



Web Shooter

Design Specification

2024.12.1.

Introduction to Software Engineering 41
TEAM 1

Team Leader	이유진
Team Member	권서진
Team Member	김문수
Team Member	김현진
Team Member	박경일

CONTENTS

1. Preliminary.....	9
1.1 Readership	9
1.2 Scope	9
1.3 Objective.....	9
1.4 Document Structure	9
2. Introduction.....	11
2.1 Objective.....	11
2.2 Applied Diagrams	11
2.2.1 Used Tools	11
2.2.2 Diagram Types.....	11
2.2.3 Project Scope.....	11
2.2.4 References.....	12
3. System Architecture - Overall	13
3.1 Objectives	13
3.2 System Organization	13
3.2.1 System Diagram.....	14
3.3 Use Case Diagram	15
4. System Architecture - Frontend	16
4.1 Objective	16
4.2 Subcomponents.....	16
4.2.1 Authentication Page.....	16
4.2.1.1 State	16
4.2.1.2 Methods	17
4.2.1.3 Class Diagram	18
4.2.1.4 Sequence Diagram.....	19
4.2.2 Problem List	19
4.2.2.1 State	19
4.2.2.2 Methods	20
4.2.2.3 Class Diagram	20
4.2.2.4 Sequence Diagram.....	21
4.2.3 Code Editor	21
4.2.3.1 State	21
4.2.3.2 Methods	21
4.2.3.3 Class Diagram	22
4.2.3.4 Sequence Diagram.....	22
4.2.4 Live Preview	22
4.2.4.1 State	22
4.2.4.2 Methods	23
4.2.4.3 Class Diagram	23

4.2.4.4 Sequence Diagram	23
4.2.5 API Test Tool	23
4.2.5.1 State	24
4.2.5.2 Methods	24
4.2.5.3 Class Diagram	24
4.2.5.4 Sequence Diagram	25
4.2.6 Chat Bot	25
4.2.6.1 State	25
4.2.6.2 Methods	25
4.2.6.3 Class Diagram	26
4.2.6.4 Sequence Diagram	26
4.2.7 Grading Panel	26
4.2.7.1 State	27
4.2.7.2 Methods	27
4.2.7.3 Class Diagram	27
4.2.7.4 Sequence Diagram	28
4.2.8 Discussion Panel & CodeReview Panel	28
4.2.8.1 State	28
4.2.8.2 Methods	29
4.2.8.3 Class Diagram	29
4.2.8.4 Sequence Diagram	30
4.2.9 User Dashboard	30
4.2.9.1 State	30
4.2.9.2 Methods	31
4.2.9.3 Class Diagram	32
4.2.9.4 Sequence Diagram	33
4.2.10 Admin User Management	33
4.2.10.1 State	33
4.2.10.2 Methods	34
4.2.10.3 Class Diagram	35
4.2.10.4 Sequence Diagram	36
4.2.11 Admin Problem Management	36
4.2.11.1 State	37
4.2.11.2 Methods	37
4.2.11.3 Class Diagram	37
4.2.11.4 Sequence Diagram	38
5. System Architecture - Backend	39
5.1 Objectives	39
5.2 Overall architecture	39
5.3 Subcomponents	41
5.3.1 User system	41
5.3.1.1 state	41
5.3.1.2 methods	42
5.3.1.3 Class Diagram	42
5.3.1.4 Sequence Diagram	43
5.3.2 Problem System	46
5.3.2.1 State	46
5.3.2.2 Methods	47
5.3.2.3 Class Diagram	47

5.3.2.4 Sequence Diagram.....	48
5.3.3 Comment System.....	50
5.3.3.1 Sequence Diagram.....	50
5.3.4 Judge System	52
5.3.4.1 Sequence Diagram.....	52
5.3.5 LLM System	53
5.3.5.1 State.....	53
5.3.5.2 Methods.....	53
5.3.5.3 Class Diagram.....	54
5.3.5.4 Sequence Diagram.....	55
6. Protocol Design.....	57
6.1 Objective	57
6.2 HTTPS (Hypertext Transfer Protocol Secure)	57
6.3 CSRF (Cross-Site Request Forgery) Token	57
6.4 Session Handling	57
6.5 JSON (JavaScript Object Notation)	57
6.6 Authentication	58
6.6.1 Register.....	58
6.6.2 Login & Logout	59
6.6.2.1 Login	59
6.6.2.2 Logout.....	60
6.6.3 Reset Password (send email).....	61
6.6.4 Reset Password (verification code).....	62
6.6.5 Reset Password (send new password)	63
6.6.6 Submit Solution	64
6.6.7 Provide Problem	65
6.6.8 Judge Submission.....	66
6.6.9 Chatbot Interaction.....	67
6.6.10 Provide Problem Comments.....	68
6.6.11 Provide User Profile	69
6.6.12 Provide Learning Progress.....	70
6.6.13 Provide User Information	71
6.6.14 Provide Problem Comments.....	72
7. Database Design.....	73
7.1 Objectives.....	73
7.2 ER Diagram	73
7.2.1 User-Profile, User-Submission	73
7.2.2 Submission-Problem, Problem-Comment.....	74
7.2.3 Dashboard-Problem_statistics	75
7.3 Relational Schema.....	76
7.4 SQL DDL	76
7.4.1 User	76
7.4.2 Profile	77
7.4.3 Problem	77
7.4.4 Submission.....	77

7.4.5	Comment	78
7.4.6	Dashboard	78
7.4.7	Problem Statistic	78
8.	Testing Plan	79
8.1.	Objective.....	79
8.2.	Testing Policy	79
8.2.1.	Release Testing	79
8.2.1.1.	End-to-End Testing	79
8.2.1.2.	Performance Testing	83
8.2.1.3.	Compatibility Testing.....	87
8.2.1.4.	Security Testing	90
8.2.1.5.	Failover and Recovery Testing.....	92
8.2.2.	User Testing.....	94
8.2.2.1.	Beta Testing	94
8.2.2.2.	Error Tolerance Testing.....	95
8.2.2.3.	Interaction Testing.....	95
8.3	Manual Test Case Document example.....	97
9.	Development Plan	103
9.1	Objective.....	103
9.2	Frontend Environment	103
9.2.1	React.js	103
9.2.2	Next.js.....	103
9.2.3	TypeScript.....	104
9.2.4	상태 관리 테스트.....	104
9.3	Backend Environment.....	105
9.3.1	SpringBoot	105
9.3.2	MySQL	105
9.3.3	GitHub.....	105
9.4	Integration Testing	106
9.4.1	프론트엔드와 백엔드 통합 테스트	106
9.5	Automated Testing.....	106
9.5.1	테스트 도구 활용	106
10.	Supporting Information	108
10.1	Software design specification.....	108
10.2	Document history	108

LIST OF FIGURES

[Figure 1] Context Diagram - Overall	14
[Figure 2] Use Case Diagram	15
[Figure 3] Class diagram - Authentication Page	18
[Figure 4] Sequence Diagram - Authentication Page	19
[Figure 5] Class Diagram – Problem List.....	20
[Figure 6] Sequence Diagram – Problem List	21
[Figure 7] Class Diagram – Code Editor	22
[Figure 8] Sequence Diagram – Code Editor.....	22
[Figure 9] Class Diagram – Live Preview	23
[Figure 10] Sequence Diagram – Live Preview.....	23
[Figure 11] Class Diagram – API Test Tool.....	24
[Figure 12] Sequence Diagram – API Test Tool	25
[Figure 13] Class Diagram – Chat Bot	26
[Figure 14] Sequence Diagram – Chat Bot.....	26
[Figure 15] Class Diagram – Grading Panel.....	27
[Figure 16] Sequence Diagram – Grading Panel	28
[Figure 17] Class Diagram - Discussion Panel & CodeReview Panel	29
[Figure 18] Discussion Panel & CodeReview Panel	30
[Figure 19] Class Diagram - User Dashboard.....	32
[Figure 20] Sequence Diagram - User Dashboard.....	33
[Figure 21] Class Diagram - Admin User Management.....	35
[Figure 22] Sequence Diagram - Admin User Management	36
[Figure 23] Class Diagram - Admin Problem Management	37
[Figure 24] Sequence Diagram - Admin Problem Management	38
[Figure 25] overall architecture - backend.....	39
[Figure 26] Class Diagram – User System	42
[Figure 27] Sequence Diagram – User System : Auth	43
[Figure 28] Sequence Diagram – User System : change password	44
[Figure 29] Sequence Diagram – User System : upload profile image	45
[Figure 30] Sequence Diagram – User System : dashboard	45
[Figure 31] Sequence Diagram – User System : Admin User Mangement	46
[Figure 32] Class Diagram – Problem System	47
[Figure 33] Sequence Diagram – Problem System : Solving problem	48
[Figure 34] Sequence Diagram – Problem System : Sharing Problem.....	48
[Figure 35] Sequence Diagram – Problem System : Admin Problem Management	49
[Figure 36] Sequence Diagram – Comment System : User Comment	50
[Figure 37] Sequence Diagram – Comment System : Admin Comment Management	51
[Figure 38] Sequence Diagram – Judge System : Judge Problem	52
[Figure 39] Sequence Diagram – Judge System : API Test Tool	52
[Figure 40] Class Diagram – LLM System.....	54
[Figure 41] Sequence Diagram – LLM System : Provide Problem.....	55
[Figure 42] Sequence Diagram – LLM System : Problem Feedback	56
[Figure 43] Sequence Diagram – LLM System : User Q&A	56
[Figure 44] User-Profile, User-Submission	73
[Figure 45] Submission-Problem, Problem-Comment	74
[Figure 46] Dashboard-Problem_statistics	75

[Figure 47] Entity Relational Schema.....	76
[Figure 48] User SQL Schema.....	76
[Figure 49] Profile SQL Schema	77
[Figure 50] Problem SQL Schema.....	77
[Figure 51] Submission SQL Schema.....	77
[Figure 52] Comment SQL Schema	78
[Figure 53] Dashboard SQL Schema.....	78
[Figure 54] Problem Statistic SQL Schema.....	78
[Figure 55] testcase1 : 로그인 처리 시간 확인	97
[Figure 56] testcase2 : 비밀번호 재설정 및 이메일 인증 동작 확인	98
[Figure 57] testcase3: 문제 생성 시간 확인	99
[Figure 58] testcase4 : AI 챗봇 응답 시간 및 품질 확인.....	100
[Figure 59] testcase5: 채점 결과 및 피드백 제공	101
[Figure 60] testcase6 : 대시보드 업데이트 및 시각화 검증	102
[Figure 61] React.js	103
[Figure 62] Next.js	104
[Figure 63] TypeScript	104
[Figure 64] SpringBoot	105
[Figure 65] MySQL	105
[Figure 66] Github	106
[Figure 67] Jest	107
[Figure 68] JUnit.....	107
[Figure 69] Postman.....	107

LIST OF TABLES

[Table 1] Table of Register Request	58
[Table 2] Table of Register Response	58
[Table 3] Table of Login Request	59
[Table 4] Table of Login Response.....	59
[Table 5] Table of Logout Request	60
[Table 6] Table of Logout Response	60
[Table 7] Table of Reset Password(send email) Request.....	61
[Table 8] Table of Reset Password(send email) Response	61
[Table 9] Table of Reset Password(verification code) Request	62
[Table 10] Table of Reset Password(verification code) Response	62
[Table 11] Table of Reset Password(send new password) Request.....	63
[Table 12] Table of Reset Password(send new password) Response	63
[Table 13] Table of Submit Solution Request.....	64
[Table 14] Table of Submit Solution Response	64
[Table 15] Table of Provide Problem Request.....	65
[Table 16] Table of Provide Problem Response	65
[Table 17] Table of Judge Submission Request.....	66
[Table 18] Table of Judge Submission Response	66
[Table 19] Table of Chatbot Interaction Request.....	67
[Table 20] Table of Chatbot Interaction Response	67
[Table 21] Table of Problem Comments Request	68
[Table 22] Table of Problem Comments Response	68
[Table 23] Table of Provide User Profile Request	69
[Table 24] Table of Provide User Profile Response	69
[Table 25] Table of Provide Learning Progress Request	70
[Table 26] Table of Provide Learning Progress Request	70
[Table 27] Table of Provide User Information Request.....	71
[Table 28] Table of Provide User Information Response	71
[Table 29] Table of User Comments Response	72
[Table 30] Table of User Comments Response	72

1. Preliminary

해당 챕터는 본 문서의 예상 독자들, 문서의 개괄적 내용, 문서 목적, 문서 구조에 대한 간단한 서술을 제공한다.

1.1 Readership

본 문서는 다음과 같은 독자들을 대상으로 한다.

- 시스템 설계 (1조 학생)
- 시스템 개발자 (가상 독자)
- 소프트웨어 공학 개론 팀프로젝트 평가자 (교수, 조교)

1.2 Scope

본 문서는 요구사항 명세서에 기입한 요구사항들에 기반하여 WebShooter 실습 프로그램의 구현의 기반이 되는 소프트웨어의 프론트엔드, 백엔드, 데이터베이스 측면에서의 설계를 정의한다.

1.3 Objective

본 문서는 요구사항 명세서에 기입한 요구사항들을 바탕으로 WebShooter에 대한 기술적 설계사항들을 명확히 기술하고, 이를 바탕으로 추후 구현 및 유지보수를 진행 할 수 있도록 작성되었다.

1.4 Document Structure

본 디자인 명세서는 WebShooter 시스템을 앞선 요구사항을 수행할 수 있도록 그에 기반하여 설계사항을 서술한다. 본 문서는 다음과 같은 주요 챕터들로 구성되어 있다.

Chapter 1. Preliminary : 해당 챕터는 본 문서의 예상 독자들, 문서의 개괄적 내용, 문서 목적, 문서 구조에 대한 간단한 서술을 제공한다.

Chapter 2. Introduction : 해당 챕터는 프로젝트 구현에 있어서 기반이 될수 있는 핵심 기능, 설계 목표 등 기본적인 설계사항을 제공한다.

Chapter 3. System Architecture - Overall : 해당 챕터는 프론트엔드 설계에서부터 백엔드 설계까지의 프로젝트 전반의 시스템에 대한 상위 아키텍처와 주요 컴포넌트 구성에 대한 정보를 제공한다.

