Software Design Specification

디자인 명세서

Step with me

강은비, 박지연, 신윤성, 이일규, 이채헌, 정정환

SWE3002-41

Team 5

1 Purpose

1.1 Readership

본 문서는 다음과 같은 독자들을 위해 작성되었다. 본 시스템의 개발자들, 의뢰인, 사용자들이다. 시스템 개발자는 Frontend, Backend 개발자와 교육 콘텐츠를 제공하는 개발자 및 LLM 개발자도 포함된다.

1.2 Scope

본 문서는 Step with me 시스템의 설계를 다룬다. 시스템의 전체적인 구조뿐만 아니라 Frontend, Backend, 그리고 Database 설계의 세부 사항을 정의하며, 이를 기반으로 추후 진행될 개발 및 테스트 계획을 지원한다. 또한, 시스템의 주요 기능과 상호작용 방식을 기술하여 구현 과정의 명확성을 확보한다.

1.3 Objectives

이 문서의 목적은 Step with me 시스템의 기술적 설계를 상세히 설명함으로써, 개발자와 이해관계자들이 시스템의 구조와 설계 의도를 명확히 이해하도록 돕는 것이다. 이를 통해 개발 과정에서 오해를 줄이고, 효율적인 협업과 구현이 가능하도록 지원한다.

1.4 Document Structure

1) Purpose : 본 문서의 목적, 예상 독자 및 문서의 구조에 대해 설명한다.
2) Introduction : 시스템 설계에 사용된 다이어그램, 툴, 참고 자료에 대해 설명한다.
3) System architecture - Overall : 시스템의 전반적인 구조를 다이어그램으로 설명한다.
4) System architecture - Frontend : Frontend 시스템의 전반적인 구조를 다이어그램으로 설명한다.
5) System architecture - Backend : Backend 시스템의 전반적인 구조를 다이어그램으로 설명한다.
6) Protocol design : 클라이언트와 서버의 커뮤니케이션을 프로토콜 디자인으로 설명한다.

7) Database design : Database 시스템의 전반적인 구조를 다이어그램으로 설명한다.
8) Testing plan : 시스템의 테스트 계획을 설명한다.
9) Development plan : 설계를 토대로 시스템의 구현을 위한 계획과 환경을 설명한다.

2 Introduction

2.1 Objectives

해당 섹션에서는 시스템의 설계에 사용된 다이어그램, 툴을 설명하고, 프로젝트의 범위를 설명한다. 또한 문서 작성에 참고한 참조 자료 등을 나열한다.

2.2 Applied Diagrams

2.2.1 Used Tools

1. Figma/FigJam : 웹 기반 UI/UX 디자인 및 프로토타이핑 도구인 Figma/FigJam의 디자인 기능을 통해 다이어그램을 작성한다.
2. Microsoft PowerPoint : 발표용 자료 제작 도구인 PowerPoint의 도형 삽입 기능을 통해 다이어그램을 작성한다.

2.2.2 Class Diagram

Class diagram은 시스템을 구성하는 클래스와, 그 클래스 간의 관계로 시스템의 정적인 구조를 표현한다. 클래스는 이름, 클래스가 갖는 속성(attribute), 클래스가 수행할 수 있는 동작이나 기능을 정의하며, 객체의 행동을 구현하는 메소드(method)로 구성된다.

2.2.3 Sequence Diagram

Sequence diagram은 시스템의 actor와 객체 간의 상호작용의 순서를 표현한다. 여기에서 actor는 시스템과 상호작용하는 사용자, 외부 시스템 등의 개체(entity)이다. 시스템 내, 외부의 객체와 어떤 데이터를 어떻게 주고받는지를 시간 순서대로 화살표를 통해 표시한다.

2.2.4 Use Case Diagram

Use Case Diagram은 시스템에서 제공해야 하는 기능 및 서비스와 actor 사이의 상호작용을 표현한다. 각 Use case는 actor가 시스템을 어떻게 활용하는지에 대한 사용 사례이다. 여기에서 actor는 사용자가 될 수도 있고, 다른 시스템이 될 수도 있다. 외부에서 본 시스템의 기능을 표현하기 때문에,

실제 내부의 비즈니스 로직이 아닌, 사용자가 수행하는 기능을 파악하고자 할 때 작성한다.

2.2.5 Entity Relationship Diagram

ER diagram은 요구사항으로부터 데이터베이스를 설계할 때 활용되는 표현 방식으로, 개체(entity)와 개체의 속성(attribute), 그리고 개체 간의 관계를 통해 데이터베이스의 구조를 표현한다. 개체는 표현할 정보를 가지고 있는 데이터의 주요 대상이며, 독립적으로 존재할 수 있는 실체이고, 속성은 개체가 가지는 구체적인 특성이나 정보를 나타내는 요소이다.

2.3 Project Scope

본 문서에서 설계하는 Step with me는 프로그래밍을 공부하는 사용자들이 LLM을 통한 맞춤형 환경 속에서 학습을 진행하도록 돕기 위한 프로젝트이다. 사용자들이 LLM과의 단계별 학습을 통해 효과적으로 프로그래밍 능력을 기르고, 자기주도 학습 능력을 고양시키고자 한다.

2.4 References

- IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements
- Guide to the Software Engineering Body of Knowledge (SWEBOK Guide), Version 4.0
- OMG Unified Modeling Language TM (OMG UML) Version 2.5
- CloudFlare 학습센터 https://www.cloudflare.com/ko-kr/learning/ssl/what-is-https/
- Next.js Docs https://nextjs.org/docs/pages/building-your-application/routing/api-routes
- Guide to Preparing the SYSTEM TEST SPECIFICATION Version 3.30.05 © R. Buckley, California State University, Sacramento
- Software Testing and Analysis: Process, Principles and Techniques

3 System Architecture - Overall

3.1 Objectives

해당 섹션에서는 Step with me의 frontend부터 backend까지의 전반적인 시스템 구성에 대한 개요를 제공한다.

3.2 System Organization

Step with me는 클라이언트 - 서버 패턴 기반으로 설계되어, frontend와 backend로 나뉜다. Frontend는 사용자와의 직접적인 상호작용을 담당하는 역할로, 사용자 입력을 받아 이를 서버로 전달하거나, 서버로부터 수신한 데이터를 시각화하여 사용자에게 제공한다. 사용자가 입력값을 작성하거나 특정 요청을 보내면, frontend는 이를 backend로 전달한다. backend는 이러한 요청을 처리하는 핵심 로직과 데이터 관리 기능을 담당하며 frontend로부터 요청을 받아 작업을 수행하고, 필요한 경우 데이터베이스와 연동하여 결과 데이터를 생성한 뒤, 이를 frontend로 반환한다. 이 과정에서 backend와 frontend는 TLS 보안이 적용된 HTTPS를 통해 통신한다.

3.2.1 System Diagram

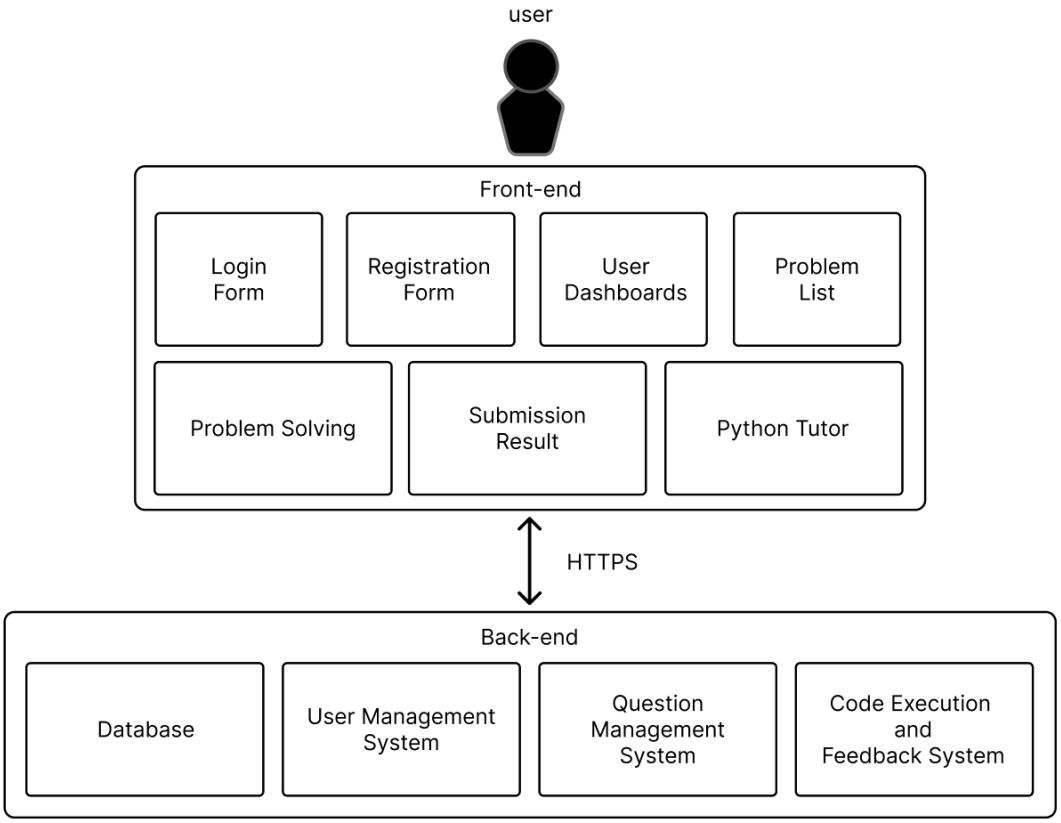

Figure 3.1: System diagram

3.3 Use Case Diagram

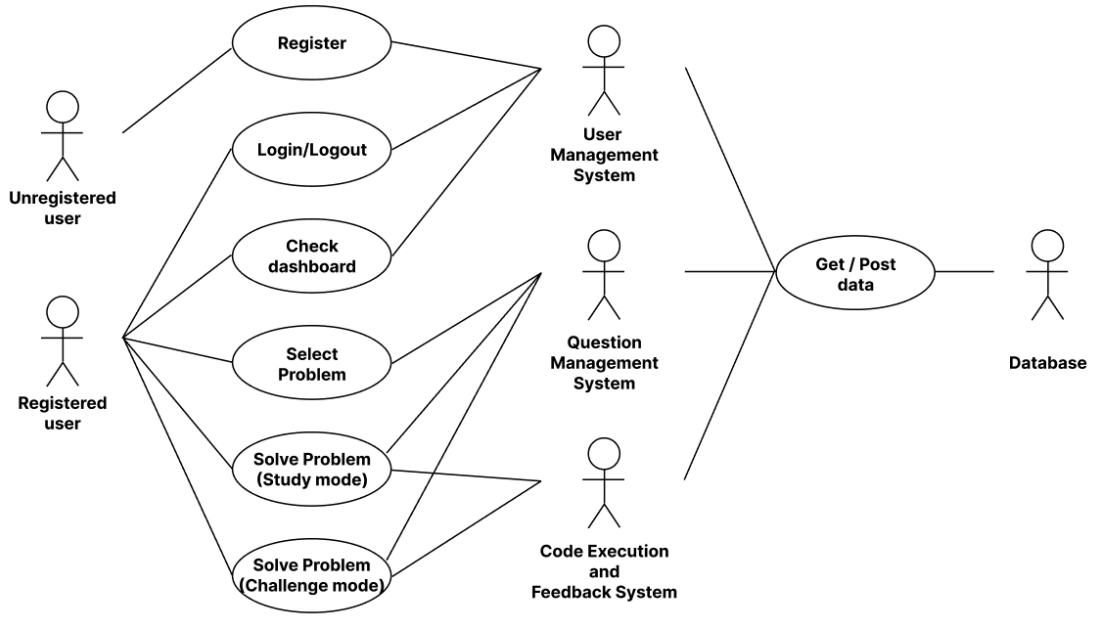

Figure 3.2: Use Case diagram

4 System Architecture - Frontend

4.1 Objectives

이 장에서는 frontend 시스템의 구조, 속성 및 기능을 설명하고, Step with me에서 각 구성요소의 관계를 설명한다.

4.2 Components

4.2.1 Registration Form

새로운 사용자가 서비스를 이용하기 위해 회원가입을 통해 기본 정보를 입력해야 한다. 입력해야 하는 기본 정보에는 이메일, 이름, 비밀번호, 나이, 프로그래밍 실력이 있다. 입력된 정보를 서버에 전달해 회원가입을 요청한다.

