. METHODS .....	12
4.2.3. STATE DIAGRAM .....	13
<b>4.3. LOGIN/SIGN UP PAGE .....</b>	<b>14</b>
4.3.1. ATTRIBUTES.....	14
4.3.2. METHODS .....	14
4.3.3. STATE DIAGRAM .....	15
<b>4.4. CURRICULUM PAGE.....</b>	<b>16</b>
4.4.1. ATTRIBUTES.....	16

4.4.2. METHODS .....	17
4.4.3. STATE DIAGRAM .....	18
<b>4.5. WORKBOOK.....</b>	<b>18</b>
4.5.1. ATTRIBUTES.....	18
4.5.2. METHODS .....	19
4.5.3. STATE DIAGRAM .....	19
<b>4.6. EDITOR PAGE.....</b>	<b>20</b>
4.6.1. ATTRIBUTES.....	21
4.6.2. METHODS .....	22
4.6.3. STATE DIAGRAM .....	22
<b>5. SYSTEM ARCHITECTURE – AI .....</b>	<b>23</b>
<b>5.1. OBJECTIVES.....</b>	<b>23</b>
<b>5.2. OVERALL ARCHITECTURE .....</b>	<b>23</b>
<b>5.3. SUBCOMPONENTS .....</b>	<b>23</b>
5.3.1. WORKBOOK GENERATE/ANSWER ANALYSIS SYSTEM .....	23
5.3.2. DATA PARSER/RECOMMENDER SYSTEM .....	26
<b>6. SYSTEM ARCHITECTURE – BACKEND .....</b>	<b>28</b>
<b>6.1. OBJECTIVES.....</b>	<b>28</b>
<b>6.2. OVERALL ARCHITECTURE .....</b>	<b>28</b>
<b>6.3. SUBCOMPONENTS .....</b>	<b>29</b>
6.3.1. CODING TEST SYSTEM.....	29
6.3.2. STORY TELLING SYSTEM.....	30
6.3.3. LOGIN/REGISTERG SYSTEM .....	31
6.3.4. QUESTION LOG SYSTEM.....	32
<b>7. PROTOCOL.....</b>	<b>34</b>
<b>7.1. HTTP .....</b>	<b>34</b>
<b>7.2. JSON .....</b>	<b>34</b>
<b>7.3. REST .....</b>	<b>34</b>
<b>7.4. PROTOCOL DESCRIPTION.....</b>	<b>34</b>
7.4.1. 학습지 조회 프로토콜.....	34
7.4.2. 문제 조회 프로토콜.....	34
7.4.3. 커리큘럼 제너레이션 프로토콜.....	35
7.4.4. 학습지 제너레이션 프로토콜 .....	35
7.4.5. 페이지 조회 프로토콜.....	35

7.4.6. 학습 진행도 프로토콜.....	36
7.4.7. 강의 상태 저장 프로토콜 .....	36
7.4.8. 커리큘럼 요청 프로토콜 .....	36
7.4.9. 학습지 요청 프로토콜.....	37
7.4.10. 문제 로그 조회 프로토콜.....	37
7.4.11. 코드 제출 프로토콜 .....	37
7.4.12. 로그인 요청 프로토콜 .....	38
7.4.13. 회원가입 프로토콜.....	38
<b>8. DATABASE DESIGN.....</b>	<b>39</b>
<b>8.1. OBJECTIVES.....</b>	<b>39</b>
<b>8.2. ER DIAGRAM.....</b>	<b>39</b>
<b>8.3. RELATIONAL SCHEMA.....</b>	<b>39</b>
<b>8.4. MODELS.PY.....</b>	<b>40</b>
8.4.1. USER DATA.....	40
8.4.2. USER INFO .....	40
8.4.3. LEETCODE INTEGRATED .....	41
8.4.4. QUESTIONS RECOMMEND RES .....	41

## 그림 목차

Figure 1 System Organization.....	9
Figure 2 Overall Frontend Architecture.....	9
Figure 3. Overall AI Architecture.....	9
Figure 4. Overall Backend Architecture .....	10
Figure 5. State Diagram of Main Page .....	13
Figure 6. State Diagram of Login/Sign up.....	16
Figure 7. State Diagram of Curriculum page .....	18
Figure 8. State Diagram of Workbook page .....	20
Figure 9. State Diagram of Editor page.....	22
Figure 10. Overall AI Architecture .....	23
Figure 11. Workbook Generate/Answer Analysis System Class Diagram.....	24
Figure 12. Workbook Generate System Sequence Diagram.....	25
Figure 13. Workbook Generate System Sequence Diagram.....	25
Figure 14. Answer Analysis System Sequence Diagram.....	25
Figure 15. Data parser / Recommender System Sequence Diagram.....	26
Figure 16. Data parser / Recommender System Class Diagram.....	27
Figure 17. Data parser / Recommender System Sequence Diagram .....	27
Figure 18. Overall Backend Architecture.....	28
Figure 19. Code Test System Class Diagram .....	29
Figure 20. Code Test System Sequence Diagram .....	29
Figure 21. Storytelling System Class Diagram.....	30
Figure 22. Code Management System Sequence Diagram .....	30
Figure 23. Log Register System Class Diagram .....	31
Figure 24. Log Register System Sequence Diagram .....	31
Figure 25. Log Register System Sequence Diagram .....	32
Figure 26. Log System Class Diagram.....	32
Figure 27. Log System Sequence Diagram .....	33
Figure 28. Log System Sequence Diagram .....	33
Figure 29. ER Diagram .....	39
Figure 30. Relational Schema .....	40

## 표 목차

Table 1. 학습지 조회 프로토콜 표.....	34
Table 2. 코드 채점 프로토콜 표 .....	34
Table 3. 커리큘럼 제너레이션 프로토콜 표.....	35
Table 4. 학습지 제너레이션 프로토콜 표.....	35
Table 5. 페이지 조회 프로토콜 표.....	36
Table 6. 학습 진행도 프로토콜 표.....	36
Table 7. 강의 상태 저장 프로토콜 표 .....	36
Table 8. 커리큘럼 요청 프로토콜 표.....	36
Table 9. 학습지 요청 프로토콜 표.....	37
Table 10. 문제 로그 조회 프로토콜 표.....	37
Table 11. 코드 제출 프로토콜 표.....	37
Table 12. 로그인 프로토콜 표.....	38
Table 13. 회원가입 프로토콜 표.....	38

# **1. Preface**

## **1.1. Objective**

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

## **1.2 Readership**

본 문서는 독자층을 AI 코딩 교육 플랫폼을 개발 및 유지 보수하는 소프트웨어 엔지니어, 시스템의 구조를 설계하는 시스템 아키텍처 등으로 상정하여 요구사항과 구성 요소를 이해하기 쉽도록 돋기 위해 작성되었다.

## **1.3 Document Structure**

### **1.3.1 Introduction**

이 장에서는 전반적인 그림 목차 항목을 찾을 수 없습니다. 시스템의 구조에 대하여 기술한다. 각 서브 시스템 구조에 대한 소개와 함께 전체 시스템 구조에 대한 소개를 통해 서브 시스템 간의 상호작용 관계를 나타낸다.

### **1.3.2 Overall System Architecture**

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

### **1.3.3 System Architecture - Frontend**

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

### **1.3.4 System Architecture – Backend**

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

### **1.3.5 System Architecture – AI**

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

### **1.3.6 Protocol Design**

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

### **1.3.7 Database / SQL**

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

## 2 Introduction

### 2.1. Applied Diagram

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

- State Diagram

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

- Sequence Diagram

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

- Class Diagram

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

### 2.2 System Overview

본 문서는 “StoryTelling-based Personalized Learning Platform” 서비스를 위한 디자인 명세서이다. 이 서비스는 스토리텔링을 기반으로 한 코딩 학습 플랫폼으로, 사용자에게 개인화된 학습 경험을 제공한다. 사용자는 자신이 원하는 캐릭터를 선택하고, 스토리텔러가 이야기를 들려주면서 캐릭터와 함께 학습할 수 있다는 점이 큰 차별화된 특징이다.

사용자는 학습 목적에 따라 프로그래밍 언어로 Python, C++, Java, Javascript를 선택할 수 있으며, 서비스는 언어에 따라 초급, 중급, 고급의 난이도를 제공한다. 학습은 workbook과 problem solving으로 나누어진다. Workbook에서는 GPT를 기반으로 하여 curriculum generate, workbook generate, text style transfer 과정을 거쳐 사용자에게 원하는 언어, 난이도, 텔러(ex. 인어공주)를 기반으로 만들어진 학습지를 제공한다. Problem solving에서는 추천 시스템을 기반으로 하여 개별화된 문제를 제공하여, 코드 제출 후 AI model response를 기반으로 한 코드 분석 및 결과를 제공받는다. 따라서, 사용자는 단순히 코딩을 배우고 문제를 푸는 것이 아닌, 사용자 맞춤화된 학습 서비스를 제공함으로써 더욱 효과적인 학습 경험이 가능해진다.

이러한 서비스의 기능들을 사용자가 편리하고 직관적으로 사용할 수 있도록 UI 및 UX를 설계한다. 서비스를 이용하는 데에 사용자는 별다른 UI/UX적 경험을 요구하지 않는다. 즉, 서비스를 처음 이용하는 사람도 편리하게 본 서비스의 기능들을 이용할 수 있도록 하는 것을 목표로 한다.

본 서비스는 현재 명시된 시스템의 구현 이후에도 시스템 확장 및 유지보수가 용이하도록 설계한다. front-end, AI model, back-end 간의 상호작용을 명시하고 체계화한다.

### 3 System Architecture – Overall

#### 3.1 System Organization

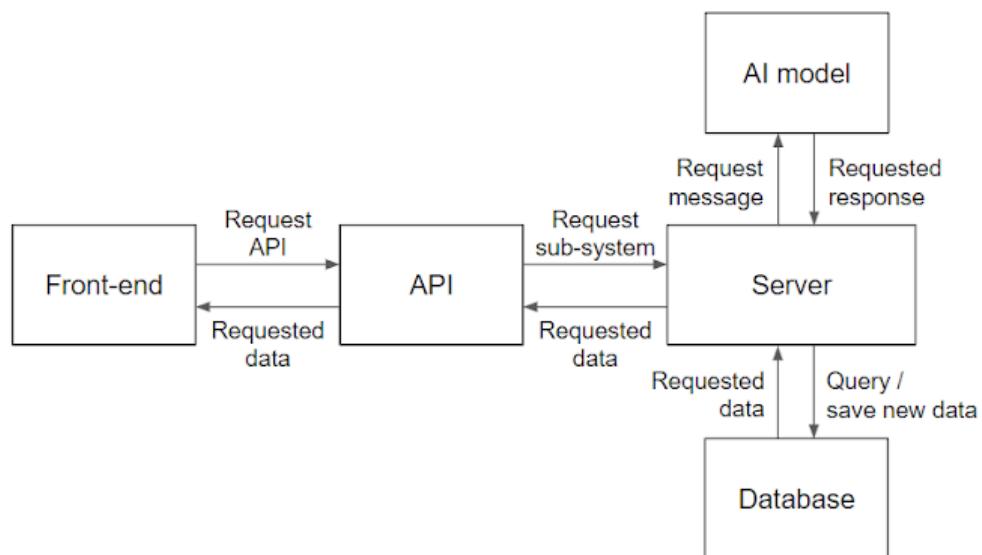


Figure 1 System Organization

본 서비스의 시스템은 크게 front-end 와 back-end(Web-server, DB), AI model로 이루어져 있다. 사용자는 front-end와 상호작용함으로써 본 서비스의 기능들을 이용할 수 있다. front-end는 server에 정보를 입력하거나 받아올 수 있다. Back-end에는 server를 구성하는 sub-system들이 있다. 여기서 front-end는 작성된 API를 통해 back-end를 호출한다. Server는 database 및 AI model과 연결되어 있으며, request 발생시 요청된 정보는 database와 AI model에서 server 측으로 전송된다.

