# Software Design Specification

of

CodeCraft


Team 9

2023.05.21

Team Leader 2018313469 박주봉

Team Member 2018312663 김민성

2019311036 신새별

2018311428 이하은

2017312010 최현진

2019312589 권혁준

# Contents

# List of Figures

## List of Tables

# 1. Preface

## 1.1. Readership

본 문서는 설계 및 개발에 관여하는 예상 독자들을 위해 작성되었다. 본 문서의 주요 독자는 6 명으로 이루어진 Team 9 와 소프트웨어공학개론 강의 조교님, 이은석 교수님이며 2023 년 1 학기 소프트웨어공학개론 강의 수강자 또한 독자가 될 수 있다. 본 문서를 상업적 용도로 활용할 경우 개발팀의 허가를 얻어야 하며 학습 및 교육의 용도로 활용 시 재배포 및 수정하는 것에 제약은 없다.

## 1.2. Scope / Objective

본 문서는 성균관대학교 학생을 대상으로 하는 AI 를 활용한 프로그래밍 교육 플랫폼, Code Craft 서비스를 제공하기 위해 작성된 문서이다. 이 서비스는 성균관대학교 이은석 교수님의 2023 년 1 학기 소프트웨어공학개론 수업 Team 9 에 의해 설계되었다. 현재 성균관대학교를 비롯한 여러 대학에서 프로그래밍 과목이 필수 수강 과목으로 지정되었으며 여러 사교육 업체나 기업에서도 프로그래밍 교육에 대한 수요가 증가하고 있다. 이에 따라 사용자에게 흥미를 불러일으킬 수 있는 게임형 프로그래밍 교육 플랫폼을 제공하고자 한다. 또한 대화형 인공지능 서비스인 ChatGPT 가 맞춤형 프로그래밍 선생님 역할을 하도록 하여 더욱 심화된 학습과 실습을 가능하게 하고 질문에 대한 다양한 답변을 제공한다. 본 서비스를 통해 이용자들은 다른 사람의 도움 없이 효과적인 프로그래밍 교육을 제공받을 수 있으며 더 나아가 게임 형식으로 이루어져 있기 때문에 재미있게 프로그래밍 역량을 기를 수 있다.

본 문서는 목표로 하는 AI 를 활용한 프로그래밍 교육 플랫폼 Code Craft 서비스를 시행하기 위해 사용되었거나 사용될 디자인에 대한 설명을 담고 있다. 디자인은 개발하기 위하여 클라이언트와 서버를 디자인하는 과정에서 도출한 요구 사항 명세서의 내용을 토대로 만들어졌다.

### 1.3. Document Structure

1. Preface: 본 문서의 목적, 예상 독자 및 문서의 구조에 대해 설명한다.

2. Introduction: 본 문서를 작성하는데 사용된 다양한 도구들과 다이어그램들, 참고 자료들에 대해 설명한다. 또한 본 프로젝트의 시스템의 범위를 서술한다.

3. Overall System Architecture: 시스템의 전체적인 구조를 Context 다이어그램, Use-Case 다이어그램, Sequence 다이어그램을 이용하여 서술한다.

4. System Architecture - Frontend: Frontend 시스템의 구조를 Class 다이어그램과 Sequence 다이어그램을 이용하여 서술한다.

5. System Architecture - Backend: Backend 시스템의 구조를 CodeCraft Software Design Specification 다이어그램과 Class 다이어그램을 다이어그램을 이용하여 서술한다.

6. Protocol Design: 클라이언트와 서버의 커뮤니케이션을 위한 프로토콜 디자인을 서술한다.

7. Database Design: 시스템의 데이터베이스 요구 사항을 기반으로 ER 다이어그램과 Relation Schema 를 이용하여 시스템의 데이터베이스 디자인을 서술한다.

8. Testing Plan: 시스템을 위한 테스트 계획을 서술한다.

9. Development Plan: 시스템의 구현 계획 및 그것을 위한 개발 도구,
라이브러리 등의 개발 환경을 설명한다.

10. Supporting Information: 본 문서의 작성 및 수정 기록을 기술한다.

## 2. Introduction

### 2.1. Applied Diagrams

#### 2.1.1. UML

Unified Modeling Language 는 통합 다이어그램 집합으로 구성된 표준화된 통합 모델링 언어다. UML 은 시스템의 모든 것을 문서화, 지정, 구축하는데 사용되는 표준 언어다. UML 은 여러 개의 표준화된 다이어그램을 제공한다. 그 목록은 다음과 같다: use-case diagram, class diagram, sequence diagram, communication diagram, activity diagram, state diagram, component diagram,

deployment diagram, package diagram. UML 다이어그램을 이용함으로써 프로그래밍을 단순화시켜 표현하여 커뮤니케이션의 활성화가 가능하며 대규모 프로젝트 구조의 로드맵과 개발을 위한 시스템 구축의 기반을 설립 가능하다. UML 은 시각화, 명세화, 구축, 문서화 언어다. 즉, 시각적인 형태로 표현하며 표준화된 다이어그램을 제공한다. 또한 소프트웨어 개발의 분석, 설계 단계의 각 과정에 필요한 모델을 명세화한다. 그리고 UML 은 다양한 프로그래밍 언어로 표현할 수 있으며 UML 모델을 코드로 자동 변환할 수 있다. 마지막으로 UML 의 설계한 내용을 자동으로 문서화 가능하다.

### 2.1.2. Context Diagram

Context Diagram 은 Data Flow Diagram 에서 가장 높은 수준의 다이어그램으로, 시스템 전체와 외부 요인의 입력 및 출력을 보여준다. Context diagram 은 시스템과 그 시스템과 상호작용할 수 있는 모든 외부의 엔티티들을 나타내며, 그 사이의 정보의 흐름을 표현한다. 시스템 내부 구조에 대한 세부 정보는 명시적으로 제외한다. Context diagram 은 비즈니스 계획 또는 프로젝트 요구 사항에서 누락 및 오류를 확인하는데 적합하며, 프로젝트의 위험을 크게 줄일 수 있다. 또한 Context diagram 은 개발 담당자가 어떤 사용자 그룹을 고객으로 간주하는지 명확히 하는 데 도움이 된다. 이 다이어그램은 프로젝트에서 설계할 시스템의 세부 정보와 영역을 이해하기 위해 널리 사용된다. 다른 프로젝트 다이어그램과 달리, Context diagram 은 엔지니어 또는 개발자가 사용하기보다는 프로젝트의 이해 관계자가 사용한다. 따라서 이해관계자가 항목을 분석할 때 쉽게 이해할 수 있도록 간단하고 이해하기 쉬운 언어로 작성해야 한다.

### 2.1.3. Sequence Diagram

Sequence Diagram 은 특정한 행동이 어떠한 순서로 어떤 객체와 어떻게 상호작용을 하는지 표현하는 행위 다이어그램이다. Event Diagram 이라고도

불리는 이것은 시나리오와 관련된 객체와 시나리오의 기능 수행에 필요한 객체 간에 교환되는 메시지를 순서대로 표현한다. 모든 커뮤니케이션은 시간 순으로 표현된다. 현재 존재하는 시스템이 어떤 시나리오로 작동하고 있는지를 나타낼 수 있다. Sequence Diagram 을 위해 필요한 필수적인 구성 요소로는 생명선(life line), 메시지, 활성 박스 (Activation box)가 있다. 생명선은 상호작용에 참여하는 오브젝트를 의미하며, 메시지는 서로 다른 객체 간의 상호작용 혹은 의사소통 통신을 정의하는 요소다. 메시지의 종류로는 동기 메시지, 비동기 메시지, 자체 메시지, 반환 메시지가 있다. 동기 메시지는 해당 메시지에 대한 응답을 기다린다. 비동기 메시지는 이와 반대로, 메시지 전송 객체가 계속하기 전까지 응답을 요구하지 않는 메시지다. 즉 전송 객체의 호출만을 표시한다. 자체 메시지는 자기 자신에게 보낸 메시지이며, 반환 메시지는 이전 호출의 반환을 기다리는 객체에 다시 반환되는 메시지다. 활성 박스란 객체의 호출이 이루어지는 박스다.

#### 2.1.4. Use-case Diagram

Use case Diagram 은 시스템에서 제공해야 하는 기능이나 서비스를 명세한 다이어그램이다. Use case Diagram 은 사용자와 그 사용자가 사용하는 use case 들 간의 관계를 보여주며, 사용자 – 시스템 간 상호작용을 표현한다. 이 다이어그램을 통해 각기 다른 종류의 시스템 사용자와 각 사용자 별 use case 를 알 수 있다. 외부에서 본 시스템의 기능을 표현하기 때문에, 사용자가 수행하는 기능을 파악하기 위해 사용된다. Use case Diagram 의 기본적인 구성 요소로는 scope, use case, relationship 그리고 actor 가 있다. Use case 는 시스템이 제공하는 서비스와 기능이며, scope 는 시스템이 제공하는 기능의 범위를 나타낸다. Actor 는 시스템과 상호작용하는 시스템 외적 존재다. Actor 는 사람일 수도 있으며, 사람이 아닌 외부적 시스템일 수도 있다. Relationship 은 actor 와 use case, 또는 두 개의 use case 사이에 나타내며, association, include, generalization, extended 라는 4 가지 종류로 구성된다.

### 2.1.5. Class Diagram

Class Diagram 은 시스템을 구성하는 클래스, 그 속성, 기능 및 객체들 간의 관계를 표현하여 시스템의 정적인 부분을 보여준다. 클래스란 동일한 속성과 행위를 수행하는 객체의 집합이자 실제 객체를 생성하는 설계도다. 이 다이어그램은 실제로 구현될 소스코드와는 다를 수 있으며 그 의미나 해석 또한 경우에 따라 달라질 수 있다. 전체 시스템의 구조 및 클래스의 의존성을 파악하고 다른 사람들과 커뮤니케이션, 설계 논의를 하기 위해 해당 다이어그램이 사용될 수 있으며, 유지보수를 위한 디자인의 back-end 문서에서도 사용된다. (본 문서에서도 back-end 설계에 사용되었다.) Class Diagram 은 불필요한 내용을 제외하고 간결한 모델로 그려야 하며 직관성을 위해 의미 있고 명확한 이름을 사용해야 한다. 클래스는 3 가지 부분 (클래스 이름, 멤버 변수, 메서드)으로 나누어 표현된다. 클래스 이름은 가장 상단에 작성되며 중간에 멤버 변수, 가장 하단에 메서드가 작성된다.

### 2.1.6. Entity Relationship Diagram

ER Diagram 은 구조화된 데이터와 그들 간의 관계를 사람이 이해할 수 있는 형태로 표현하는 다이어그램이며, 현실에서의 요구사항들을 이용한 데이터베이스 설계 과정에서 활용된다. 해당 다이어그램은 엔티티, 속성 (attribute), 관계성(Relationship) 으로 구성된다. 엔티티란 관리할 정보의 실체로, 둥근 사각형으로 작성한다. 모든 엔티티는 하나 이상의 식별자를 가진다. 속성은 엔티티를 구성하고 있는 구성 요소이며, 관계성은 엔티티들 간의 관계이다. 강한 관계성은 개체가 다른 개체를 통해 존재할 수 있는 의존적인 관계를 뜻하며, 약한 관계성은 개체가 다른 개체와 독립적으로 존재할 수 있는 독립적인 관계를 뜻한다. 이 3 가지 구성 요소들을 표시하여 ER Diagram 은 데이터베이스의 논리적 구조를 설명한다.

## 2.2. Tools

### 2.2.1. Microsoft PowerPoint

Diagram 및 figure 의 작성을 위해 PowerPoint 가 사용되었다. PowerPoint 는 다양한 도형, 그림 및 텍스트의 삽입과 편집을 지원하며, 표와 그래프의 생성 또한 가능하기 때문에 문서의 작성에 필요한 Diagram 및 figure 를 생성할 때 유용하게 사용된다.