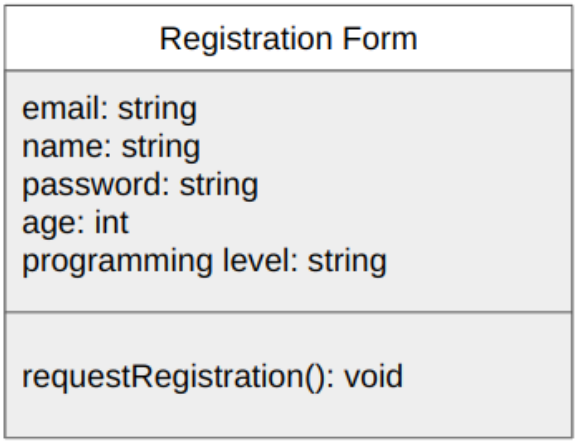

Figure 4.1: Class Diagram - Registration Form

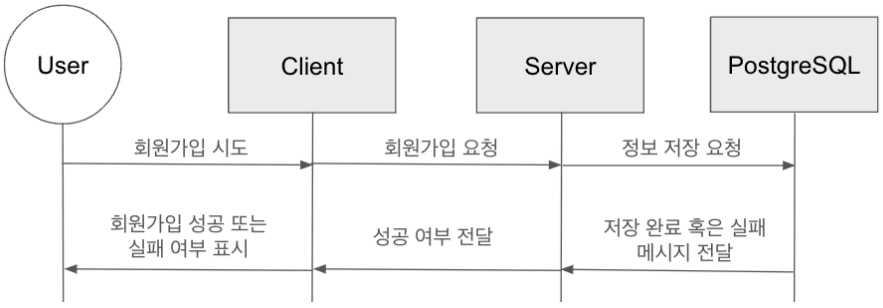

Figure 4.2: Sequence Diagram - Registration Form

4.2.2 Login Form

사용자는 회원가입 시 등록한 이메일과 비밀번호를 입력해 로그인할 수 있다. 입력된 정보를 서버에 전달해 로그인을 요청한다.

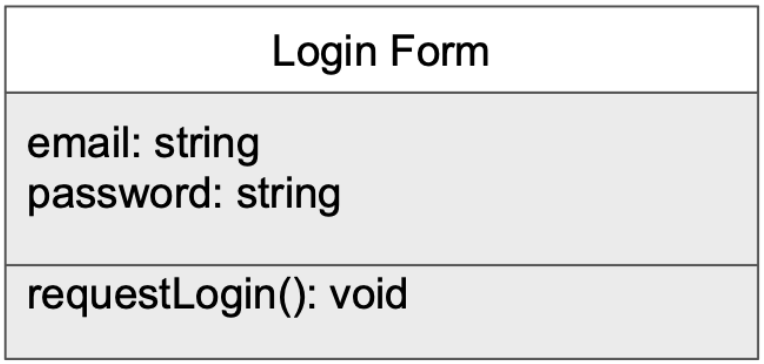

Figure 4.3: Class Diagram - Login Form

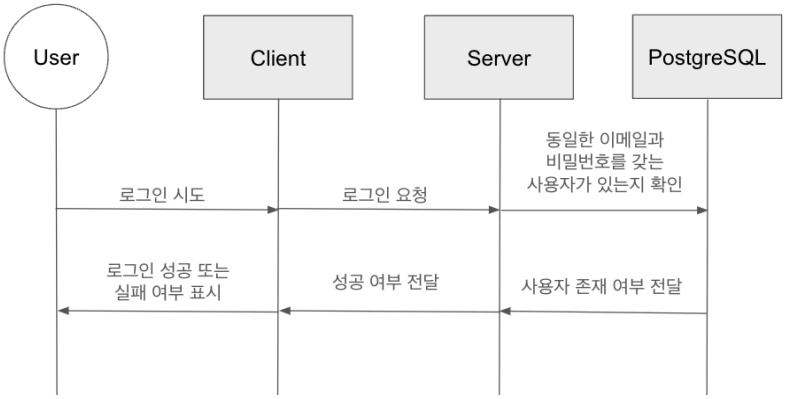

Figure 4.4: Sequence Diagram - Login Form

4.2.3 User Dashboard

사용자가 대시보드 페이지 접근 시 최근 완료한 문제, 자주 푼 유형, 추천 문제 목록을 보여주는 컴포넌트이다.

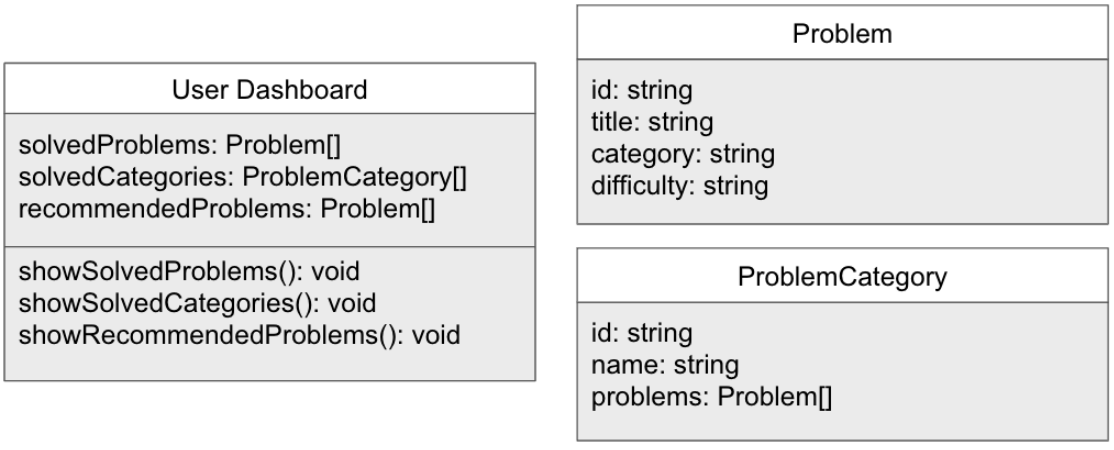

Figure 4.5: Class Diagram - User Dashboard

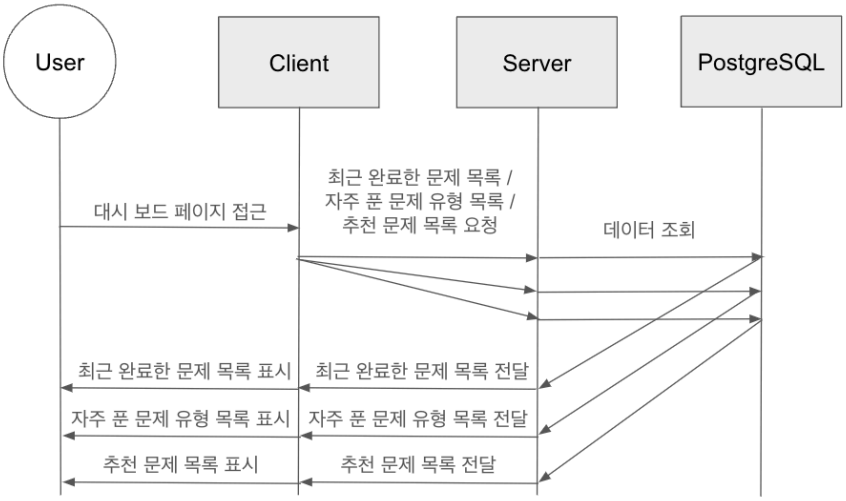

Figure 4.6: Sequence Diagram - User Dashboard

4.2.4 Problem List

사용자가 문제 목록 페이지에 접근 시 추천 문제 목록을 보여주는 컴포넌트이다.

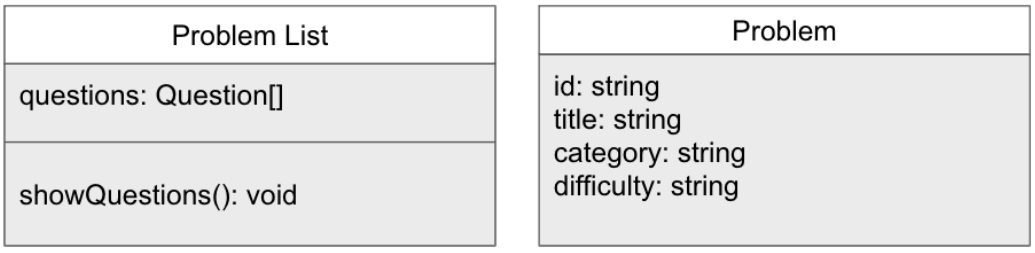

Figure 4.7: Class Diagram - Problem List

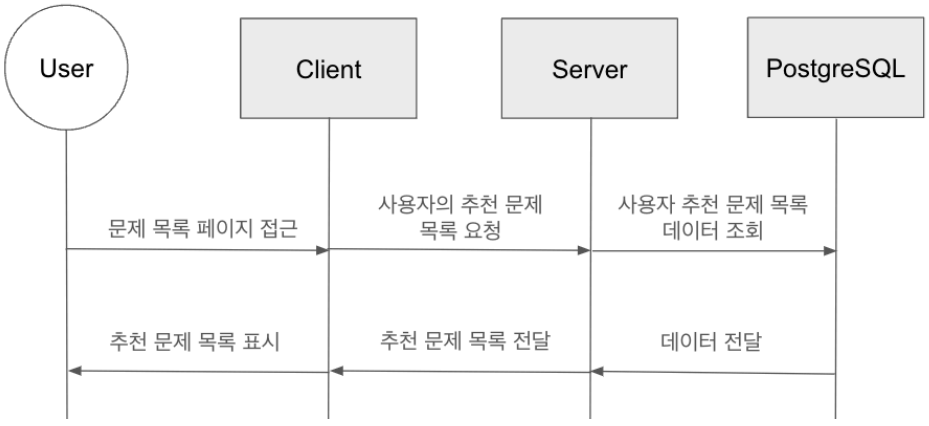

Figure 4.8: Sequence Diagram - Problem List

4.2.5 Problem Solving

사용자가 입력한 코드를 저장하고, 사용자가 코드 제출 시 서버에게 실행을 요청하며, 결과 및 피드백을 전달받은 후 사용자에게 표시하는 컴포넌트이다. 챌린지 모드일 때는 서버로부터 피드백을 전달받거나 사용자에게 피드백을

표시하지 않고 실행 결과와 에러 메시지만 표시한다. 스터디 모드일 때는 실행 결과와 에러 메시지뿐만 아니라 서버로부터 전달받은 피드백을 사용자에게 표시한다.

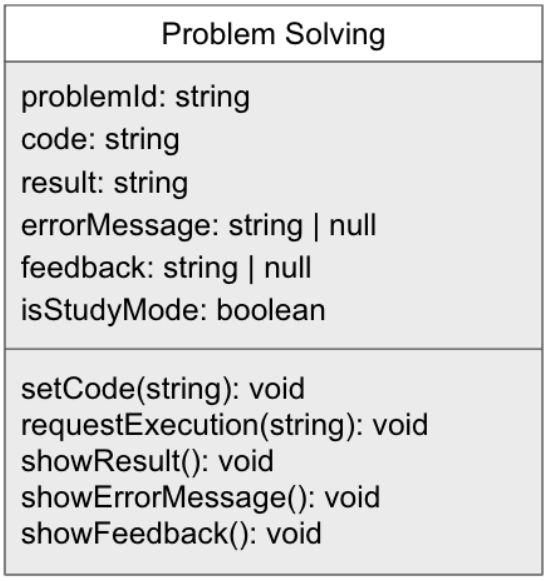

Figure 4.9: Class Diagram - Problem Solving

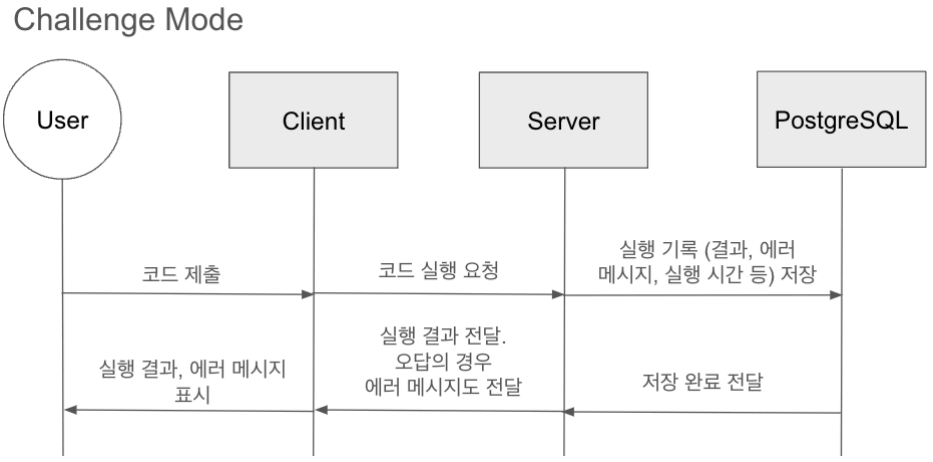

Figure 4.10.1: Sequence Diagram - Problem Solving (Challenge Mode)