#### 3.2. System Architecture – Frontend Application

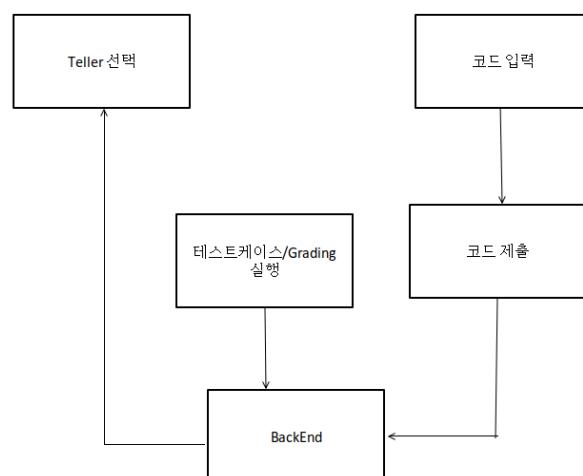


Figure 2. Overall Frontend Architecture

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

### 3.3. System Architecture – AI Application

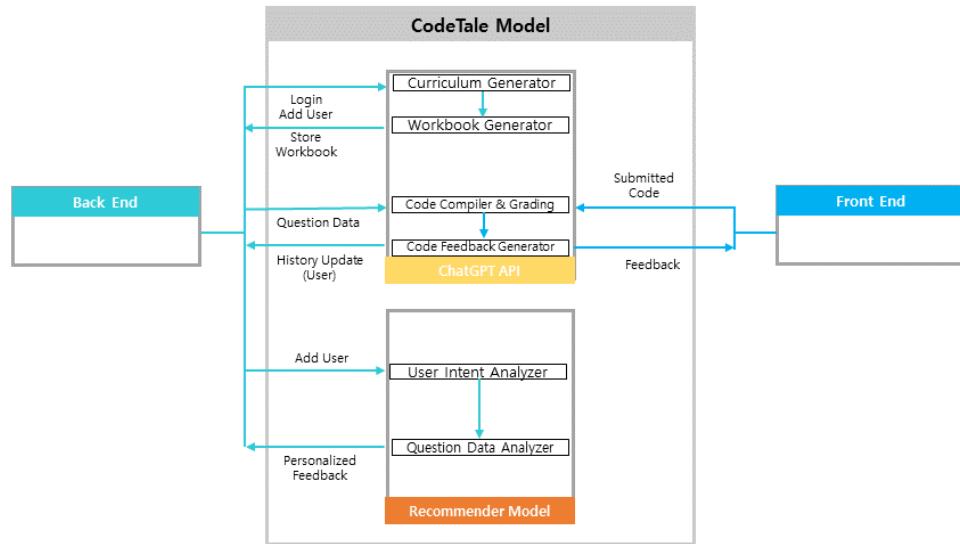
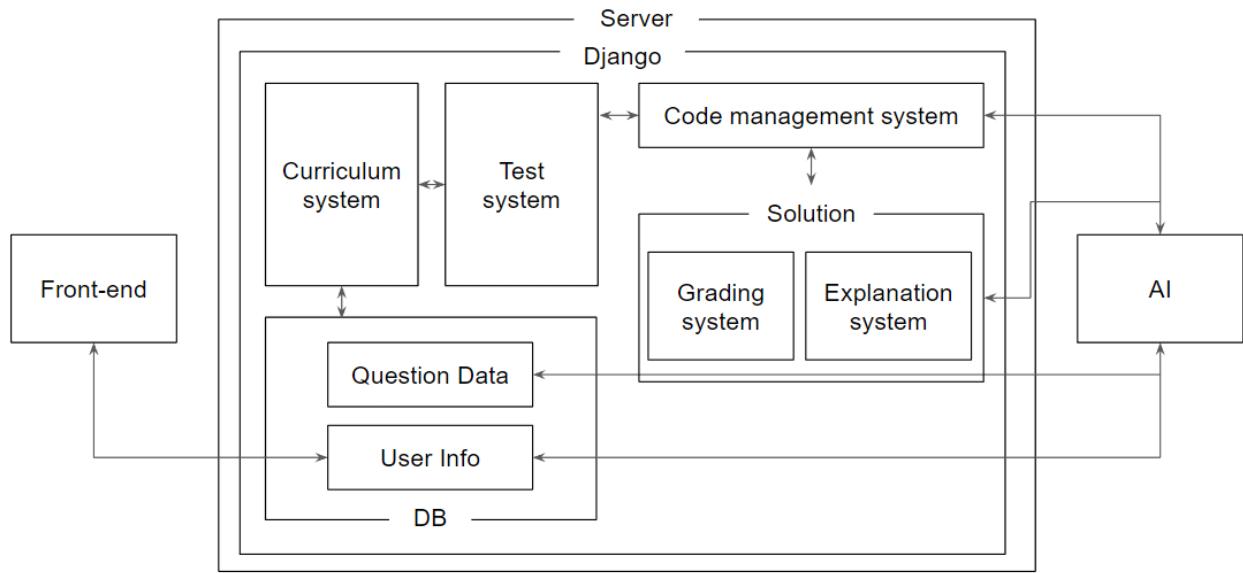


Figure 3. Overall AI Architecture

본 서비스의 AI 모델은 back-end와 front-end와 상호작용하는 방식으로 구성된다. 사용자가 사용자 등록 후 사용자 맞춤형 커리큘럼과 그에 따른 학습지를 ChatGPT API가 생성해주고, 로그인시 입력한 정보를 바탕으로 사용자의 선호도를 파악하고, 해당 정보를 바탕으로 사용자의 문제 풀이 데이터를 추천 모델을 사용하여 도출한다. 이렇게 생성된 문제풀이 추천 정보를 바탕으로 사용자에게 문제를 보여주고, 사용자가 제출한 코드는 문제에 해당되는 테스트 케이스와 함께 ChatGPT API가 컴파일 및 채점해준다. 그리고, 해당 코드의 피드백(리팩토링, 오류점 등)을 front-end에 보내준다.

### 3.4. System Architecture – Backend



**Figure 4. Overall Backend Architecture**

Backend 의 Django 는 크게 5개의 sub-system으로 구성되어 있다. Curriculum system, Test system, Code management system, Grading system, Explanation system 이에 해당한다.

- Curriculum system: DB에서 AI model이 생성한 Question Data를 받아 이를 토대로 Curriculum을 구성한다.
- Test system: Curriculum의 Question data를 토대로 학습지를 구성한다.
- Code management system: code 의 저장 및 제출을 담당한다.
- Grading System: code 의 효율성 및 가독성 채점을 담당한다.
- Explanation system: 기타 code 에 대한 분석 및 설명을 담당한다.

이렇게 구성된 서버는 AI model과 DB, front-end를 연결한다. Front-end에서 서버에 API를 호출하면 Django에서 작성된 API에 따라 해당하는 sub-system 이 호출된다.

## 4. System Architecture – Frontend

### 4.1. Objectives

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

### 4.2. Main page

CodeTale의 메인 페이지로 처음 시스템에 접속하면 볼 수 있는 랜딩 페이지이다. 이 페이지에서 유저는 전반적인 서비스에 관한 설명과 기능 그리고 웹 페이지에서 만날 수 있는 페이지들에 대한 구성을 찾을 수 있다.

#### 4.2.1. Attributes

메인 페이지는 크게 Navbar, Content, Team description, Footer로 구성되어 있다.

(1) Navbar는 아래의 속성을 가지고 있다.

- Home: 메인 페이지로 돌아오는 버튼이다.
- Pages: 웹 페이지에서 볼 수 있는 페이지들을 나열한 list이다. Page list에는 강의를 들을 수 있는 Curriculum, Code 작업을 할 수 있는 Code Test 페이지와 로그인과 회원가입을 할 수 있는 Sign In Page로 구성되어 있다.
- Docs: 서비스와 관련된 문서를 볼 수 있는 list이다.

(2) Content 파트는 Body와 Description 파트로 구성되어 있다.

- Body: Codetale의 서비스를 한 줄로 나타낸 표어와 부제, 그리고 서비스 이용을 시작할 수 있는 Get Started 버튼과 Docs를 읽을 수 있는 Read More 버튼으로 구성되어 있다.
- Description: Codetale의 서비스 기능을 4가지로 나누어 설명하는 카드 구성으로 되어 있다.

(3) Team Description 파트는 Team Description Card로 구성되어 있다.

- Team Description Card: 서비스 제작에 참여한 팀원의 이름과 기술 스택에 대한 설명이 기술되어 있다.

(4) Footer 파트는 인사말과 커뮤니티 파트로 구성되어 있다.

- 인사말: 방문한 고객에 대한 감사 인사를 나타낸다.
- 커뮤니티: 서비스와 관련된 자료를 찾아볼 수 있는 커뮤니티로 이동할 수 있는 버튼들로 구성되어 있으며 트위터, 페이스북, 깃허브, 드리블 파트로 구성되어 있다

#### 4.2.2. Methods

메인 페이지가 가지고 있는 method를 파트에 따라 나누면 다음과 같다.

(1) Navbar 가 가지고 있는 method 는 아래와 같다

moveToMainPage(): 홈 버튼 클릭 시 메인 페이지로 이동한다.

moveToCurriculumPage(): Page 메뉴에서 Lecture 버튼 클릭 시 커리큘럼을 볼 수 있는 커리큘럼 페이지로 이동한다.

moveToLoginPage(): Page 메뉴에서 Sign In 버튼 클릭 시 로그인 및 회원가입 절차를 진행할 수 있는 로그인 페이지로 이동한다.

moveToCodeEditor(): Page 메뉴에서 Code Test 버튼 클릭 시 코드 테스트를 진행할 수 있는 코드 에디터 페이지로 넘어간다,

moveToDocs(): Docs 메뉴에서 Docs 버튼 클릭 시 문서를 볼 수 있는 Docs 페이지로 이동한다.

(2) Content가 가지고 있는 method 는 아래와 같다.

moveToCodeEditor(): Body Part에 있는 Get started 버튼 클릭시 Code Test 페이지로 넘어간다,

moveToDocs(): Body Part에 있는 Read More 버튼 클릭 시 문서를 볼 수 있는 Docs 페이지로 이동한다.

(3) Team Description에는 구현된 method가 존재하지 않는다.

(4) Footer가 가진 method는 아래와 같다.

moveToTwitterUrl(): Twitter 버튼 클릭 시 서비스와 관련된 트위터 링크로 이동한다.

moveToFackbookUrl(): Facebook 버튼 클릭 시 서비스와 관련된 페이스북 링크로 이동한다

moveToGithubUrl(): Github 버튼 클릭 시 서비스와 관련된 깃허브 링크로 이동한다

moveToDribbbleUrl(): 드리블 버튼 클릭 시 서비스와 관련된 드리블 링크로 이동한다.

#### 4.2.3. State Diagram

아래는 사용자가 Main Page에서 수행 가능한 기능에 대한 state diagram이다.

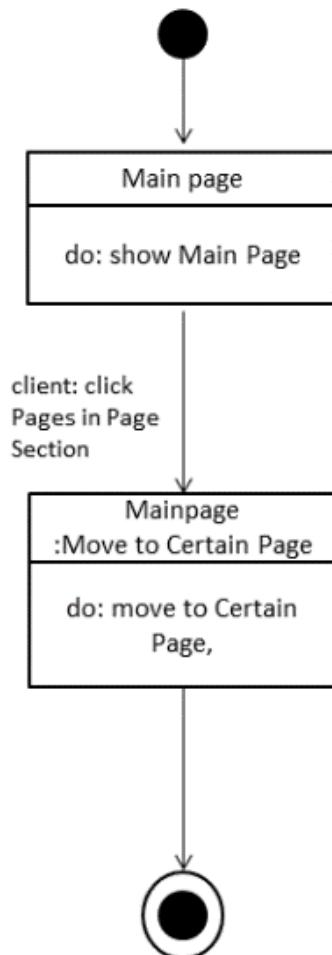


Figure 5. State Diagram of Main Page

사용자가 Main Page에서 수행할 수 있는 주요한 기능은 다른 페이지로의 이동이다. 사용자가 Navbar, Content에서 버튼을 누르면 해당하는 Page로의 이동이 수행된다. 이동할 수 있는 페이지들과 작동방식은 Method 파트에 기술되어 있다.Explain, Requirements, Testcase, Skeleton code 를 가져와 화면에 표시한다.