### 2.2.2. Figma

웹 사이트 제작시 디자인을 빠르고 쉽게 할 수 있도록 도와주는 웹 기반 모델링 도구이며, 기존 사용되는 앱인 XD 보다 간단한 component 로 이루어져 있다. Export 또한 잘 이루어져 있어 Frontend 와 UI/UX 디자인과의 협업에 유용하게 쓰인다.

## 2.3. Project Scope

본 문서에서 설계하는 소프트웨어는 사용자가 인공지능의 도움을 받으며 코딩 학습을 할 수 있는 온라인 서비스이다. 사용자는 코딩 교육 어플리케이션을 통해 다양한 난이도의 문제를 경험하고 도중에 어려움을 느끼면 ChatGPT 가 힌트를 제공하여 도움을 줄 수 있다. 또한 오답노트 기능을 통해 사용자가 이해하지 못하거나 틀린 문제에 대해 해답을 정리해보고 자기 주도적으로 학습을 할 수 있다.

이 서비스는 게임을 하면서 코딩 교육을 할 수 있기 때문에 사용자는 게임의 재미를 느끼며 코딩 학습을 할 수 있다. 또한, 게임 속에서 코딩 학습을 하면서 게임을 클리어하면 점수를 얻는 등의 재미 요소가 추가되어 있으며 사용자는 게임의 재미와 코딩 학습의 효과를 모두 느낄 수 있다.

## 2.4. References


- Django Software Foundation, “Django Project”. https://www.djangoproject.com/
- OpenJS Foundation, “About Node.js” https://nodejs.org/
- https://github.com/skkuse/2021spring_41class_team9
- https://github.com/skkuse/2022fall_41class_team2

# 3. System Architecture - Overall

## 3.1. Objective

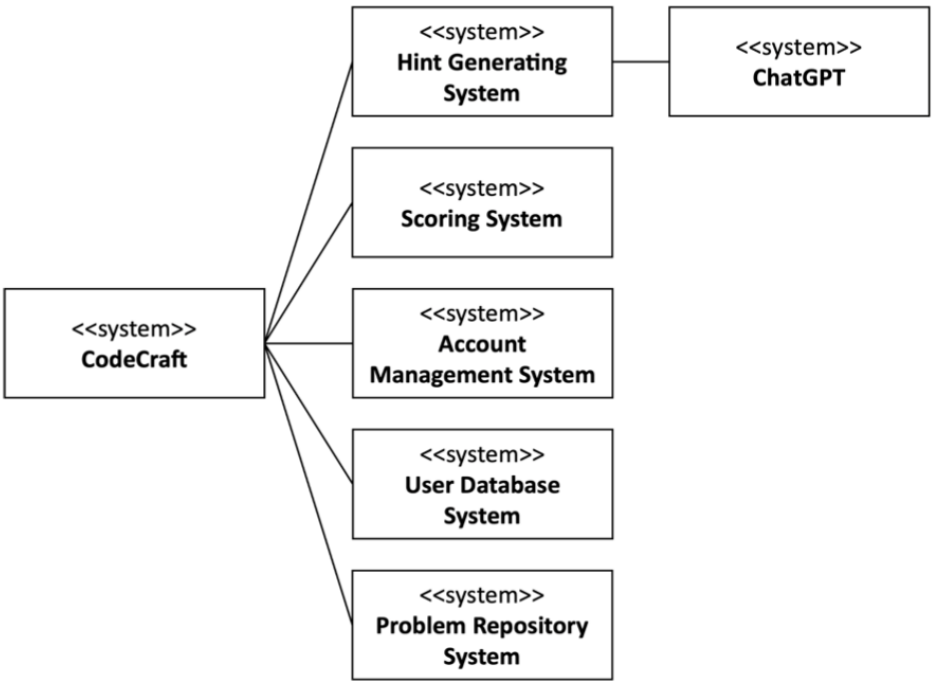
본 chapter 에서는 frontend 설계에서 backend 설계에 이르는 프로젝트 application 의 system architecture 에 대해 설명한다.

## 3.2. System Organization

이 서비스는 클라이언트 - 서버 모델을 적용하여 설계되었으며, frontend application 은 사용자와의 모든 상호작용을 담당하며, frontend application 및 backend application 은 JSON 기반의 HTTP 통신을 통해 데이터를 주고받는다. backend application 은 설계 사양 요청을 frontend 로부터 컨트롤러로 배포하고, 오라클 데이터베이스에서 필요한 객체 정보를 얻어서 데이터베이스에서 처리한 후 JSON 형식으로 전달한다. backend application 은 사용자 정보를 수신 받고 문제를 전달한다. 이후 사용자가 문제 풀이를 진행하고 나면 문제 풀이 과정에서 발생한 과정 및 결과를 전달한다.

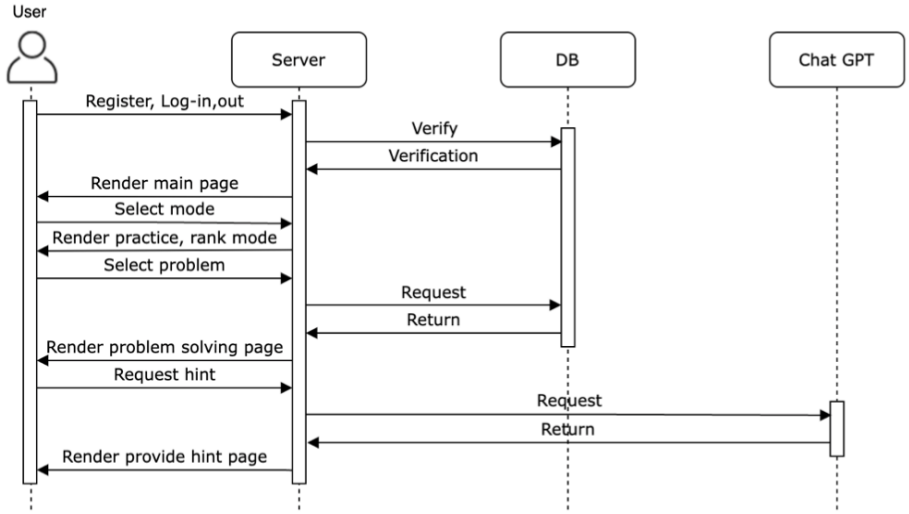
### 3.2.1. Context Diagram

*[Figure 1] Context Diagram*



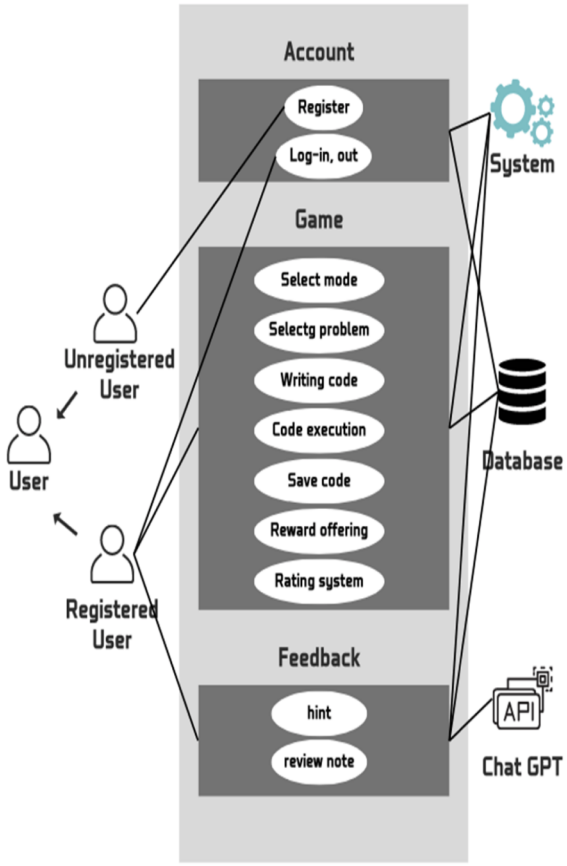
### 3.2.2. Sequence Diagram

*[Figure 2] Sequence Diagram*




### 3.2.3. Use Case Diagram

*[Figure 3] Use Case Diagram*



# 4. System Architecture - Frontend

## 4.1. Objective

본 Chapter 에서는 CodeCraft 의 시스템 중, 유저와의 상호작용을 담당하는 Frontend 시스템의 구조와 속성, 그리고 이 시스템이 포함하는 기능을 서술하며 이를 구성하는 각 구성요소들 간의 관계를 기술한다.

## 4.2. Subcomponents

### 4.2.1. Profile

기본적인 사용자 정보를 처리하고 관리하는 객체이다. 개인 정보에는 아이디와 비밀번호, 이메일 계정 정보 등이 있다. 회원가입 이후 사용자는 정보를 변경할 수 없다.

#### 4.2.1.1. Attribute

프로파일 객체에 있는 속성들은 다음과 같다.

- User id: 사용자 아이디
- Password: 비밀번호
- Email: 이메일 주소

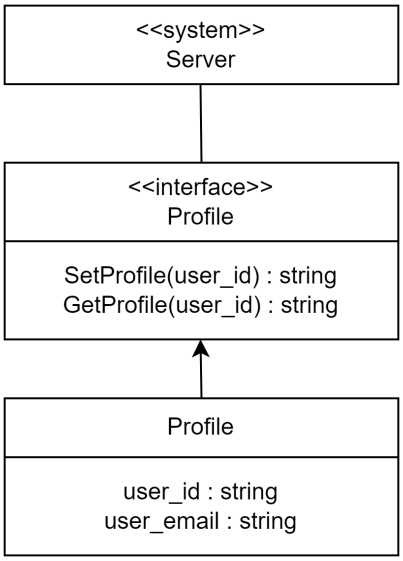
#### 4.2.1.2. Methods

프로파일 클래스가 가지는 메서드는 다음과 같다.

- SetProfile() : 서버로부터 가져온 사용자 정보를 보여주는 함수
- GetProfile() : 서버로부터 사용자 프로필 정보를 가져오는 함수

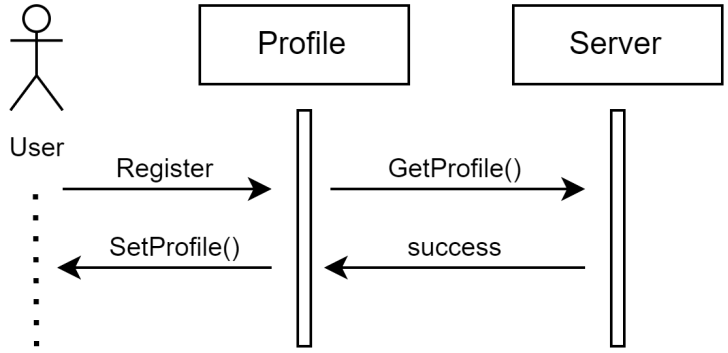
#### 4.2.1.3. Context Diagram

*[Figure 4] Profile Context Diagram*




#### 4.2.1.4. Sequence Diagram

*[Figure 5] Profile Sequence Diagram*




### 4.2.2. Score

Score Component 는 사용자가 코드를 입력하고 정답과 비교하여 점수를 산출하는 프로세스를 담당한다.

#### 4.2.2.1. Attribute

Score 개체에 있는 속성들은 다음과 같다.

- User id : 사용자의 id
- Score : 코드 문제 풀이 점수

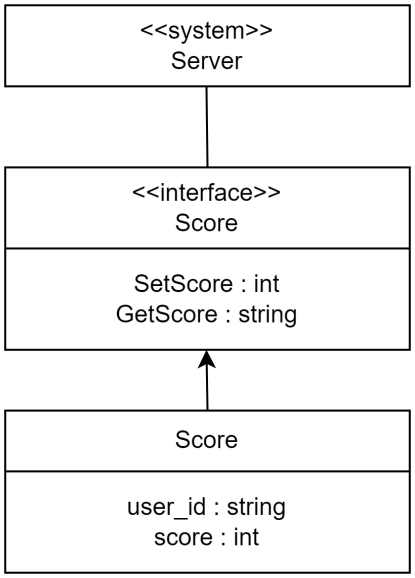
#### 4.2.2.2. Methods

Score 클래스가 가지는 메서드는 다음과 같다.