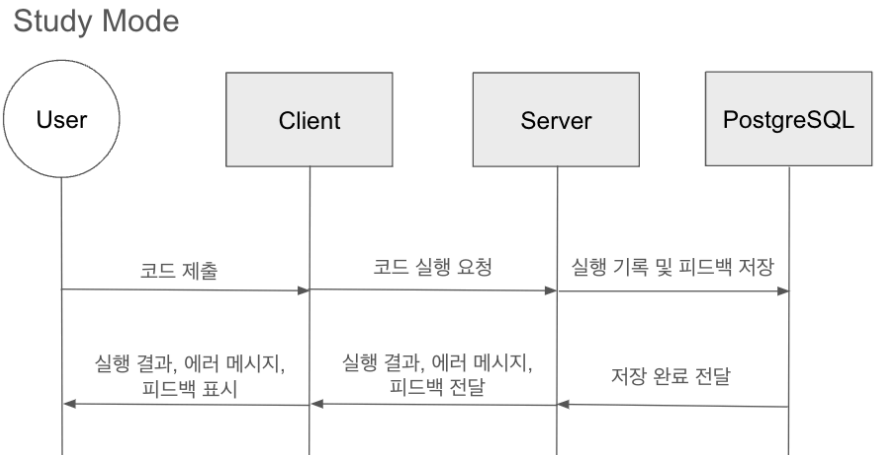

Figure 4.10.2: Sequence Diagram - Problem Solving (Study Mode)

4.2.6 Submission Result

챌린지 모드에서 코드 제출 상세 페이지 접근 시 코드 제출 결과를 보여주는 컴포넌트이다. 제출된 코드, 실행 결과, 실행 시간, 메모리 사용량, 백분위를 표시한다.

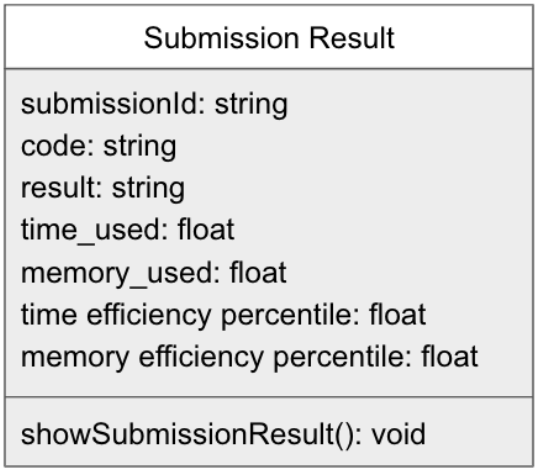

Figure 4.11: Class Diagram - Submission Result

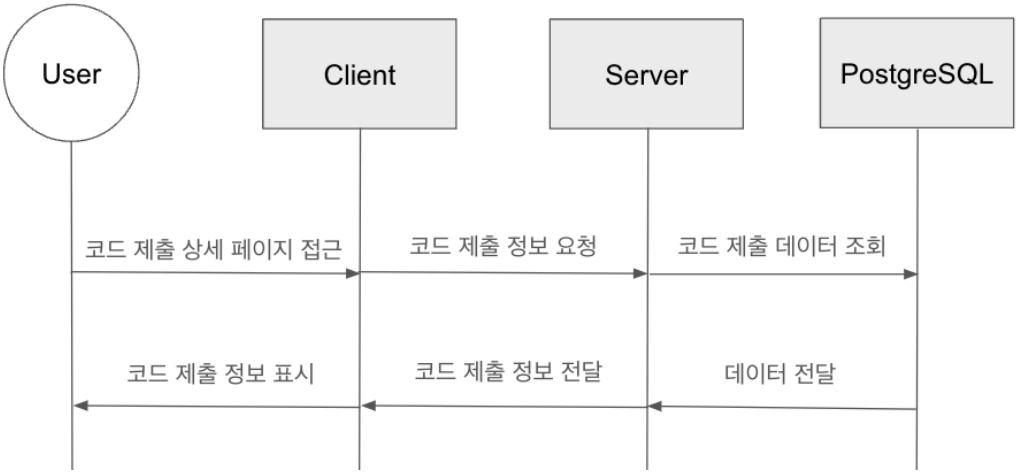

Figure 4.12: Sequence Diagram - Submission Result

4.2.7 Python Tutor

입력한 코드의 알고리즘을 단계적으로 시각화해 주는 오픈소스 컴포넌트이다. 정상적인 알고리즘의 전체 동작 과정을 표시하거나, 오류가 있는 입력의 경우 실패 직전의 지점까지 시각화함으로써 문제 발생 지점 및 원인의 이해를 도울 수 있다. Python, Java, C, C++, 그리고 JavaScript 언어에 대한 시각화를 제공한다. Step with me는 해당 기능을 활용하여, 사용자의 코드를 실행시켜 이에 대한 trace를 수집 후, 해당 trace들을 바탕으로 사용자 인터페이스에 시각화 정보를 제공한다. Python Tutor 시각화 피드백은 스터디 모드에서만 제공된다.

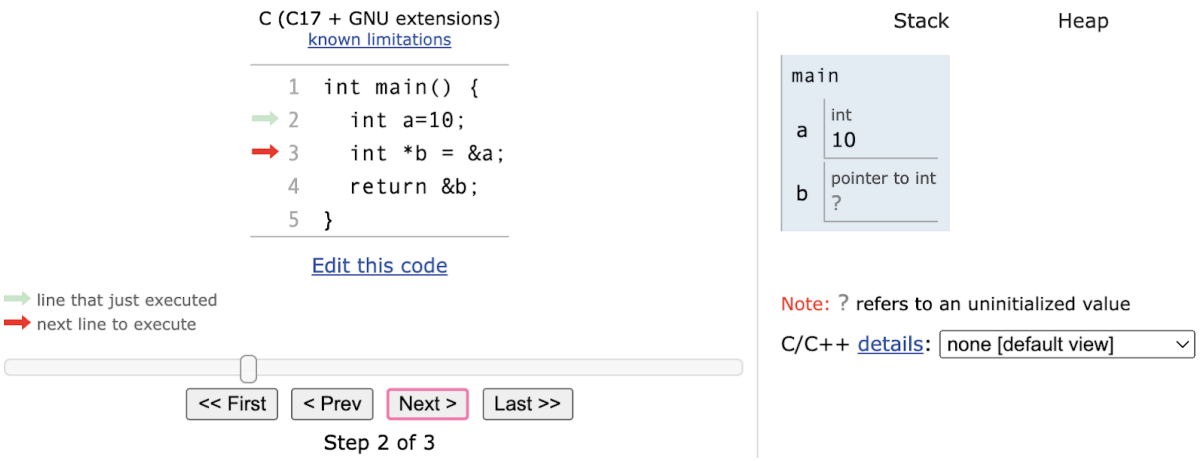

Figure 4.13.1: Example of Python Tutor (1)

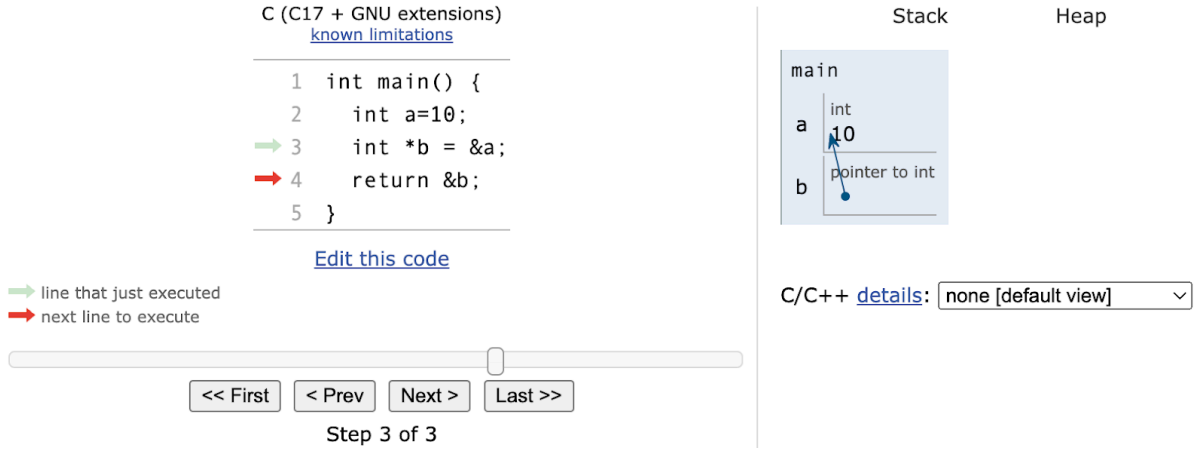

Figure 4.14.2: Example of Python Tutor (2)

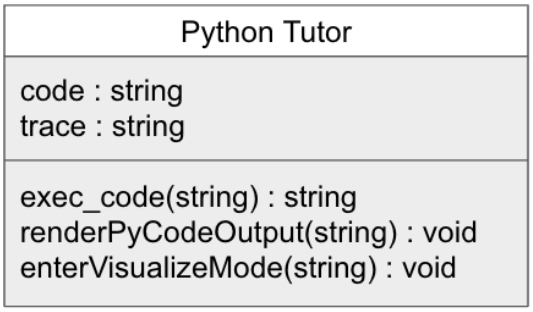

Figure 4.15: Class Diagram - Python Tutor

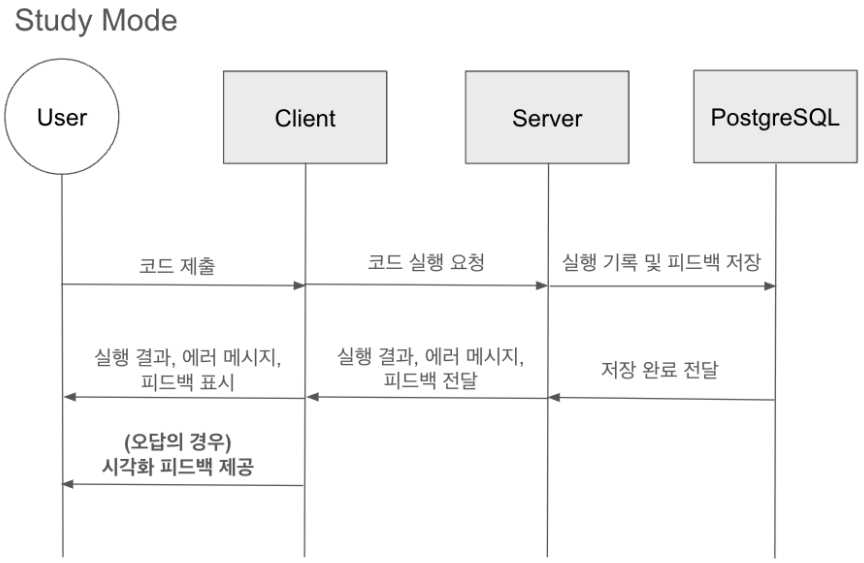

Figure 4.16: Sequence Diagram - Python Tutor

5 System Architecture - Backend

5.1 Objectives

이 장에서는 Backend 시스템의 구조와 이를 구성하는 컴포넌트에 관해 설명한다.