### 4.3. Login/Sign Up page

Codetale의 로그인/회원가입 페이지로 처음 시스템에 접속하면 볼 수 있는 페이지다. 이 페이지에서는 로그인 기능과 회원가입 기능을 제공하고, 사용자가 회원가입을 진행하면 그 데이터를 저장하여 로그인 할 수 있도록 한다. 회원가입을 완료하면 성공적으로 회원가입을 완료했다는 문구가 뜨고 로그인 창으로 넘어간다. 로그인을 진행하면 다음 페이지인 메인 페이지가 나온다.

#### 4.3.1. Attributes

로그인/회원가입 페이지는 크게 두 가지 부분으로 구성되어 있다.

- Login: 사용자의 이메일과 비밀번호를 입력 받아 로그인한다.
- Sign Up: 사용자의 이름, 이메일, 비밀번호, 비밀번호 재확인, 학생/선생 선택, 언어 선택을 하고 회원가입을 진행한다.

(1) Login은 아래의 속성을 가지고 있다.

Email: 사용자가 회원가입 시 입력했던 Email ID를 입력받는 Textbox이다.

Password: 사용자가 회원가입 시 입력했던 비밀번호를 입력받는 Textbox이다.

(2) Sign Up은 아래의 속성을 가지고 있다.

User name: 사용자의 이름을 입력받는 Textbox이다.

Email ID: 사용자의 Email ID를 입력받는 Textbox이다.

Password: 사용자의 비밀번호를 입력받는 Textbox이다.

Confirm password: 사용자가 Password에 입력했던 비밀번호와 동일한지 확인하는 Textbox이다.

Student/Teacher: 사용자의 사용 목적을 알기 위해 직업 선택을 하는 radiobox다.

Select language: 사용자가 어떤 언어로 학습을 원하는지 확인하기 위한 Dropbox이다.

#### 4.3.2. Methods

로그인/회원가입 페이지가 가지고 있는 method를 파트에 따라 나누면 다음과 같다.

(1) Login이 가지고 있는 method는 아래와 같다.

mainViewLoad(): 사용자가 처음 접속했을 때, 로그인 페이지를 볼 수 있게 띄워준다.

addLoginButtonClick(): 사용자가 회원가입 페이지를 보고 있을 때, 로그인 페이지로 넘어가도록 로그인 페이지를 띄워주고, 회원가입 페이지를 숨긴다.

addSignUpButtonClick(): 사용자가 로그인 페이지를 보고 있을 때, 회원가입 페이지로 넘어가도록 회원가입 페이지를 띄워주고, 로그인 페이지를 숨긴다.

addBottomLoginClick(): 사용자가 입력한 Email ID와 Password가 회원가입 되어 있는 데이터와 일치하는지 확인한 후, 맞다면 메인 페이지로 넘어가게 해준다. 만약 틀리다면 다시 로그인 하라는 경고 문구를 띄운다.

(2) Sign Up이 가지고 있는 method는 아래와 같다.

checkFirstName(): 사용자가 User name을 입력했는지 확인한다.

checkEmail(): 사용자가 Email을 입력했는지 확인한다.

checkPassword(): 사용자가 Password를 입력했는지 확인한다.

checkConfirmPassword(): 사용자가 Confirm Password를 입력했는지 확인한다.

checkJob(): 사용자가 student/teacher을 선택했는지 확인한다.

checkLanguage(): 사용자가 사용할 언어를 입력했는지 확인한다.

firstkeyup(): 사용자가 사용자의 이름을 조건에 맞게 입력했는지 확인한다. 조건이 맞지 않다면 오류 메세지를 띄운다.

emailkeyup(): 사용자가 사용자의 Email ID를 조건에 맞게 입력했는지 확인한다. 조건이 맞지 않다면 오류 메세지를 띄운다.

passwordkeyup(): 사용자가 비밀번호를 조건에 맞게 입력했는지 확인한다. 조건이 맞지 않다면 오류 메세지를 띄운다.

confirmPasswordkeyup(): 사용자가 비밀번호 확인을 조건에 맞게 입력했는지 확인한다. 조건이 맞지 않다면 오류 메세지를 띄운다.

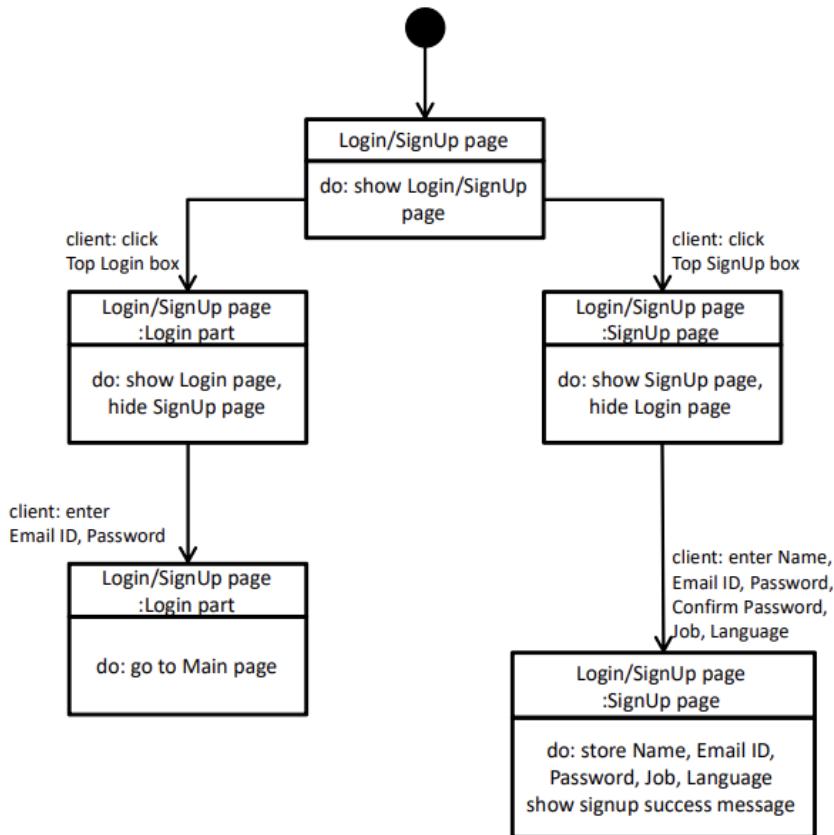
jobkeyup(): 사용자가 선택한 Job을 데이터베이스에 저장한다.

languagekeyup(): 사용자가 선택한 언어를 데이터베이스에 저장한다.

addBottomSignUpClick(): 사용자의 Email ID, Password, Job 그리고 선택한 언어를 데이터베이스에 저장하고 회원가입 성공 메세지를 띄운다.

#### 4.3.3. State Diagram

아래는 사용자가 Login 및 Sign Up을 한 뒤 나타나는 State diagram이다.



**Figure 6. State Diagram of Login/Sign Up**

#### 4.4. Curriculum page

사용자가 메인 페이지에서 커리큘럼 버튼을 눌렀을 때, 나오는 페이지다. 사용자에게 알맞는 커리큘럼이 보여지는 페이지로 각 Week마다 총 3개의 chapter로 구성되어 있다. 각 Week에 배우는 내용에 대한 짧은 설명과 그 아래 chapter 버튼이 존재하고, 사용자는 이 버튼을 눌러 각 chapter에 해당하는 내용의 학습지를 볼 수 있게 된다. Week를 끝낼 때마다 체크할 수 있는 check box를 클릭하면 오른쪽 상단에 진도율을 확인할 수 있는 progress bar가 표시된다.

##### 4.4.1. Attributes

커리큘럼 페이지는 크게 두 가지 부분으로 구성되어 있다.

- Head: Curriculum 제목과 사용자의 ID, Progress bar 그리고 뒤로가기 버튼으로 구성되어 사용자가 볼 수 있도록 표시된다.
- Body: 각 Week에 대한 간단한 설명과 Chapter 버튼이 존재하며, 버튼을 누를 경우 학습지를 열어볼 수 있다. 가장 하단에 있는 checkbox를 클릭하면 현재까지 진행한 진행률이 오른쪽 상단 progress bar에 표시된다.

(1) Head는 아래의 속성을 가지고 있다.

Email ID: 사용자가 회원가입 시 입력했던 Email ID를 표시해준다.

Progress Bar: 사용자의 현재 학습 진행률을 progress bar 형태로 표시해준다.

Backspace: 사용자가 메인 페이지로 돌아갈 수 있도록 한다.

(2) Body는 아래의 속성을 가지고 있다.

Item1: Week1에 대한 정보와 chapter 버튼 3개를 가지고 있으며, 가장 하단에는 checkbox가 있다.

Item2: Week2에 대한 정보와 chapter 버튼 3개를 가지고 있으며, 가장 하단에는 checkbox가 있다.

Item3: Week3에 대한 정보와 chapter 버튼 3개를 가지고 있으며, 가장 하단에는 checkbox가 있다.

Item4: Week4에 대한 정보와 chapter 버튼 3개를 가지고 있으며, 가장 하단에는 checkbox가 있다.

Item5: Week5에 대한 정보와 chapter 버튼 3개를 가지고 있으며, 가장 하단에는 checkbox가 있다.

#### 4.4.2. Methods

커리큘럼 페이지가 가지고 있는 method를 파트에 따라 나누면 다음과 같다.

(1) Head가 가지고 있는 method는 다음과 같다.

bringEmail(): 사용자가 로그인 했을 때의 Email ID를 가져와 표시한다.

showProgress(): progress bar에 현재 사용자가 진행한 학습 진행률을 표시해준다.

backspace(): 이미지 버튼을 누르면 사용자가 메인 페이지로 돌아갈 수 있도록 한다.

(2) Body가 가지고 있는 method는 다음과 같다.

showWeek(): 각 Week에 해당하는 간단한 설명을 보여준다.

clickChapter(): Chapter 버튼을 클릭하면 해당 챕터의 학습지를 보여주는 페이지로 넘어간다.

sendProgress(): 각 Week 하단에 있는 checkbox를 클릭하면 progress bar에 학습 진행률에 관한 정보를 넘겨준다.

#### 4.4.3. State Diagram

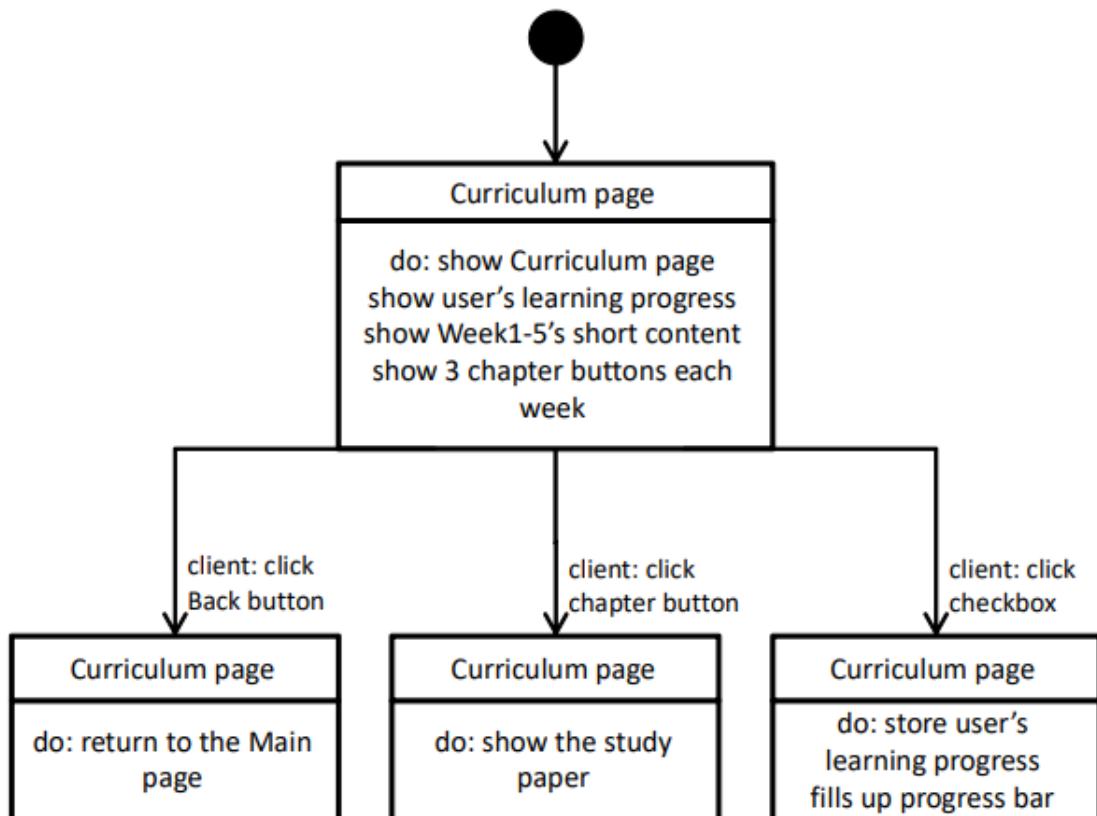


Figure 7. State Diagram of Curriculum page

## 4.5. WorkBook

사용자가 Curriculum 페이지에서 각 커리큘럼에 해당하는 Title 섹션을 눌렀을 때 이동할 수 있는 페이지로 해당 주제에 대한 설명과 학습 진행도 상황과 다음 Title로 넘어갈 수 있는 기능, 해당 강의 완료를 수행할 수 있는 페이지이다.