Chapter 4. System Architecture - Frontend : 해당 챕터는 사용자가 직접적으로 상호작

용하는 프론트엔드 시스템의 구조와 속성 및 기능을 설명하고 설계에 있어 각 컴포넌트 간의 관계 등 구체적인 기술 스택에 대한 정보를 제공한다.

Chapter 5. System Architecture - Backend : 해당 챕터는 백엔드 시스템의 구조와 서버 설계에 대한 정보를 제공한다.

Chapter 6. Protocol Design : 해당 챕터는 프론트엔드와 서버 사이의 프로토콜을 정의하고 각 인터페이스에 대한 구체적인 설계사항을 제공한다.

Chapter 7. Database Design : 해당 챕터는 시스템 데이터 구조와 데이터베이스의 구현 내용, 구체적 설계 사항에 대한 정보를 제공한다.

Chapter 8. Testing Plan : 해당 챕터는 시스템의 development testing, release testing 및 user testing에 대한 테스트 계획 및 구체적 테스트 사항에 대한 정보를 제공한다.

Chapter 9. Development Plan : 해당 챕터는 시스템의 개발 환경 및 도구에 관한 정보를 제공한다.

Chapter 10. Supporting Information : 해당 챕터는 본 문서의 작성자 정보 및 디자인 명세서의 작성양식에 대한 정보를 제공한다.

2. Introduction

2.1 Objective

해당 챕터는 시스템 설계에 사용된 다이어그램, 툴에 대해 서술하고 개발 범위에 대한 정보를 제공한다.

2.2 Applied Diagrams

2.2.1 Used Tools

본 문서에서 사용하고 있는 다이어그램은 온라인 기반 다이어그램 작성 툴인 draw.io을 통해 제작되었다.

2.2.2 Diagram Types

1) Use Case Diagram : Use Case Diagram은 시스템의 주요 기능과 사용자에 대한 정보를 담고 있고, 시스템이 사용자와 상호작용하는 부분에 대한 다이어그램이다. 특히 시스템의 주요 기능을 사용자가 어떻게 이용할 수 있는지에 초점을 맞추고 있다.

2) Class Diagram : Class Diagram은 시스템을 구성하는 클래스, 그 속성, 기능, 그리고 객체들 간의 관계를 표현하여 시스템의 정적인 구조를 보여주며 객체지향 설계의 핵심 요소를 시각화하고, 클래스 간의 상호작용과 종속성을 이해하는 데 도움을 주는데 그 목적이 있다.

3) Sequence Diagram : Sequence Diagram은 시간 순서에 따라 행동별로 객체가 어떻게 상호작용하는지를 나타내는 다이어그램으로, 특정 시나리오와 관련된 객체들(사용자, 서버, 클라이언트) 간의 메시지 교환을 순서대로 표현한다.

4) Context Diagram : Context Diagram은 시스템 전체와 외부 요인간의 입력 및 출력을 시각화하며 시스템과 해당 시스템과 상호작용할 수 있는 모든 외부 엔터티를 나타내며, 그 사이의 Data Flow을 표현한다.

5) Entity Relationship Diagram : Entity Relationship Diagram은 구조화된 데이터와 그들 간의 관계를 사람이 이해할 수 있는 형태로 표현하며, 현실의 요구사항을 기반으로 데이터베이스 설계 과정에서 활용되며, 데이터베이스의 논리적 구조를 설명하는 데 사용된다.

2.2.3 Project Scope

본 문서에서 설계하는 WebShooter 실습 프로그램은 웹 개발 학습을 희망하는 모든 사람을 대상으로 하며, LLM을 이용하여 프론트엔드부터 백엔드까지 웹 전반에 걸친 실습을 제공하는 것을 목표로 한다. 사용자는 LLM을 통해 자신이 원하는 유형

과 난이도에 맞는 학습 문제를 생성하여 실시간 실습 환경에서 웹 개발을 단계별로 학습할 수 있도록 설계되었다. 사용자는 실습을 통해 웹 개발의 기초부터 고급 문제 해결 능력까지 점진적으로 향상시킬 수 있으며, 학습 진척도는 데이터베이스에 저장된 문제 풀이 기록을 통해 관리되도록 기획하였다.

2.2.4 References

AWS Documentation : <https://docs.aws.amazon.com/>

OpenAI API Documentation : <https://platform.openai.com/docs/overview>

IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements

3. System Architecture - Overall

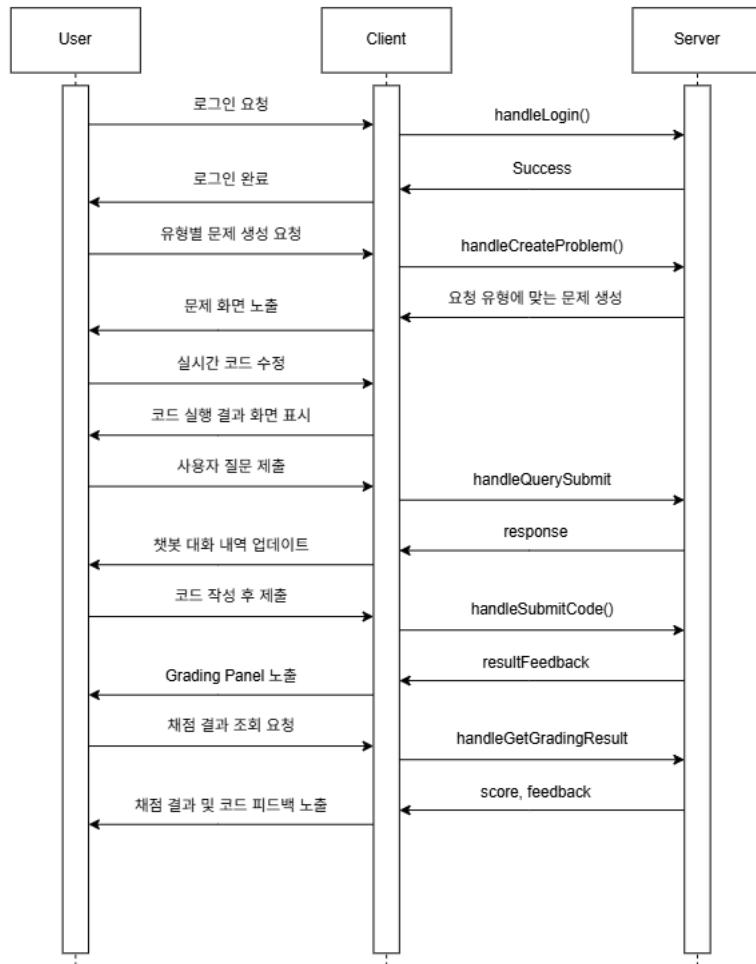
3.1 Objectives

해당 챕터는 프론트엔드 설계에서부터 백엔드 설계까지의 프로젝트 전반의 시스템에 대한 상위 아키텍처와 주요 컴포넌트 구성에 대한 정보를 제공한다. 서비스는 사용자가 입력한 웹 프로그래밍 키워드를 기반으로 실시간으로 프로그래밍 문제를 생성하고, 이를 해결하기 위한 힌트를 제공하며, 학습 진행 상황 추적 등 다양한 기능을 통해 사용자에게 맞춤형 학습 환경을 제공하는 것을 목표로 한다.

3.2 System Organization

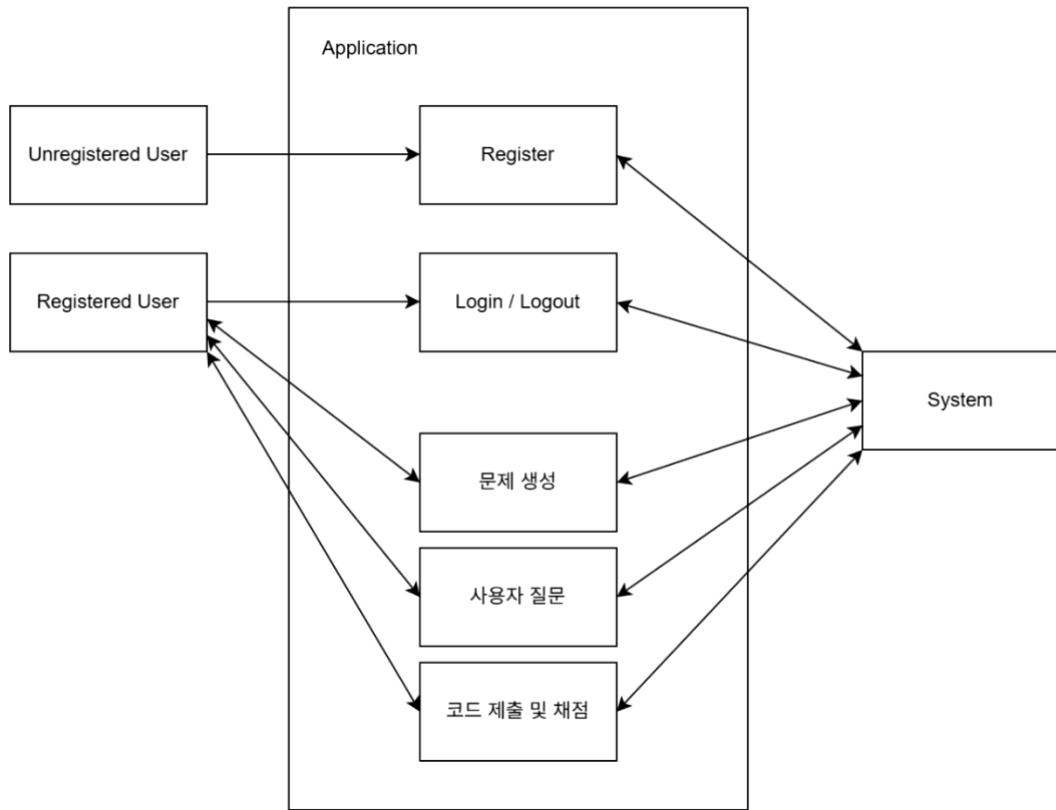
이 시스템은 클라이언트-서버 모델을 적용하여 설계되었다. 전체 아키텍처는 프론트엔드 어플리케이션과 백엔드 어플리케이션으로 구분되며, 이들은 JSON 기반의 HTTPS 통신을 통해 데이터를 전달한다. 프론트엔드 어플리케이션은 사용자와의 모든 상호작용을 담당하며, 사용자에게 학습 환경을 제공한다. 사용자가 웹 프로그래밍 키워드를 선택하고, 생성된 문제를 풀며, 문제 해결에 필요한 힌트를 요청할 수 있는 사용자 인터페이스를 제공한다. 백엔드와 JSON 형식의 HTTPS 요청을 주고받아 데이터를 처리하고, 사용자에게 결과를 표시한다. 백엔드 어플리케이션은 사용자 요청 처리와 데이터 관리를 담당하며, 시스템 설계 명세에 따라 프론트엔드로부터 요청을 받아 해당 기능을 실행한다. 문제 생성 및 힌트 제공을 위한 LLM 기반 서비스를 실행하며, 문제를 해결하기 위한 코드나 분석 결과를 처리한다. 백엔드는 사용자가 제출한 코드를 받아 실행하고, 그 실행 결과를 JSON 형식으로 프론트엔드로 반환한다. 실행 결과는 정확도와 정답 여부를 포함하며, 이 데이터는 MySQL 데이터베이스에 기록되어 이후 업데이트된 학습 정보를 사용자에게 제공할 수 있도록 한다. DB 시스템에서는 MySQL3를 사용하여 사용자 계정 정보, 문제 기록, 코드 실행 결과 등을 저장한다. 사용자는 개인 계정을 생성하여 시스템을 이용할 수 있으며, 계정 생성 시 기존 데이터베이스와의 중복 체크를 거친 후 새로운 계정이 생성된다. 중복 계정 방지를 위해 계정 생성 시 기존 사용자 정보와 대조 작업을 진행하고, 새로운 계정만을 등록한다. 프로토콜은 모든 데이터는 JSON 형식으로 전달되며, 서버는 해당 정보를 처리하고 필요한 데이터를 데이터베이스에서 조회하여 응답을 제공한다. 서버는 문제 풀이 결과와 관련된 정보를 지속적으로 업데이트하여, 사용자에게 최신 정보를 제공하며 학습을 지원합니다.

3.2.1 System Diagram



[Figure 1] Context Diagram - Overall

3.3 Use Case Diagram



[Figure 2] Use Case Diagram

4. System Architecture - Frontend

4.1 Objective

본 챕터에서는 플랫폼의 핵심 기능을 담당하는 Subcomponents에 대해 다룬다. 이 Subcomponents는 사용자 인증, 문제 관리, 코드 작성 및 실행 환경 제공, API 테스트, 챗봇 상호작용 등 다양한 작업을 수행하며, 백엔드 서비스와의 연동을 통해 원활한 사용자 경험을 제공한다.

4.2 Subcomponents

4.2.1 Authentication Page

사용자가 플랫폼에 접근할 수 있도록 계정을 생성하고, 로그인, 로그아웃, 비밀번호 조회 관리하는 기능을 제공한다. 이 기능은 사용자의 신원을 확인하고, 계정과 관련된 중요한 정보를 보호하기 위해 필요하다.

4.2.1.1 State

Session:

- **session**: NextAuth 의 useSession 흑을 통해 얻는 현재 사용자 세션 정보. 로그인된 사용자에 대한 정보를 담고 있으며, 로그인 상태에서 authenticated로 설정되고, 로그아웃 시 unauthenticated로 변경된다.