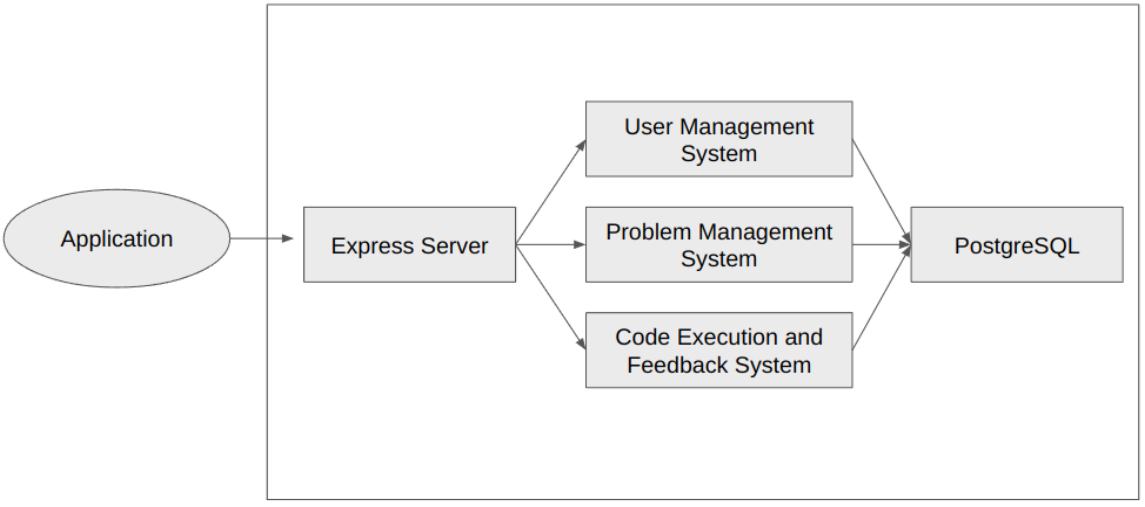

Figure 5.1: Backend Overall Architecture

1. Application ↔ Express Server
   Express 서버는 User Management System과 Problem Management System, Code Execution and Feedback System을 통해 사용자 요청을 처리한다. 클라이언트가 사용자 정보 수정, 코드 제출 등 요청을 보내면, Express 서버가 이를 처리하여 PostgreSQL 데이터베이스에 반영한다.
2. Systems ↔ PostgreSQL
   User Management System과 Problem Management System, Code Execution and Feedback System은 PostgreSQL 데이터베이스를 사용하여 데이터를 저장하고 관리한다. Express 서버를 통해 데이터베이스와 상호작용을 하며, 필요한 데이터를 읽고 쓴다.

5.2 Subcomponents

5.2.1 User Management System

Class Diagram

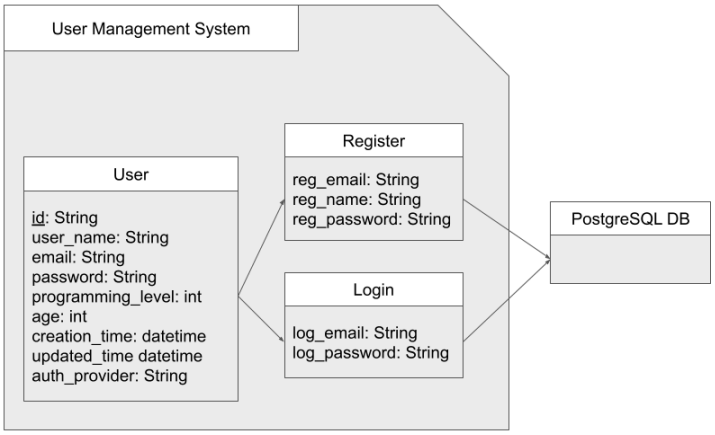

Figure 5.2: Class Diagram - User Management System

- Register class: 입력받은 email이 중복되지 않았는지 확인하고, 입력받은 정보를 데이터베이스에 추가한다.
- Login class: 입력받은 데이터가 데이터베이스에 존재하는지 확인하고, 정보가 일치하면 해당 class의 정보를 불러온다.

Sequence Diagram

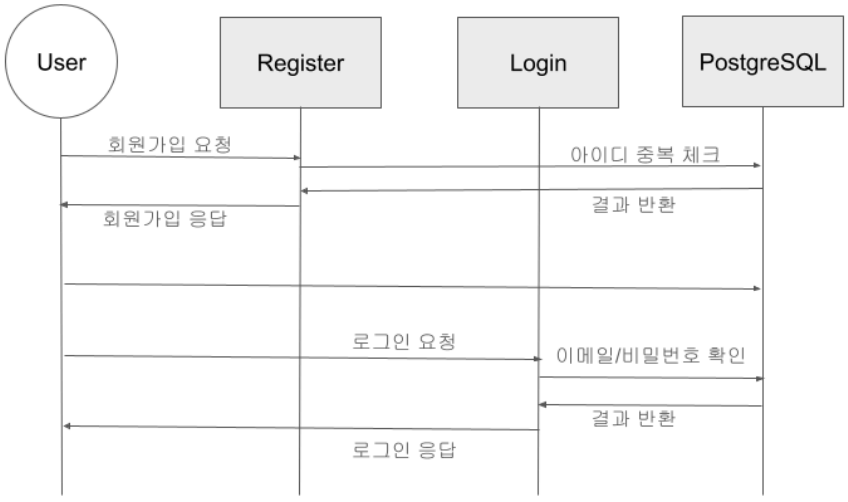

Figure 5.3: Sequence Diagram - User Management System

5.2.2 Problem Management System

Class Diagram

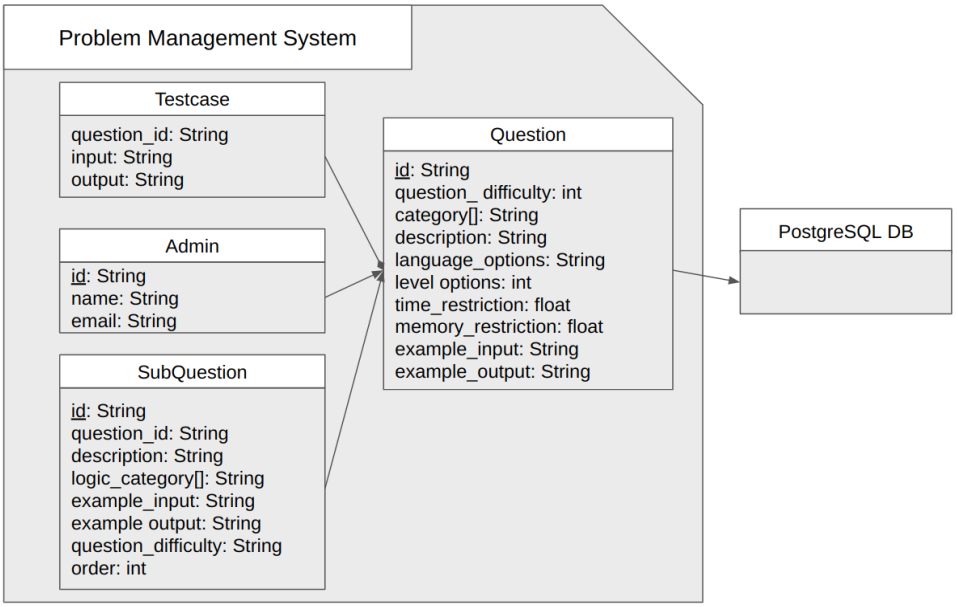

Figure 5.4: Class Diagram - Problem Management System

- Question class: 관리자(Admin)가 추가한 문제 데이터를 저장하고, Testcase를 포함하여 문제 해결에 필요한 모든 데이터를 관리한다.
- Testcase class: problem_id를 통해 특정 문제와 연결되며, 사용자가 제출한 코드 실행 결과와 output을 비교하여 결과를 반환한다.
- Admin class: 새로운 문제 데이터를 생성하고 데이터베이스에 저장하며, 기존 문제를 수정하거나 삭제할 수 있다.

Sequence Diagram

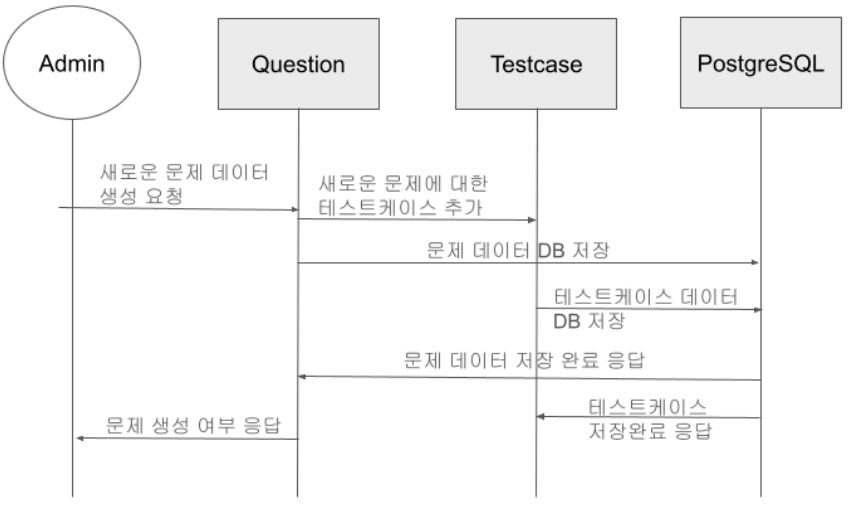

Figure 5.5: Sequence Diagram - Problem Management System

5.2.3 Code Execution and Feedback System

Class Diagram

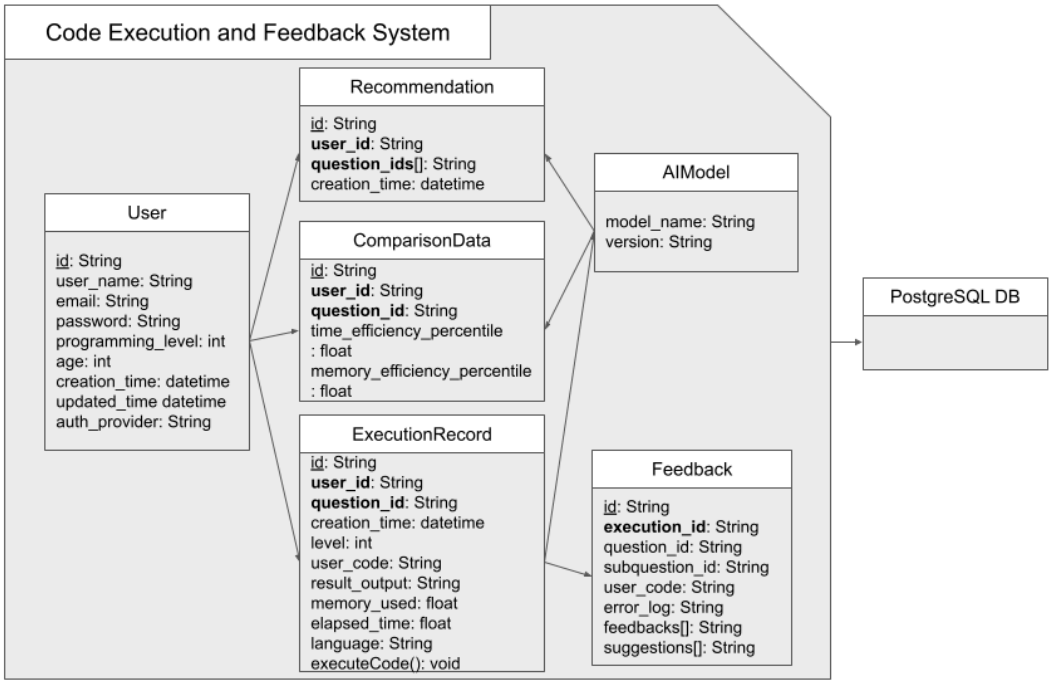

Figure 5.6: Class Diagram - Code Execution and Feedback System

- User class: 코드 제출 시 사용자 ID와 연결된 실행 기록이 생성되며, 사용자는 자신의 피드백, 추천 데이터, 비교 데이터를 조회할 수 있다.
- AIModel class: 실행기록 데이터를 분석하여 코드의 오류와 개선점을 파악하고, 사용자 수준에 맞는 문제를 추천하며, 사용자 간의 성과 데이터를 비교하여 백분위를 계산한다.
- ExecutionRecord class: 코드 실행 메서드가 호출되면 실행기록이 생성되고, 데이터베이스에 저장된다. 이후 AIModel에 실행 기록 분석을 요청한다. 해당 기록 역시 데이터베이스에 저장된다.
- Feedback class: AIModel이 생성한 각 피드백(코드의 오류와 개선점)은 데이터베이스에 저장된다.
- Recommendation class: Execution record와 Feedback을 기반으로 AIModel이 생성한 추천 문제 목록을 데이터베이스에 저장한다.
- ComparisonData class: AIModel이 생성한 각 사용자에 대한 풀이 성과 비교 데이터를 데이터베이스에 저장한다.