### 4.5.1. Attributes

WorkBook의 구성은 크게 Navbar, Progress bar, Content, interaction section으로 나누어져 있다.

#### (1) Navbar

Main Page의 attribute의 Navbar와 동일하다.

## (2) Progress bar

Progress bar의 구성은 학습 진행도 현황 Bar으로 구성되어 있다.

- 학습 진행도 현황 Bar: 사용자가 수강하고 있는 커리큘럼의 총 진행상황을 볼 수 있는 progress Bar이다

## (3) Content

Content는 Title에 대한 강의 내용을 볼 수 있는 Body로 구성되어 있다.

- Body: Title에 관한 개념 설명 및 강의 내용을 볼 수 있는 섹션이다.

## (4) Interaction Section

Interaction Section은 Previous버튼과 Next 버튼, Complete 버튼으로 이루어져 있다.

- Previous 버튼: 이전 강의로 돌아갈 수 있는 버튼이다.

### 4.5.2. Methods

Workbook 페이지가 가지고 있는 Method 를 파트 별로 나누면 다음과 같다.

1. Navbar 가 가지고 있는 Method 는 다음과 같다.

Main Page 의 Navbar 와 동일

2. Progress Bar 가 가지고 있는 Method 는 다음과 같다.

- ShowCurrentProgress(): 총 커리큘럼에서 사용자가 수강 완료한 title 의 퍼센테이지를 보여준다.

3. Content 가 가지고 있는 Method 는 다음과 같다.

- GetWorkbookContent(): 해당 타이틀에 따른 강의 내용을 Database 에서 가져온다.

- ParseContentJson(): Database 에서 가져온 json 데이터를 웹 페이지의 형식에 맞게 parsing 한다.

- ShowWorkbookContent(): 파싱된 데이터를 Body 파트에 보여준다.

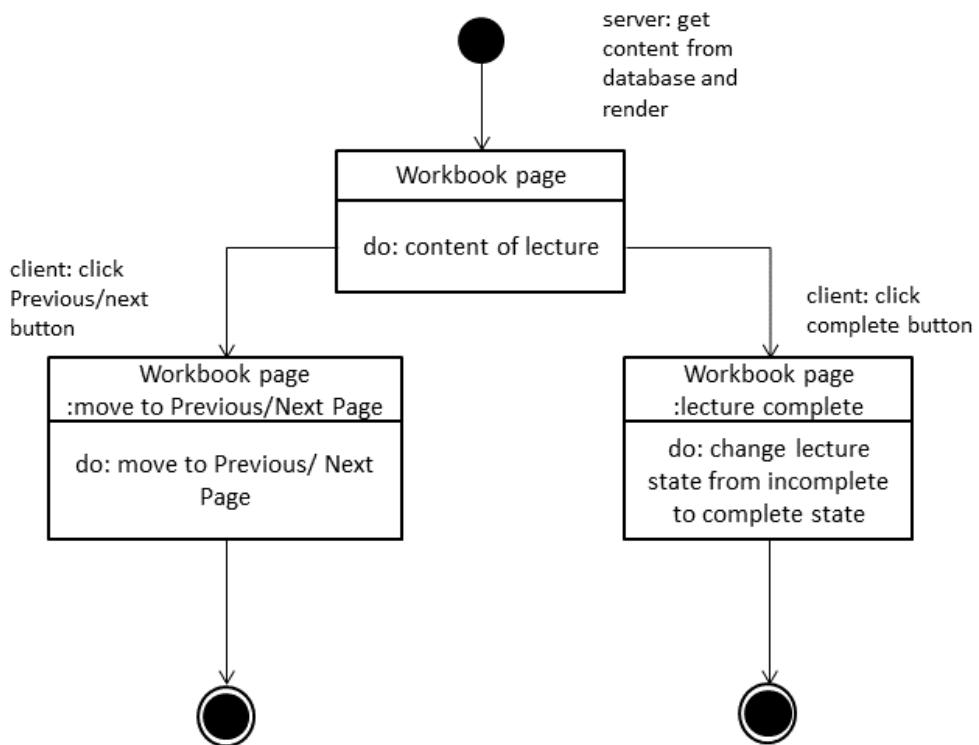
4. Interaction Section 이 가지고 있는 Method 는 다음과 같다.

- moveToPreviousTitle(): 이전 title 로 이동한다.

- moveToNextTitle(): 다음 title 로 이동한다.

### 4.5.3. State Diagram

아래는 사용자가 WorkBook Page에서 Interaction버튼들을 눌렀을 때의 WorkBook Page상태를 그린 state diagram이다



**Figure 8. State Diagram of Workbook Page**

사용자가 Previous/Next 버튼을 누를 경우, 이전/다음 페이지로 넘어간다. 혹은 사용자가 Complete 버튼을 누를 경우, 강의가 imcomplete 상태에서 complete 상태로 변하며 페이지 새로고침 시 progress bar에 적용된다.

## 4.6. Editor page

Codetale에서 커리큘럼을 클릭할 시, 또는 test 시작 시 나오는 페이지이다. 해당 페이지는 유저가 코드를 작성할 수 있는 code editor이 구현되어 있으며, 원하는 프로그래밍 언어를 선택할 수 있는 dropdown list가 있다. 유저는 submit 버튼을 통해 코드 제출이 가능하며, 결과값은 오른쪽 output 아래에 출력된다. 해당 페이지의 method는 다음과 같다.

### 4.6.1. Attributes

메인 페이지는 크게 아래의 네 가지 부분으로 구성되어 있다.

- LanguageSelect: 유저가 원하는 프로그래밍 언어를 선택한다
- QuestionInfo: 과제 정보를 표시한다.
- CodeEditor: 사용자가 코드를 작성할 수 있는 코드 에디터이다.
- Result: 사용자의 코드 실행, 채점, 제출 결과를 표시하고 관리한다.

(1) LanguageSelect 는 다음과 같은 속성을 가지고 있다.

- Dropdown: 유저가 선택할 수 있는 프로그래밍 언어들의 리스트가 나오고 유저는 해당 언어를 선택할 수 있다.

(2) QuestionInfo 는 아래의 속성을 가지고 있다.

- ProblemExplain: 과제 내용에 대해 설명한다.
- Requirements: 과제의 참조/제약사항에 대해 설명한다.
- Testcase: 과제에 속한 testcase 들을 표시한다.
- Test: testcase 를 검증하는 버튼이다.

(3) CodeEditor 는 아래의 속성을 가지고 있다.

- Code: 사용자가 작성하는 코드이다.

(4) Result 는 아래의 속성을 가지고 있다.

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

#### 4.6.2. Methods

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

1. LanguageSelect 가 가지고 있는 method 는 아래와 같다.
  - OnSelectChange(): 해당 dropdown 은 javascript 로 초기화되어 있다. 이후 유저는 해당 dropdown 을 클릭하여 원하는 언어를 선택할 수 있다,
2. QuestionInfo 가 가지고 있는 method 는 아래와 같다.
  - setProblemExplain(): 사용자가 선택한 QuestionId 에 속하는 ProblemExplain 내용을 가져와 표시한다.
  - setRequirements(): 사용자가 선택한 QuestionId 에 속하는 Requirements 내용을 가져와 표시한다.
  - setTestcase(): 사용자가 선택한 QuestionId 에 속하는 공개 Testcase 내용을 가져와 표시한다.
  - runTestcase(): 사용자가 주어진 공개 testcase 의 검증 버튼을 누르면 사용자가 작성한 코드에 해당 testcase input 값을 넣어 결과를 알려준다.
3. CodeEditor 가 가지고 있는 method 는 아래와 같다.
  - checkStatus(): 사용자가 코드 에디터의 Compile and Execute 버튼을 누르면 작성중인 코드를 실행하여 실행 결과창에 그 결과가 표시되도록 한다.
  - handleCompile(): 사용자가 코드 에디터의 Compile and Execute 버튼을 누르면 해당 요청이 하루 제한인 100 회를 넘어가는지를 판단하여 넘을 경우 에러 메시지를 출력한다.
  - showSuccessToast(): 사용자가 코드 에디터의 Compile and Execute 를 누르고 compile 이 성공적으로 될 경우 "Complie Success!" 가 toast 된다.

- showErrorToast(): 사용자가 코드 에디터의 Compile and Execute 를 누르고 compile 에 문제가 발생할 경우 "Something went wrong!! Please try again." 가 toast 된다.
  - refreshCode(): 사용자가 코드 에디터의 초기화 버튼을 누르면 작성중인 코드가 사라지고 초기 상태의 코드 에디터로 돌아간다.
  - downloadCode(): 사용자가 코드 에디터의 다운로드 버튼을 누르면 작성중인 코드를 사용자의 컴퓨터에 다운로드할 수 있도록 한다.
4. Result 가 가지고 있는 method 는 아래와 같다.
- showResult(): 사용자가 작성한 코드를 실행, 채점, 제출한 결과가 화면에 표시되도록 한다.