- GetScore()
- SetScore()

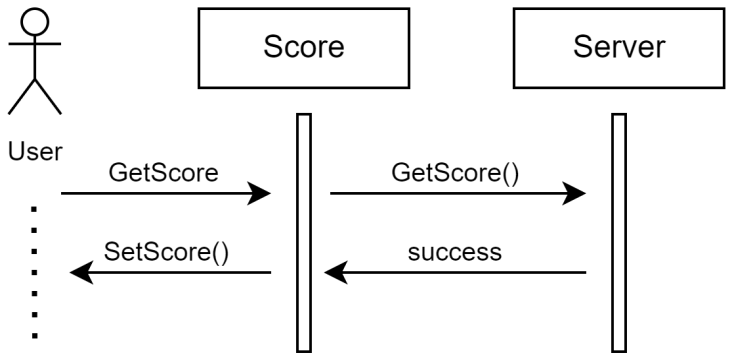
#### 4.2.2.3. Context Diagram

*[Figure 6] Score Context Diagram*




#### 4.2.2.4. Sequence Diagram

*[Figure 7] Score Sequence Diagram*



#### 4.2.3. Hint

힌트 클래스는 사용자가 문제를 풀 때 요청하는 힌트를 제공하는 프로세스를 담당한다. 5 가지 종류로 이루어진 힌트를 사용자는 최대 3 회까지 선택하여 요청할 수 있다.

##### 4.2.3.1. Attribute

Hint 개체에 있는 속성들은 다음과 같다.

- Hint type : 힌트의 종류
- Hint count : 힌트를 사용한 횟수

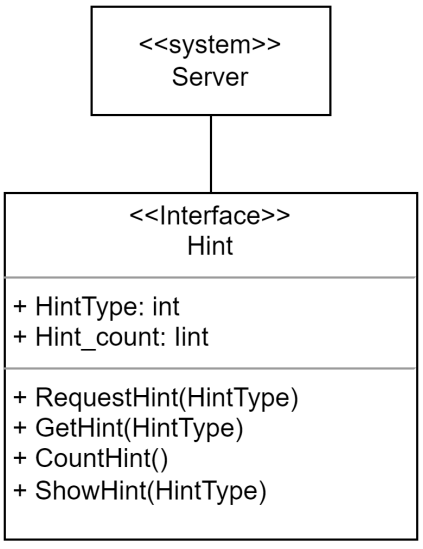
##### 4.2.3.2. Methods

Hint 클래스가 가지는 메서드는 다음과 같다.

- RequestHint(): 사용자가 힌트를 요청한다
- GetHint(): 타입별 힌트를 요청하여 받아온다.
- HintCount(): 사용자가 사용한 힌트의 총 갯수를 저장한다.
- ShowHint(): 힌트 클래스를 렌더링하여 사용자 화면에 시각화한다.

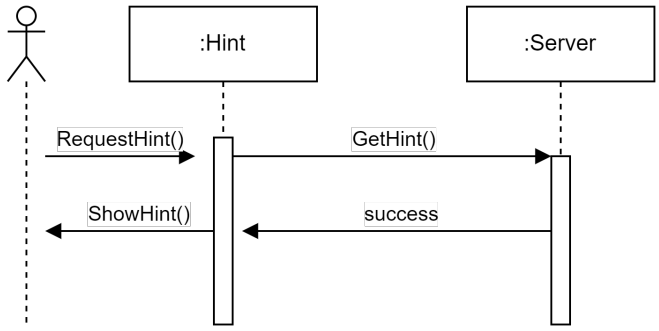
##### 4.2.3.3. Context Diagram

*[Figure 8] Class Context Diagram*



#### 4.2.3.4. Sequence Diagram

*[Figure 9] Class Sequence Diagram*




### 4.2.4. Coding

사용자가 코드 입력을 하면 정답과 비교하는 프로세스를 담당하는 클래스이다.

#### 4.2.4.1. Attribute

- problem_id: 문제 (단계)의 id 값이다.

- repo_id: Coding 클래스의 고유한 id 이다.

- result: 해당 Coding 클래스의 결과 값을 저장하는 클래스이다.

- code: 해당 Coding 클래스의 코드 입력 값을 저장하는 클래스이다.

- code_content: 사용자가 입력한 코드 내용이다.

- code_id: 코드 식별 id 이다.

- success : 문제의 정답 여부를 알려주는 값이다.

#### 4.2.4.2. Methods

- SaveCode(): 현재까지 입력된 사용자의 코드 입력 값을 서버로 전송하여 저장한다.

- GetCode(): 서버에 저장되어 있는 코드 입력 값을 요청하여 받아온다.

- ExecuteCode(): 현재까지 입력된 사용자의 코드 입력 값을 서버로 보낸 후 실행하여 결과 값을 받는다.

- GradeCode(): 서버에 저장되어 있는 코드를 채점하여 결과 값을 받아온다.

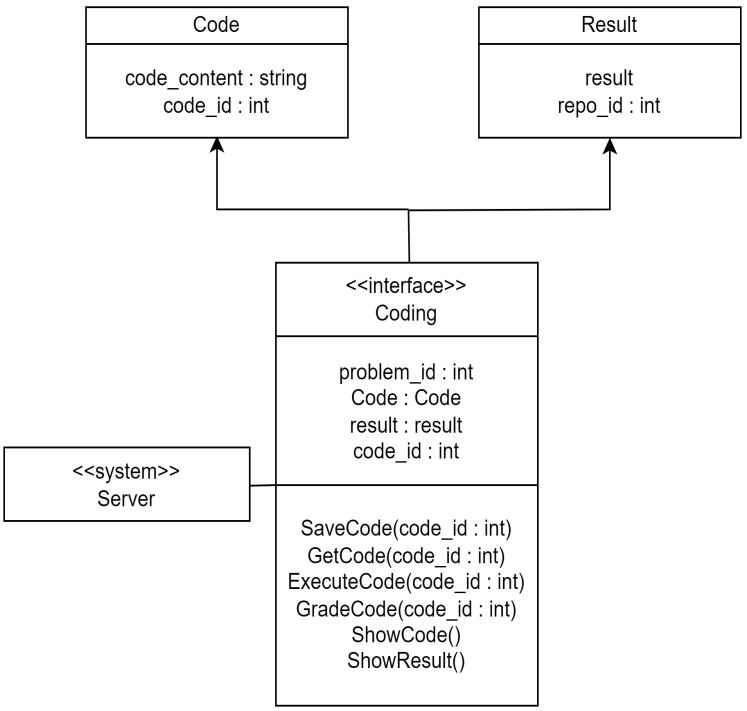
- ShowCode(): Coding 클래스를 렌더링하여 시각화한다.

- ShowResult(): Coding 결과값을 보여준다.

- IsSuccessful() : Coding 결과값이 문제의 결과값과 일치하는지의 여부를 판단한다.

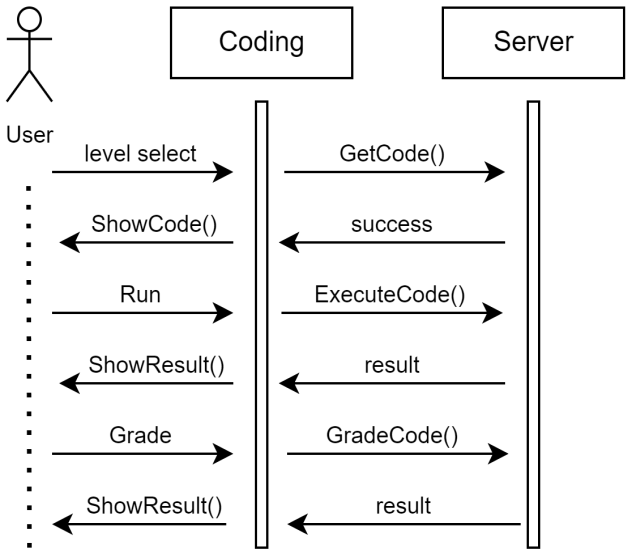
#### 4.2.4.3. Context Diagram

*[Figure 10] Coding Context Diagram*



#### 4.2.4.4. Sequence Diagram

*[Figure 11] Sequence Context Diagram*




### 4.2.5. Review Note

사용자가 실전모드에서 틀린 문제를 다시 볼 수 있도록 문제를 다시 보여주는 프로세스를 담당한다. 이와 함께, 내가 이전에 썼던 코드, 런타임 최단시간 답변 및 인공지능 수정 답변을 제공한다.

#### 4.2.5.1. Attribute

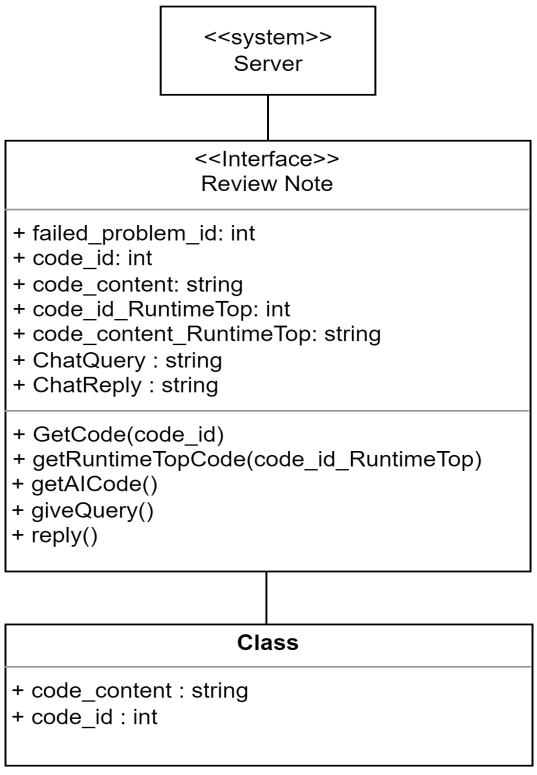
- failed_problem_id: 실전모드에서 실패한 문제 (단계)의 id 값이다.
- code_id: 코드 식별 id 이다.
- code_content: 사용자가 실전모드에서 입력한 코드 내용이다.
- code_id_RuntimeTop: 런타임 최대시간 코드 식별 id 이다.
- code_content_RuntimeTop: 런타임 최단시간 코드 내용이다.
- ChatQuery : 사용자가 인공지능 챗봇에게 보낸 질의 내용이다.
- ChatReply : 사용자의 질의에 대한 인공지능 챗봇의 응답 내용이다.

#### 4.2.5.2. Methods

- GetCode(): 서버에 저장되어 있는 사용자의 코드 입력 값을 요청하여 받아온다.
- getRuntimeTopCode(): 해당 문제의 런타임 시간이 가장 짧은 코드를 요청하여 받아온다.
- getAICode(): 해당 문제의 인공지능이 해답으로 작성한 코드를 요청하여 받아온다.
- ShowCode(): 요청된 코드 값을 보여준다.
- giveQuery(): 사용자가 인공지능 챗봇에게 해당 문제에 대해 궁금한 사항을 입력하는 프로세스를 수행한다.
- reply(): 인공지능 챗봇이 사용자의 질의에 응답하는 프로세스를 수행한다.