Sequence Diagram

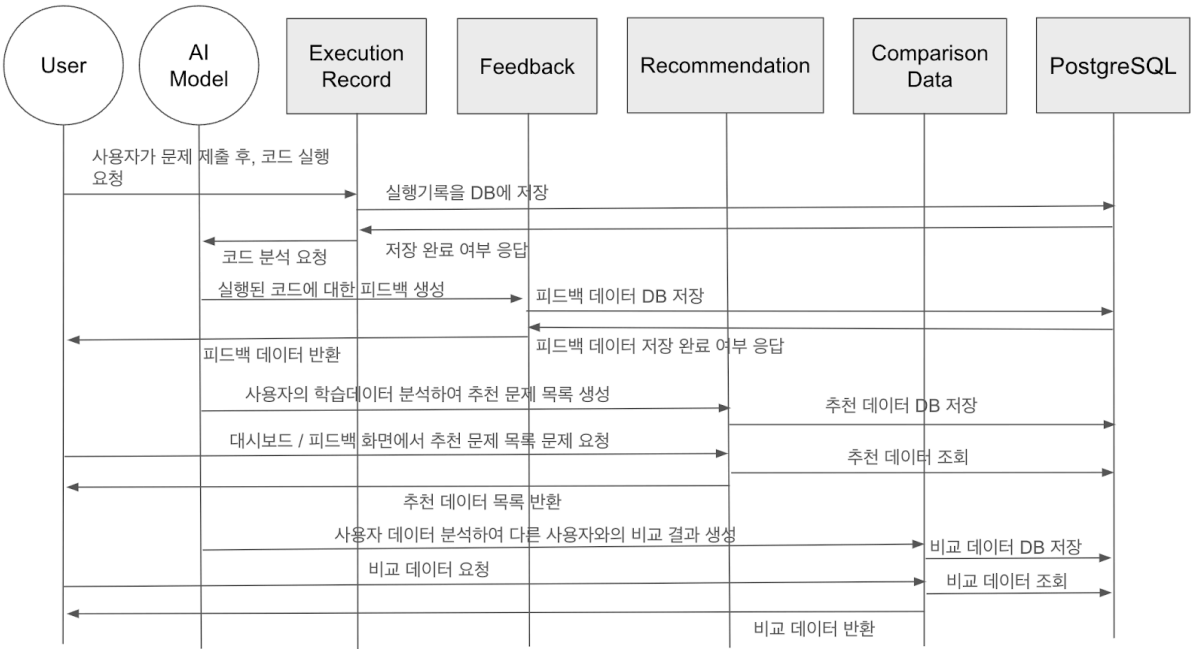

Figure 5.7: Sequence Diagram - Code Execution and Feedback System

6 Protocol Design

6.1 Objectives

이 섹션은 Frontend단과 Backend단이 어떤 프로토콜을 사용하여 통신하는지를 기술한다. 또한 요청-응답별 세부 명세사항을 기술한다.

6.2 Next.js API Route

Next.js의 API route를 사용하면 frontend 프로젝트 내에서 서버사이드로 API endpoint를 생성할 수 있다. CORS(Cross origin resource sharing)에러는 서버와 서버 간 통신에서는 일반적으로 발생하지 않는데, 이 프로젝트에서는 CORS 에러를 우회하여 해결하고자 API route가 사용되었다.

6.3 TLS

TLS(Transport Layer Security)는 인터넷상의 커뮤니케이션을 위한 개인 정보와 데이터 보안을 용이하게 하기 위해 설계되어 널리 채택된 보안 프로토콜이다. 웹 애플리케이션과 서버 간의 커뮤니케이션을 암호화하는 것에 주로 사용된다.

6.4 HTTPS

HTTPS(Hypertext Transport Protocol Secure)는 TLS(SSL)를 사용하여 요청과 응답에 디지털 서명을 통해 보안이 추가된 HTTP이다. 그 결과로 HTTPS는 HTTP보다 훨씬 더 안전하고, 현대에 운영되는 웹 서비스들에 필수적이다.

6.5 Authentication Request/Response

6.5.1 Register

Request

Description	User email register

URI	/api/v1/user/add
Method	POST
HEADERS	Content-type: application/json charset: utf-8
Body example	{“email” : “example@gmail.com”, “password” : “userpass”, “user_name”: “username”}

Response

Status	201 created
Body example	{“message”: “success”, “access_token”: “eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyX2FjY291bnQiOjExfQ.ZTag-kNYIw_NAya7ql-dyRosc2Lttbwpux-tfwlUFYg..”, “refresh_token”: “eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyX2FjY291bnQiOjExfQ.ZTag-kNYIw_NAya7ql-dyRosc2Lttbwpux-tfwlUFYg..”}

6.5.2 Login

Request

Description	User email login
URI	/api/v1/user
Method	POST
HEADERS	Content-type: application/json charset: utf-8
Body example	{“email” : “example@gmail.com”, “password” : “userpass”}

Response

Status	200 ok

Body example	{
	“user”: {id: “1”, “email”:”example@gmail.com”, “user_name”:”junghwan”,age:”25”,”programming_level”:”intermediate”,”auth_provider”:”kakao” }
	}

Status	401 unauthorized
Body example	{“message”: “user not found”}

6.5.3 Token refresh

Request

Description	Refresh access token with refresh token
URI	/api/v1/token/refresh
Method	POST
HEADERS	Content-type: application/json charset: utf-8
Body example	{“refresh_token”: “eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyX2FjY291bnQiOjExfQ.ZTag-kNYIw_NAya7ql-dyRosc2Lttbwpux-tfwlUFYg..”}

Response

Status	200 ok
Body example	{“access_token”: “eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyX2FjY291bnQiOjExfQ.ZTag-kNYIw_NAya7ql-dyRosc2Lttbwpux-tfwlUFYg..”, “refresh_token”: “eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyX2FjY291bnQiOjExfQ.ZTag-kNYIw_NAya7ql-dyRosc2Lttbwpux-tfwlUFYg..”}

6.5.4 Update User Information

Request

Description	Update user information
URI	/api/v1/token/user/update
Method	PUT
HEADERS	Content-type: application/json charset: utf-8 Authorization: bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyX2FjY291bnQiOjExfQ.ZTag-kNYIw_NAya7ql-dyRosc2Lttbwpux-tfwlUFYg.
Query params	userId
Body example	{“user_name”: “new_user_name”}

Response

Status	200 ok
Body example	{“message”: “success”}

6.5.6 Get User Execution Record

Request

Description	Get all execution history of user
URI	/api/v1/executions
Method	GET
HEADERS	Content-type: application/json charset: utf-8 Authorization: bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyX2FjY291bnQiOjExfQ.ZTag-kNYIw_NAya7ql-dyRosc2Lttbwpux-tfwlUFYg.
Query params	userId

Response

Status	200 ok

Body example	{data: [
	{
	"id": 10,
	"user_id": 2,
	"problem_id": 42,
	"creation_time": "2024-07-26T00:00:00",
	"level": "Intermediate",
	"memory_used": "150MB",
	"elapsed_time": 100
	"user_code": “#include <stdio.h> int main(){ double A,B; scanf("%lf %lf",&A,&B); printf("%.10lf",A/B); return 0;}"
	},
	{
	"id": 11,
	"user_id": 2,
	"problem_id": 55,
	"creation_time": "2024-07-26T00:00:00",
	"level": "Intermediate",
	“language: "Python",
	"memory_used": "150MB",
	"elapsed_time":
	"user_code": “#include <iostream> #include <string> using namespace std; int main(){ string num; cin >> num; string cur(num); int cnt = 0; int temp; do{ temp = 0; for(int i=0;i<cur.length();i++){ temp += cur[i]-'0';} cur = cur.back(); cur.push_back(to_string(temp).back()); cur = to_string(stoi(cur)); cnt++; }while(num != cur); cout<<cnt; return 0;}”
	}]
	}

6.5.6 Add User Execution Record

Request

Description	Post new execution history
URI	/api/v1/execution/add
Method	POST
HEADERS	Content-type: application/json charset: utf-8 Authorization: bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyX2FjY291bnQiOjExfQ.ZTag-kNYIw_NAya7ql-dyRosc2Lttbwpux-tfwlUFYg.
Query params	

Body	{
	"id": 10,
	"user_id": 2,
	"problem_id": 42,
	"creation_time": "2024-07-26T00:00:00",
	"level": "Intermediate",
	"memory_used": "150MB",
	"elapsed_time": 100
	"user_code": “#include <stdio.h> int main(){ double A,B; scanf("%lf %lf",&A,&B); printf("%.10lf",A/B); return 0;}"
	}

Response

Status	201 created
Body example	{“message”:”success”}

6.5.7 Get User Feedbacks

Request

Description	Get all feedbacks for user

URI	/api/v1/feedbacks
Method	GET
HEADERS	Content-type: application/json charset: utf-8 Authorization: bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyX2FjY291bnQiOjExfQ.ZTag-kNYIw_NAya7ql-dyRosc2Lttbwpux-tfwlUFYg.
Query params	userId

Response

Status	200 okay
Body example	{“body”: [{ "id": 1, "executionId": 101, "problemId": 201, "subproblemId": 301, "userCode": "def solve(): return sum(range(10))", "errorLog": "IndexError: list index out of range", "feedbacks": ["Consider checking the bounds of your loop.", "You may be accessing elements outside the array's range."], "suggestions": ["Use a try-except block to handle potential exceptions.", "Validate input before processing."] }, {

	"id": 2, "executionId": 102, "problemId": 202, "subproblemId": 302, "userCode": "def solve(a, b): return a / b", "errorLog": "ZeroDivisionError: division by zero", "feedbacks": ["Ensure denominator is not zero before dividing.", "Test edge cases for zero inputs."], "suggestions": ["Add a condition to check if b == 0 before performing division.", "Provide a default value or throw a custom error for invalid inputs."] }]}

6.5.8 Get User Recommendation Records

Request

Description	Post new execution history
URI	/api/v1/recommendations
Method	GET
HEADERS	Content-type: application/json charset: utf-8 Authorization: bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyX2FjY291bnQiOjExfQ.ZTag-kNYIw_NAya7ql-dyRosc2Lttbwpux-tfwlUFYg.
Query params	userId

Response

Status	200 okay

Body example	{“body”: [
	{
	"id": 1,
	"userId": 501,
	"problemId": 201,
	"creationTime": "2024-11-30T12:00:00Z"
	},
	{
	"id": 2,
	"userId": 502,
	"problemId": 202,
	"creationTime": "2024-11-30T12:05:00Z"
	}
	]}

6.5.9 Get User Comparison data

Request

Description	Get user comparison data
URI	/api/v1/comparisons
Method	GET
HEADERS	Content-type: application/json charset: utf-8 Authorization: bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyX2FjY291bnQiOjExfQ.ZTag-kNYlw_NAya7ql-dyRosc2Lttbwpux-tfwlUFYg.
Query params	userId

Response

Status	200 okay

Body example	{“body”: [
	{
	"id": 1,
	"userId": 501,
	"problemId": 201,
	"timeEfficiencyPercentile": 85.5,
	"memoryEfficiencyPercentile": 92.3
	},
	{
	"id": 2,
	"userId": 502,
	"problemId": 202,
	"timeEfficiencyPercentile": 73.4,
	"memoryEfficiencyPercentile": 81.7
	}
	]}

6.5.10 Add new problem

Request

Description	Post new problem
URI	/api/v1/problem/add
Method	POST
HEADERS	Content-type: application/json charset: utf-8 Authorization: bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyX2FjY291bnQiOjExfQ.ZTag-kNYIw_NAya7ql-dyRosc2Lttbwpux-tfwlUFYg.
Query params	
Body	{ "id": 1, "problemDifficulty": "Medium", "category": "Mathematics", "description": "Given a list of integers, find the

	sum of all even numbers in the list.",
	"languageOptions": ["Python", "Java", "C++"],
	"levelOptions": ["Beginner", "Intermediate"],
	"timeRestriction": 2000,
	"exampleInput": "[1, 2, 3, 4, 5]",
	"exampleOutput": "6",
	"memoryRestriction": 256,
	"subproblem": [
	{
	"id": 101,
	"description": "Identify all even numbers from the input list.",
	"logicCategory": "Conditional Statement",
	"exampleInput": "[1, 2, 3, 4, 5]",
	"exampleOutput": "[2, 4]",
	"problemDifficulty": "Easy",
	"testcases": [
	{
	"problemId": 1,
	"input": "[10, 15, 20, 25]",
	"output": "[10, 20]"
	},
	{
	"problemId": 1,
	"input": "[1, 3, 5]",
	"output": "[]"
	}
	]
	},
	{
	"id": 102,
	"description": "Calculate the sum of the filtered even numbers.",
	"logicCategory": "For Loop",
	"exampleInput": "[2, 4]",
	"exampleOutput": "6",
	"problemDifficulty": "Medium",
	"testcases": [
	{
	"problemId": 1,
	"input": "[2, 8, 10]",
	"output": "20"

	},
	{
	"problemId": 1,
	"input": "[]",
	"output": "0"
	}
	]
	}
	]
	}

Response

Status	201 created
Body example	{“message”:”success”}

Status	401 unauthorized
Body example	{“message”: “not an admin user”}

7 Database Design

7.1 Objectives

이 섹션에서는 서비스에서 사용되는 데이터 모델의 구조와 데이터베이스에서 각 모델이 어떻게 구현되어 있는지에 대해 설명한다. ER 다이어그램을 사용하여 데이터 모델을 시각화하고, 각 엔티티에 대한 설명을 기술한다.