#### 4.6.3. State Diagram

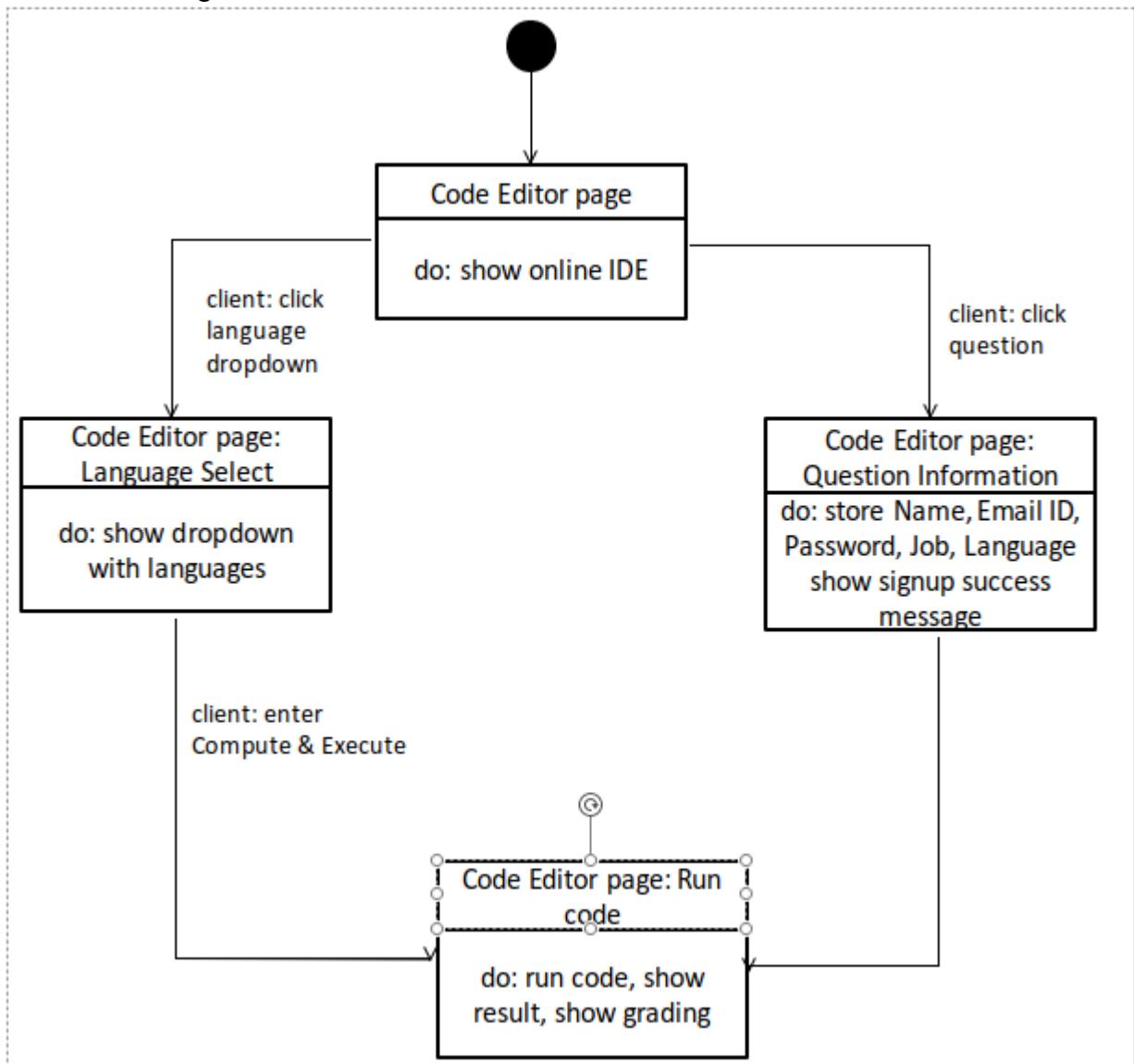


Figure 9. State Diagram of Editor page

## 5. System Architecture – AI

### 5.1. Objectives

AI모델과 ChatGPT-API를 활용한 개인화된 추천 모델의 전반적인 구조와 세부 시스템 구조를 설명한다. 각각의 sub-system은 sequence diagram을 통해 표현되며, 해당 diagram을 활용하여 데이터의 흐름을 파악한다. 각 database에 대한 자세한 설명은 8 장에서 확인할 수 있다.

### 5.2. Overall Architecture

AI model의 전체적인 구조는 다음과 같다.

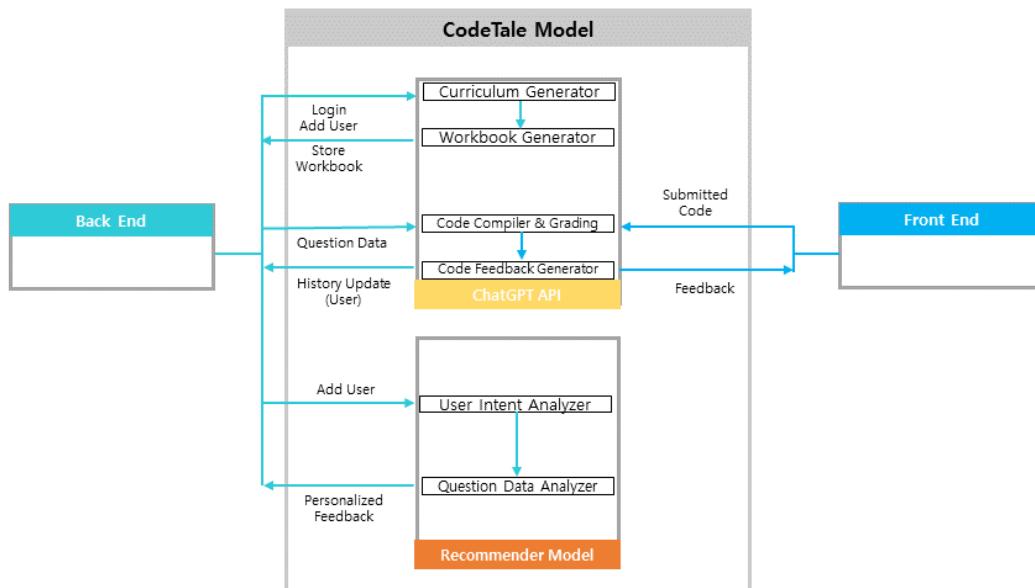


Figure 10. Overall AI Architecture

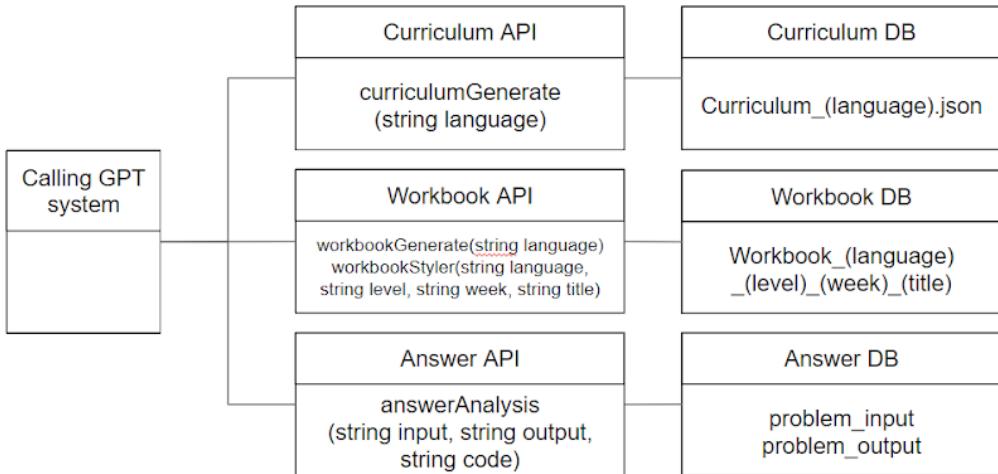
먼저, ChatGPT API를 활용한 모델이 있다. Back-end에서 Login이나 사용자가 회원가입 후 추가될 시에, 사용자의 커리큘럼 정보가 없다면 Curriculum generator를 활용하여 개인화된 Curriculum을 생성하고, 커리큘럼에 따라 Workbook을 만들어준다. 해당 Workbook은 storage에 저장된다. 그리고, 사용자가 문제풀이를 진행 시 Storage에서 question data를 보내주고, 해당 question을 사용자가 풀면 front end에서 submitted code를 가져온다. 그 후, code compiler & grading을 활용하여 코드와 함께 테스트 케이스를 채점하고, 해당 코드에 대한 피드백을 Code feedback generator를 활용하여 생성한 뒤, front end를 활용하여 사용자에게 보여준다.

다음으로, 개인화된 추천시스템을 활용한 모델이 있다. 사용자가 회원가입 후 기입한 정보를 바탕으로 User Intent Analyzer가 사용자의 선호도를 파악한 뒤, Question Data Analyzer가 사용자에 맞는 문제풀이 데이터를 골라서 해당 feedback을 storage에 저장한다.

### 5.3. Overall Architecture

#### 5.3.1. Workbook Generate/Answer Analysis System

GPT에 대한 요청 및 response를 기반으로 한 curriculum, workbook을 생성하는 시스템이다. 또한, GPT에 요청하는 구조는 problem solving 단계에서 코드 제출 이후, 답변 채점 및 분석하는 데에도 동일한 구조로 쓰인다. 따라서, class diagram은 다음과 같다.



**Figure 11. Workbook Generate/Answer Analysis System Class Diagram**

#### - Curriculum API Class

Front-end로부터 요청을 받으면 지정된 language에 대해 초급, 중급, 고급의 level, 5 week 및 3 chapter에 해당하는 curriculum을 json 형태로 제너레이트한 후, curriculum DB에 저장하는 역할을 한다. 해당 클래스에는 curriculumGenerate가 존재한다. 이것은 language 지정 시 지정된 조건에 따라 커리큘럼을 생성하는데, 다음과 같은 형태로 저장된다: Curriculum\_(language).json

#### - Workbook API Class

Front-end로부터 요청을 받으면 curriculum의 정보들을 가져와 GPT response(workbook)를 workbook DB에 저장하는 역할을 한다. workbookGenerate() 함수 호출 시, 지정된 language에 대한 workbook을 GPT에 request 한다. 이때 받은 response를 다음과 같은 형태로 저장한다: Workbook\_(language)\_(\_level)\_(\_week)\_(\_title).

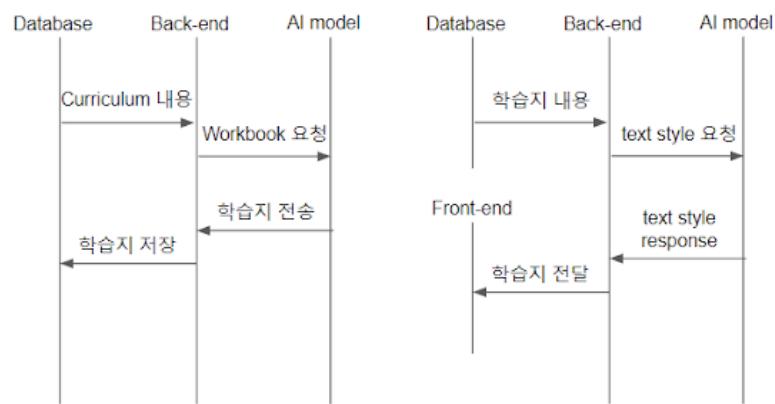
Front-end로부터 요청을 받으면 workbook을 텔러(ex. 인어공주)의 text style대로 변환해주는 text style transfer의 역할을 한다. workbookStyler() 함수 호출 시, 지정된 학습지에 대한 workbook 및 말투를 GPT에 request 한다. 이때 받은 response를 다시 Front-end로 전달한다.

#### - Answer API Class

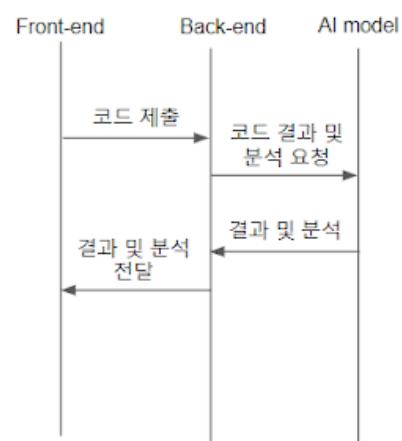
Front-end로부터 요청을 받으면 사용자가 작성한 code 및 문제의 input, output 테스트케이스를 GPT로 보낸 후, 코드 채점 및 분석에 대한 response를 가져오는 역할을 한다. 따라서, answerAnalysis() 함수는 input, output를 기반으로 code에 대한 분석 string을 반환하는 함수이다.



**Figure 12. Workbook Generate System Sequence Diagram - Curriculum Generate**



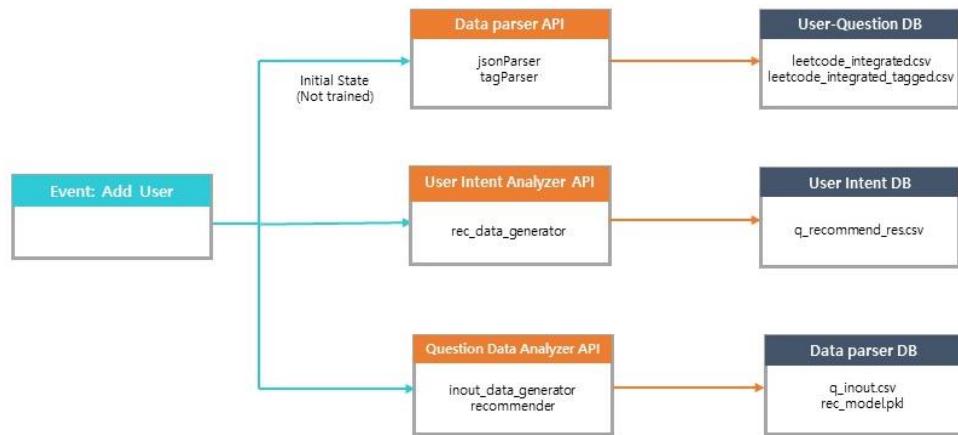
**Figure 13. Workbook Generate System Sequence Diagram - Workbook Generate, Workbook Style Transfer**



**Figure 14. Answer Analysis System Sequence Diagram**

### 5.3.2. Workbook Generate/Answer Analysis System

사용자 맞춤형 문제풀이 시스템 구축을 위한 추천 모델이다. 사용자가 신규 등록되었을 때 사용자에 대한 정보를 바탕으로 기존 사용자의 문제풀이 history를 기반으로 한 추천 모델이 사용자에 맞는 문제를 추천해준다. Class diagram은 다음과 같다.