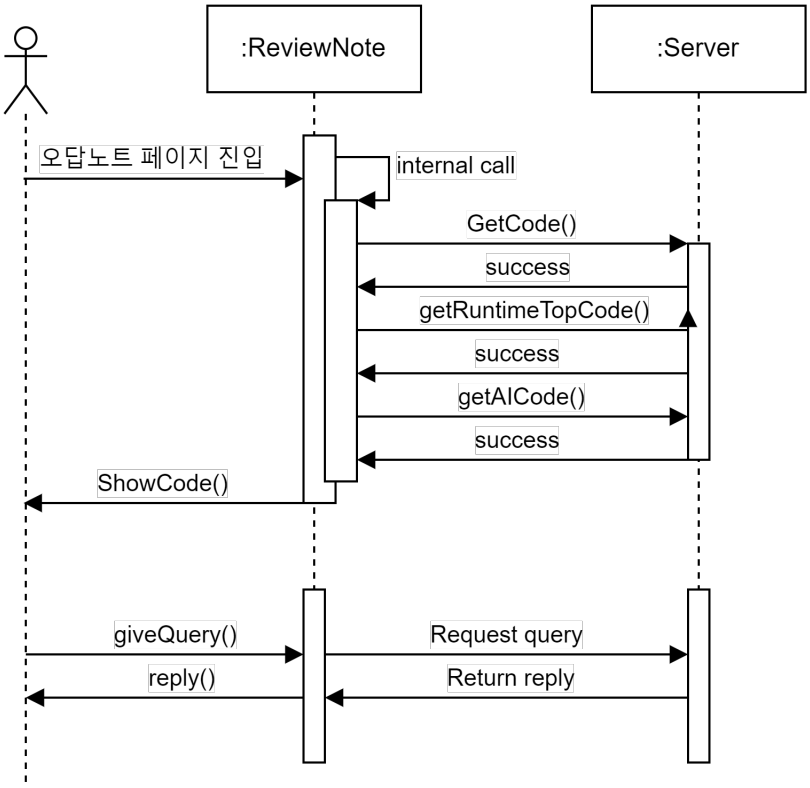
#### 4.2.5.3. Context Diagram

*[Figure 12] Review Note Context Diagram*



#### 4.2.5.4. Sequence Diagram

*[Figure 13] Review Note Sequence Diagram*



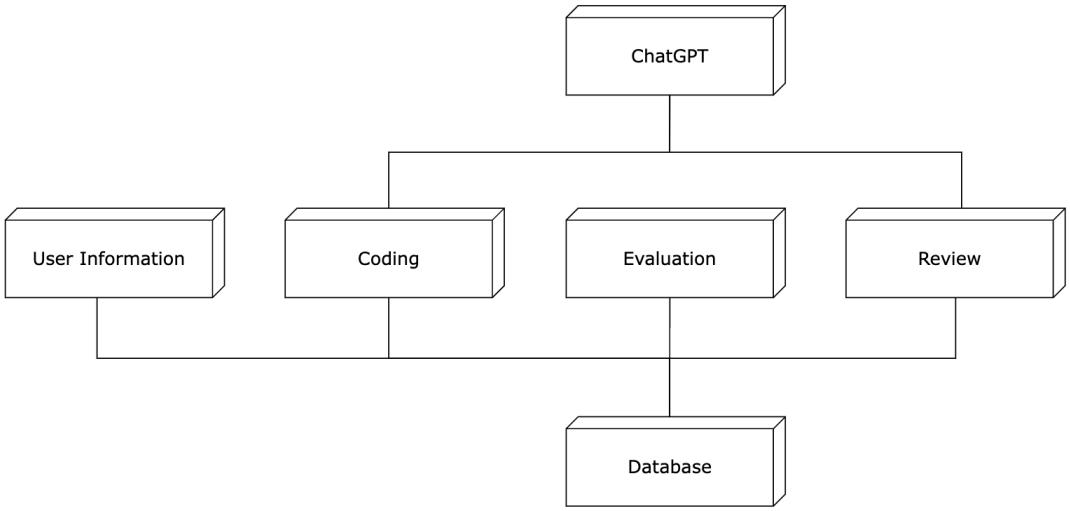

# 5. System Architecture - Backend

## 5.1. Objective

이 장에서는 본 프로젝트의 시스템에 대한 전반적인 구조에 대해 설명한다. 시스템의 구성 요소와 각 구성 요소 간의 관계 그리고 어떻게 동작하는지에 대한 흐름을 설명하기 위해 다이어그램을 사용한다.

## 5.2. Overall Architecture

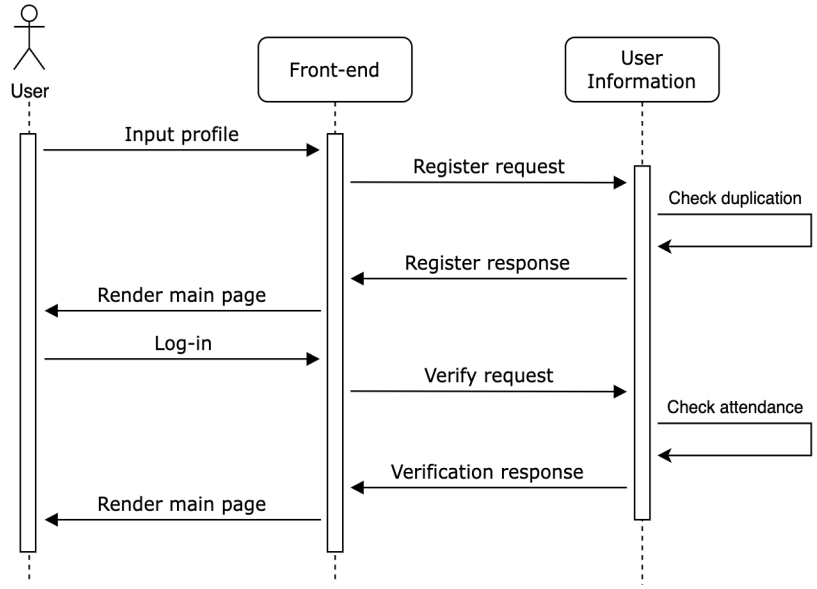
*[Figure 14] Backend Overall Architecture*




## 5.3. Sub Systems

### 5.3.1. User Information System

*[Figure 15] User Information System Sequence Diagram*

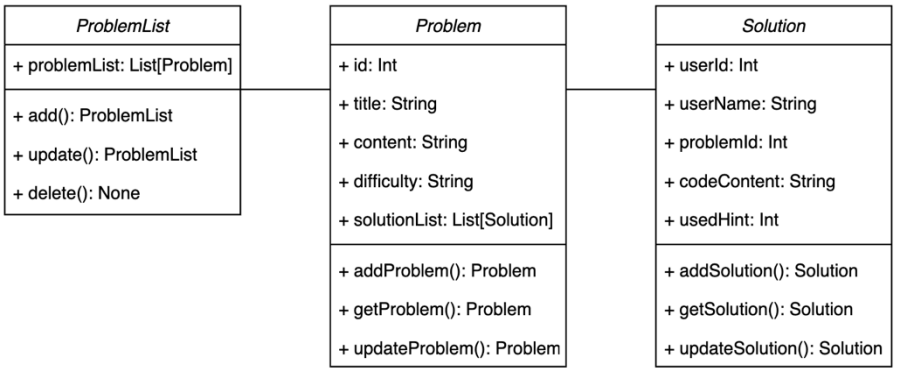



사용자 정보를 관리하는 시스템으로 사용자 정보를 등록하는 요청을 보낼 경우에 시스템은 중복 여부를 확인하고 중복된 정보가 존재하지 않다면 데이터베이스에

저장한다. 사용자가 로그인을 하면 시스템은 로그인 기록을 바탕으로 attendance 값을 갱신한다

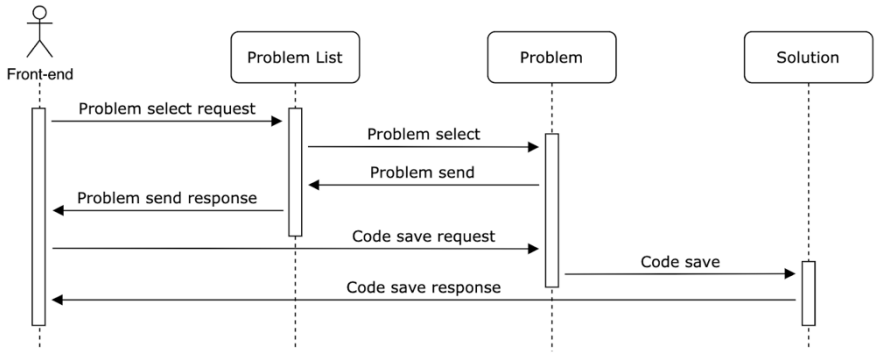
### 5.3.2. Coding System

*[Figure 16] Coding System Class Diagram*




Problem List 에 대한 객체가 존재하며 개별 Problem 들에 대한 객체들을 다룬다. Problem 객체에서는 다시 문제 풀이에 대한 객체들을 다루는 Solution List 가 존재하고 Solution 에 대한 객체는 사용자 정보와 코드들을 포함한다.

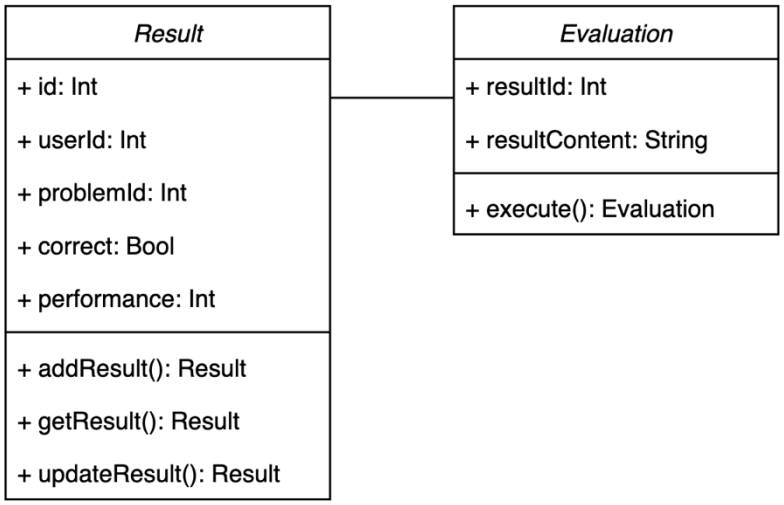
*[Figure 17] Coding System Sequence Diagram*




사용자가 Problem List 에서 문제를 선택하면 해당 문제 정보를 불러온다. 문제 풀이를 진행하며 코드를 작성할 경우 Problem 으로 저장을 위한 요청을 보내고 Solution List 에 사용자의 id 와 problem id 에 해당하는 Solution 을 저장한다.

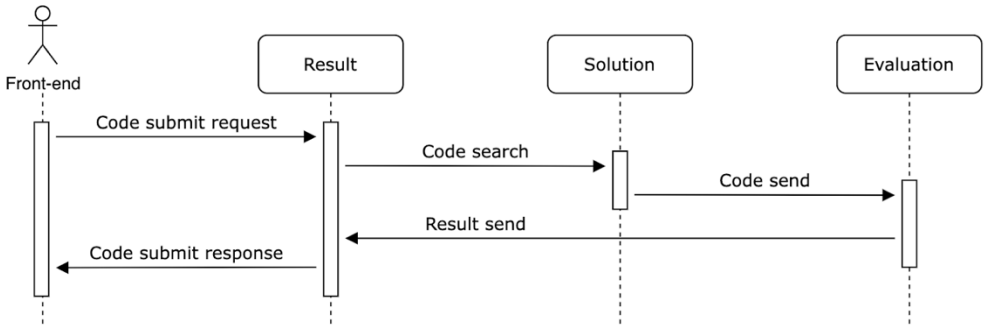
### 5.3.3. Evaluation System

*[Figure 18] Evaluation System Class Diagram*



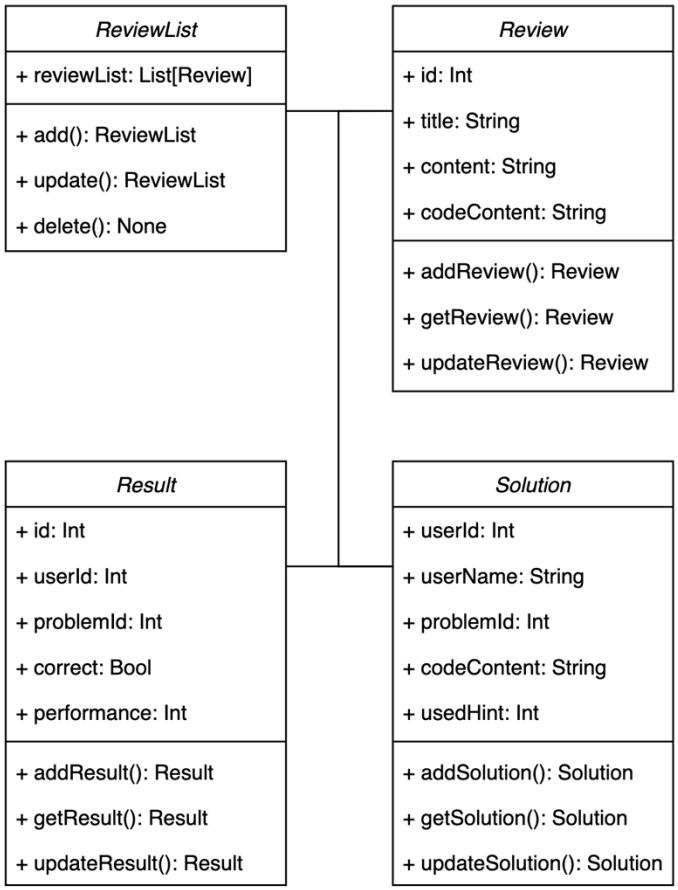

Result 는 사용자와 문제를 식별하는 id 정보를 포함한다. 해당 정보를 바탕으로 Evaluation 에 풀이코드를 보내고 Evaluation 내에서 execute() 메서드를 통해 정답 여부를 확인한다. 정답 여부에 대한 결과는 Result 의 correct 값으로 저장된다.