7.2 E-R Diagram

Step with me의 데이터베이스 구조는 크게 User, Admin, Feedbacks, ExecutionRecord, RecommendationRecord, ComparsionData, problem, Subproblem, Testcase 총 9개의 테이블로 이루어져 있다.

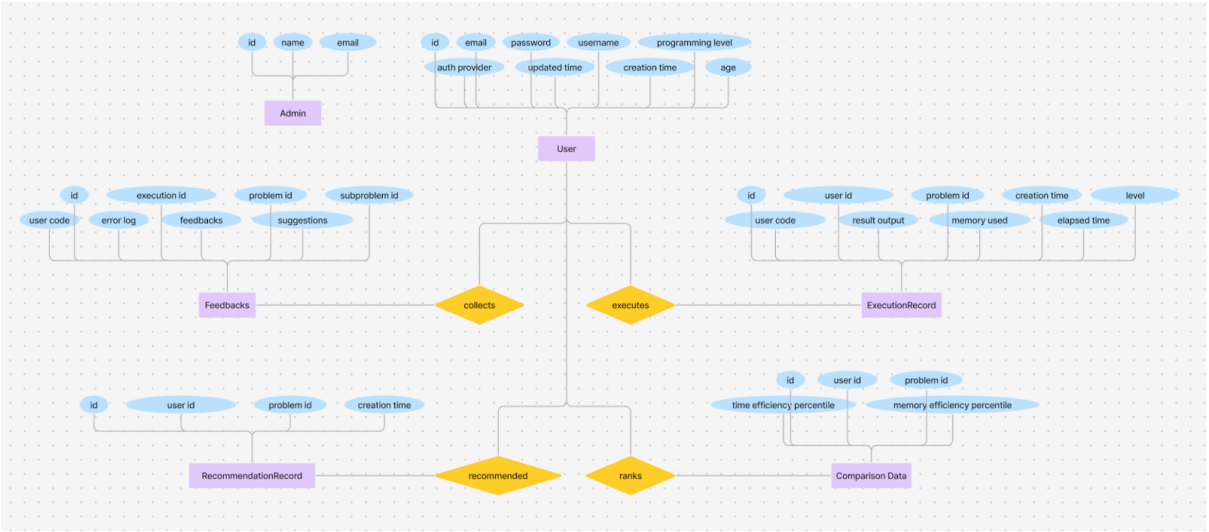

Figure 7.1.1: ER-diagram

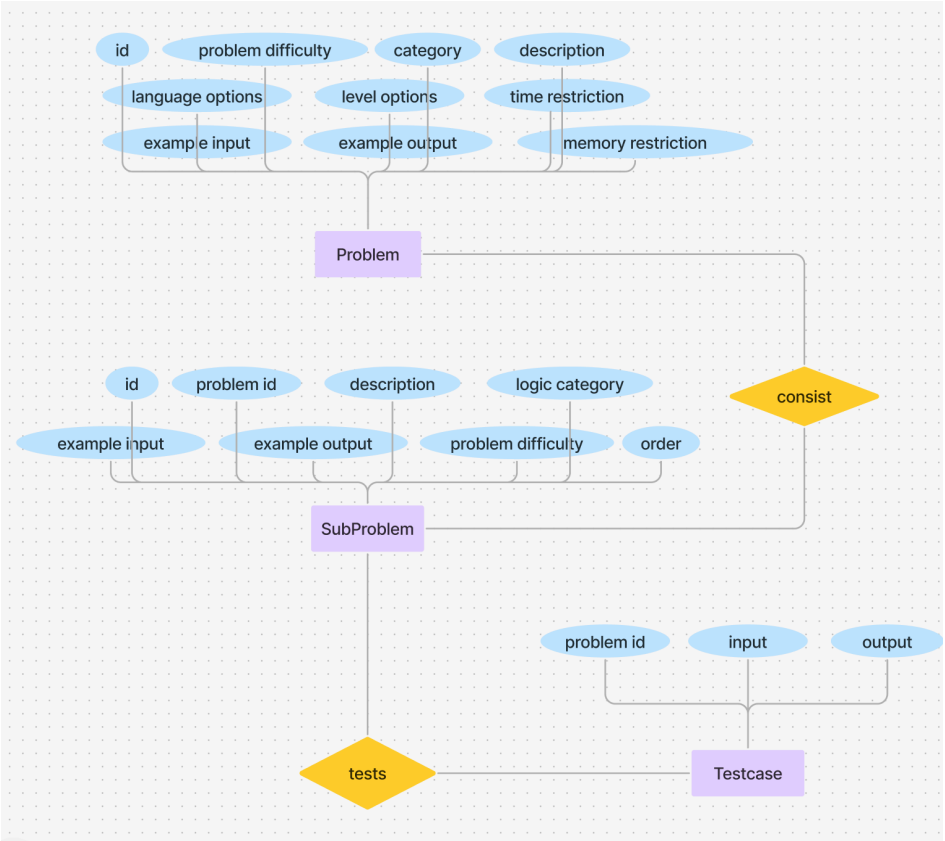

Figure 7.1.2: ER-diagram

7.2.1 User Table

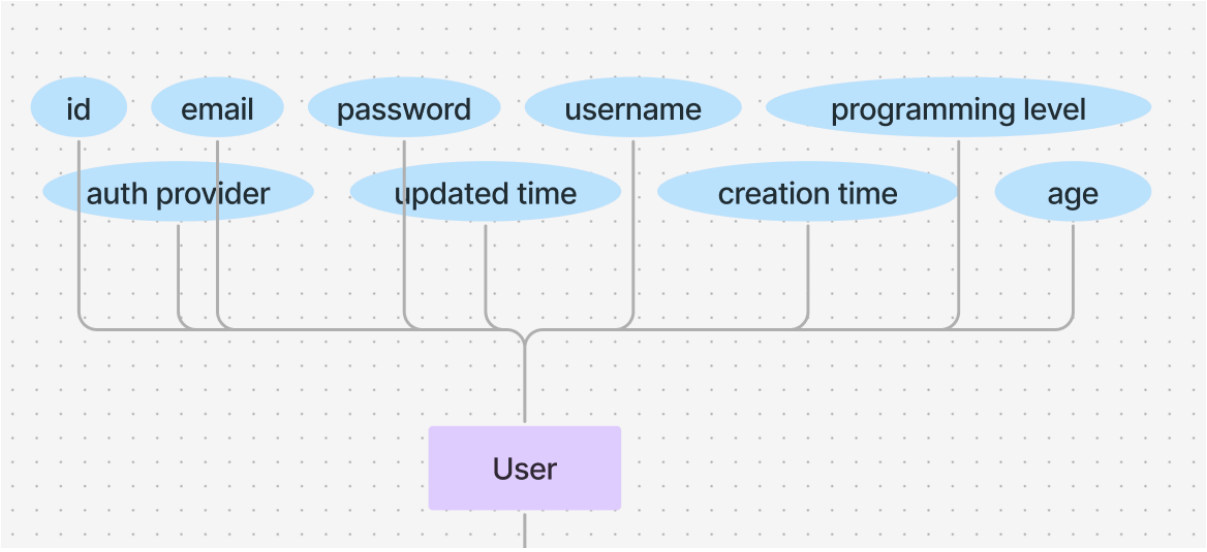

Figure 7.2: User ER-diagram

User entity는 첫 회원가입 때 생성되며, 회원 정보 수정을 통해 정보가 수정된다.

- id: user entity 식별자
- user name: 사용자가 애플리케이션 내에서 사용하는 이름

- programming level: 회원가입 질문에서 수집하는 사용자의 대략적인 프로그래밍 실력
- auth provider: 소셜 회원가입 시 인증을 제공하는 OAuth 호스트

7.2.2 Admin Table

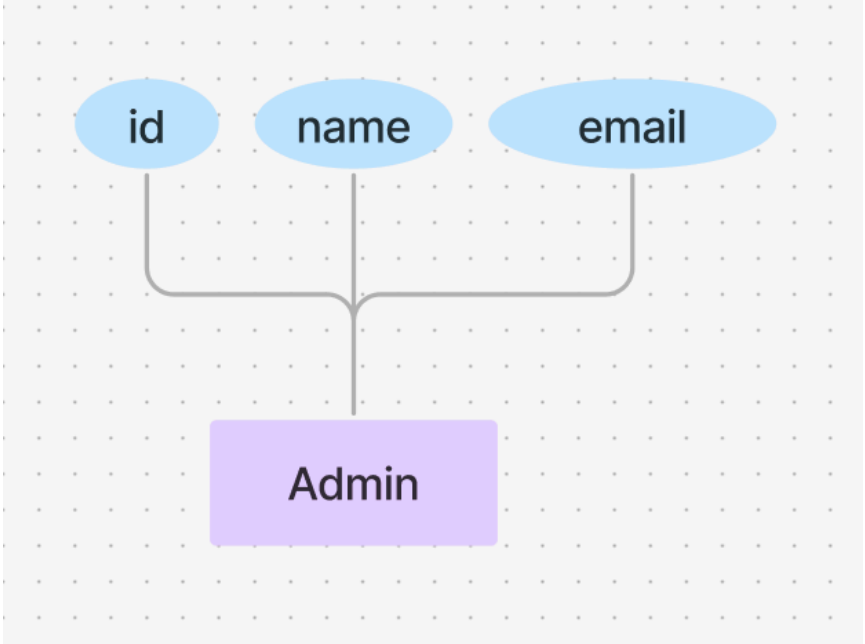

Figure 7.3: Admin ER-diagram

Admin 유저는 문제를 추가, 수정, 제거할 수 있는 권한이 있다.

7.2.3 Problem Table

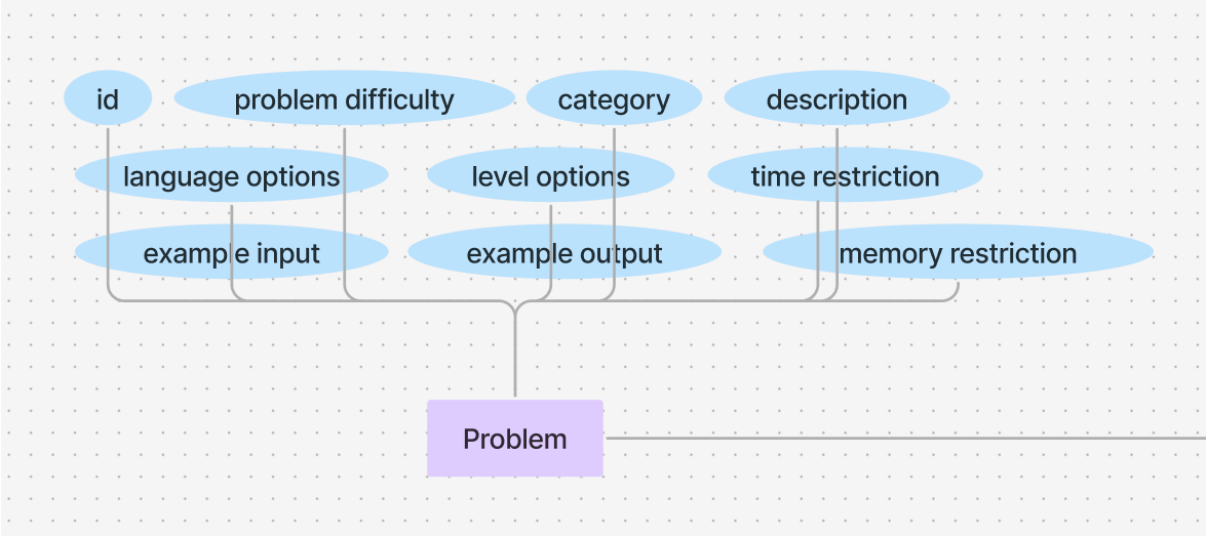

Figure 7.4: Problem ER-diagram

Step with me의 알고리즘 문제들은 전부 사전 제작된 problem table에서 제공된다. 이에는 문제 상세, 문제 메타정보, 난이도별 알고리즘 분할 정보 등이 포함되어 있다.