Register

- **email**: 사용자가 입력한 이메일 주소
- **username**: 사용자가 입력한 사용자 이름
- **password**: 사용자가 입력한 비밀번호
- **confirmPassword**: 비밀번호 확인을 위한 입력 값
- **interestLanguage**: 사용자가 관심 있는 프로그래밍 언어
- **interestField**: 사용자가 관심 있는 기술 분야
- **error**: 회원가입 중 발생한 오류 메시지

Login

- **email**: 사용자가 입력한 이메일 주소
- **password**: 사용자가 입력한 비밀번호
- **error**: 로그인 중 발생한 오류 메시지

Logout

- **error:** 로그아웃 중 발생한 오류 메시지

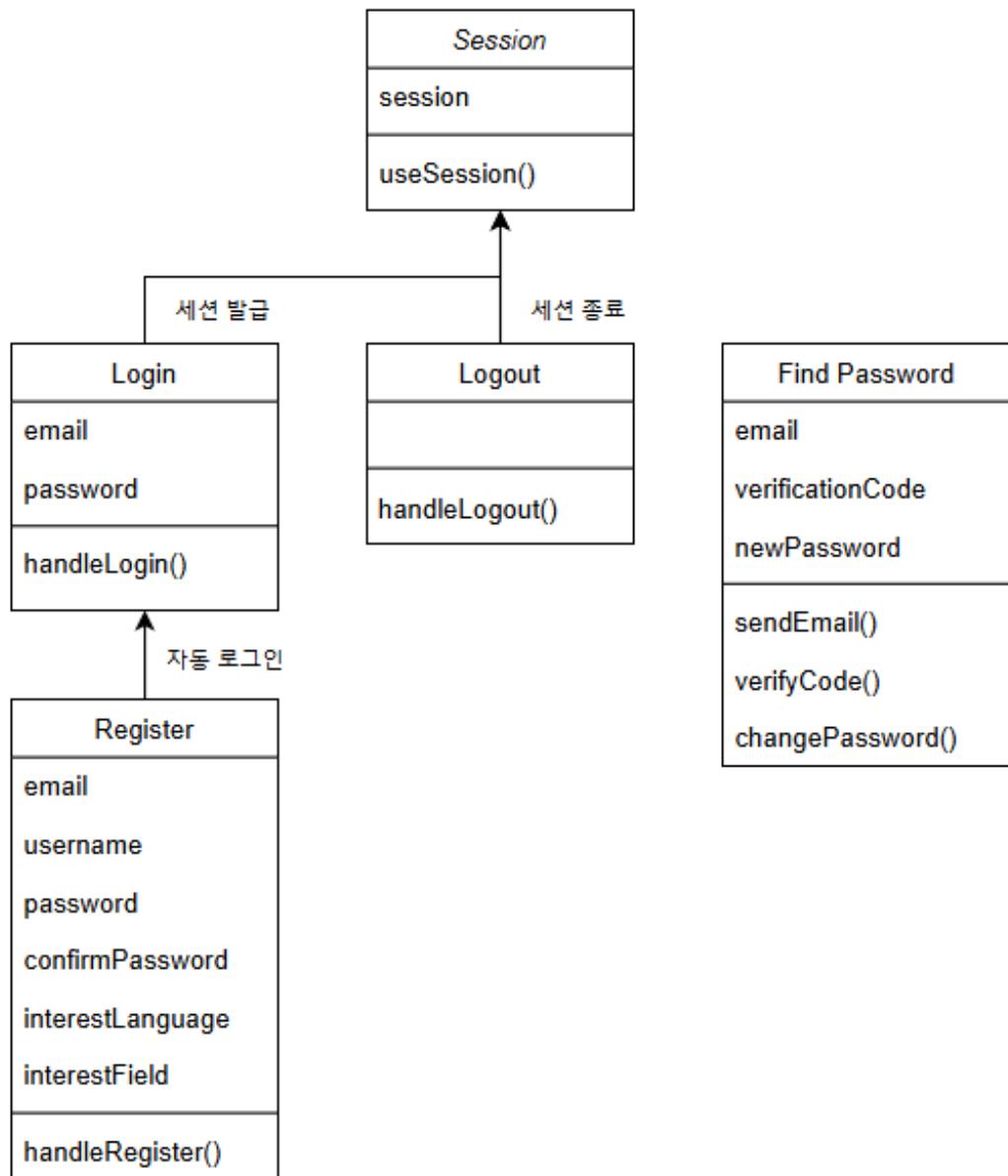
Find Password

- **email:** 사용자가 입력한 이메일 주소
- **verificationCode:** 사용자가 입력한 인증 코드
- **newPassword:** 사용자가 입력한 새로운 비밀번호
- **error:** 비밀번호 변경 중 발생한 오류 메시지

4.2.1.2 Methods

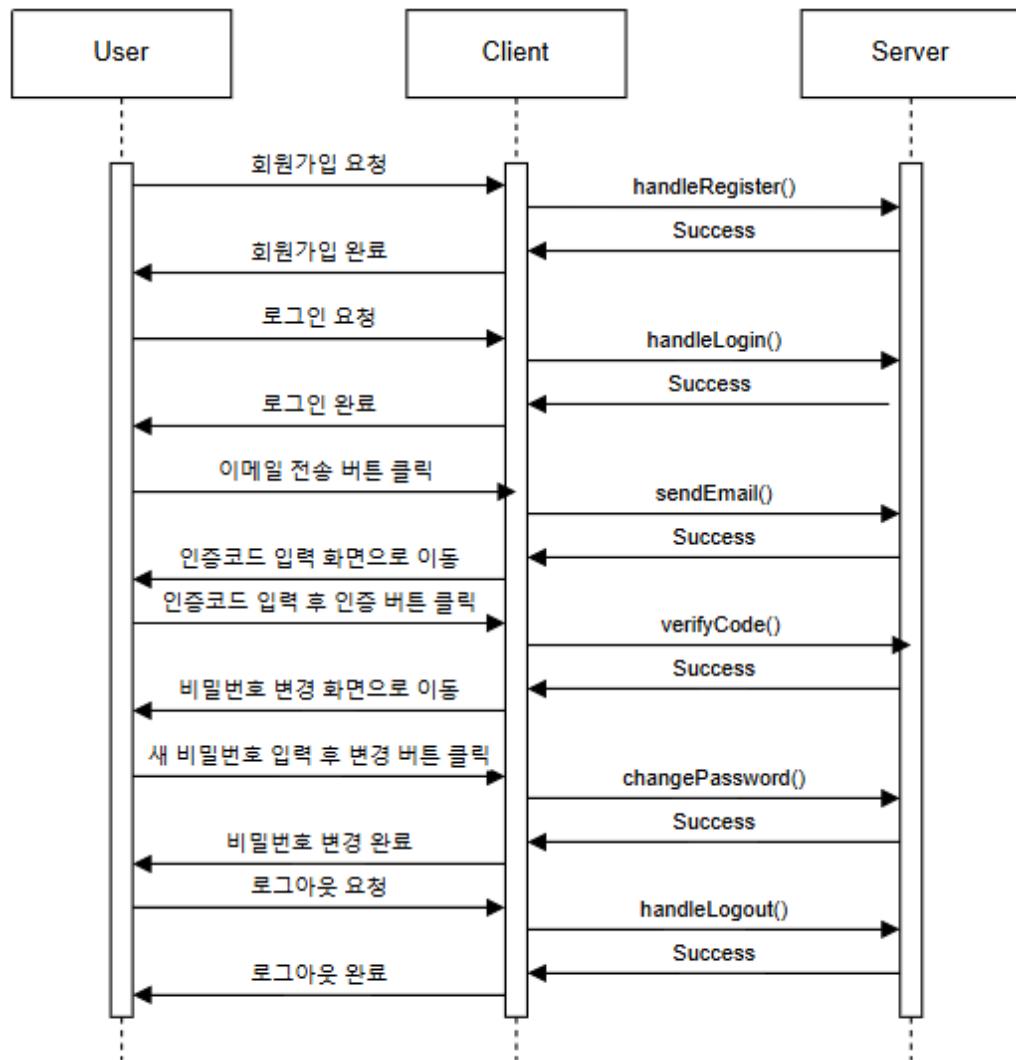
- **handleRegister:** 사용자 입력값 검증 후, 서버로 회원가입 요청을 보낸다. 성공 시, NextAuth 의 signIn 메소드를 사용하여 자동 로그인 처리를 한다. NextAuth 의 세션 관리를 위해 signIn 을 사용하여 로그인한 사용자 세션을 활성화시킨다.
- **handleLogin:** 로그인 시, 사용자가 입력한 이메일과 비밀번호를 NextAuth 의 signIn 메소드를 통해 서버로 전송한다. 인증이 성공하면 NextAuth 에서 세션을 자동으로 관리하며, useSession 혹은 사용하여 세션 상태를 확인할 수 있다. 실패 시 error 상태를 설정하여 오류 메시지를 사용자에게 제공한다.
- **handleLogout:** 로그아웃 시 NextAuth 의 signOut 메소드를 사용하여 세션을 종료하고, 로그인된 상태를 useSession 을 통해 확인할 수 없다. 로그아웃 성공 후, 사용자 인터페이스를 갱신하여 로그아웃 상태로 UI 를 업데이트한다.
- **sendEmail:** 사용자가 이메일 주소를 입력 후 전송 버튼을 눌렀을 때, 이메일을 서버로 전송한다. 서버에서는 사용자가 입력한 이메일 주소로 verificationCode 를 발송한다.
- **verifyCode:** 사용자가 이메일로 받은 verificationCode 를 입력하고, 인증 버튼을 누르면 인증코드를 서버로 전송한다. 서버에서는 사용자가 입력한 인증코드의 유효성 검사를 한다
- **changePassword:** 사용자가 새로운 비밀번호를 입력하여, 변경 버튼을 눌렀을 때, 서버로 새로운 비밀번호를 전송하여 계정의 비밀번호를 변경한다.

4.2.1.3 Class Diagram



[Figure 3] Class diagram - Authentication Page

4.2.1.4 Sequence Diagram



[Figure 4] Sequence Diagram - Authentication Page

4.2.2 Problem List

사용자가 원하는 유형과 난이도에 맞는 문제를 생성하고 조회할 수 있는 기능을 제공하는 컴포넌트이다. 사용자는 문제의 유형과 난이도를 설정하여, 해당 조건에 맞는 문제를 생성하거나, 기존에 등록된 문제들을 조회할 수 있다.

4.2.2.1 State

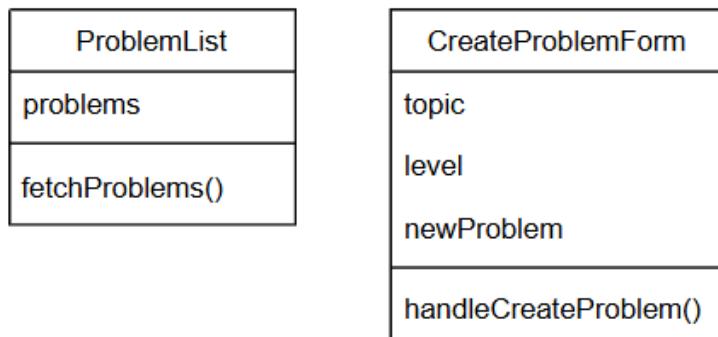
- **problems**: 등록된 모든 문제들의 목록이다. 서버에서 가져온 문제 데이터를 저장하며, 문제 조회 시 사용된다.
- **topic**: 사용자가 설정한 문제의 주제
- **level**: 사용자가 설정한 문제의 난이도

- **newProblem**: 사용자가 새 문제를 생성할 때 입력한 문제 제목, 설명, 난이도, 유형 등의 정보를 담는 상태이다.
- **error**: 문제 조회 또는 생성 시 오류가 발생하면 오류 메시지를 담는 상태이다

4.2.2.2 Methods

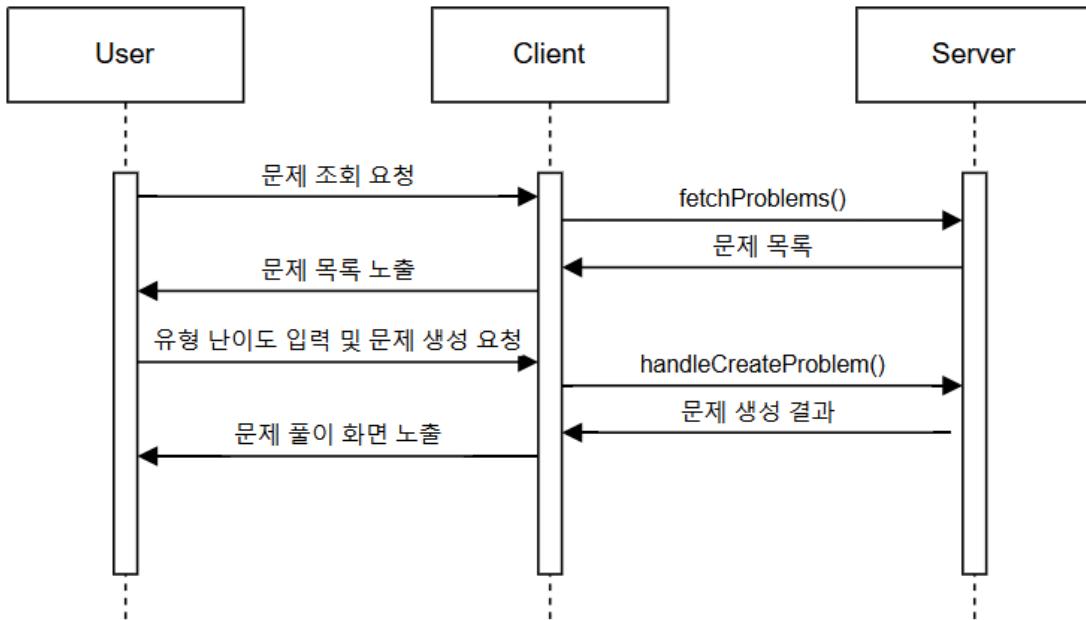
- **fetchProblems**: 서버에서 문제 목록을 가져오는 메소드이다. 서버에 GET 요청을 보내어 모든 문제를 조회한다. 성공 시, 가져온 문제 목록을 problems 상태에 저장한다. 실패 시, error 상태에 오류 메시지를 저장한다.
- **handleCreateProblem**: 사용자가 입력한 유형, 난이도에 맞는 문제를 생성하는 메소드이다. 서버에 POST 요청을 보내어 새 문제를 생성한다. 성공 시, 문제 풀이 화면으로 리디렉션한다. 사용자가 생성한 문제를 바로 풀 수 있게 한다. 실패 시, error 상태에 오류 메시지를 표시한다.

4.2.2.3 Class Diagram



[Figure 5] Class Diagram – Problem List

4.2.2.4 Sequence Diagram



[Figure 6] Sequence Diagram – Problem List

4.2.3 Code Editor

사용자가 문제를 해결하기 위해 코드를 작성하고 제출할 수 있는 기능을 제공하는 컴포넌트이다. 이 컴포넌트는 코드 작성, 코드 제출, 실시간 코드 실행 등을 지원하며, 문제 풀이 환경을 제공한다. 사용자는 코드 에디터를 통해 문제를 해결하고, 이를 제출하여 채점 결과를 받을 수 있다. 이 컴포넌트는 CodeMirror 라이브러리를 사용하여 코드 작성 편의성을 극대화하며, 코드 하이라이팅, 자동 완성 등의 기능을 제공한다.