*[Figure 19] Evaluation System Sequence Diagram*



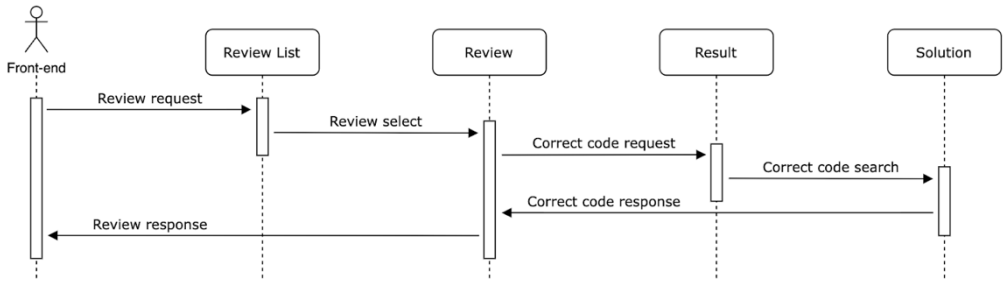
### 5.3.4. Review System

*[Figure 20] Review System Class Diagram*




Review LIst 는 사용자가 틀린 문제들에 대한 리스트이다. 각 Review 는 다른 사용자들의 코드를 참고하여 진행하기 때문에 Result 가 사용되며 Result 와 함께 코드가 저장되어 있는 Solution 도 사용된다.

*[Figure 21] Review System Sequence Diagram*



사용자가 Review 를 불러오는 것을 요청하면 Review List 에서 해당 문제에 대한 Review 를 선택한다. 선택된 Review 에 해당하는 문제 id 값을 Result 로 전달하고 Result 에서 문제 id 값에 해당하는 Solution 들을 performance 에 따라 정렬한 뒤에 가장 높은 performance 를 갖는 Solution 을 다시 Review 로 전달한다. 전달받은 정답 코드와 함께 사용자에게 선택된 문제에 대한 Review 를 전달한다.

# 6. Protocol Design

## 6.1. Objective

이 장에서는 각 sub system 들 사이에 사용된 프로토콜의 구조에 대해서 설명한다. 본 시스템을 제공하기 위한 환경은 Django 프레임워크를 사용해서 제공된다. 사용자의 코드와 같은 사용자 정보는 데이터베이스안에 저장되며 통신할 때 사용되는 데이터의 형태는 JSON 을 사용한다.

## 6.2. HTTP

HTTP(Hypertext Transfer Protocol)는 월드 와이드 웹(World Wide Web)에서 데이터를 전송하기 위해 사용되는 protocol 이다. HTTP 는 클라이언트와 서버 간의 통신을 담당하며, 요청(Request)과 응답(Response)의 형태로 이루어진다. 사용자가 코딩 교육 플랫폼의 웹 페이지를 로드하거나, 기능을 실행하려고 할 때 클라이언트는 서버에 HTTP 요청을 보낸다. GET 요청은 서버로부터 콘텐츠(HTML, CSS, JavaScript)를 가져올때 사용되며 POST 요청은 서버로 데이터를 보낼 때 사용된다. 서버는 클라이언트로부터의 요청을 받고, 해당 요청을 처리한 후 적절한 응답을 생성한다. 서버는 HTTP 응답을 통해 클라이언트에게 데이터를 전달하며 응답은 HTML, JSON, 파일 등의 형태로 클라이언트에게 전송될 수 있다.

## 6.3. RESTful API

RESTful API(Representational State Transfer API)는 웹 서비스에서 리소스를 조작하기 위한 architecture 스타일이다. REST 는 클라이언트와 서버간의 통신을 위한 규칙과 제약

조건을 정의하며, 리소스를 고유한 URI(Uniform Resource Identifier)로 식별한다. RESTful API 는 클라이언트와 서버 간의 통신을 단순하고 표준화된 방식으로 처리할 수 있도록 하며 이를 통해 다양한 플랫폼과 기기에서 동작하는 웹 서비스를 구축할 수 있다. 본 CodeCraft 서비스에서는 RESTful API 를 사용하여 사용자 등록 및 인증, 코딩 문제와 답안 관리, 사용자 점수와 진행 상황 관리등의 클라이언트와 서버 간의 효율적인 데이터 통신을 구현할 수 있다.

### 6.4. JSON

JSON(JavaScript Object Notation)은 데이터를 표현하기 위한 경량의 데이터 교환 형식이다. JSON 은 텍스트 기반의 format 으로, 사람과 기계가 모두 이해하기 쉬운 방식이다. 주로 웹 서비스에서 데이터를 전송하고 저장하는데 사용된다. 클라이언트는 사용자 입력이나 상태를 JSON 형식으로 변환하여 서버에 전송하고 서버는 해당 JSON 데이터를 수신하고 처리한다. 이를 통해 사용자 등록, 답안 제출, 점수 변화 등 다양한 데이터를 전송할 수 있다. 서버에서는 데이터베이스로부터 필요한 데이터를 조회한 후, 데이터를 JSON 형식으로 변환하여 클라이언트에게 전송한다. 클라이언트는 해당 JSON 을 수신하고 화면에 표시하거나 필요한 처리를 수행하며 이를 통해 사용자 정보, 코딩 문제, 점수 등 다양한 데이터를 수신할 수 있다.

### 6.5. Django

장고(Django)는 파이썬으로 작성된 무료 오픈 소스 웹 프레임워크이다. 장고는 웹 개발을 빠르고 쉽게 진행할 수 있도록 도와주는 도구와 기능을 제공한다. MVC 패턴을 기반으로 하여 모델(Model)에서는 데이터를 관리하고 비즈니스 로직을 처리한다. 뷰(View)는 사용자에게 정보를 표시하고 요청을 처리하며 컨트롤러는 Model 과 View 를 연결하여 데이터를 처리하고 흐름을 제어한다. 또한 장고는 객체 관계 매핑(Object-Relational Mapping)을 제공하여 데이터베이스와 상호작용을 쉽게 관리할 수 있다. 장고는 웹 개발에 필요한 많은 기능을 내장하고 있어 개발자가 웹 애플리케이션을 신속하게 구축할 수 있으며 확장성이 뛰어나고 다양한 기능을 추가할 수 있다.

## 6.6. API

### 6.6.1. Register

*[Table 1] Register API*

| API | Register | | |
|---|---|---|---|
| **Method** | POST | **URI** | /register |
| **Parameter** | name | 사용자 이름 | |
| | id | 사용자 아이디 | |
| | password | 사용자 비밀번호 | |
| | email | 사용자 이메일 | |
| **Request** | Authentication Token | | |
| **Status Code** | 1 | 회원가입 성공 | |
| | 0 | 회원가입 실패 | |

### 6.6.2. Log-in

*[Table 2] Log-in API*

| API | Log-in | | |
|---|---|---|---|
| **Method** | GET | **URI** | /login |
| **Parameter** | id | 사용자 아이디 | |
| | password | 사용자 비밀번호 | |
| **Request** | Authentication Token | | |
| **Status Code** | 1 | 로그인 성공 | |
| | 0 | 로그인 실패 | |

#### 6.6.3. Select Mode

*[Table 3] Select Mode API*

| **API** | **Select mode** | | |
|---|---|---|---|
| **Method** | GET | **URI** | /practice, /rank |
| **Parameter** | mode | 사용자가 설정한 모드 | |
| **Request** | Authentication Token | | |
| **Status Code** | 1 | 모드  선택 성공 | |
| | 0 | 모드 선택 실패 | |

#### 6.6.4. Select Map

*[Table 4] Select Map API*

| **API** | **Select map** | | |
|---|---|---|---|
| **Method** | GET | **URI** | /practice/map_number |
| **Parameter** | mode | 일반 모드 | |
| | map_nubmer | 맵 번호 | |
| **Request** | Authentication Token | | |
| **Status Code** | 1 | map 선택 성공 | |
| | 0 | map 선택 실패 | |

#### 6.6.5. Problem

*[Table 5] Problem API*

| API | Problem | | |
|---|---|---|---|
| **Method** | GET | **URI** | /problem |
| **Parameter** | problem_id | 문제 아이디 | |
| | problem_name | 문제 이름 | |
| | map_nubmer | 맵 번호 | |
| | deadline | 제출 기한 | |
| | requirements | 요구사항 | |
| **Request** | Authentication Token | | |
| **Status Code** | 1 | 문제 조회 성공 | |
| | 0 | 문제 조회 실패 | |

#### 6.6.6. Save Code

*[Table 6] Save Code API*

| API | Save code | | |
|---|---|---|---|
| **Method** | POST | **URI** | /code_saved |
| **Parameter** | problem_id | 문제 아이디 | |
| | problem_name | 문제 이름 | |
| | map_nubmer | 맵 번호 | |
| | code | 사용자가 작성한 코드 | |

| | | |
|---|---|---|
| | code_id | 사용자가 작성한 코드 아이디 |
| **Request** | Authentication Token | |
| **Status Code** | 1 | 코드 저장 성공 |
| | 0 | 코드 저장 실패 |

### 6.6.7. Load Saved Code

*[Table 7] Load Saved Code API*

| **API** | **Load saved code** | | |
|---|---|---|---|
| **Method** | GET | **URI** | /code_saved/problem_id |
| **Parameter** | problem_id | 문제 아이디 | |
| | code | 사용자가 작성한 코드 | |
| | code_id | 사용자가 작성한 코드 아이디 | |
| **Request** | Authentication Token | | |
| **Status Code** | 1 | 저장된 코드 불러오기 성공 | |
| | 0 | 저장된 코드 불러오기 실패 | |

### 6.6.8. Submit Code

*[Table 8] Submit Code API*

| **API** | **Submit code** | | |
|---|---|---|---|
| **Method** | POST | **URI** | /code_submitted |

| | | |
|---|---|---|
| **Parameter** | problem_id | 문제 아이디 |
| | code | 사용자가 작성한 코드 |
| | code_id | 사용자가 작성한 코드 아이디 |
| | input | 사용자가 작성한 코드에 대한 입력 |
| | result | 사용자가 작성한 코드의 실행결과 |
| **Request** | Authentication Token | |
| **Status Code** | 1 | 코드 저장 성공 |
| | 0 | 코드 저장 실패 |

### 6.6.9. Execute Code

*[Table 9] Execute Code API*

| **API** | **Execute code** | | |
|---|---|---|---|
| **Method** | POST | **URI** | /problem/problem_id |
| **Parameter** | problem_id | 문제 아이디 | |
| | code | 사용자가 작성한 코드 | |
| | code_id | 사용자가 작성한 코드 아이디 | |
| **Request** | Authentication Token | | |
| **Status Code** | 1 | 코드 실행 성공 | |
| | 0 | 코드 실행 실패 | |