- problem difficulty: 문제의 난이도. 1부터 5 사이의 값이 사용.
- category: 알고리즘 문제 유형
- description: 사용자가 문제를 해결하기 위해 읽어야 하는 문제 설명
- language options: 사용자가 선택할 수 있는 프로그래밍 언어 선택지
- level options: 문제 세부 난이도 선택지. 어려울수록 문제 분할이 적음.
- time restriction: 제한 시간
- memory restriction: 메모리 제한

7.2.4 Subproblem Table

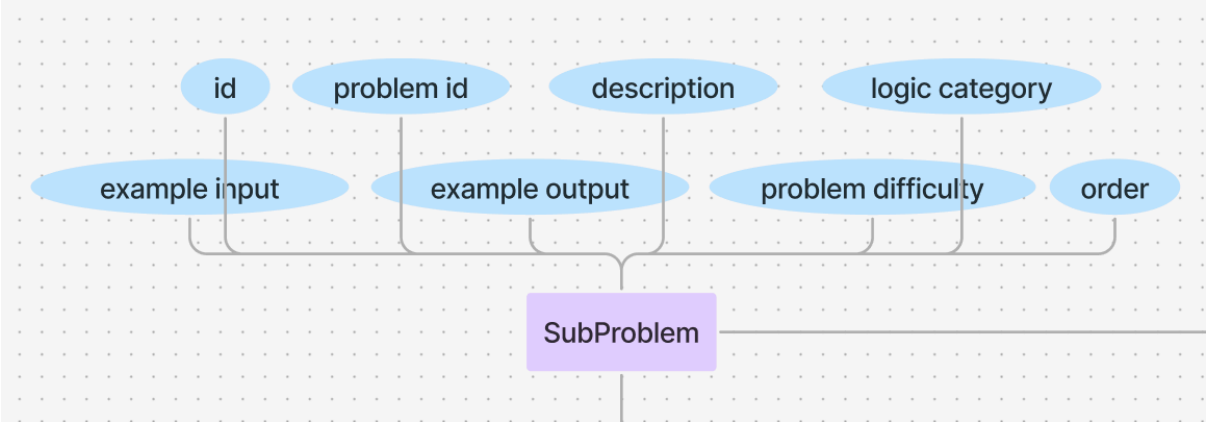

Figure 7.5: Subproblem ER-diagram

하나의 문제에는 난이도에 따라 여러 개의 subproblem으로 분할된다.

- logic category: 알고리즘 유형 (반복문, 조건문, 대입 연산 등)
- problem difficulty: 문제의 분할 정도
- order: 하위 문제가 표시되는 순서

7.2.5 Testcase Table

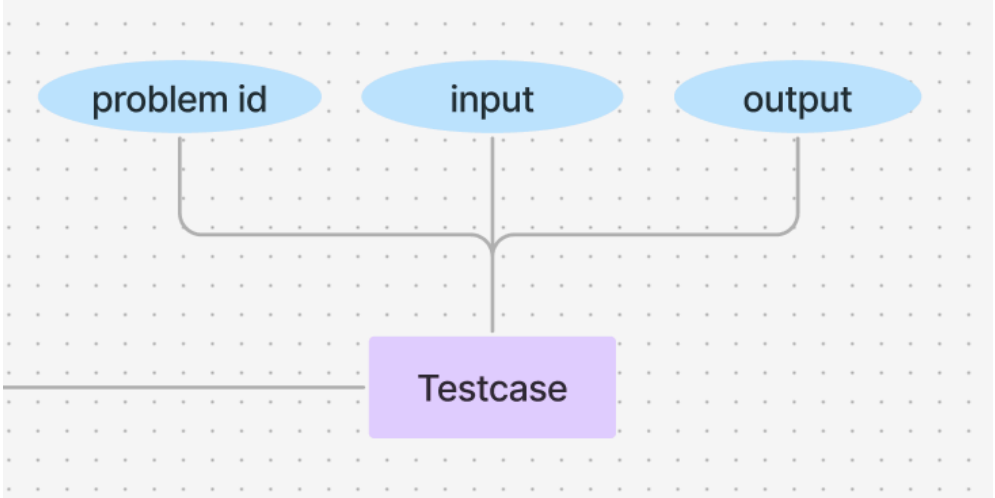

Figure 7.6: Testcase ER-diagram

하나의 subproblem에는 여러 개의 testcase들이 존재한다.

7.2.6 ExecutionRecord Table

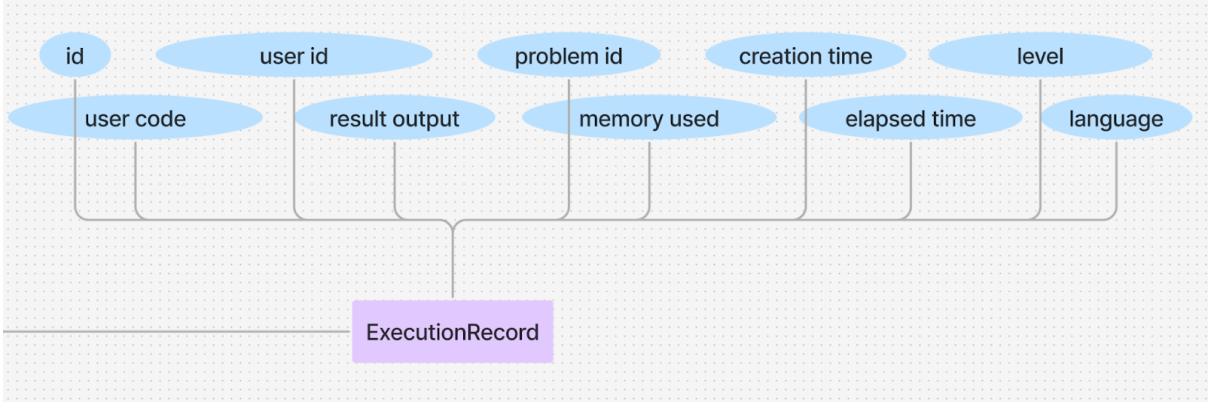

Figure 7.7: ExecutionRecord ER-diagram

사용자가 문제를 제출한 기록은 ExecutionRecord table에 저장된다.

- result output: 코드 제출 결과
- level: 해결한 문제의 난이도 (알고리즘 분할 정도)
- user code: 사용자가 제출한 코드
- elapsed time: 소요 시간
- memory used: 소요 메모리 크기
- language: 사용 언어

7.2.7 ComparisionDataTable

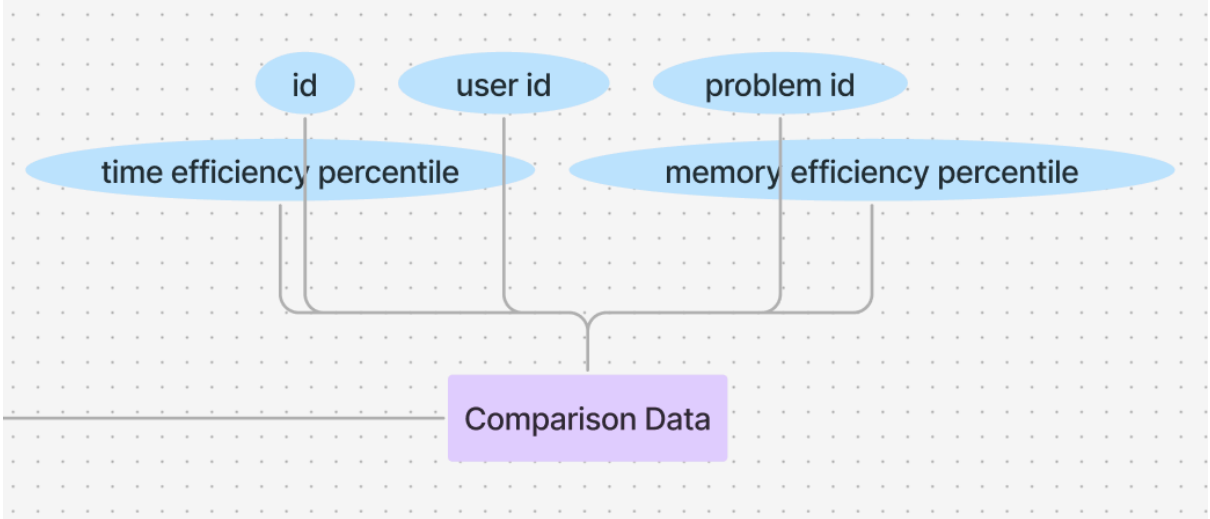

Figure 7.8: ComparisonData ER-diagram

문제가 해결됐을 때 유저의 시간/메모리 사용의 백분위를 저장하는 테이블이다.

- time efficiency percentile: 소요 시간 백분위
- memory efficiency percentile: 메모리 사용 백분위

7.2.8 RecommendationRecord Table

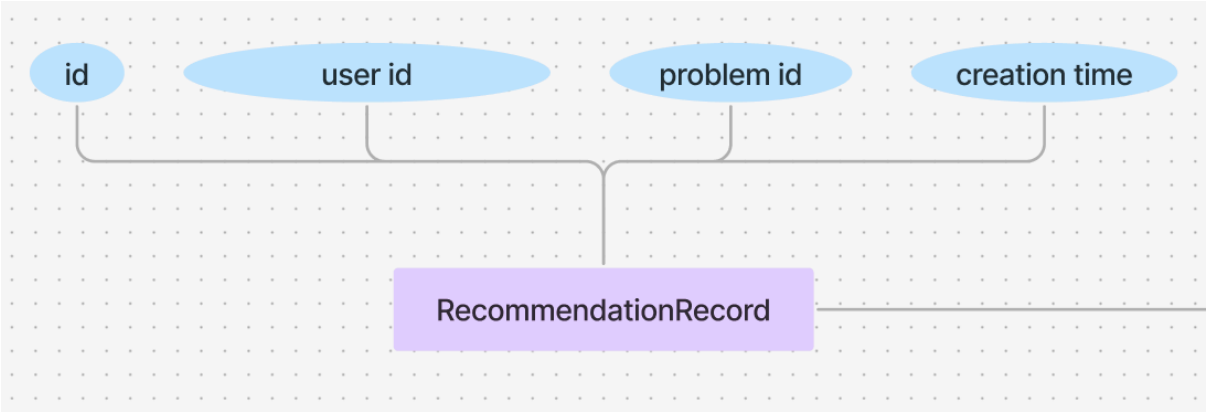

Figure 7.9: RecommendationRecord ER-diagram

각 유저에게 추천된 문제가 기록된 테이블이다.

7.2.9 Feedback Table

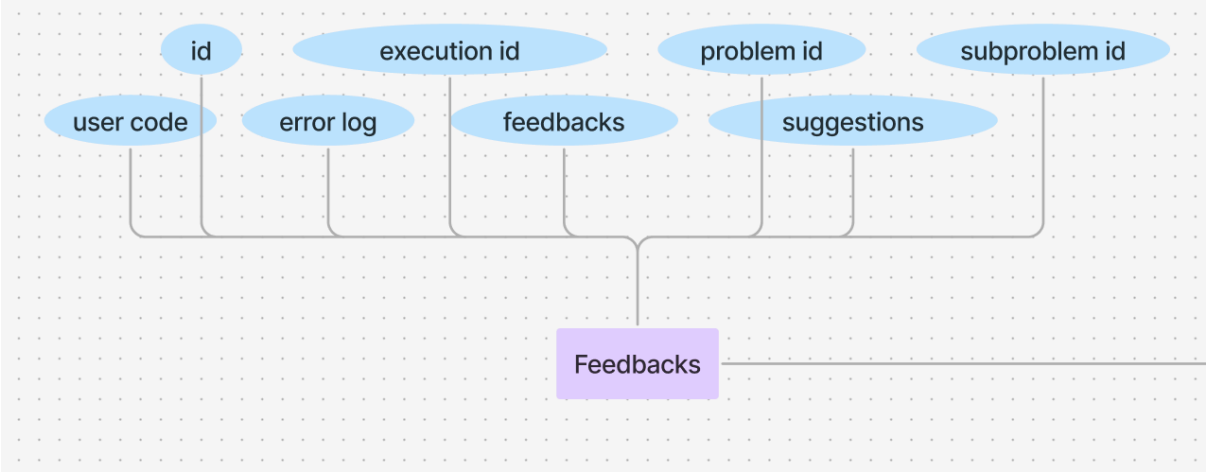

Figure 7.10: Feedback ER-diagram

유저가 받은 피드백 정보들이 저장되는 테이블이다.