4.2.3.1 State

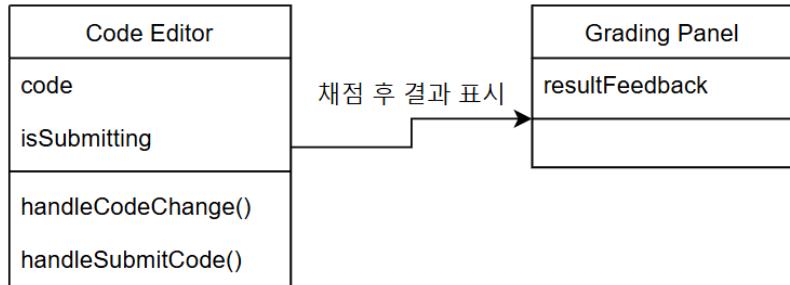
- **code**: 사용자가 작성한 코드를 저장하는 상태이다. 이 상태는 코드 에디터에서 실시간으로 업데이트된다.
- **error**: 코드 제출 또는 실행 중 발생한 오류 메시지를 저장하는 상태이다. 코드 제출 시 발생한 서버 오류나 실행 오류 메시지를 표시한다.
- **isSubmitting**: 코드 제출 상태를 나타내는 불리언 값이다. 코드 제출 버튼을 클릭하면 이 값이 true로 설정되어 제출 요청 중임을 나타낸다.
- **resultFeedback**: 제출된 코드에 대한 피드백을 저장하는 상태이다. 채점 결과나 코드에 대한 피드백을 받아서 이 상태에 저장한다.

4.2.3.2 Methods

- **handleCodeChange**: 사용자가 코드 에디터에서 코드를 작성할 때마다 호출되는 메소드이다. 사용자가 입력한 코드를 code 상태에 업데이트한다.

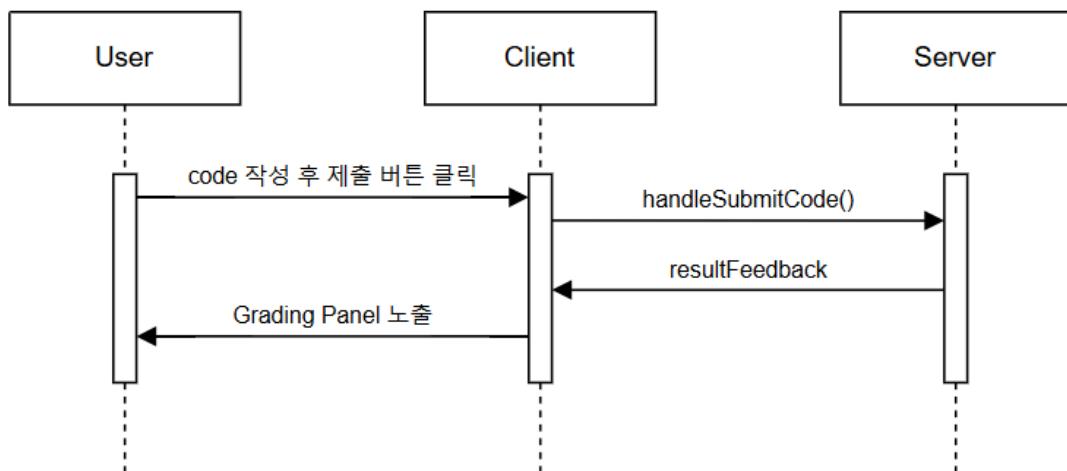
- **handleSubmitCode**: 사용자가 코드를 제출할 때 호출되는 메소드이다. 코드가 제출되면 `isSubmitting` 상태를 `true`로 설정하고, 서버로 코드 제출 요청을 보낸다. 제출 결과는 `resultFeedback` 상태에 저장된다.

4.2.3.3 Class Diagram



[Figure 7] Class Diagram – Code Editor

4.2.3.4 Sequence Diagram



[Figure 8] Sequence Diagram – Code Editor

4.2.4 Live Preview

사용자가 작성한 코드를 실시간으로 실행하여 결과를 확인할 수 있는 환경을 제공한다. 이 기능은 react-runner 라이브러리를 사용하여 구현된다. 사용자는 코드 에디터에서 코드를 작성하고, 이를 실시간으로 실행해 결과를 확인할 수 있다. 실행된 결과는 사용자에게 즉시 반영되어 코드 수정이 즉각적으로 피드백을 받을 수 있다.

4.2.4.1 State

- **code**: 사용자가 작성한 코드를 저장하는 상태로, Code Editor에서 작성한 코드 상태를 그대로 공유하여 실시간으로 실행된다.

- **error**: 코드 실행 중 발생한 오류 메시지를 저장하는 상태이다. 코드 실행 시 발생한 오류 메시지가 이 상태에 저장된다.

4.2.4.2 Methods

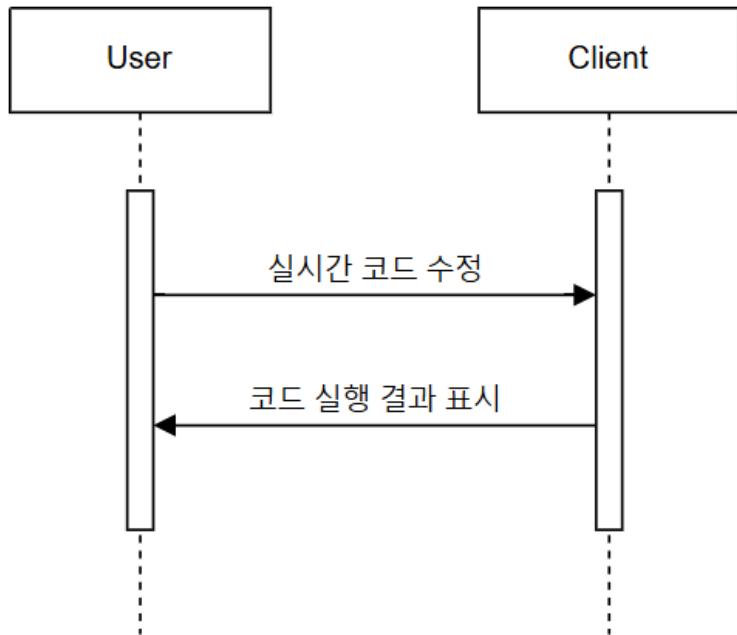
- **handleCodeChange**: Code Editor에서 사용자가 입력한 코드 변경을 code 상태에 업데이트하는 메소드를 공유한다. Live Preview에서도 동일한 방식으로 코드 변경을 실시간으로 반영한다.
- **handleRendered**: react-runner에서 코드가 렌더링된 후 실행되는 콜백 함수. 코드 실행 후 DOM이 갱신된 시점에 호출되어, 렌더링 결과에 대한 후속 작업이나 상태 업데이트를 처리하는 데 사용된다.

4.2.4.3 Class Diagram



[Figure 9] Class Diagram – Live Preview

4.2.4.4 Sequence Diagram



[Figure 10] Sequence Diagram – Live Preview

4.2.5 API Test Tool

사용자가 API 요청을 보내고, 그에 대한 응답을 조회할 수 있는 기능을 제공하는

컴포넌트이다. 사용자는 요청을 보내기 위해 필요한 body, headers, endpoint를 설정하고, 해당 요청을 백엔드로 전송한다. 백엔드는 요청을 처리한 후, 그에 대한 응답을 반환하고, 사용자는 이를 실시간으로 확인할 수 있다.

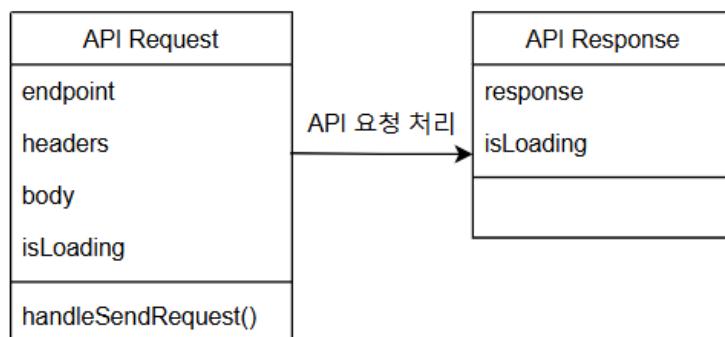
4.2.5.1 State

- **endpoint**: 사용자가 요청을 보낼 API의 엔드포인트 URL을 저장하는 상태이다. 사용자는 이 값을 입력하여 API 요청의 대상 URL을 설정한다.
- **headers**: API 요청에 포함될 헤더를 저장하는 상태이다. 예를 들어, Content-Type, Authorization 등의 헤더를 설정할 수 있다.
- **body**: API 요청의 본문 데이터를 저장하는 상태이다. JSON 형식으로 데이터를 입력하고, 이 데이터를 백엔드로 전송한다.
- **response**: API 요청 후 받은 응답 데이터를 저장하는 상태이다. 서버로부터 반환된 데이터를 사용자에게 표시한다.
- **error**: API 요청 중 발생한 오류 메시지를 저장하는 상태이다. 요청이 실패하거나 오류가 발생한 경우, 이 상태에 오류 메시지를 표시한다.
- **isLoading**: API 요청이 진행 중인지 여부를 나타내는 불리언 값이다. 요청이 진행 중일 때 true로 설정된다.

4.2.5.2 Methods

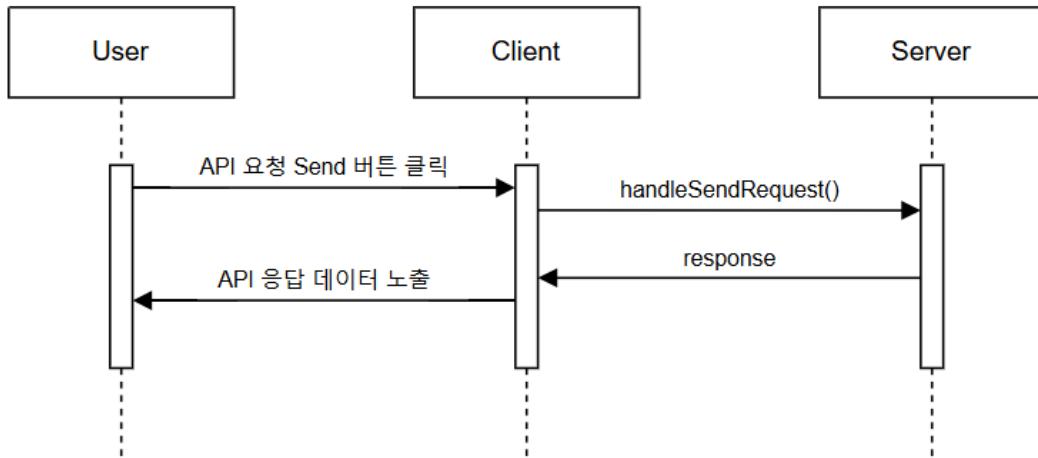
- **handleSendRequest**: 사용자가 Send 버튼을 클릭할 때 호출되는 메소드이다. 이 메소드는 endpoint, headers, body 상태를 서버로 전송하여 API 요청을 처리하고, 응답을 받아 response 상태에 저장한다. 요청 중에는 isLoading을 true로 설정하고, 완료되면 false로 설정한다.

4.2.5.3 Class Diagram



[Figure 11] Class Diagram – API Test Tool

4.2.5.4 Sequence Diagram



[Figure 12] Sequence Diagram – API Test Tool

4.2.6 Chat Bot

사용자가 입력한 질문에 대해 챗봇이 응답을 제공하는 기능을 제공하는 컴포넌트이다. 사용자는 질문을 입력하고, 챗봇은 해당 질문에 대한 답변을 반환한다. 질문과 답변은 1대1로 매칭되어 저장되며, 모든 질문과 답변은 배열에 저장되어 화면에 순차적으로 표시된다.

4.2.6.1 State

- **query**: 사용자가 입력한 질문을 저장하는 상태이다. 사용자가 챗봇에 질문을 입력하고 제출하면 이 상태에 저장된다.
- **response**: 챗봇이 제공한 답변을 저장하는 상태이다. 챗봇이 반환한 답변을 이 상태에 저장하고, 질문에 대한 응답을 화면에 표시한다.
- **messages**: 질문과 답변을 순차적으로 저장하는 배열이다. query 와 response 를 포함하는 객체가 배열에 저장되며, 이 배열은 사용자와 챗봇의 대화를 저장하고 화면에 표시하는 데 사용된다.
- **isLoading**: 질문을 제출한 후 챗봇의 응답이 올 때까지의 대기 상태를 나타내는 불리언 값이다. 응답이 도착하기 전에는 true, 응답을 받은 후에는 false 로 설정된다.
- **error**: 챗봇 응답을 처리하는 도중 발생한 오류 메시지를 저장하는 상태이다. 예를 들어, 서버 요청 실패나 네트워크 오류가 발생할 경우 이 상태에 오류 메시지가 저장된다.

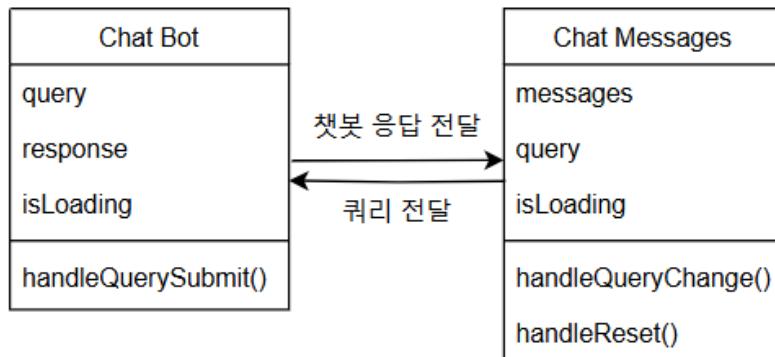
4.2.6.2 Methods

- **handleQuerySubmit**: 사용자가 질문을 제출할 때 호출되는 메소드이다. 이 메소드는 사용자가 입력한 query 를 messages 배열에 추가하고, 챗봇

API 를 통해 답변을 요청한다. 답변이 오면 response 상태에 저장되고, messages 배열에 질문과 답변을 함께 저장한다.