**Figure 15. Data parser / Recommender System Class Diagram**

- Data parser API Class

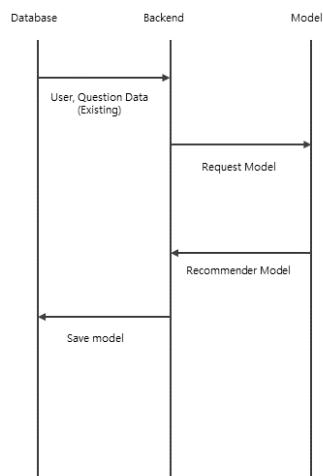
새로운 User 가 들어왔을때 user 와 question 에 대한 preprocessed 된 데이터가 없으면(초기 상태) 기존에 존재하던 user 와 question dataset 를 활용하여 데이터 전처리를 진행한다. jsonParser 는 문제 데이터셋인 leet10k-alpaca.json 을 입력으로 받아 json 형식의 데이터를 개발에 용이한 csv 형식의 파일로 변환해준다. 또한, tagParser 가 해당 문제 데이터셋에서 회사에 해당 문제가 코딩 테스트에 나왔는지에 대하여 list 형식으로 저장된 attribute 를 parsing 해준다.

- User intent analyzer API Class

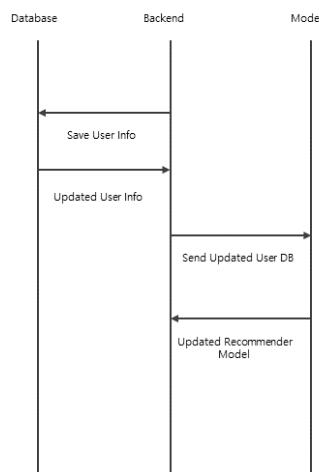
Front-end로부터 새로운 사용자가 등록되었다는 event 의 log 를 받으면 User Intent Analyzer 가 preprocessed 된 데이터셋을 기반으로 학습된 추천 모델을 활용하여 사용자의 history 가 없더라도 문제풀이 history 를 만들어 주는 역할을 rec\_data\_generator 가 수행해준다.

- Question data analyzer Class

Front-end로부터 새로운 사용자가 최초로 문제풀이 시스템을 사용한다는 요청을 받을시 system 이 initial state(데이터셋 준비가 다 되지 않을 시) 확인하고, 데이터셋이 확인되지 않을 시 inout\_data\_generator 를 활용하여 해당 문제별로 테스트 케이스를 parsing 한 뒤 q\_inout.csv 를 생성해준다. 또한, 위에서 preprocess 된 데이터셋을 기반으로 feature engineering 을 통해 사용자의 문제별로 선호도를 파악하고, recommender 가 ALS(Alternating Least Square)를 활용하여 사용자가 선호할 문제를 추천해준다.



**Figure 16. Data parser / Recommender System Class Diagram**



**Figure 17. Data Parser / Recommender System Sequence Diagram - New User Event**

## 6. System Architecture – Backend

### 6.1. Objectives

Back-end의 전반적인 구조와 세부 시스템 구조를 설명한다. 각각의 sub-system은 class diagram을 통해 객체 간의 관계를 확인하고 sequence diagram을 통해 데이터의 흐름을 파악한다. Back-end의 전반적인 구조와 Component 구조에 대해 서술한 파트이다. 각 Component는 클래스 다이어그램, 시퀀스 다이어그램을 사용하여 알아보기 쉽게 표현하도록 하였다. 또한, component의 기능과 역할을 기술하고, 사용된 기술과 프레임워크에 대한 설명을 추가한다. 이렇게 하면 back-end의 설계와 구현 과정을 이해하기 쉽고, 유지보수와 확장성을 높이는 데 도움이 될 것이다. 각 데이터베이스에 대한 추가 설명은 8장에서 서술한다.

### 6.2. Overall Architecture

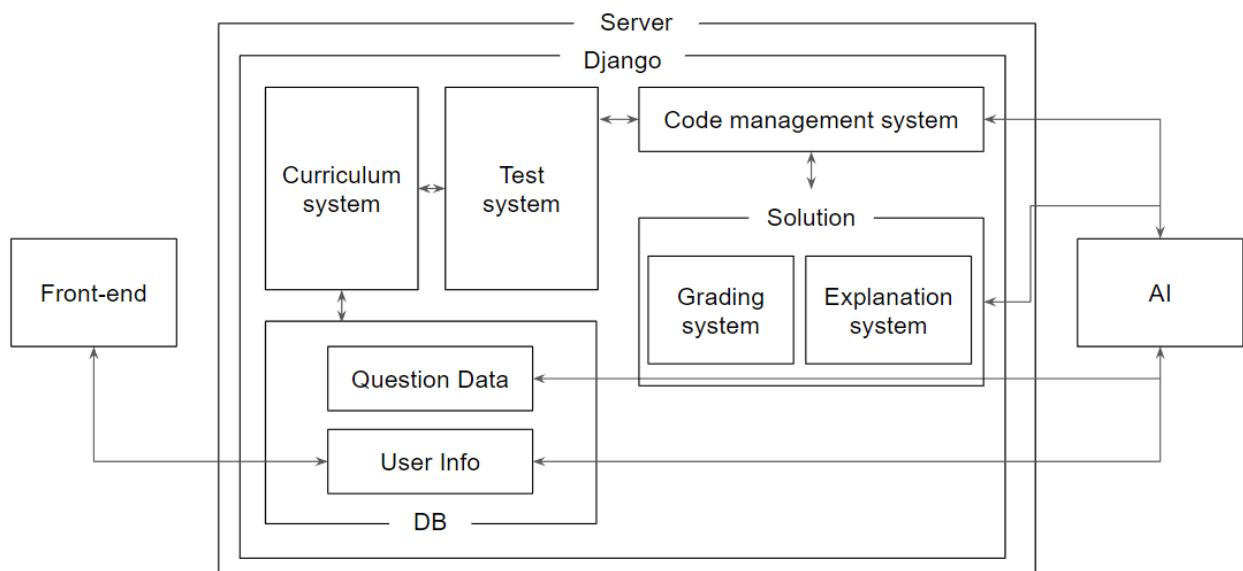


Figure 18. Overall Backend Architecture

전체적 구조는 다음과 같이 설명할 수 있다. Front-end에서는 사용자가 로그인을 하여 자신의 정보를 요청하면, REST API를 통해 Django 서버로 전달된다. Django 서버에서는 User Info를 DB에서 검색하여, 사용자에게 POST 방식으로 응답해준다. 이후에는 DB에 저장된 Question Data와 Curriculum system을 활용하여 학습 과정을 진행하게 된다. 학습 과정에서는 Test System에서 Code management 시스템을 통해 테스트 문제와 코드를 관리하고, Grading system과 Explanation system에서는 AI 파트를 이용하여 채점과 해설을 제공한다. AI 파트에서는 사용자의 코드를 분석하고 평가하며, 적절한 피드백과 설명을 생성한다. 마지막으로, Back-end에서 처리된 결과를 Front-end에 다시 전송하여, 사용자에게 보여준다.

## 6.3. Subcomponents

### 6.3.1. Coding test system

코딩 문제를 화면에 보여주는 작업을 하는 시스템



Figure 19. Code Test System Class Diagram

Front-end에서 유저의 아이디를 받아오면, 코드 정보가 있는 해당 유저의 정보와 맞는 문제의 아이디와 문제 정보를 Front-end에 전달합니다. 이렇게 하면 Front-end에서 유저가 문제를 볼 수 있고, 문제를 풀 수 있는 환경을 제공할 수 있다. 이 기능은 유저의 학습 효과를 높이고, 문제의 난이도와 유형을 다양화할 수 있게 도와주는 역할을 한다.



Figure 20. Code Test System Sequence Diagram

### 6.3.2. Code Management System

학습지를 받아 유저에게 보여주는 시스템

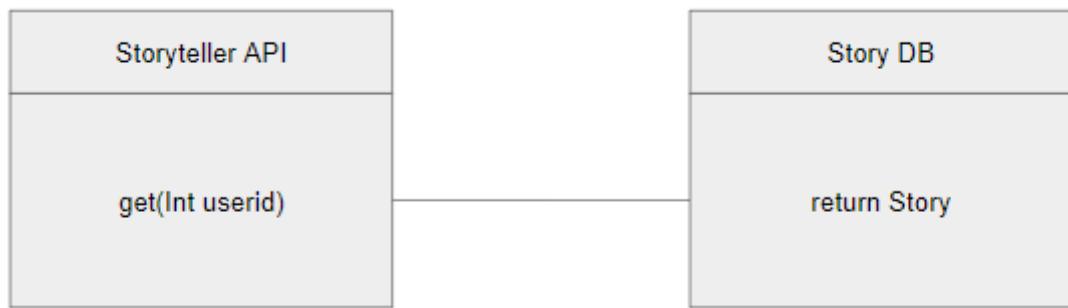


Figure 21. Storytelling System Class Diagram

- Storyteller API Class

스토리텔러 시스템이라는 프로그램은, 유저가 로그인을 하면, 유저의 아이디를 인식하고, 유저의 개인 정보와 학습 수준에 맞춰서 AI 시스템이 자동으로 생성한 맞춤형 학습지를 DB에서 가져와서, 시스템의 내부 로직을 통해 Front-end에 전송하고, 유저에게 보여주는 기능을 가지고 있습니다.

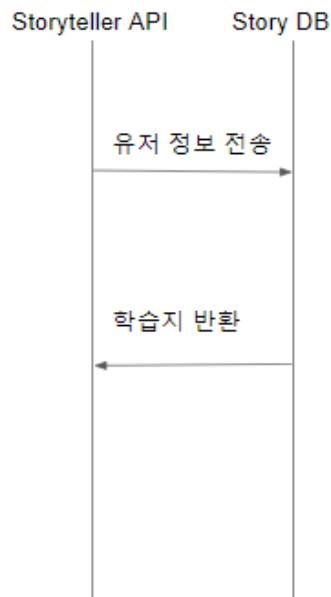


Figure 22. Code Management System Sequence Diagram

### 6.3.3. Code Management System

DB를 통하여 로그인과 회원가입을 관장하는 시스템이다.

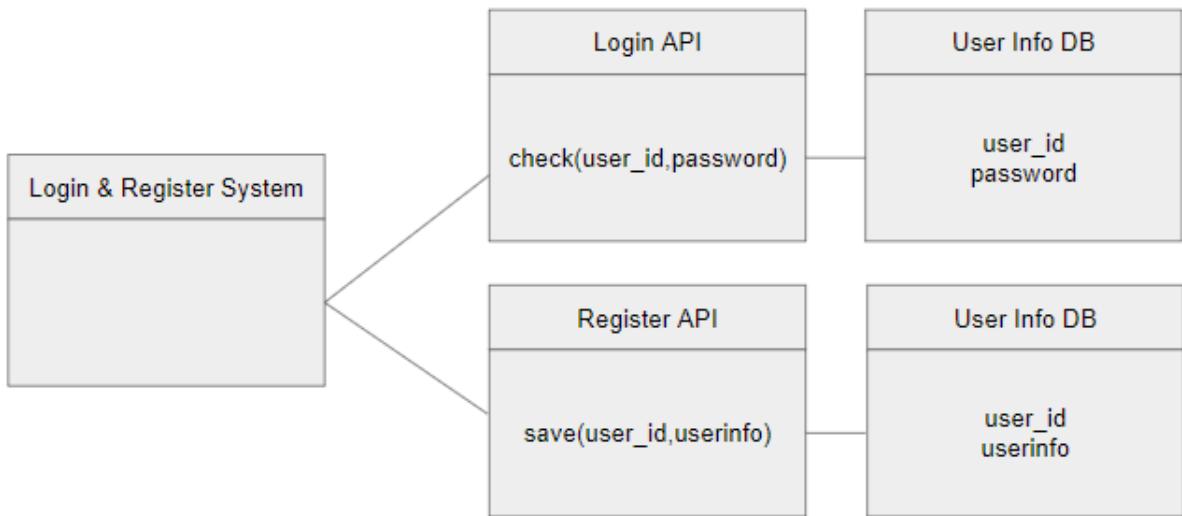


Figure 23. Login Register System Class Diagram

- Login API

프론트엔드에서 유저가 로그인을 요청하면, 백엔드에서는 데이터베이스에 접근하여 유저의 아이디와 비밀번호가 User Info DB에 저장된 값과 일치하는지 검사하는 API를 실행한다. 이 API는 로그인의 성공 여부를 판단하고, 성공한 경우에는 프론트 단에게 로그인이 완료되었다는 정보를 보내고, 실패한 경우에는 어떤 이유로 로그인이 실패하였는지를 프론트엔드에 전달하는 기능을 수행한다.