- error log: 컴파일러가 출력한 에러 메시지
- feedbacks: LLM이 error log와 user code를 바탕으로 생성한 자연어 피드백 배열
- suggestions: LLM이 error log와 user code를 바탕으로 생성한 자연어 코드 수정 제안 배열

8 Testing Plan

8.1 Objectives

이번 장에서는 Step with me의 testing 계획을 설명한다. 본 과정은 development, release, user testing, 의 세 단계로 나누어 진행된다. Testing은 본 애플리케이션의 문제점을 조기에 발견하며, 요구 사항을 만족하였는지를 확인함으로써 안정적인 작동을 보장한다. 이를 통해 사용자에게 제공하는 제품의 품질을 극대화하는 것을 최종 목표로 한다.
해당 애플리케이션은 다음과 같은 조건을 충족해야 한다. 코딩 교육이라는 주목적에 따라 사용자가 코드를 제출할 경우 검토 및 오류 발견이 평균 3초 이내로 완료되어야 한다. Python Tutor를 이용한 시각화 및 개인 맞춤형 피드백의 생성 역시 유저의 요청 이후 3초 이내로 완료될 수 있어야 한다. 또한, 이후 사용자 맞춤형 피드백 및 추천 생성을 위해 이용 기록은 5초 이내로 데이터베이스에 저장될 수 있어야 한다.
웹 브라우저를 통해 동시 평균 10,000명의 원활한 사용을 지원하기 위해 다양한 브라우저, 운영체제, 네트워크 환경에서 테스트해야 한다.

8.2 Testing Policy

Step with me의 성능, 안정성 등을 개별 모듈에서 전체 시스템 및 실제 사용까지 단계적으로 오류를 감지하고 수정한다. 개발 단계에서부터 testing을 실행함으로써 조기 감지 및 수정, 개발 팀원을 비롯하여 실제 사용자의 피드백을 반영하여 애플리케이션의 성능 및 안정성을 보장한다.

8.2.1 Development Testing

8.2.1.1 Unit Testing

개별 객체와 함수가 설계대로 동작하는지 확인한다. 모의 객체(mock object)를 이용하여 다른 컴포넌트와의 상호 작용을 제한함으로써 각 컴포넌트의 작동을 확인할 수 있다. 해당 애플리케이션에서는 사용자가 입력한 코드를 정확하게 실행하는지, 코드의 에러를 감지하는지, 이를 기반으로 올바른 에러 메시지를 출력하는지 등을 검증한다.

8.2.1.2 Component Testing
System testing을 위해 모든 컴포넌트가 통합되기 전, 각각의 동작성 및 의존성을 확인한다. 각 컴포넌트를 이루는 모듈 간의 상호작용이 원활하고, 데이터 흐름과 기능의 연결이 적절한지 스터빙(stubbing), 모의 객체 등을 이용하여 점검한다. 또한, 서비스-UI 간의 작용, 외부 시스템과의 상호 작용 등도 검증한다.

8.2.1.3 System Testing
통합된 시스템을 종합적으로 확인한다. 사용자 인터페이스, LLM, 피드백 시스템 등을 포함한다. 요구 사항의 충족 및 구성 컴포넌트 간의 상호작용, 시스템 전체의 emergent properties를 점검한다.

8.2.2 Release Testing
개발 팀 외부의 인원이 진행하게 되며, 완성된 전체 시스템을 테스트한다. 유저에게 배포 전 버그를 수정하고, 요구사항이 충족되었는지 확인하는 것이 주 목적이다. 따라서 외부 배포를 전제로 한 완성된 시스템을 바탕으로 테스트를 진행한다.

8.2.3 User Testing
시스템에 대한 유저의 포용성, 시스템의 직관성, 완성도, 사용성 등을 평가한다. 또한, 애플리케이션의 특성상 실제 사용자를 대상으로 학습 효율의 증대에 필요한 개선 사항을 확인한다. 해당 단계에서 사용되는 애플리케이션은 시스템 테스팅을 통해 기초적인 동작 사항을 모두 충족할 수 있는 상태여야 한다. 개발팀 내부에서 소통하며 잠재적 케이스에 대한 피드백을 제공하는 알파 테스팅 이후, 베타 버전을 배포함으로써 외부 유저의 사용으로 일어나는 개별적인 시나리오를 통해 오류를 확인한다.

8.3 Test Case
해당 애플리케이션에 관한 테스트 케이스는 기능성, 사용성, 성능, 보안의 확인에 초점을 맞추어 설계한다. 각각에 대한 테스트 환경을 명확히 정의하고, 경계 조건과 예외 상황을 포함하여 다양한 상황에 대비할 수 있도록 한다. 각 영역에 대해 5개 이상의 시나리오를 통해 검증을 진행한다.

Test Case Example ID	SWM001
Test Case Description	사용자가 등록되지 않은 계정 정보로 Step with me웹 페이지에 로그인할 수 없음을 확인한다.

Preconditions	Step with me의 데이터베이스에 해당 정보를 가진 유효한 계정이 존재하지 않는다.
Test Steps	1. 웹 브라우저를 열고 Step with me의 웹 페이지에 접속한다. 2. 로그인 페이지에서 아이디와 비밀번호를 입력하거나 소셜 로그인 방식을 선택한다. - 일반 로그인: 유효하지 않은 아이디와 비밀번호를 입력한다. - 소셜 로그인: Step with me에 등록되지 않은 소셜 계정을 선택한다. 3. “로그인” 버튼을 클릭하거나 소셜 로그인의 경우 해당 소셜 플랫폼의 인증을 완료한다. 4.
Expected Results	“존재하지 않는 계정입니다.”라는 메시지가 표시되며 로그인에 실패한다.

Test Case Example ID	SWM002
Test Case Description	사용자가 비밀번호 없이 로그인 시도를 할 경우, 경고 메시지가 출력된다.
Preconditions	사용자가 Step with me에 계정을 보유하고 있으나 비밀번호 입력 없이 로그인을 시도한다.
Test Steps	1. 웹 브라우저를 열어 Step with me의 웹 페이지에 접속한다. 2. 아이디 필드에 유효한 이메일 주소를 입력하고 비밀번호 필드를 비워둔다. 3. 로그인 버튼을 클릭한다.
Expected Results	“비밀번호를 입력해 주세요”라는 경고 메시지가 표시되고 로그인이 진행되지 않는다.

Test Case Example ID	SWM003
Test Case Description	사용자가 유효한 코드 제출 후 올바른 결과를 받는다.
Preconditions	사용자가 Step with me의 문제 중 하나를 선택하여 실행할 수 있는 올바른 코드를 작성한다.
Test Steps	1. 사용자가 logic block에 올바르게 입력한다. 2. 입력한 코드에 대해 컴파일 및 분석을 진행한다. 3. 결과를 표시한다.
Expected Results	코드가 성공적으로 실행되며 올바른 출력과 함께 “정답입니다!”라는 메시지가 표시된다.

Test Case Example ID	SWM004
Test Case Description	사용자가 런타임 에러가 발생하는 코드를 입력하여 시각적 피드백을 받는다.
Preconditions	컴파일러 언어 (C, C++)를 선택했을 경우 컴파일이 가능해야 한다.
Test Steps	1. 사용자가 logic block에 부분적으로 잘못된 코드를 입력한다. 2. 입력한 코드에 대해 분석 및 오류 검출을 진행한다. 3. 입력한 코드에 대한 시각화를 실행한다. 4. 결과를 표시하고 사용자 피드백을 기다린다.
Expected Results	오류 발생 부분에 오류 메시지가 표시된다. 오류 발생 부분 직전까지 시각화를 통해 사용자 입력 코드의 알고리즘 흐름이 표시된다. 사용자가 오류 메시지에 대해 추가적인 자연어 피드백을 요청한다.

Test Case Example ID	SWM005
Test Case Description	사용자의 이전 기록이 존재하지 않을 경우 추천 문제를 요청하면, 메시지가 표시된다.
Preconditions	Step with me 데이터베이스에 사용자의 이전 실행 데이터가 없으며, 추천 문제 기록이 비어 있다.
Test Steps	1. 사용자 계정으로 로그인한다. 2. 추천 문제 탭으로 이동한다.
Expected Results	추천 문제 탭이 비어 있다는 메시지가 표시된다.

9 Development Plan

9.1 Objectives

시스템의 개발 환경 및 기술에 관해 설명한다.

9.2 Frontend Environment

9.2.1 JavaScript

웹 페이지에 동적 기능을 제공하는 객체 기반 언어로, 페이지의 상호작용을 실제로 구현한다. 사용자의 실제 동작, 웹 서버와의 통신을 가능하게끔 한다. 해당 애플리케이션에서는 웹 페이지를 매개로 한 사용자와 AI 모델 간의 상호작용에 사용되며, Next.js를 사용하여 구현한다.

9.2.2 HTML (HyperText Markup Language)

웹 페이지의 구조를 정의하는 언어로, 페이지의 콘텐츠를 나타내는 요소들을 마크업하고 사용자에게 표시되는 텍스트와 이미지, 비디오 등을 정의한다. JavaScript를 통해 구현된 기능을 사용자에게 나타내는 역할을 한다. 사용자의 입력을 받을 코드 입력 창, 제출 버튼, 피드백 출력 창 등의 정의에 사용된다. W3C 기준을 따른다.

9.2.3 CSS (Cascading Style Sheets)

HTML로 구조화된 콘텐츠의 스타일을 지정한다. 색상이나 글꼴, 애니메이션 등을 정의하여 인터페이스의 디자인과 사용성을 향상함으로써 사용자 경험에 기여한다. W3C에 의해 정의되어 있다.

9.3 BackEnd Environment

9.3.1 PostgreSQL

오픈 소스 관계형 데이터베이스 관리 시스템(RDBMS)으로, SQL을 사용하여 데이터를 관리하고 저장한다. 해당 애플리케이션에서는 사용자에게 제공되는 코딩 문제를 관리하고 각 코드의 세부 해결 등을 저장한다.

9.3.2 Amazon E2C (Amazon Elastic Compute Cloud)

클라우드를 기반으로 하는 가상 서버이며, 다양한 인스턴스를 선택할 수 있음과 동시에 AWS의 다른 서비스와 통합하여 높은 가용성을 지닌다. 또한, 필요에 따라 서버의 용량을 조정할 수 있어 유연한 리소스 관리가 가능하다. 해당 애플리케이션의 경우 AWS의 머신러닝 서비스를 이용하여 사용자의 이용 기록을 바탕으로 LLM을 학습시킬 수 있다.

9.3.3 WebSocket

HTTP/1을 사용하는 TCP 기반 프로토콜로, 이중 및 양방향 데이터 통신을 지원하여 다중 사용자의 안전한 서비스 사용에 적합하다. 지속적인 연결을 보장하여 유연하고 다양한 종류의 데이터 교환에 유리하다. 다만, scaling에 취약하며 안정성의 문제 등이 존재한다.

9.3.4 gRPC

API 아키텍처의 일종으로, 오픈 소스 RPC(Remote procedure call) 프레임워크이다. 데이터 전송에 HTTP/2 프로토콜과 protobuf(protocol buffers)를 사용하며, 양방향 통신을 지원하기 때문에 효율적인 통신이 가능하며, 웹에 적합하다. 또한, 기본적으로 SSL과 load balancing을 지원한다. 웹 브라우저에 따라 해당 애플리케이션에서는 WebSocket의 백업으로 사용될 수 있다.

9.3.5 CodeBERT

프로그래밍 언어를 위해 훈련된 LLM으로, 현재 Python, Java, JavaScript를 포함한 6개 코딩 언어를 지원한다. 자연어-프로그래밍 언어 데이터 쌍이 데이터셋에 포함되어 있어 두 언어 형식 간의 연결에 높은 이해도를 보이며, natural language code search를 비롯한 두 형식 간 전환에 높은 성능을 나타낸다.

9.4 Constraints

본 애플리케이션의 제약 사항으로는 다음이 존재한다.

- 외부 소프트웨어 사용 시 최대한 오픈 소스 소프트웨어를 사용한다.
- 서버의 응답은 1초 이내, AI 모델의 응답 시간은 2초 이내로 제한한다.
- 사용자 데이터 및 코드 실행 환경의 보안이 유지되어야 한다.
- 소프트웨어 구성 요소는 관련 저작권 및 라이선스를 준수하여 사용한다.
- 1.0 GHz CPU, 4GB RAM, 10Mbps 인터넷 속도의 최소 사양을 가정한다.
- 최신 웹 표준을 준수하는 브라우저(Chrome, Firefox, Safari, Edge)를 지원하며, Internet Explorer는 지원하지 않는다

9.5 Assumptions and Dependencies

해당 애플리케이션을 사용함에 있어 데스크탑 환경에서 최신 웹 표준을 준수하는 브라우저에서 접속함을 가정한다. 또한, 코드 학습을 목적으로 하는 만큼 코드를 입력하고 피드백을 확인할 수 있는 I/O 장치가 존재함을 가정한다. 위에서 명시된 최소 사양을 충족하지 않는 환경에서 사용할 경우 해당 애플리케이션의 사용에 어려움을 겪을 수 있다.