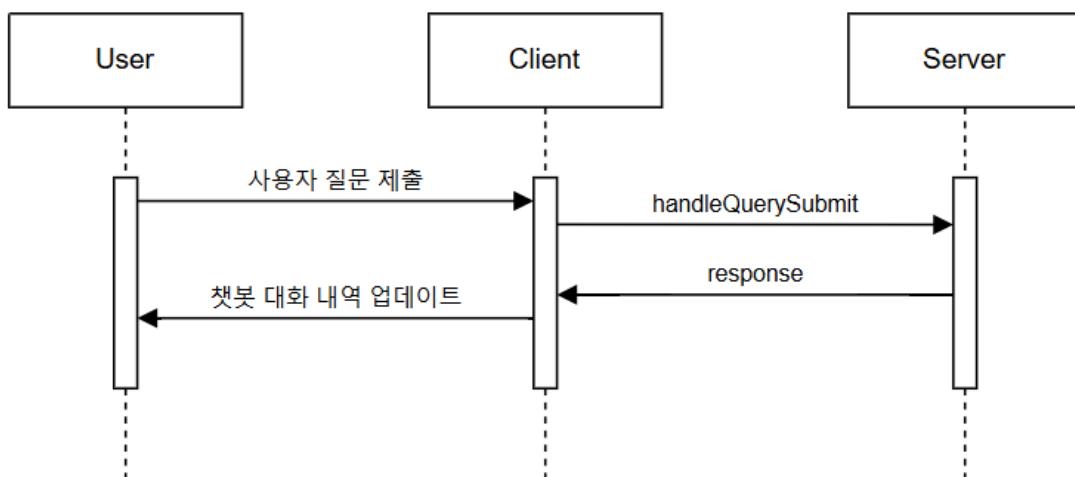
- **handleQueryChange:** 사용자가 질문을 입력할 때마다 호출되는 메소드이다. 이 메소드는 사용자가 입력한 질문을 query 상태에 업데이트한다.
- **handleReset:** 사용자가 대화를 초기화하고 싶을 때 호출되는 메소드이다. messages 배열을 초기화하여 대화 내역을 비운다.

4.2.6.3 Class Diagram



[Figure 13] Class Diagram – Chat Bot

4.2.6.4 Sequence Diagram



[Figure 14] Sequence Diagram – Chat Bot

4.2.7 Grading Panel

사용자가 제출한 코드에 대한 채점 결과와 피드백을 확인하는 기능을 제공하는 컴포넌트이다. 이 컴포넌트는 코드 제출 후 채점 결과를 보여주며, 해당 코드에 대한 개선 사항이나 피드백을 사용자에게 전달한다. 또한, 사용자는 문제가 제출된 후 해당 문제를 등록할 수도 있다.

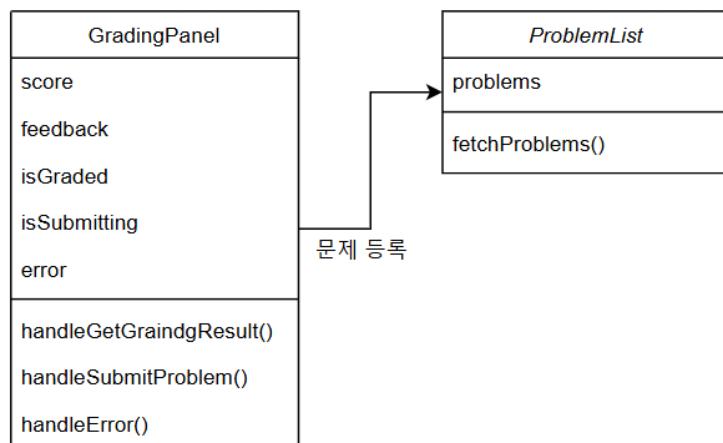
4.2.7.1 State

- **score**: 제출된 코드의 채점 점수를 저장하는 상태이다. 채점된 코드에 대한 점수 결과를 이 상태에 저장하여 표시한다.
- **feedback**: 제출된 코드에 대한 피드백을 저장하는 상태이다. 채점 결과와 함께 제공되는 피드백을 이 상태에 저장하고, 사용자가 이를 확인할 수 있도록 한다.
- **isGraded**: 코드가 채점되었는지 여부를 나타내는 불리언 값이다. 채점이 완료되면 true로 설정되고, 그렇지 않으면 false로 설정된다.
- **isSubmitting**: 문제를 등록할 때의 상태를 나타내는 불리언 값이다. 문제 등록 요청이 진행 중일 때 true로 설정되고, 등록이 완료되면 false로 설정된다.
- **error**: 채점 또는 문제 제출 중 발생한 오류 메시지를 저장하는 상태이다. 서버 요청 실패나 네트워크 오류가 발생하면 이 상태에 오류 메시지가 저장된다.

4.2.7.2 Methods

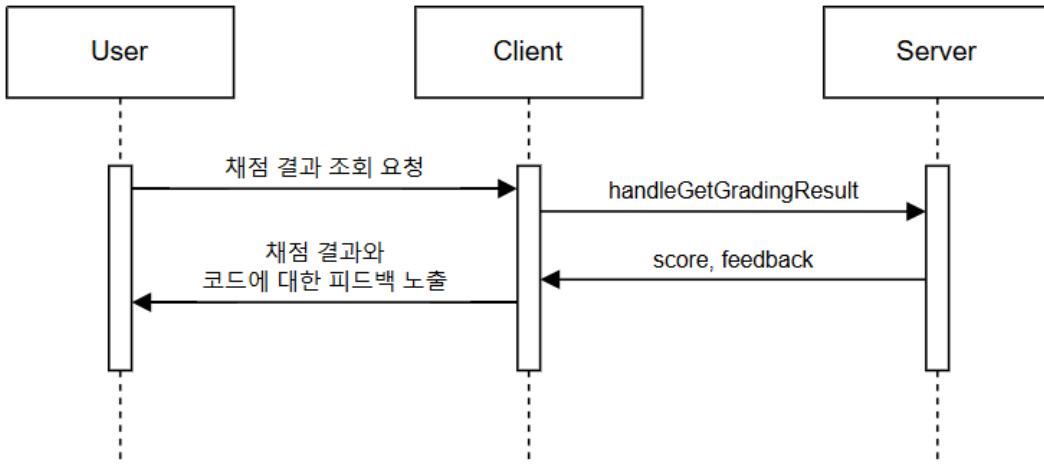
- **handleGetGradingResult**: 제출된 코드의 채점 결과를 서버에서 받아오는 메소드이다. 서버에 채점 결과를 요청하고, 그 결과를 score와 feedback 상태에 저장한다. 채점이 완료되면 isGraded를 true로 설정한다.
- **handleSubmitProblem**: 사용자가 새로운 문제를 등록할 때 호출되는 메소드이다. 문제 제목, 설명, 난이도 등을 포함한 문제 정보를 서버로 제출하고, 문제가 성공적으로 등록되면 해당 문제를 문제 목록에 추가한다.
- **handleError**: 채점이나 문제 제출 중 발생한 오류를 처리하는 메소드이다. 오류 메시지를 error 상태에 저장하여 사용자에게 해당 오류를 알려준다.

4.2.7.3 Class Diagram



[Figure 15] Class Diagram – Grading Panel

4.2.7.4 Sequence Diagram



[Figure 16] Sequence Diagram – Grading Panel

4.2.8 Discussion Panel & CodeReview Panel

사용자가 코드나 문제에 대해 댓글(코드리뷰)을 달고, 다른 사용자들의 댓글(코드리뷰)를 조회할 수 있는 기능을 제공하는 컴포넌트이다. 사용자는 다른 사람들의 댓글(코드리뷰)를 보고 자신의 의견을 댓글(코드리뷰)로 남길 수 있으며, 기존 댓글(코드리뷰)을 삭제할 수도 있다. 댓글(코드리뷰) 목록은 서버에서 가져와 화면에 표시하며, 사용자는 새 댓글(코드리뷰)을 추가할 때마다 실시간으로 반영된다.

4.2.8.1 State

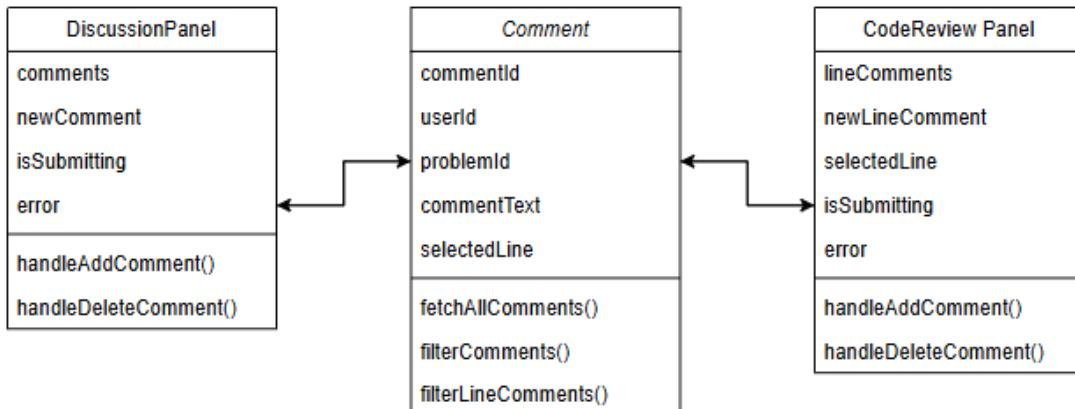
- **comments**: 다른 사용자들이 달아 놓은 코드에 대한 댓글을 저장하는 상태이다. 이 배열에는 각 댓글의 내용, 작성자, 작성 시간 등의 정보가 포함된다.
- **newComment**: 사용자가 새로 작성하려는 댓글 내용을 저장하는 상태이다. 사용자가 댓글을 작성하고 제출할 때 이 상태에 저장된다.
- **lineComments**: 다른 사용자들이 달아 놓은 코드에 대한 코드 리뷰를 저장하는 상태이다. 이 배열에는 각 코드 리뷰의 내용, 작성자, 작성 시간 등의 정보가 포함된다.
- **newLineComments**: 사용자가 새로 작성하려는 코드 리뷰 내용을 저장하는 상태이다. 사용자가 코드 리뷰를 작성하고 제출할 때 이 상태에 저장된다.
- **selectedLine**: 현재 선택된 코드의 라인 번호를 저장하는 상태이다. null 이면 일반 댓글 목록을 표시하고, 값이 있으면 해당 라인에 대한 코드 리뷰 댓글을 표시한다.
- **isSubmitting**: 댓글을 제출 중인 상태를 나타내는 불리언 값이다. 제출 요청이 진행 중일 때 true로 설정되며, 완료되면 false로 설정된다.

- **error:** 댓글을 추가하는 동안 발생한 오류 메시지를 저장하는 상태이다. 서버 요청 실패나 입력 오류가 발생할 경우 이 상태에 오류 메시지가 저장된다.

4.2.8.2 Methods

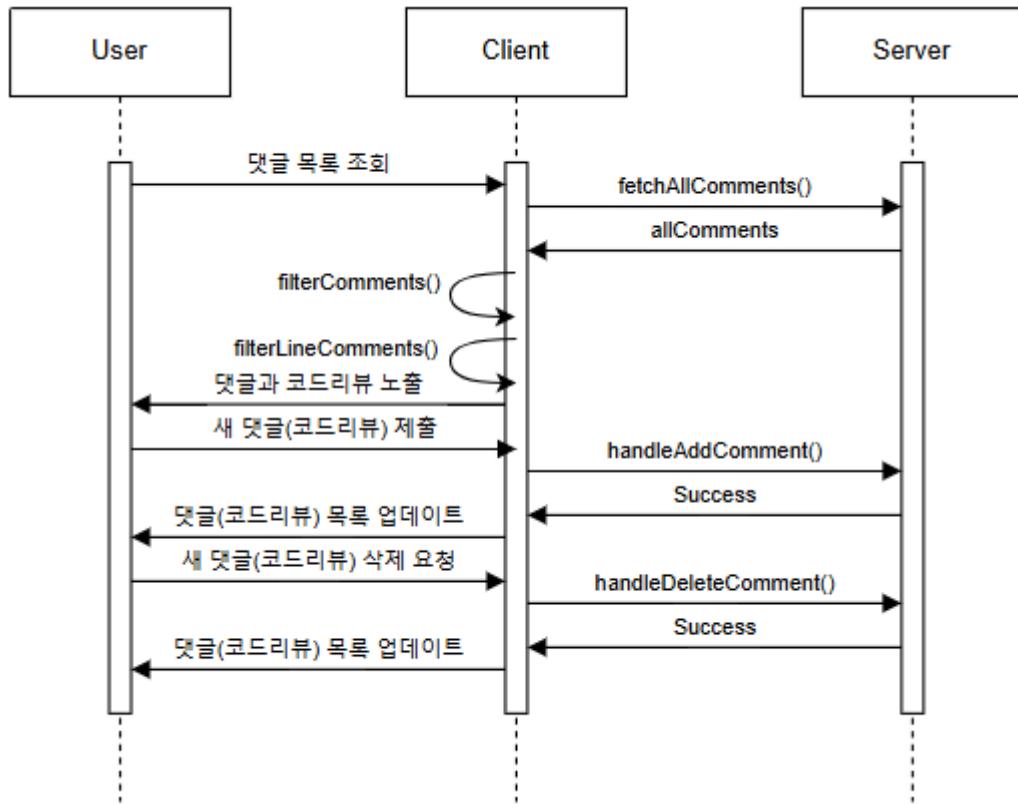
- **fetchAllComments:** 서버에서 댓글 및 코드 리뷰 목록을 가져오는 메소드이다. 페이지가 로드될 때 또는 특정 조건에 맞춰 댓글 및 코드 리뷰 목록을 서버에서 받아온다.
- **filterComments:** 일반 댓글만 필터링하는 메소드이다. 댓글 목록에서 일반 댓글을 분리하여 comments 상태에 저장한다.
- **filterLineComments:** 코드 리뷰 댓글을 필터링하는 메소드이다. 댓글 목록에서 코드 리뷰만 필터링하여 lineComments 상태에 저장한다.
- **handleAddComment:** 사용자가 새로운 댓글(코드리뷰)을 작성하고 제출할 때 호출되는 메소드이다. 사용자가 입력한 newComment를 서버에 제출하고, 성공적으로 추가되면 해당 배열에 반영한다.
- **handleDeleteComment:** 기존의 댓글(코드리뷰)을 삭제할 때 호출되는 메소드이다. 삭제할 댓글(코드리뷰)의 commentId를 서버에 전송하고, 성공적으로 삭제되면 해당 배열에서 해당 댓글(코드리뷰)을 제거한다.