### 6.6.10. Test Case

*[Table 10] Test Case API*

| API | Test case | | |
|---|---|---|---|
| **Method** | POST | **URI** | /problem/problem_id/testcase |
| **Parameter** | problem_id | 문제 아이디 | |
| | code | 사용자가 작성한 코드 | |
| | code_id | 사용자가 작성한 코드 아이디 | |
| | test_case | 문제 테스트 케이스 | |
| | result | 테스트 케이스 결과 | |
| **Request** | Authentication Token | | |
| **Status Code** | 1 | 테스트 케이스 결과 전달 성공 | |
| | 0 | 테스트 케이스 결과 전달 실패 | |

### 6.6.11. Select Hint

*[Table 11] Select Hint API*

| API | Select hint | | |
|---|---|---|---|
| **Method** | POST | **URI** | /problem/problem_id/hint |
| **Parameter** | problem_id | 문제 아이디 | |
| | hint_type | 힌트 타입 | |
| **Request** | Authentication Token | | |

| API | Select hint | |
|---|---|---|
| **Status Code** | 1 | 힌트 타입 선택 성공 |
| | 0 | 힌트 타입 선택 실패 |

### 6.6.12. Hint

*[Table 12] Hint API*

| API | Hint | | |
|---|---|---|---|
| **Method** | GET | **URI** | /problem/problem_id/hint |
| **Parameter** | problem_id | 문제 아이디 | |
| | hint_type | 힌트 타입 | |
| | hint_result | 힌트 결과 | |
| **Request** | Authentication Token | | |
| **Status Code** | 1 | 힌트 결과 전달 성공 | |
| | 0 | 힌트 결과 전달 실패 | |

### 6.6.13. Score

*[Table 13] Score API*

| API | Score | | |
|---|---|---|---|
| **Method** | GET | **URI** | /score |
| **Parameter** | id | 사용자 아이디 | |

| | | |
|---|---|---|
| | name | 사용자 이름 |
| | score | 사용자 점수 |
| **Request** | Authentication Token | |
| **Status Code** | 1 | 점수 조회 성공 |
| | 0 | 점수 조회 실패 |

#### 6.6.14. Ranking

*[Table 14] Ranking API*

| **API** | **Ranking** | | |
|---|---|---|---|
| **Method** | GET | **URI** | /rank |
| **Parameter** | id | 사용자 아이디 | |
| | name | 사용자 이름 | |
| | ranking | 사용자 랭킹 | |
| **Request** | Authentication Token | | |
| **Status Code** | 1 | 랭킹 조회 성공 | |
| | 0 | 랭킹 조회 실패 | |

#### 6.6.15. Reward

*[Table 15] Reward API*

| **API** | **Reward** | | |
|---|---|---|---|
| **Method** | GET | **URI** | /reward |

| | | |
|---|---|---|
| **Parameter** | reward | 보상 |
| | name | 사용자 이름 |
| | score | 사용자 점수 |
| **Request** | Authentication Token | |
| **Status Code** | 1 | 점수 조회 성공 |
| | 0 | 점수 조회 실패 |

#### 6.6.16. Review Note

*[Table 16] Review Note API*

| **API** | **Review note** | | |
|---|---|---|---|
| **Method** | GET | **URI** | /review |
| **Parameter** | problem_id | 오답노트 문제 아이디 | |
| | id | 사용자 아이디 | |
| | name | 사용자 이름 | |
| | code | 사용자가 작성한 코드 | |
| | code_id | 사용자가 작성한 코드 아이디 | |
| | input | 사용자가 작성한 코드에 대한 입력 | |
| | result | 사용자가 작성한 코드의 실행결과 | |
| **Request** | Authentication Token | | |
| **Status Code** | 1 | 오답노트 문제 조회 성공 | |

| | | |
|---|---|---|
| | 0 | 오답노트 문제 조회 실패 |

### 6.6.17. Review Note - User Code

*[Table 17] Review Note - User Code API*

| **API** | **Review note - user code** | | |
|---|---|---|---|
| **Method** | GET | **URI** | /review/user_id |
| **Parameter** | problem_id | 오답노트 문제 아이디 | |
| | id | 사용자 아이디 | |
| | name | 사용자 이름 | |
| | user_id | 다른 사용자 아이디 | |
| | code | 다른 사용자가 작성한 코드 | |
| | code_id | 다른 사용자가 작성한 코드 아이디 | |
| **Request** | Authentication Token | | |
| **Status Code** | 1 | 다른 user 풀이 조회 성공 | |
| | 0 | 다른 user 풀이 조회 실패 | |

### 6.6.18. Add To Review Note

*[Table 18] Add To Review Note API*

| **API** | **Add to review note** | | |
|---|---|---|---|
| **Method** | POST | **URI** | /review |

| | | |
|---|---|---|
| **Parameter** | problem_id | 오답노트 문제 아이디 |
| | id | 사용자 아이디 |
| | name | 사용자 이름 |
| | code | 사용자가 작성한 코드 |
| | code_id | 사용자가 작성한 코드 아이디 |
| | input | 사용자가 작성한 코드에 대한 입력 |
| | result | 사용자가 작성한 코드의 실행결과 |
| **Request** | Authentication Token | |
| **Status Code** | 1 | 오답노트 문제 추가 성공 |
| | 0 | 오답노트 문제 추가 실패 |

### 6.6.19. Q&A - User Question

*[Table 19] Q&A - User Question API*

| **API** | **Q&A - user question** | | |
|---|---|---|---|
| **Method** | POST | **URI** | /review/q&a |
| **Parameter** | problem_id | 오답노트 문제 아이디 | |
| | id | 사용자 아이디 | |
| | name | 사용자 이름 | |
| | question | 사용자 질문 | |
| **Request** | Authentication Token | | |

| | | |
|---|---|---|
| **Status Code** | 1 | Q&A 질문 전달 성공 |
| | 0 | Q&A 질문 전달 실패 |

#### 6.6.20. Q&A - GPT Answer

*[Table 20] Q&A - GPT Answer API*

| **API** | **Q&A - user question** | | |
|---|---|---|---|
| **Method** | GET | **URI** | /review/q&a |
| **Parameter** | problem_id | 오답노트 문제 아이디 | |
| | id | 사용자 아이디 | |
| | name | 사용자 이름 | |
| | question | 사용자 질문 | |
| | answer | GPT 답변 | |
| **Request** | Authentication Token | | |
| **Status Code** | 1 | GPT 답변 전달 성공 | |
| | 0 | GPT 답변 전달 실패 | |

## 7. Database Design

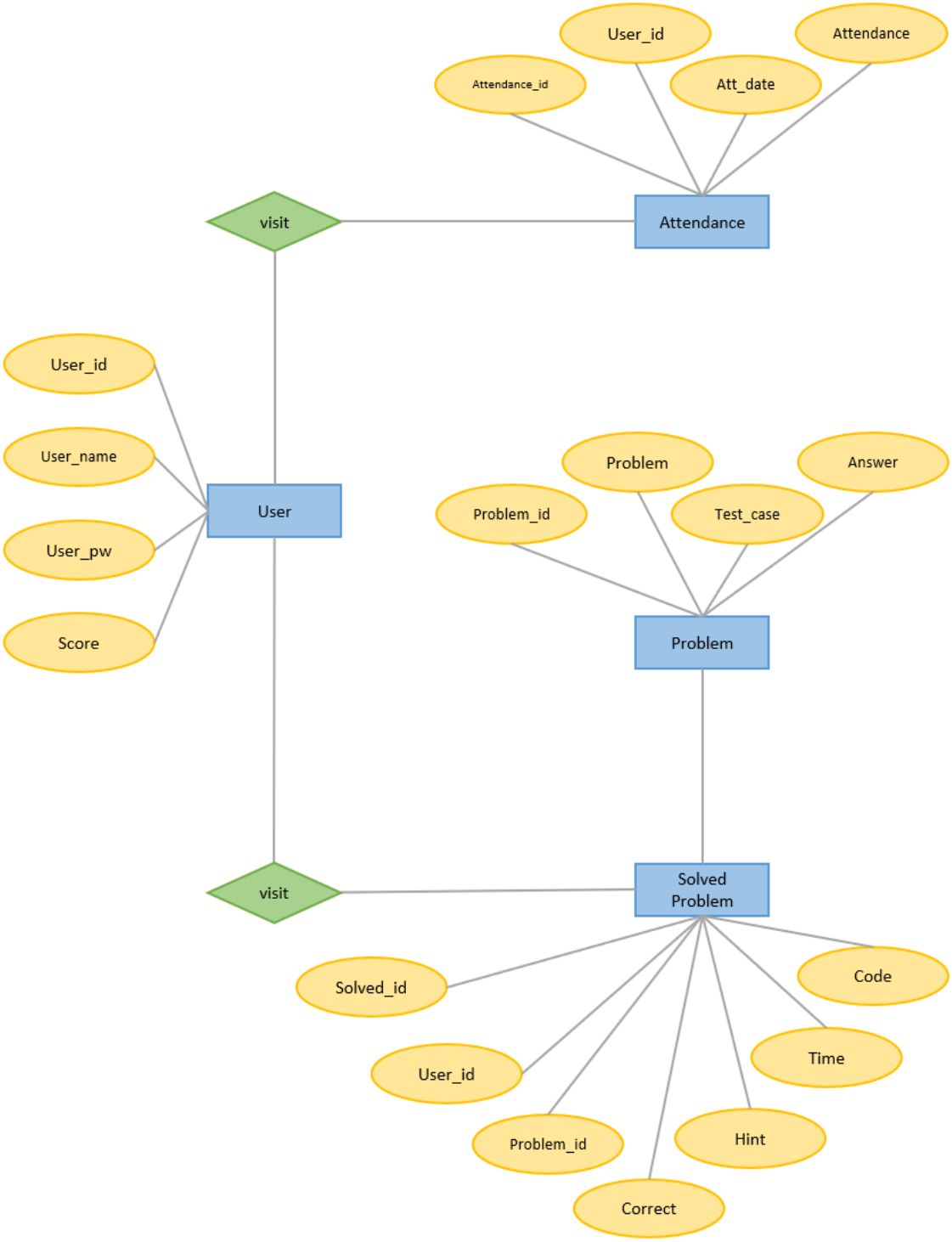
### 7.1. Objective

7 장에서는 시스템 데이터 구조와 해당 구조가 어떻게 Database 로 구현되었는지에 대해 설명한다. 먼저 ER Diagram 을 통해 entity 들 간의 관계를 식별하고 후에 Schema 및 SQL Data Definition Language(DDL)을 작성한다.

## 7.2. Database Entities

### 7.2.1. ER Diagram

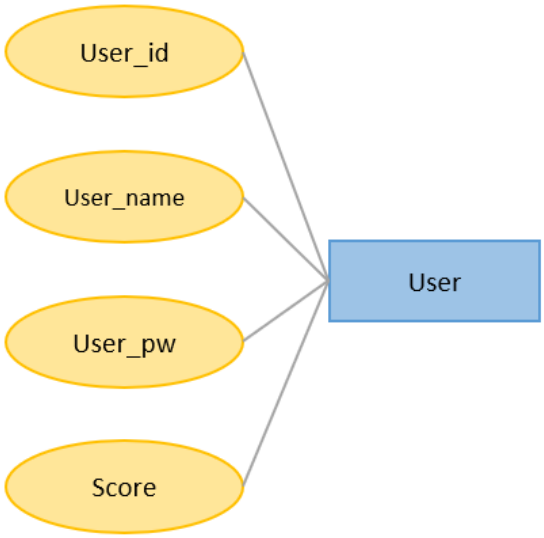
*[Figure 22] ER Diagram*



### 7.2.2. Entities

#### 7.2.2.1. User

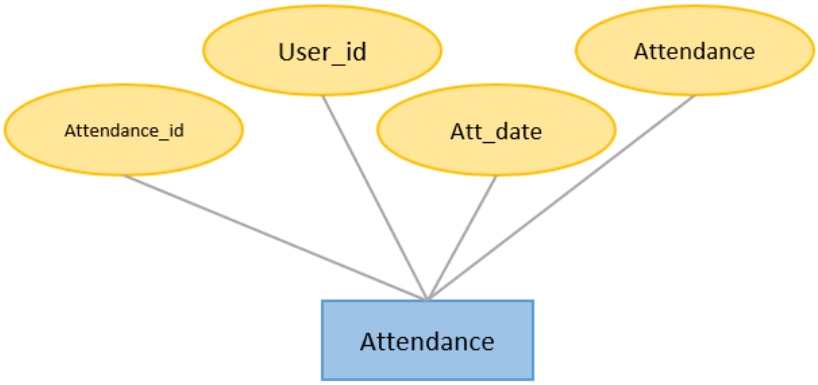
*[Figure 23] User Entity*




User Entity 는 CodeCraft 의 사용자들을 이야기한다. User Entity 는 User_id, Name, Password, Score 의 attribute 를 가지며, User_id 가 primary key 이다. Score 를 제외한 attribute 들은 사용자들이 회원가입 시 기입한 정보들이며 Score 는 각 user 가 획득한 점수를 나타낸다.