- Register API

프론트엔드에서 유저가 회원가입을 요청하면, 백엔드에서는 데이터베이스에 접근하여 유저의 아이디가 User Info DB에 이미 존재하는지 확인하는 API를 실행한다. 이 API는 회원가입의 가능 여부를 판단하고, 가능한 경우에는 유저의 아이디와 비밀번호를 User Info DB에 저장하고, 프론트 단에게 회원가입이 성공적으로 이루어졌다는 정보를 보낸다.

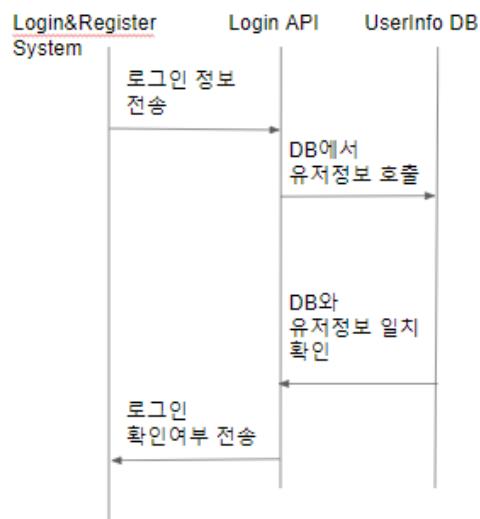


Figure 24. Login Register System Sequence Diagram

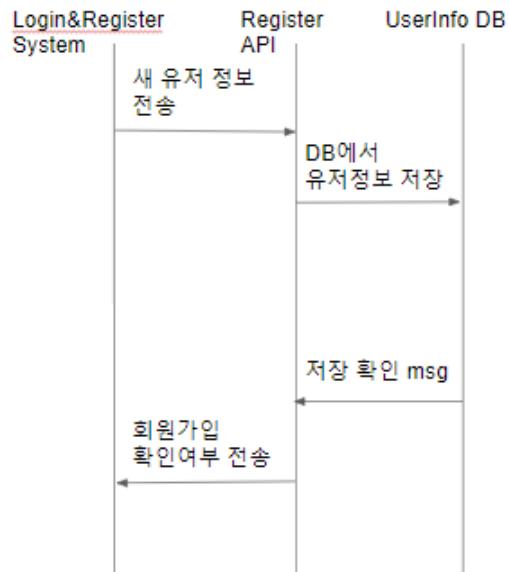


Figure 25. Login Register System Sequence Diagram

#### 6.3.4. Question Log System

DB를 풀어낸 문제에 대한 로그를 관리하는 시스템이다.

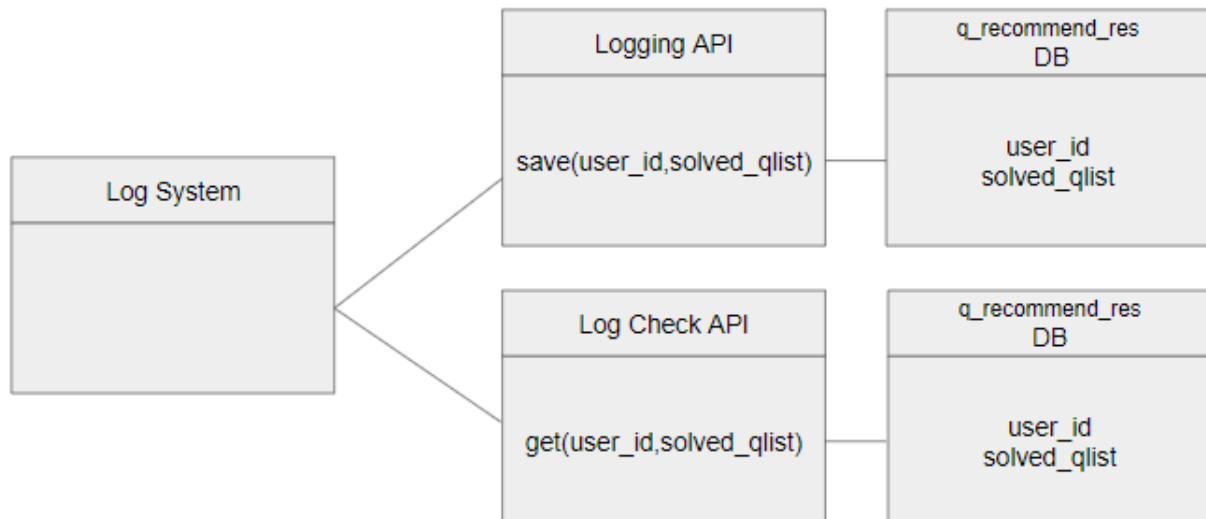


Figure 26. Log System Class Diagram

##### - Logging API

유저의 id 정보를 이용하여, 유저가 해결한 문제에 대한 정보를 list 형태로 DB에 저장하는 API이다.

##### - Log Check API

해당 클래스에는 한 개의 메소드가 존재한다. Test API로부터 codeID를 받고 이를 통해 해당

question에 속하는 testcase를 가져와 테스트케이스 검증을 한 후 결과를 TestAPI로 보내주는 unittest() 이다.

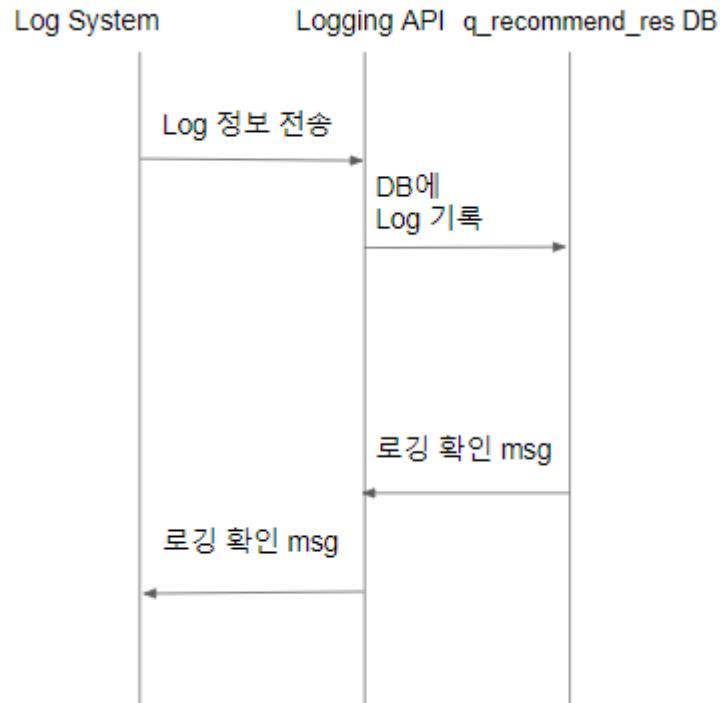


Figure 27. Log System Sequence Diagram

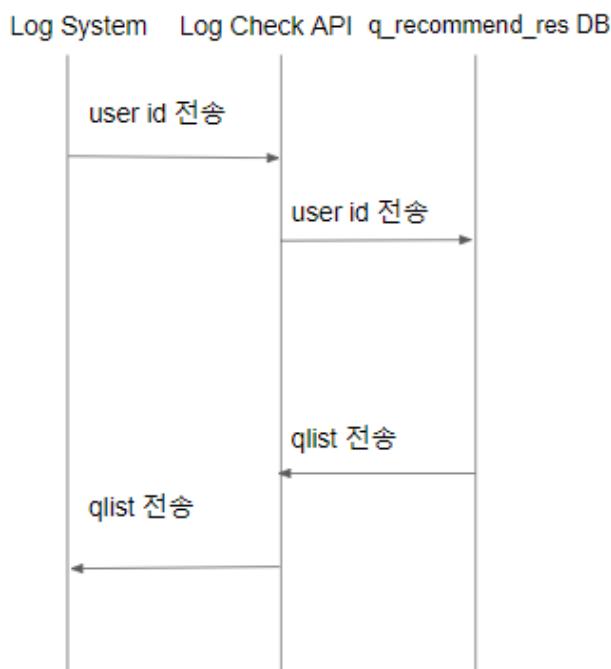


Figure 28. Log System Sequence Diagram

## 7. Protocol

### 7.1. HTTP

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

### 7.2. JSON

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

### 7.3. REST

REST는 네트워크 애플리케이션을 설계하기 위한 지침을 제공하는 아키텍처 스타일이다. REST는 기존의 HTTP 프로토콜을 사용하여 확장 가능하고 상호 운용성 있는 통신 모델을 만드는데 RESTful API(응용 프로그래밍 인터페이스)는 이러한 원칙을 준수하며, 웹 서비스 구축에 널리 사용된다.

### 7.4. Protocol Description

#### 7.4.1. 학습지 조회 프로토콜

GET	workbook/<id>	
사용자가 workbook 을 조회할 때 요청하는 주소		
분류	항목명	설명 및 제약사항
Parameter	languageId	workbook language 값
	levelId	workbook level 값
	weekId	workbook week 값
	chapterId	workbook chapter 값
Response	workbook	workbook 내용값
Status Code	200	Response 불러오기 성공
	400	존재하지 않음

Table 1. 학습지 조회 프로토콜 표

#### 7.4.2. 코드 채점 프로토콜

GET	code_submitted/<id>	
사용자가 코드 제출 이후 채점 완료 페이지를 요청할 때 요청하는 주소		
분류	항목명	설명 및 제약사항
Parameter	codeId	제출한 code 값

	inputId	테스트케이스 input 값
	outputId	테스트케이스 output 값
Response	codeCompiled	컴파일 성공값
	codeAnalysis	코드 분석 내용값
Status Code	200	Response 불러오기 성공
	400	존재하지 않음

Table 2. 코드 채점프로토콜 표

#### 7.4.3. 커리큘럼 제너레이션 프로토콜

GET	generation/<id>	
사용자가 문제를 조회할 때 요청하는 주소		
분류	항목명	설명 및 제약사항
Parameter	languageId	workbook language 값
Response	curriculumGenerationSuccess	커리큘럼 제너레이션 성공값
Status Code	200	제너레이션 성공
	400	존재하지 않음

Table 3. 커리큘럼 제너레이션 프로토콜 표

#### 7.4.4. 학습지 제너레이션 프로토콜

GET	generation/<id>	
사용자가 테스트케이스를 실행할 때 요청하는 주소		
분류	항목명	설명 및 제약사항
Parameter	languageId	workbook language 값
	levelId	workbook level 값
	chapterId	workbook chapter 값
Response	workbookGenerationSuccess	학습지 제너레이션 성공값
Status Code	200	제너레이션 성공
	400	존재하지 않음

Table 4. 학습지 제너레이션 프로토콜 표

#### 7.4.5. 페이지 조회 프로토콜

GET	page/<id>	
사용자가 이동하고자 하는 페이지를 조회할 때 요청하는 주소		
분류	항목명	설명 및 제약사항
Parameter	id	페이지 id 값
Response	pageId	페이지 id 값
	pageName	페이지 제목
Status Code	200 OK	페이지 불러오기 성공
	404 Not Found	페이지 존재하지 않음

	500 Internal Server Error	내부 서버 오류
--	---------------------------	----------

Table 5. 페이지 조회 프로토콜 표

#### 7.4.6. 학습 진행도 조회 프로토콜

GET	progress/<user_id>/<progress_id>	
사용자의 학습 진행도 현황을 조회할 때 요청하는 주소		
분류	항목명	설명 및 제약사항
Parameter	user_id	사용자 id 값
	progress_id	현황 id 값
Response	message	"조회 성공/실패"
	progress value	학습 현황도(%)
Status Code	200 OK	조회 성공
	404 Not Found	사용자 없음/조회 실패
	500 Internal Server Error	내부 서버 오류