4.2.8.3 Class Diagram



[Figure 17] Class Diagram - Discussion Panel & CodeReview Panel

4.2.8.4 Sequence Diagram



[Figure 18] Discussion Panel & CodeReview Panel

4.2.9 User Dashboard

사용자의 프로필 정보, 생성한 문제 목록, 학습 현황을 확인할 수 있는 기능을 제공하는 컴포넌트이다. 이 대시보드는 사용자가 자신의 학습 진행 상황과 생성한 문제들을 한눈에 볼 수 있게 하며, 학습 목표를 확인하고 평가할 수 있는 유용한 정보를 제공한다.

4.2.9.1 State

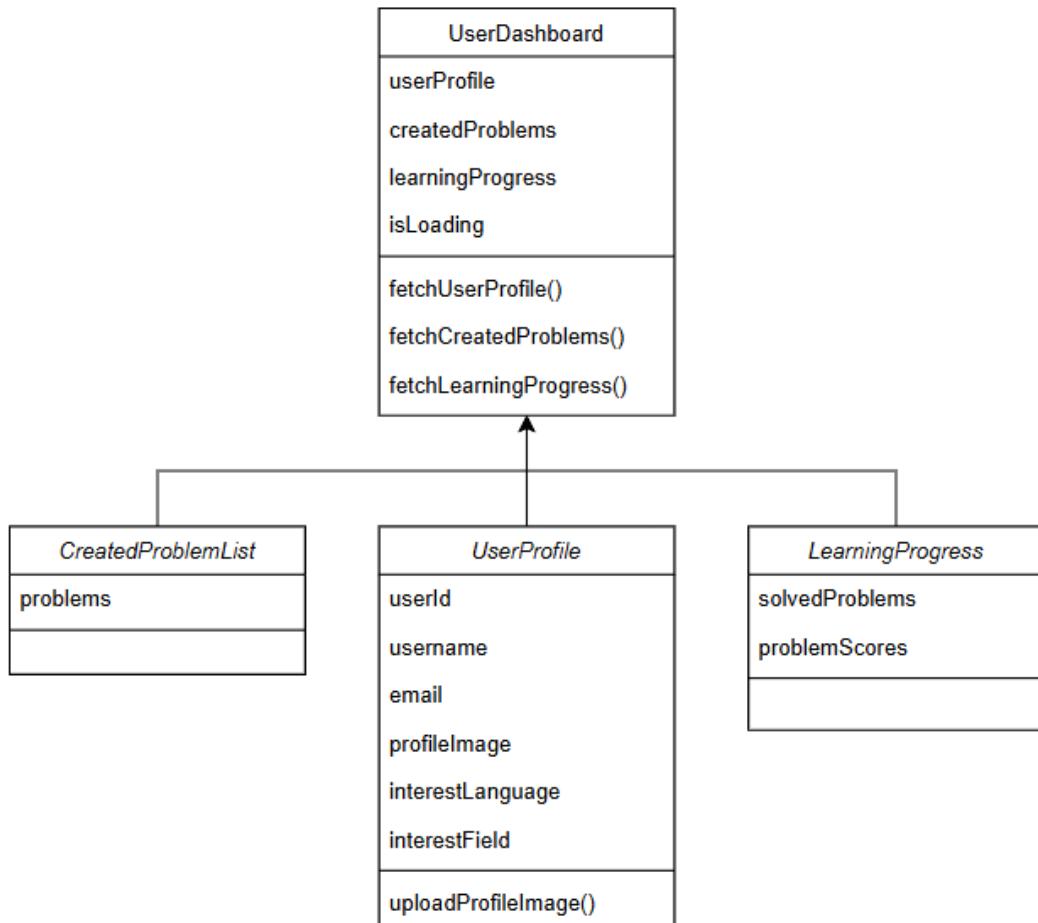
- **userProfile**: 사용자의 프로필 정보를 저장하는 상태이다. 사용자의 이름, 이메일, 가입 날짜 등과 같은 정보를 포함한다.
- **newProfileImage**: 사용자가 업로드하기 위해 올린 이미지 파일을 저장하는 상태이다.
- **createdProblems**: 사용자가 생성한 문제 목록을 저장하는 상태이다. 이 배열에는 각 문제의 제목, 난이도, 작성 날짜 등의 정보가 포함된다.
- **learningProgress**: 사용자가 해결한 문제와 각 유형별 해결한 문제 수 등의 정보를 저장하는 상태이다. 학습한 문제의 개수, 문제 유형별 성과 등을 포함한다.

- **isLoading:** 데이터를 로딩 중인 상태를 나타내는 불리언 값이다. 데이터를 요청 중일 때 true로 설정되고, 데이터가 로드되면 false로 설정된다.
- **error:** 데이터 로딩 중 발생한 오류 메시지를 저장하는 상태이다. 서버 요청 실패나 네트워크 오류가 발생하면 이 상태에 오류 메시지가 저장된다.

4.2.9.2 Methods

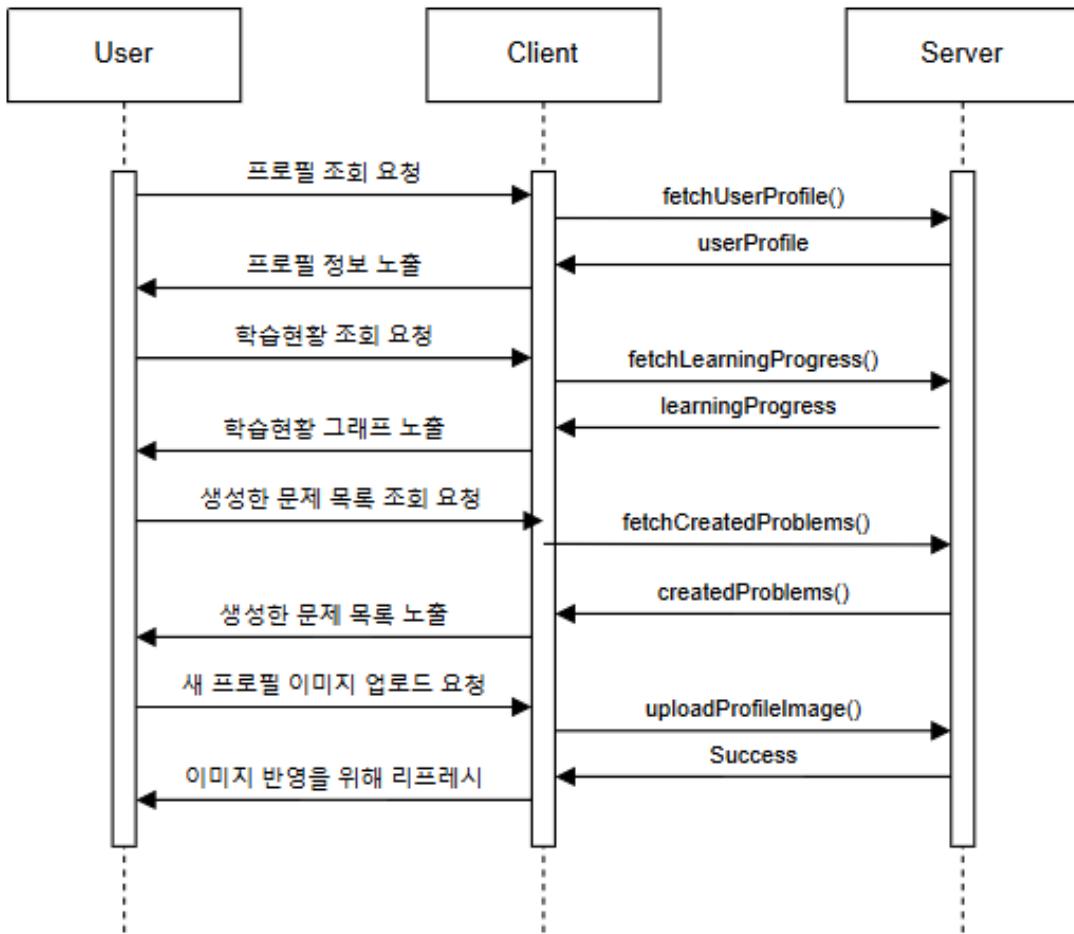
- **fetchUserProfile:** 서버에서 사용자의 프로필 정보를 가져오는 메소드이다. 페이지가 로드될 때 사용자의 프로필 정보를 요청하고, 성공적으로 데이터를 받으면 userProfile 상태에 저장한다.
- **fetchCreatedProblems:** 사용자가 생성한 문제 목록을 서버에서 가져오는 메소드이다. 페이지가 로드될 때 사용자가 생성한 문제 목록을 요청하고, createdProblems 상태에 저장한다.
- **fetchLearningProgress:** 사용자가 해결한 문제와 각 유형별 해결한 문제 수 등의 학습 현황을 서버에서 가져오는 메소드이다. 학습한 문제의 수, 문제 유형별 성과 등을 요청하고, learningProgress 상태에 저장한다.
- **uploadProfileImage:** 사용자가 선택한 프로필 이미지를 서버에 업로드하는 메소드이다. 선택된 파일을 newProfileImage 상태에 저장하고, 이를 서버로 전송하여 프로필 이미지를 업데이트한다.

4.2.9.3 Class Diagram



[Figure 19] Class Diagram - User Dashboard

4.2.9.4 Sequence Diagram



[Figure 20] Sequence Diagram - User Dashboard

4.2.10 Admin User Management

관리자가 시스템에 등록된 사용자 목록을 조회하고, 사용자가 작성한 댓글 및 코드리뷰를 조회 및 삭제할 수 있는 기능을 제공하는 컴포넌트이다. 또한, 사용자를 삭제할 수 있으며, 관리자는 사용자가 작성한 부적절한 댓글 및 코드리뷰를 삭제하는 역할을 한다.

4.2.10.1 State

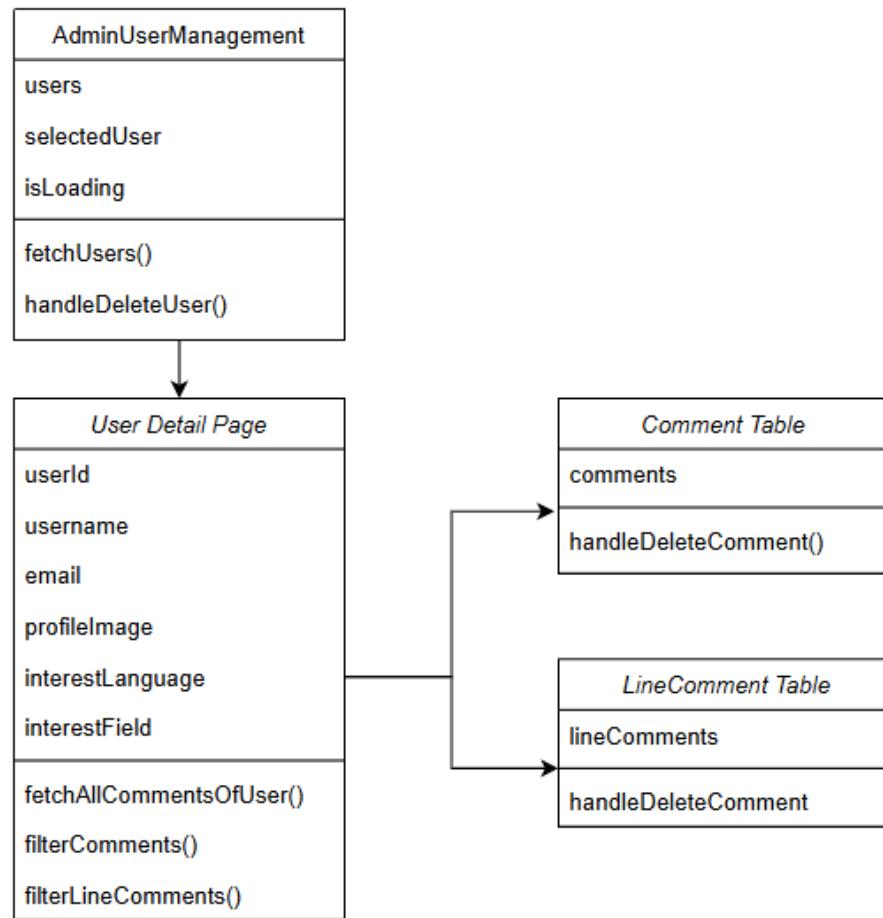
- **users:** 시스템에 등록된 사용자 목록을 저장하는 상태이다. 각 사용자의 정보가 포함된 배열로, 사용자 테이블에 표시된다.
- **comments:** 특정 사용자가 작성한 댓글 목록을 저장하는 상태이다. 댓글 내용, 작성자, 작성 시간 등의 정보가 포함된다.
- **lineComments:** 특정 사용자가 작성한 코드리뷰 목록을 저장하는 상태이다. 코드리뷰 내용, 작성자, 작성 시간 등의 정보가 포함된다.

- **selectedUser**: 선택된 사용자의 상세 정보를 저장하는 상태이다. 사용자가 클릭한 행의 상세 정보를 보여준다.
- **isLoading**: 데이터를 로딩 중인 상태를 나타내는 불리언 값이다. 데이터를 요청 중일 때 true로 설정되고, 데이터가 로드되면 false로 설정된다.
- **error**: 데이터 로딩 중 발생한 오류 메시지를 저장하는 상태이다. 서버 요청 실패나 네트워크 오류가 발생하면 이 상태에 오류 메시지가 저장된다.