#### 7.2.2.2. Attendance

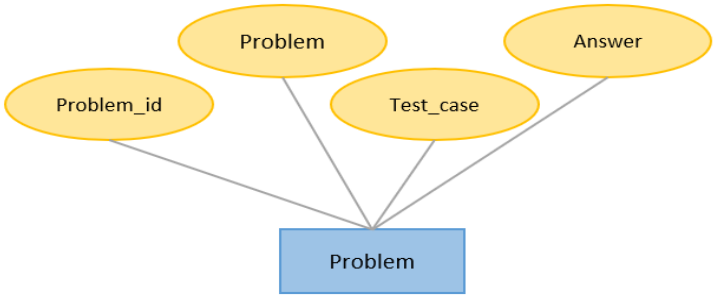
*[Figure 24] Attendance Entity*



Attendance Entity 는 CodeCraft 의 사용자들이 언제 접속했는지를 알기 위한 Entity 이며 이를 통해 사용자들의 학습을 장려하고자 한다. 해당 entity 는 User_id, Date, Attendance 의 attribute 를 가지며, User_id 가 primary key 이다. Attendance 의 경우 각 user 가 해당 Date 에 접속했는지 여부를 확인하기 위한 attribute 이다.

#### 7.2.2.3. Problem

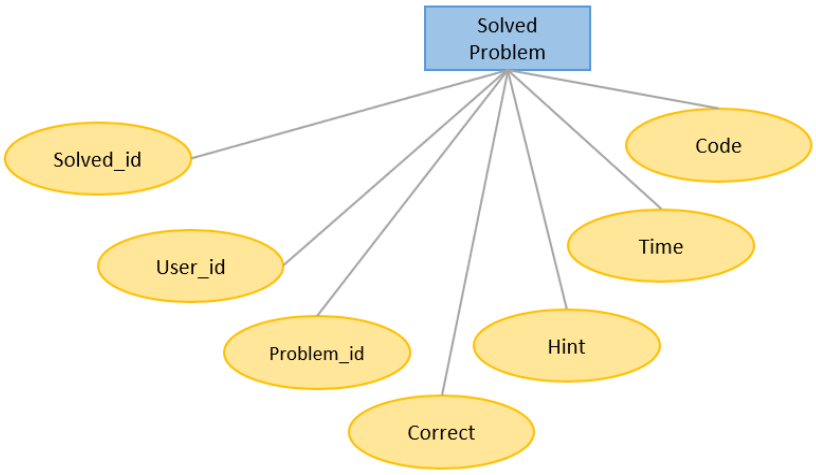
*[Figure 25] Problem Entity*




Problem Entity 는 CodeCraft 에서 제공하는 Problem 들에 대한 Entity 이며 Problem_id, Problem, Test_case, Answer 의 attribute 를 가진다. Problem_id 가 primary key 이며, 각 문제와 문제의 정답, test_case 들을 포함하고 있다.

#### 7.2.2.4. Solved Problem

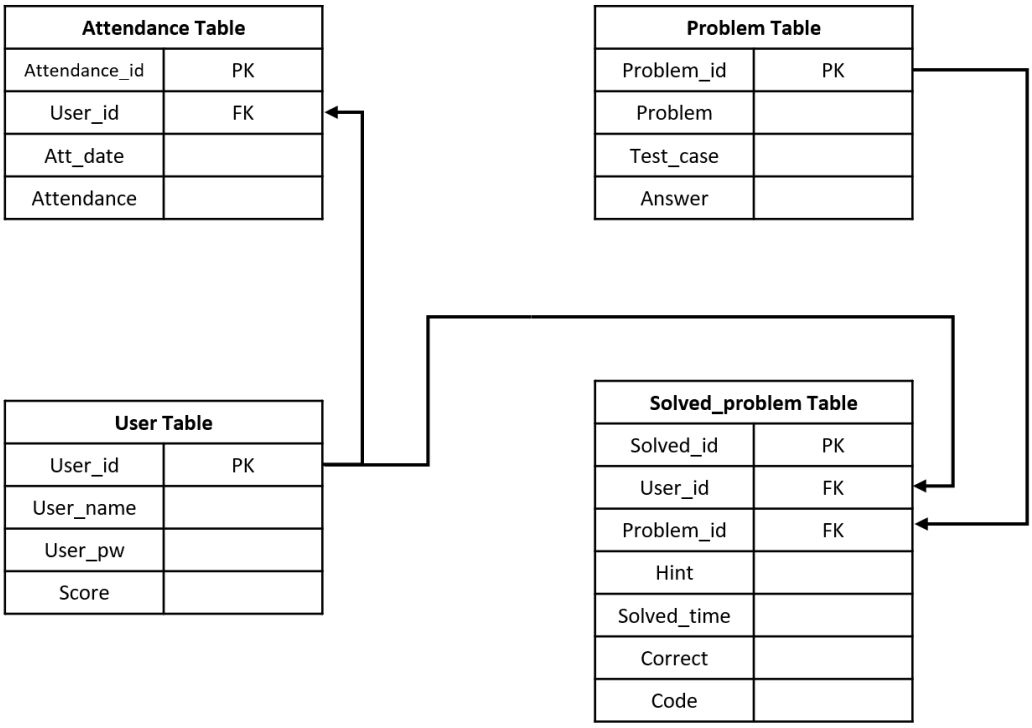
*[Figure 26] Solved Problem Entity*



Solved Problem Entity 의 경우 각 사용자가 푼 문제들에 대한 정보를 담고 있으며 Attribute 로는 User_id, Problem_id, Correct, Hint, Time, Code 가 있다. 어떤 사용자가 어떤 문제를 풀었는지 그리고 정답 여부, Hint 사용 횟수, 문제를 해결하는 데 소요된 시간, 작성한 code 등에 대한 정보가 각 attribute 에 저장됩니다.

## 7.3. Schema

*[Figure 27] Schema*




## 7.4. SQL Data Definition Language

### 7.4.1. User

```
CREATE TABLE User (
    user_id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
    user_name INT Not Null,
    user_pw INT Not NULL,
    score INT DEFAULT 0
)
```

#### 7.4.2. Attendance

```
CREATE TABLE Attendance (
    attendance_id INTEGER NOT NULL PRIMARY KEY AUTO_INCREMENT,
    user_id INTEGER NOT NULL,
    att_date DATE Not Null,
    attendance BOOLEAN DEFAULT False
)
```

#### 7.4.3. Problem

```
CREATE TABLE Problem (
    problem_id INTEGER NOT NULL PRIMARY KEY AUTO_INCREMENT,
    problem VARCHAR(65535) NOT NULL,
    test_case_1 VARCHAR(255),
    test_case_2 VARCHAR(255),
    test_case_3 VARCHAR(255),
    answer VARCHAR(255)
)
```

#### 7.4.4. Solved problem

```
CREATE TABLE Solved_problem (
    solved_id INTEGER NOT NULL PRIMARY KEY AUTO_INCREMENT,
    user_id INTEGER NOT NULL,
    problem_id INTEGER NOT NULL,
    hint INTEGER DEFAULT 3,
    solved_time TIME,
    correct BOOLEAN,
    code VARCHAR(65535)
)
```

## 8. Plan

### 8.1. Testing

#### 8.1.1. Objective

본 장에서는 요구사항 명세서에서 기술한 시나리오를 바탕으로 통합 시스템이 동작하는 지 확인한다. 유닛 테스트, 통합 테스트 과정을 거쳐 시스템 출시 전에

발생할 수 있는 잠재적인 오류 혹은 결함을 찾아내고자 한다. 이를 위해 세가지 관점으로 나누어 테스트 과정을 설계하며, 세부 테스트 케이스를 설명한다.

### 8.1.2. Testing Policy

#### 8.1.2.1. Development Testing

본 시스템 개발 과정 중에 광범위하게 발생할 수 있는 리스크와 비용을 줄이는 것을 목표로 하며, 소프트웨어가 예상대로 기능하는지 확인하고, 시스템을 구성하는 서브 시스템 혹은 컴포넌트 단위의 독립적인 유닛 테스트를 먼저 진행한다. 유닛 테스트는 개별 모듈이나 기능들이 예상대로 동작하는지 확인한다. 각 함수의 입력과 출력을 정확한 연산을 수행하는지 중점적으로 확인한다. 이와 더불어 구현 컴포넌트의 유지보수성을 높이고 잠재적인 결함을 발견하기 위해 상호 간의 코드 리뷰를 진행한다. 유닛 테스트 진행 후, 각 컴포넌트들을 순차적으로 통합하여 통합 테스트를 진행한다. 통합 테스트는 여러 모듈이 결함될 때 서로 올바르게 상호작용하는지 확인한다. 이는 각 개별 시스템 간의 인터페이스가 적절하게 작동하는지 확인한다. 기능적인 항목 뿐만 아니라 비기능적인 성능, 보안, 안정성 등을 검증해야 한다. 이를 위해 정적 코드 분석, 데이터 흐름 분석 등을 수행할 수 있다.

#### 8.1.2.2. Release Testing

본 시스템이 사용자 환경에서 안정적으로 작동하고, 모든 요구사항이 충족되며 예상되는 품질 표준을 충족하는지 확인한다. 또한 새로운 버전 출시에 따른 배포 과정이 정상적으로 수행될 수 있는 지를 검증한다. 시스템이 최신 버전의 온전한 상태로 배포되어 작동에 어떠한 하자가 없어야 한다. 일반적으로 목표로 하는 기능을 포함한 알파 테스트를 거쳐, 그 과정에서 발견한 피드백 및 결함을 반영하여 베타 테스트를 진행한다. 알파 테스트의 경우 개발 진행 조직 내부에서 진행하며, 베타 테스트의 경우 제한된 사용자에게 공개하여 진행한다.