Table 6. 학습 진행도 프로토콜 표

#### 7.4.7. 강의 상태 저장 프로토콜

POST	progress/<user_id>/<progress_state>	
사용자의 강의 완료/미완료 상태를 저장할 때 요청을 보내는 주소		
분류	항목명	설명 및 제약사항
Parameter	user_id	사용자 id 값
	progress_id	현황 id 값
Response	message	저장 성공/실패
Status Code	200 OK	저장 성공
	404 Not Found	사용자 없음/저장 실패
	500 Internal Server Error	내부 서버 오류

Table 7. 강의 상태 저장 프로토콜 표

#### 7.4.8. 커리큘럼 요청 프로토콜

GET	<user_id>	
사용자의 처음 커리큘럼을 받아올 때, 요청을 보내는 주소		
분류	항목명	설명 및 제약사항
Parameter	id	사용자 id 값

Response	curriculum	커리큘럼 id 값
Status Code	200 OK	커리큘럼 불러오기 성공
	404 Not Found	커리큘럼 존재하지 않음
	500 Internal Server Error	내부 서버 오류

Table 8. 커리큘럼 요청 프로토콜 표

#### 7.4.9. 학습지 요청 프로토콜

GET	<user_id>/<level>/<week>/<title>	
사용자가 각 Week 에 Chapter 버튼을 눌렀을 때, 학습지를 불러오는 요청을 보내는 주소		
분류	항목명	설명 및 제약사항
Parameter	id	사용자 id 값
	level	학습지 난이도 값
	week	학습지 week 값
	title	학습지 id 값
Response	workbook	학습지 id 값
Status Code	200 OK	학습지 불러오기 성공
	404 Not Found	학습지 존재하지 않음
	500 Internal Server Error	내부 서버 오류

Table 9. 학습지 요청 프로토콜 표

#### 7.4.10. 문제 로그 조회 프로토콜

GET	log/<user_id>	
사용자가 문제 로그를 조회할 때 요청하는 주소		
분류	항목명	설명 및 제약사항
Parameter	user_id	사용자 id 값
Response	solved_qlist	해결한 문제 리스트
Status Code	200	리스트 불러오기 성공
	400	리스트 존재하지 않음
	500 Internal Server Error	내부 서버 오류

Table 10. 문제 로그 조회 프로토콜 표

#### 7.4.11. 코드 제출 프로토콜

POST	code_sumit/<qid>	
사용자의 학습 진행도 현황을 조회할 때 요청하는 주소		
분류	항목명	설명 및 제약사항

Parameter	qid	문제 id 값
Response	message	"코드 제출 성공/실패"
Status Code	200	제출 성공
	400	제출 실패
	500 Internal Server Error	내부 서버 오류

Table 11. 코드 제출 프로토콜 표

#### 7.4.12. 로그인 프로토콜

POST	login/<user_id>/<password>	
사용자가 로그인을 할 때 요청을 보내는 주소		
분류	항목명	설명 및 제약사항
Parameter	user_id	사용자 id 값
	password	패스워드
Response	message	로그인 성공/실패
Status Code	200	로그인 성공
	400	사용자 없음
	500 Internal Server Error	내부 서버 오류

Table 12. 로그인 프로토콜 표

#### 7.4.13. 회원가입 프로토콜

POST	login/<user_id>/<password>	
사용자가 회원가입을 할 때 요청을 보내는 주소		
분류	항목명	설명 및 제약사항
Parameter	user_id	사용자 id 값
	password	패스워드
Response	message	회원가입 성공/실패
Status Code	200	회원가입 성공
	400	회원가입 실패
	500 Internal Server Error	내부 서버 오류

Table 13. 회원가입 프로토콜 표

## 8. database Design

### 8.1. Objectives

Database design 을 전반적으로 설명하고 ER diagram을 통해 Entity와 Relationship, Relational Schema를 통해 각각의 관계를 설명한다. 또 Django의 models.py의 코드로 Database가 어떻게 구현되었는지 설명한다.

### 8.2. ER Diagram

현재 구현한 시스템에는 user\_info, user\_data, leetcode\_integrated, q\_recommend\_res 총 4개의 Entity가 존재한다. ER diagram에서 Entity는 노란색 직사각형, Entity 간 관계는 초록색 마름모와 숫자로 표현한다. Entity가 가지는 Attribute는 주황색 타원형으로 표현하고 Primary key는 타원형에 빨간색으로 밑줄을 그어 표현했다.

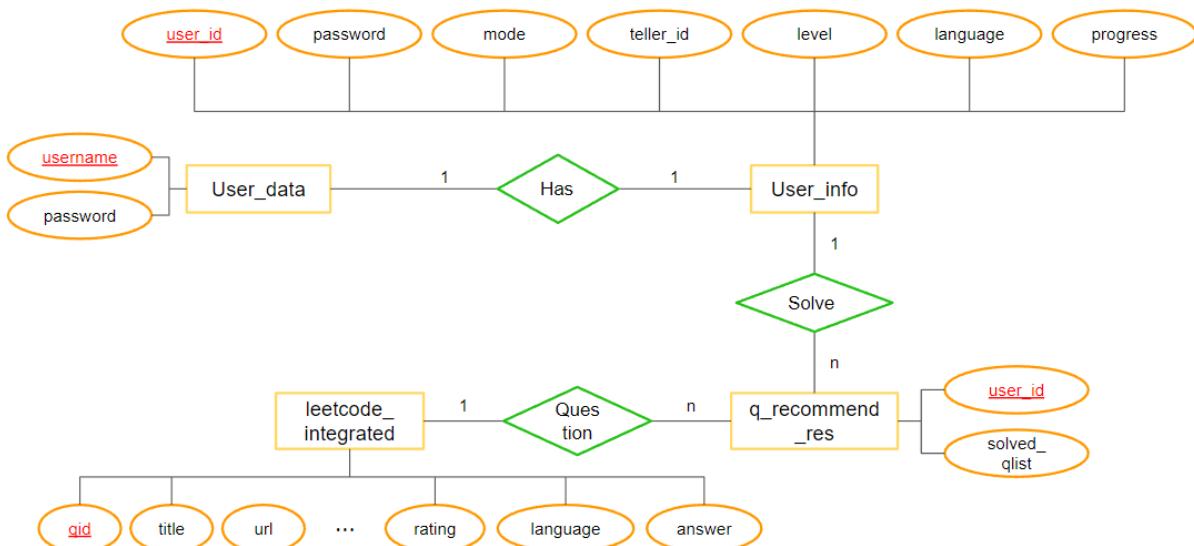


Figure 29. ER Diagram

### 8.3. Relational Schema

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

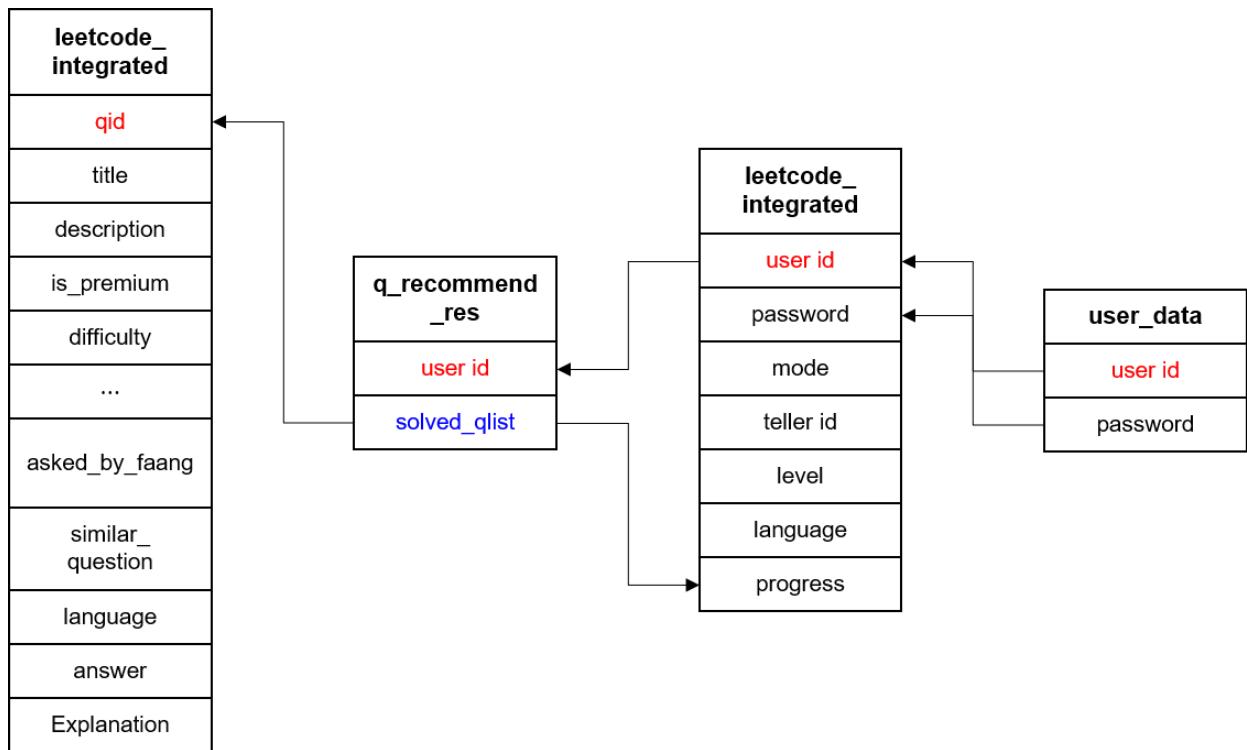


Figure 30. Relational Schema

## 8.4. Models.py

Django 내의 models.py에 저장된 코드이다. 이는 자동으로 sqlite DB에 저장된다.

### 8.4.1. User data

```
class user_data(models.Model):
    username = models.CharField(max_length=40, default='default user')
    password = models.CharField(max_length=30, default='default password')
```

### 8.4.2. User info

```
class user_info(models.Model):
    user_id = models.CharField(max_length=40, default='default user')
    password = models.CharField(max_length=30, default='default password')
    mode = models.IntegerField(default = 0)
    teller_id = models.IntegerField(default = 0)
    level = models.IntegerField(default = 0)
    language = models.IntegerField(default = 0)
    progress = models.IntegerField(default = 0)
```

#### **8.4.3. Leetcode Integrated**

```
class leetcode_integrated(models.Model):
    qid = models.IntegerField(default = 0)
    title = models.CharField(max_length=50, default='default title')
    description = models.CharField(max_length=10000, default='default description')
    is_premium = models.IntegerField(default = 0)
    difficulty = models.CharField(max_length=10, default='defficulty')
    solution_link = models.CharField(max_length=60, default='default solution')
    accpetance_rate = models.FloatField(default=0.0)
    frequency = models.FloatField(default=0.0)
    url = models.CharField(max_length=60, default='default url')
    discuss_count = models.IntegerField(default = 0)
    accepted = models.CharField(max_length=10, default='default accepted')
    submissions = models.CharField(max_length=10, default='default submission')
    companies = models.CharField(max_length=1000, default='default companies')
    related_topics = models.CharField(max_length=500, default='related topics')
    likes = models.IntegerField(default = 0)
    dislikes = models.IntegerField(default = 0)
    rating = models.IntegerField(default = 0)
    asked_by_faang = models.IntegerField(default = 0)
    similar_questions = models.CharField(max_length=1000, default='similar questions')
    language = models.CharField(max_length=20, default='default language')
    answer = models.CharField(max_length=10000, default='default answer')
    Explanation = models.CharField(max_length=10000, default='default explanation')
```

#### **8.4.4. Questions recommend res**

```
class q_recommend_res (models.Model):
    user_id = models.CharField(max_length=40, default='default user', primary_key = True)
    questionId1 = models.ForeignKey( leetcode_integrated)
    questionId2 = models.ForeignKey( leetcode_integrated)
    ...
    questionId21 = models.ForeignKey( leetcode_integrated)
```