4.2.10.2 Methods

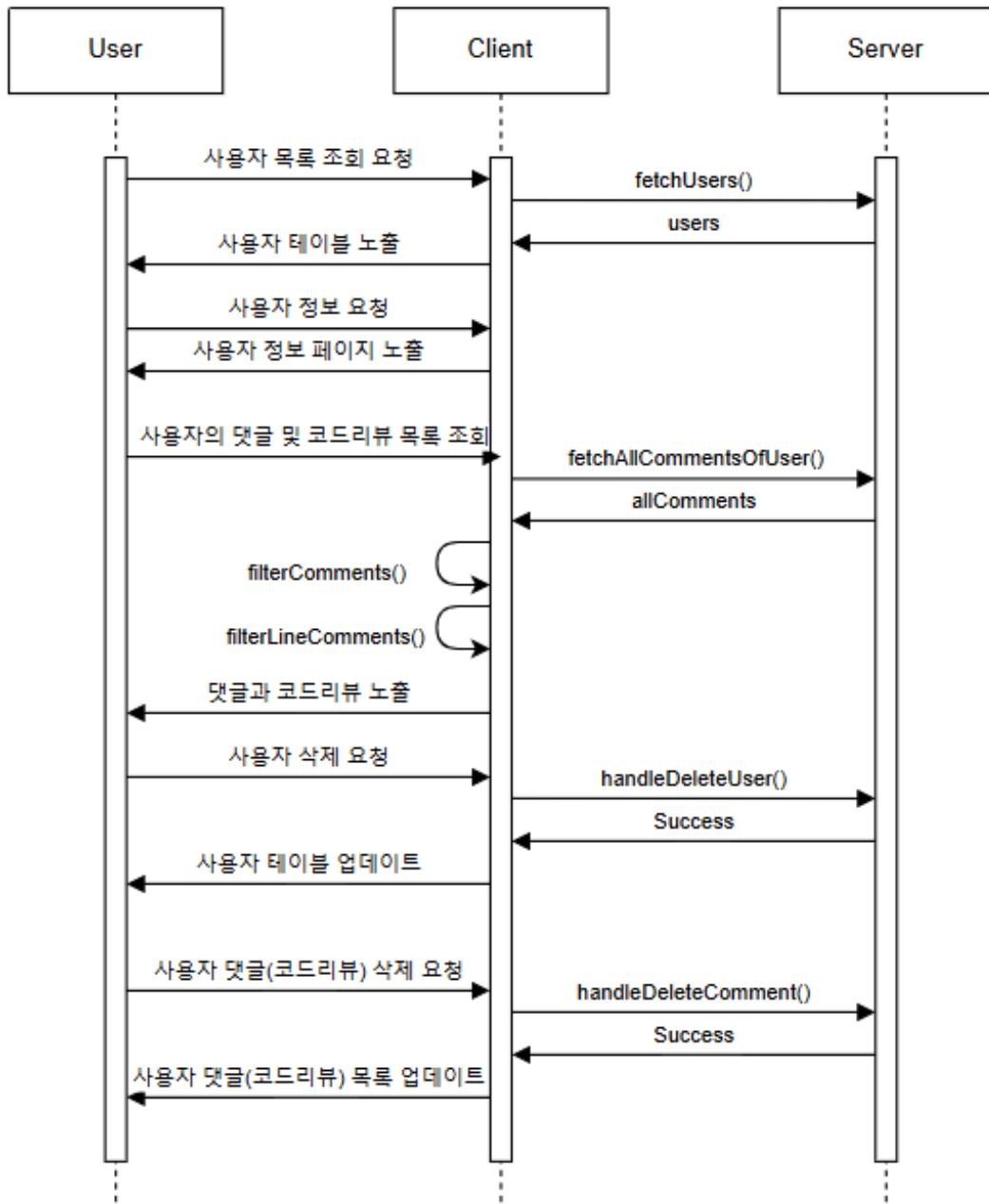
- **fetchUsers**: 시스템에 등록된 사용자 목록을 서버에서 가져오는 메소드이다. 페이지가 로드될 때 사용자 목록을 요청하고, users 상태에 저장한다.
- **handleDeleteUser**: 선택된 사용자를 삭제하는 메소드이다. selectedUser에 저장된 사용자 ID를 기반으로 서버에 삭제 요청을 보내고, 삭제가 완료되면 사용자 목록에서 해당 사용자를 제거한다.
- **fetchAllCommentsOfUser**: 특정 사용자가 작성한 댓글 및 코드리뷰 목록을 서버에서 가져오는 메소드이다. 사용자의 ID를 기반으로 해당 사용자가 작성한 댓글 및 코드리뷰 데이터를 받아온다.
- **handleDeleteComment**: 특정 댓글(코드리뷰)을 삭제하는 메소드이다. commentId를 기반으로 서버에 삭제 요청을 보내고, 삭제가 완료되면 댓글(코드리뷰) 목록에서 해당 댓글을 제거한다.
- **filterComments**: 일반 댓글만 필터링하는 메소드이다. 댓글 목록에서 일반 댓글을 분리하여 comments 상태에 저장한다.
- **filterLineComments**: 코드 리뷰 댓글을 필터링하는 메소드이다. 댓글 목록에서 코드 리뷰만 필터링하여 lineComments 상태에 저장한다.

4.2.10.3 Class Diagram



[Figure 21] Class Diagram - Admin User Management

4.2.10.4 Sequence Diagram



[Figure 22] Sequence Diagram - Admin User Management

4.2.11 Admin Problem Management

관리자가 시스템에 등록된 문제 목록을 조회하고, 문제에 대한 상세 정보를 제공하는 컴포넌트이다. 문제를 클릭하면 해당 문제의 상세 정보 페이지로 이동하고, 해당 문제를 삭제할 수 있다.

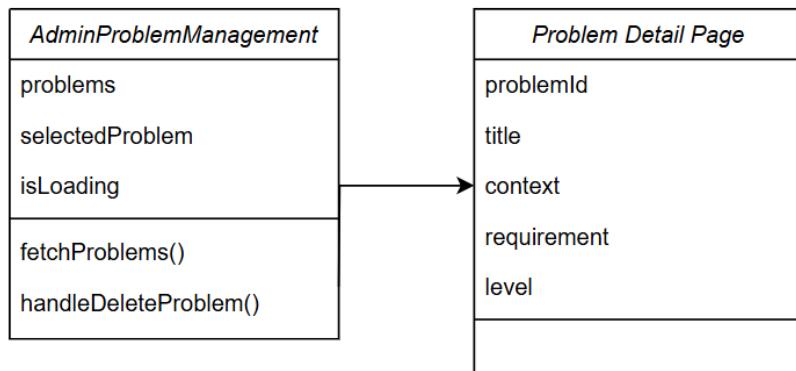
4.2.11.1 State

- **problems**: 시스템에 등록된 문제 목록을 저장하는 상태이다. 각 문제의 제목, 설명, 난이도 등의 정보가 포함된 배열로, 문제 테이블에 표시된다.
- **selectedProblem**: 선택된 문제의 상세 정보를 저장하는 상태이다. 문제를 클릭하면 해당 문제의 상세 정보를 보여준다.
- **isLoading**: 데이터를 로딩 중인 상태를 나타내는 불리언 값이다. 데이터를 요청 중일 때 true로 설정되고, 데이터가 로드되면 false로 설정된다.
- **error**: 데이터 로딩 중 발생한 오류 메시지를 저장하는 상태이다. 서버 요청 실패나 네트워크 오류가 발생하면 이 상태에 오류 메시지가 저장된다.

4.2.11.2 Methods

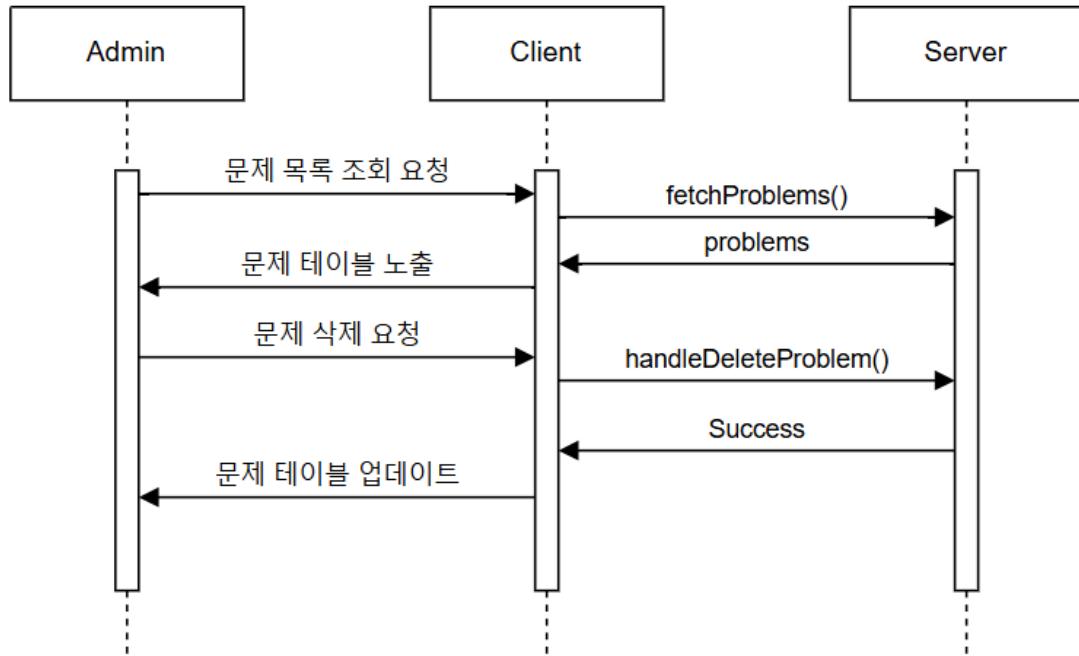
- **fetchProblems**: 시스템에 등록된 문제 목록을 서버에서 가져오는 메소드이다. 페이지가 로드될 때 문제 목록을 요청하고, problems 상태에 저장한다.
- **handleDeleteProblem**: 선택된 문제를 삭제하는 메소드이다. selectedProblem에 저장된 문제 ID를 기반으로 서버에 삭제 요청을 보내고, 삭제가 완료되면 문제 목록에서 해당 문제를 제거한다.

4.2.11.3 Class Diagram



[Figure 23] Class Diagram - Admin Problem Management

4.2.11.4 Sequence Diagram



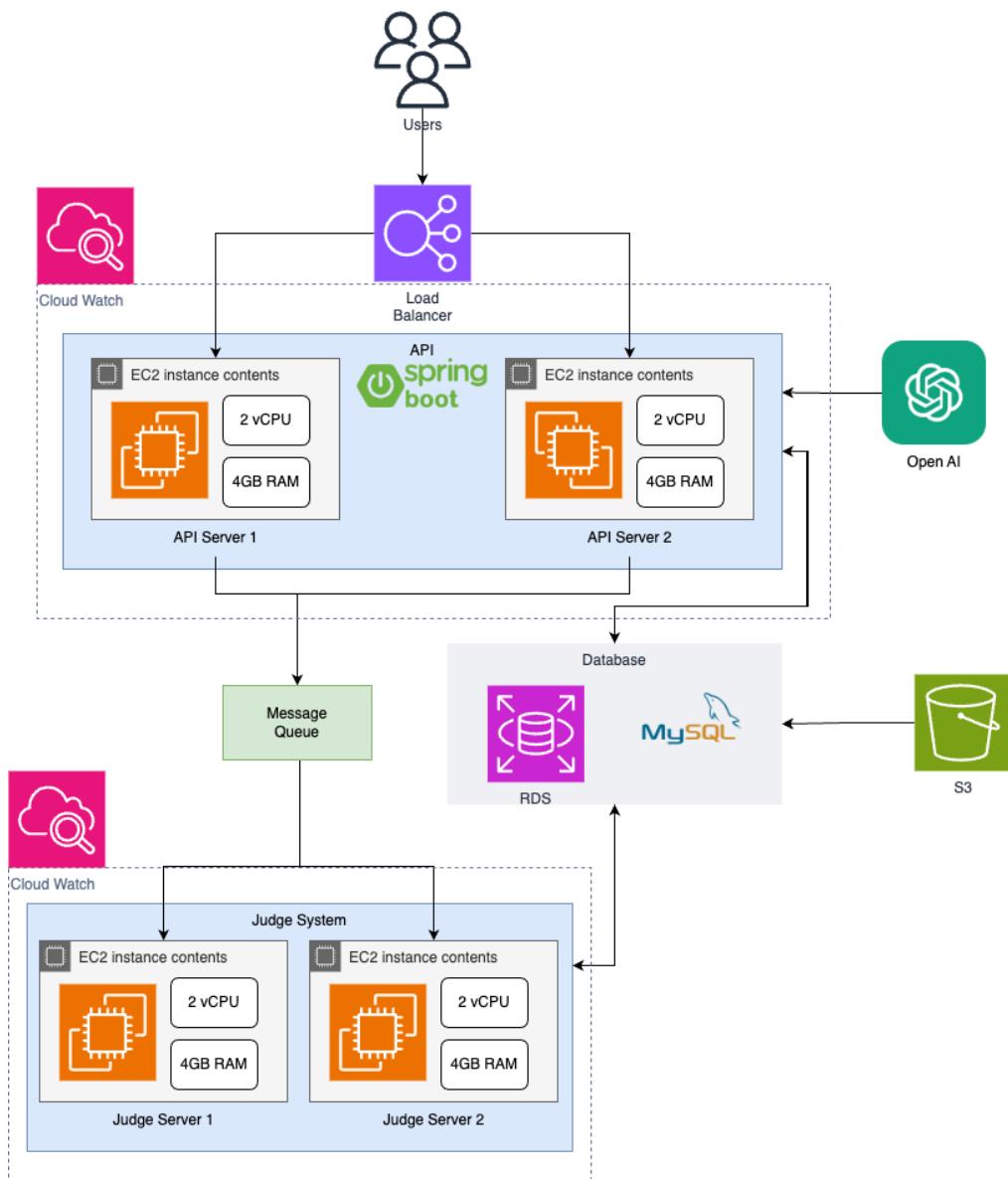
[Figure 24] Sequence Diagram - Admin Problem Management

5. System Architecture - Backend

5.1 Objectives

이 챕터는 back-end 시스템의 전반적인 구조에 대해 기술한다.

5.2 Overall architecture



[Figure 25] overall architecture - backend

1. 사용자 인터페이스 (Users):

- 사용자는 웹 브라우저를 통해 시스템에 접속하고, 다양한 기능을 이용한다.
- 기능에는 회원가입, 로그인, 문제 생성, 실시간 실습, 채점 및 피드백 확인 등이 포함된다.