#### 8.1.2.3. User Testing

본 시스템이 사용자의 기대와 요구사항을 충족하는 확인하는 작업으로, 예상 사용자가 사용하는 과정을 시험함으로써, 본 시스템이 모든 요구사항 명세를 충족하는 지 검증한다. 이를 위해 동시에 접속하는 유저의 수를 포함하는 시스템 사용 시나리오를 작성하고, 이 시나리오에서 CodeCraft 시스템이 모든 Use Case 를 충족할 수 있는지를 검증한다. 검증 결과를 바탕으로 CodeCraft 시스템의 베타 버전의 유용성을 평가하고, 최종 결과물로서 실제 환경에 Deployment 될 것인지를 결정한다

### 8.1.3. Test Case

#### 8.1.3.1. 로그인

*[Table 21] 로그인 테스트*

| Element | Description |
|---|---|
| Summary | CodeCraft 시스템을 이용하기 앞서 로그인을 진행한다. |
| Pre-Conditions | ● 사용자는 사전에 회원가입을 완료했다. |
| Test Steps | 1. 사용자는 본인 계정으로 로그인을 진행<br>2. 서버는 사용자의 정보를 토대로 사용자를 인증한 값을 반환 |
| Expected Results | ● 랜딩 페이지 상의 프로필 정보가 보여지게 된다. |

#### 8.1.3.2. 연습 모드 선택

*[Table 22] 연습 모드 선택 테스트*

| Element | Description |
|---|---|
| Summary | 사용자는 메인페이지 중앙에 위치한 연습 모드 버튼을 클릭하여 연습 모드 레벨을 확인하고, 문제를 선택할 수 있다. |
| Pre-Conditions | ● 사용자는 로그인을 완료한 상태이다.<br>● 사용자의 인터넷 연결이 원활하다. |
| Test Steps | 1. 메인페이지에서 '연습 모드' 버튼을 클릭<br>2. 연습 모드 페이지에서 지금까지 푼 문제가 잘 마킹 되어있는지 확인<br>3. 연습 모드 페이지에서 현재 나의 단계에 해당하는 '문제' 버튼을 클릭 했을 때, 페이지가 잘 이동하는지 확인<br>4. 다른 '문제' 버튼을 클릭했을 때, 페이지가 이동되지 않는지 확인 |
| Expected Results | ● 사용자는 풀고자 하는 연습 모드 페이지로 이동한다. |

#### 8.1.3.3. 연습 모드 풀이

*[Table 23] 연습 모드 풀이 테스트*

| Element | Description |
|---|---|
| Summary | 사용자는 연습 모드 페이지에서 문제 버튼을 클릭하여 문제 풀이를 진행할 수 있다. |
| Pre-Conditions | ● 사용자는 로그인을 완료한 상태이다.<br>● 사용자의 인터넷 연결이 원활하다.<br>● 연습 모드 페이지에 문제가 등록되어 있다.<br>● 사용자는 연습 모드 페이지에서 문제를 선택하였다. |
| Test Steps | 1. 연습 모드 선택 페이지에서 '문제' 버튼을 클릭하여 문제 풀이 페이지로 이동<br>2. 문제 설명 및 입출력 예시가 올바르게 표시되는지 확인<br>3. '힌트' 버튼 클릭 시 올바르게 작동하는지 확인<br>4. 코드 작성이 원활하게 진행되는지 확인 |
| Expected Results | ● 사용자는 코드를 작성하여 제출할 수 있다. |

#### 8.1.3.4. 실전 모드 선택

*[Table 24] 실전 모드 선택 테스트*

| Element | Description |
|---|---|
| Summary | 사용자는 메인페이지 중앙에 위치한 실전 모드 버튼을 |

| | |
|---|---|
| | 클릭하여 실전 모드 페이지로 연결된다. |
| Pre-Conditions | • 사용자는 로그인을 완료한 상태이다.<br>• 사용자의 인터넷 연결이 원활하다. |
| Test Steps | 1. 메인페이지에서 '실전 모드' 버튼을 클릭<br>2. 화면 중앙에 실전 모드에 대한 설명이 올바르게 표시되는지 확인<br>3. '시작하기' 버튼을 클릭했을 때 문제 풀이 페이지로 올바르게 연결되는지 확인 |
| Expected Results | • 사용자는 실전 모드 페이지로 이동한다. |

#### 8.1.3.5. 실전 모드 풀이

*[Table 25] 실전 모드 풀이 테스트*

| Element | Description |
|---|---|
| Summary | 사용자는 실전 모드 페이지에서 문제 버튼을 클릭하여 문제 풀이를 진행할 수 있다. |
| Pre-Conditions | • 사용자는 로그인을 완료한 상태이다.<br>• 사용자의 인터넷 연결이 원활하다.<br>• 실전 모드 페이지에 문제가 등록되어 있다.<br>• 사용자는 실전 모드 페이지에서 시작하기 버튼을 클릭했다. |
| Test Steps | 1. 사용자가 작성하는 코드가 제대로 입력되는지 확인 |

| | |
|---|---|
| | 2. 제한시간 막대가 제대로 작동하는지 확인<br>3. 제한시간이 끝난 이후 문제풀이가 불가한지 확인 |
| Expected Results | ● 사용자는 코드를 작성하여 제출할 수 있다. |

#### 8.1.3.6. 코드 제출

*[Table 26] 코드 제출 테스트*

| Element | Description |
|---|---|
| Summary | 문제 풀이 페이지에서 코드 작성을 완료한 사용자는 제출 버튼을 클릭하여 코드를 저장하고 채점을 진행할 수 있다. |
| Pre-Conditions | ● 사용자는 로그인을 완료한 상태이다.<br>● 사용자의 인터넷 연결이 원활하다.<br>● 연습 & 실전 모드 페이지에 문제가 등록되어 있다.<br>● 사용자는 코드 작성을 완료한 상태이다. |
| Test Steps | 1. 문제 풀이 페이지에서 '제출' 버튼 클릭<br>2. 사용자의 코드의 정답여부가 올바르게 표시되는지 확인<br>3. 일반 모드일 경우에 힌트 개수에 따라 결과가 올바르게 표시되는지 확인 |
| Expected Results | ● 사용자는 푼 문제에 대한 정답 여부를 확인할 수 있다. |

#### 8.1.3.7. 오답노트 페이지 이동

*[Table 27] 오답노트 페이지 이동 테스트*

| Element | Description |
|---|---|
| Summary | 사용자는 메인페이지 좌측에 위치한 오답노트 버튼을 클릭하여 오답노트 페이지로 연결된다. |
| Pre-Conditions | • 사용자는 로그인을 완료한 상태이다.<br>• 사용자의 인터넷 연결이 원활하다. |
| Test Steps | 1. 오답노트 버튼을 클릭<br>2. '오답노트' 버튼을 클릭했을 때 오답노트 페이지로 올바르게 연결되는지 확인 |
| Expected Results | • 오답노트 문제 페이지로 이동한다. |

#### 8.1.3.8. 오답노트 풀이

*[Table 28] 오답노트 풀이 테스트*

| Element | Description |
|---|---|
| Summary | 사용자는 오답노트 페이지에서 문제 버튼을 클릭하여 오답 문제를 다시 풀 수 있다. |
| Pre-Conditions | • 사용자는 로그인을 완료한 상태이다.<br>• 사용자의 인터넷 연결이 원활하다.<br>• 사용자는 사전에 문제 풀이를 1 회 이상 완료했다.<br>• 사용자는 사전에 오답 제출을 1 회 이상 완료했다. |
| Test Steps | 1. 오답노트 목록에서 특정 '문제' 버튼 클릭 |

| | |
|---|---|
| | 2. 첫번째 박스에 사용자가 마지막으로 작성한 코드가 올바르게 표시되는지 확인<br>3. 두번째 박스에 다른 사용자의 정답 코드가 올바르게 표시되는지 확인<br>4. 두번째 박스에 다른 사용자의 정답 코드가 존재하지 않을 경우 공백으로 표시되는지 확인<br>5. 세번째 박스에 chatGPT 가 수정한 코드가 올바르게 표시되는지 확인<br>6. 화면 상단에 'Q&A' 텍스트 입력폼에 질문 입력<br>7. 질의응답이 원활하게 진행되는지 확인 |
| Expected Results | ● 사용자는 오답노트에서 틀린문제를 복습할 수 있다. |

## 8.2. Development

### 8.2.1. Objective

본 장에서는 시스템 구현을 위한 기술 항목, 개발 환경과 개발 과정에 있어서 지켜져야 하는 제약 사항을 기술한다.

### 8.2.2. Frontend Environment

#### 8.2.2.1. Java Server Pages

*[Figure 28] JSP Logo*

JSP 는 JAVA 를 이용한 서버 사이드 스크립트 언어로 HTML 코드에 JAVA 코드를 넣어 동적 웹페이지를 생성하는 웹 어플리케이션 도구이다. JSP 가 실행되면 자바 서블릿(Servlet)으로 변환되며 웹 어플리케이션 서버에서 동작되면서 필요한 기능을 수행하고 그렇게 생성된 데이터를 웹페이지와 함께 클라이언트로 응답하게 된다.

#### 8.2.2.2. HyperText Markup Language

*[Figure 29] HTML Logo*


HTML 은 웹 페이지를 위한 지배적인 마크업 언어이며 W3C 가 HTML 과 CSS 표준의 공동 책임자이다. HTML 을 통해 제목, 단락, 목록 등 본문을 위한 구조적 의미를 나타내는 것뿐만 아니라 링크, 인용과 그 밖의 항목으로 구조적 문서를 만들 수 있는 방법을 제공받을 수 있다. 이는 태그로 되어있는 HTML 요소 형태로 작성되며 웹브라우저와 같은 HTML 처리 장치의 행동에 영향을 주는 자바스크립트를 포함하거나 불러올 수 있다.

#### 8.2.2.3. Django

*[Figure 30] Django Logo*

Django 는 Python 기반의 오픈소스 웹 애플리케이션을 위한 프레임워크이다. 웹 서버를 개발함에 있어서 상당 부분을 추상화하여 비교적 수월하게 개발할 수 있고 이를 통해 짧은 시간 주기의 개발이 가능하면서도 준수한 성능을 보여준다. 더불어 확장성을 고려하여 설계되어 다양한 추가 기능 및 미들웨어 적용이 가능하다. 본 시스템 제약사항으로 Python 기반으로 개발되어야 한다는 점과 추후 확장 가능하다는 점, 그리고 범용적으로 이용될 수 있다는 점을 고려하여 Django 를 활용한다.

### 8.2.3. Backend Environment

#### 8.2.3.1. Github

*[Figure 31] GitHub Logo*


Github 는 Git 을 이용한 소프트웨어 개발 및 버전 관리를 위해 만들어졌다. Git 의 분산 제어 및 소스 코드 관리(SCM)기능을 제공한다. 접근 제어와 버그 추적, 기능 요청, 작업 관리, 지속적인 통합 등 Management 를 향상시켜준다.

#### 8.2.3.2. MySQL

*[Figure 32] MySQL Logo*

MySQL 은 오픈소스 관계형 데이터 베이스 관리 시스템이다. 이들은 다중 스레드, 다중 사용자 형식의의 구조 질의어 형식의 데이터 베이스 관리 시스템으로 오라클에서 관리 및 지원하고 있다.

#### 8.2.3.3. Postman

*[Figure 33] Potman Logo*


Postman 은 API 개발을 도와주는 플랫폼이다. Postman 을 통해 서버가 개발 API 를 테스트 할 수 있고 더 나아가 변수 및 환경, request 설명, 테스트 및 사전 요청에 필요한 스크립트 작성등을 할 수 있는 프레임 워크이다.

## 9. Supporting Information

### 9.1. Software design specification

본 소프트웨어 디자인 명세서는 IEEE 권장 사항에 맞추어 작성되었다 (IEEE Standard for Information Technology Systems Design Software Design Descriptions, IEEE-Std-830). 다만 본 시스템의 설계 사항을 용이하게 파악할 수 있도록 본래의 양식에서 일부 추가되거나 제외된 항목이 있다

### 9.2. Document history

| Date | Version | Description | Writer |
|---|---|---|---|
| 05/06 | 0.1 | Style (IEEE-Std 830) | - |
| 05/08 | 1.1 | 1 작성 | 신새별 |
| 05/08 | 1.2 | 2.1 작성 | 이하은 |
| 05/08 | 1.3 | 2.2, 2.3 작성 | 신새별 |
| 05/11 | 1.4 | 3 작성 | 박주봉 |
| 05/11 | 1.5 | 4.2.3 작성 | 이하은 |
| 05/11 | 1.6 | 4.1, 4.2.1 작성 | 신새별 |
| 05/11 | 1.7 | 4.2.4 작성 | 이하은 |
| 05/11 | 1.8 | 4.2.2, 4.2.4 작성 | 신새별 |
| 05/15 | 1.9 | 6 작성 | 김민성 |
| 05/15 | 1.10 | 7 작성 | 박주봉 |
| 05/15 | 1.11 | 8.1.2 작성 | 이하은 |
| 05/18 | 1.12 | 5 작성 | 최현진 |
| 05/18 | 1.13 | 8.2.1 작성 | 신새별 |
| 05/18 | 1.14 | 8.1.3 작성 | 이하은, 최현진 |
| 05/18 | 1.15 | 8.2.2 작성 | 김민성 |
| 05/19 | 1.16 | 9 작성 | 박주봉 |
| 05/20 | 1.17 | 수정 및 검토 | 전원 |