2. 로드 밸런서:

- 사용자 요청을 두 개의 API 서버에 균등하게 분산하여 트래픽을 효율적으로 관리한다.

3. API 서버 (API Server 1 & API Server 2):

- AWS EC2 인스턴스에서 운영되며, 각각 2 vCPU와 4GB RAM을 갖추고 있다.
- Spring Boot 프레임워크를 사용하여 문제 생성, 피드백 제공, 사용자 요청 처리를 담당한다.
- GPT API와 통신하여 AI 기반의 문제 생성 및 피드백을 제공한다.

4. 메시지 큐 (Message Queue):

- 채점 요청을 관리하고, 두 개의 채점 서버 간에 작업을 분배하여 병목 현상을 방지한다.

5. 채점 서버 (Judge Server 1 & Judge Server 2):

- AWS EC2 인스턴스에서 운영되며, 각각 2 vCPU와 4GB RAM을 갖춘 두 대의 인스턴스를 사용한다.
- 사용자로부터 전달받은 코드를 샌드박스 환경에서 채점하고, 결과를 데이터베이스에 저장한다.

6. 데이터베이스 (RDS):

- AWS RDS를 사용하여 사용자 정보, 문제 풀이 기록, 학습 히스토리를 저장한다.
- MySQL 데이터베이스를 사용하며, 데이터의 안전성과 성능을 보장하기 위해 주기적인 백업과 최적화를 수행한다.

7. 파일 저장소 (S3):

- 이미지 파일 및 기타 정적 자산을 저장하는 데 사용된다.

8. OpenAI (GPT API):

- 문제 생성과 피드백 제공을 위해 GPT API를 활용하여 AI 기반의 기능을 지원한다.

운영 및 관리

- 관리자 페이지:** 시스템 내 사용자 관리와 문제 관리를 통해 부적절한 댓글과 리뷰를 처리하고, 문제 데이터의 품질을 유지한다.
- 서버 모니터링:** AWS CloudWatch를 통해 서버 상태를 모니터링하고, CPU 및 메모리 사용량을 실시간으로 확인한다.
- LLM 신뢰성 관리:** Prompt Generator를 사용하여 문제 생성 프롬프트의 품질을 지속적으로 점검하고 개선한다.

5.3 Subcomponents

5.3.1 User system

사용자 정보를 관리하는 시스템으로, 사용자 정보를 등록하는 요청이 들어오면 시스템은 중복 여부를 확인한다. 중복된 정보가 존재하지 않을 경우, 데이터베이스에 정보를 저장한다. 사용자 로그인 기록을 바탕으로 방문을 파악한다.

5.3.1.1 state

User

- user_id: 사용자의 고유 식별 ID
- email: 사용자의 이메일 주소
- username: 사용자의 사용자 이름
- password: 사용자의 비밀번호
- interest_language: 사용자가 관심 있는 프로그래밍 언어
- interest_field: 사용자가 관심 있는 기술 분야
- role: 사용자의 역할 (user/admin)

Profile

- profile_id: 프로필 식별 ID
- original_img: 업로드 전 원본 이미지
- uploaded_img: S3에 업로드된 이미지의 URL

Dashboard

- dashboard_id: 대시보드 식별 ID
- total_problems: 사용자가 해결한 총 문제 수

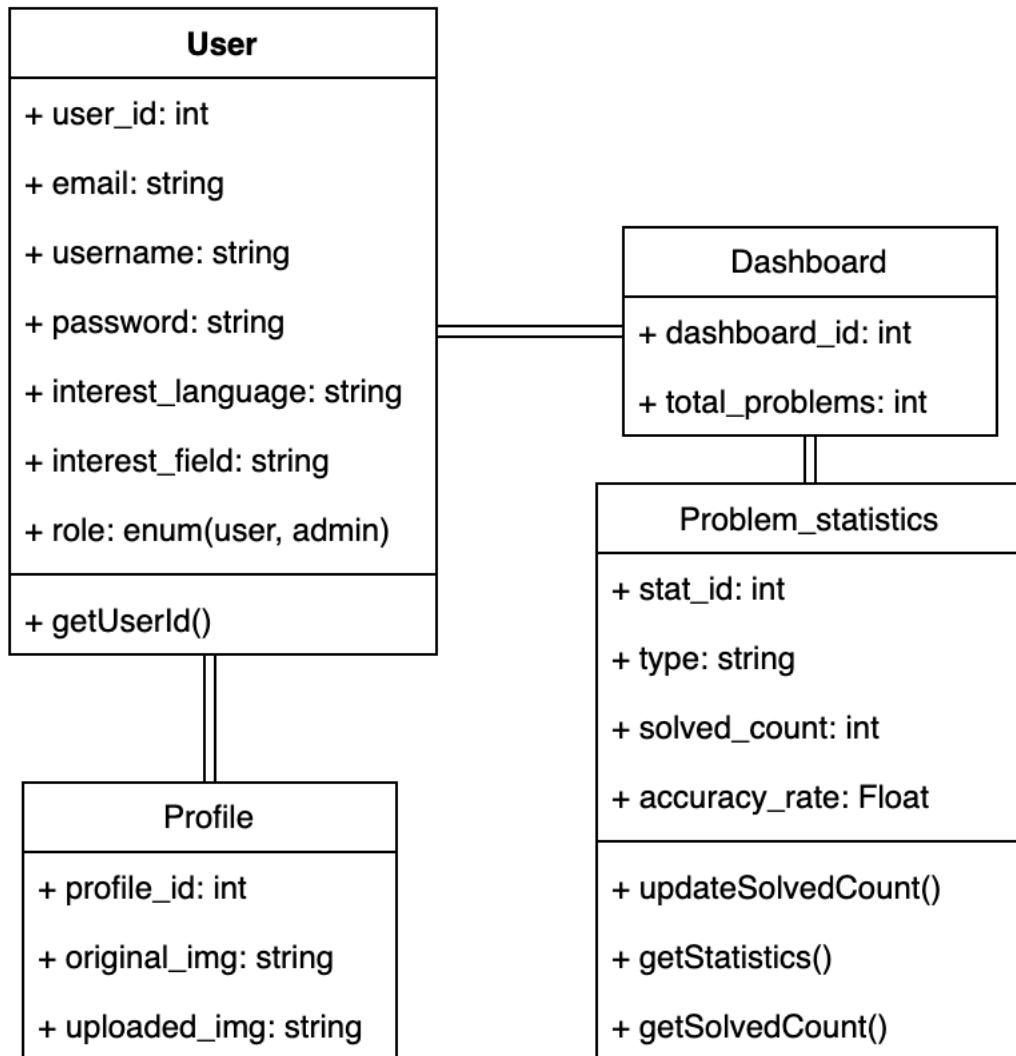
Problem_statistics

- stat_id: 통계 고유 ID
- type: 문제 유형
- solved_count: 해결된 문제 수
- accuracy_rate: 해결 정확도

5.3.1.2 methods

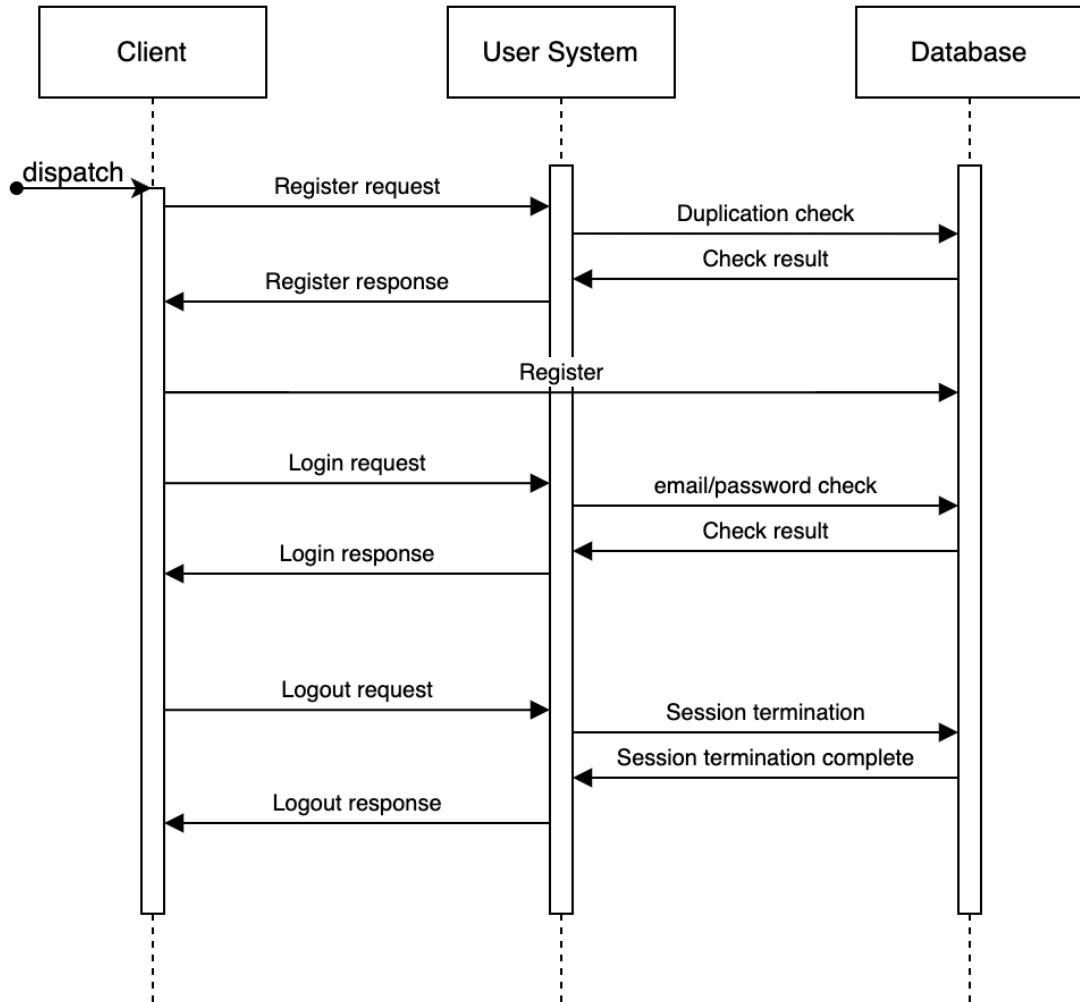
- getUserId(): 사용자 ID 반환
- updateSolvedCount(): 해결된 문제 수 업데이트
- getStatistics(): 통계 정보 반환
- getSolvedCount(): 해결된 문제 수 반환

5.3.1.3 Class Diagram

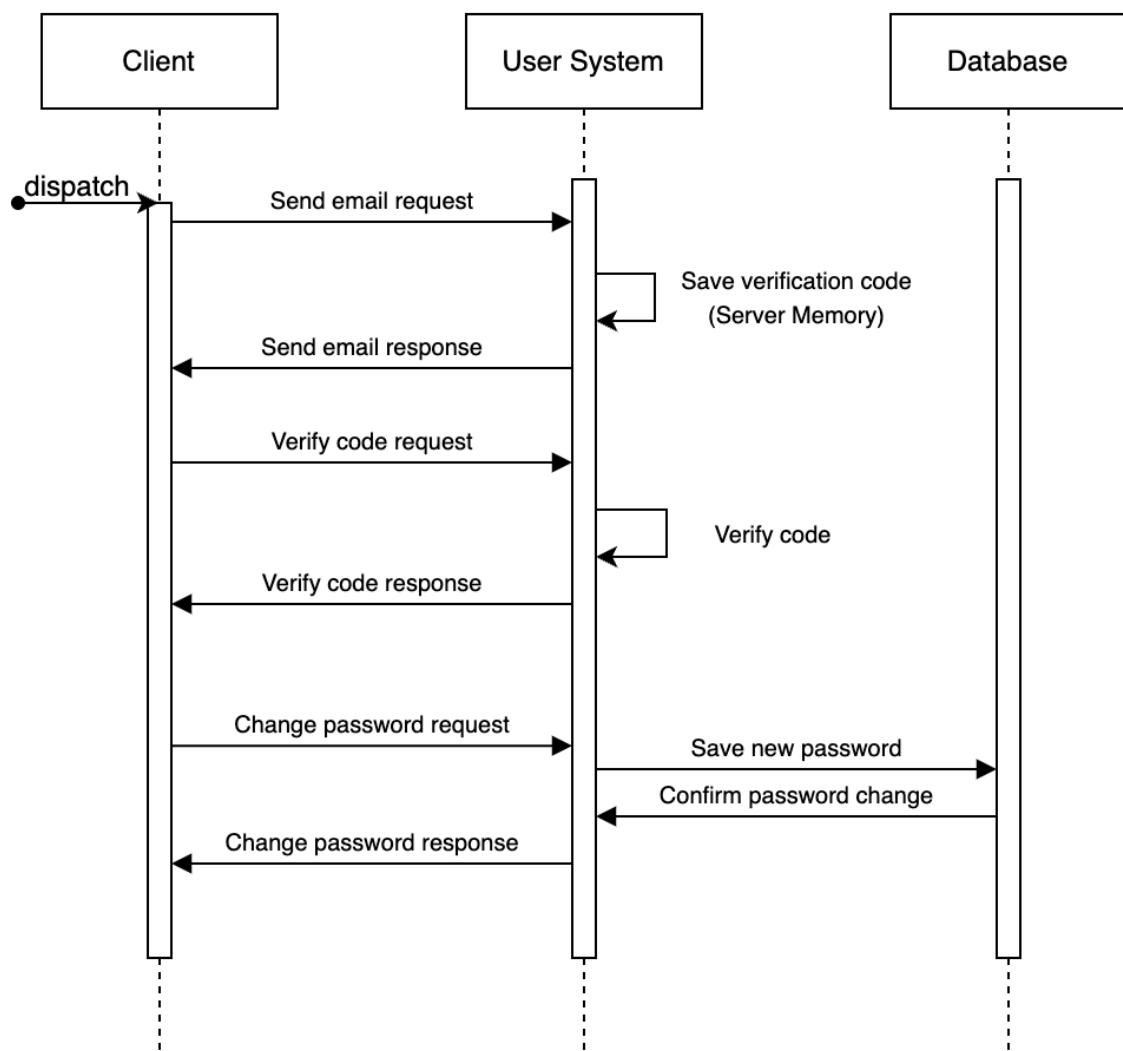


[Figure 26] Class Diagram – User System

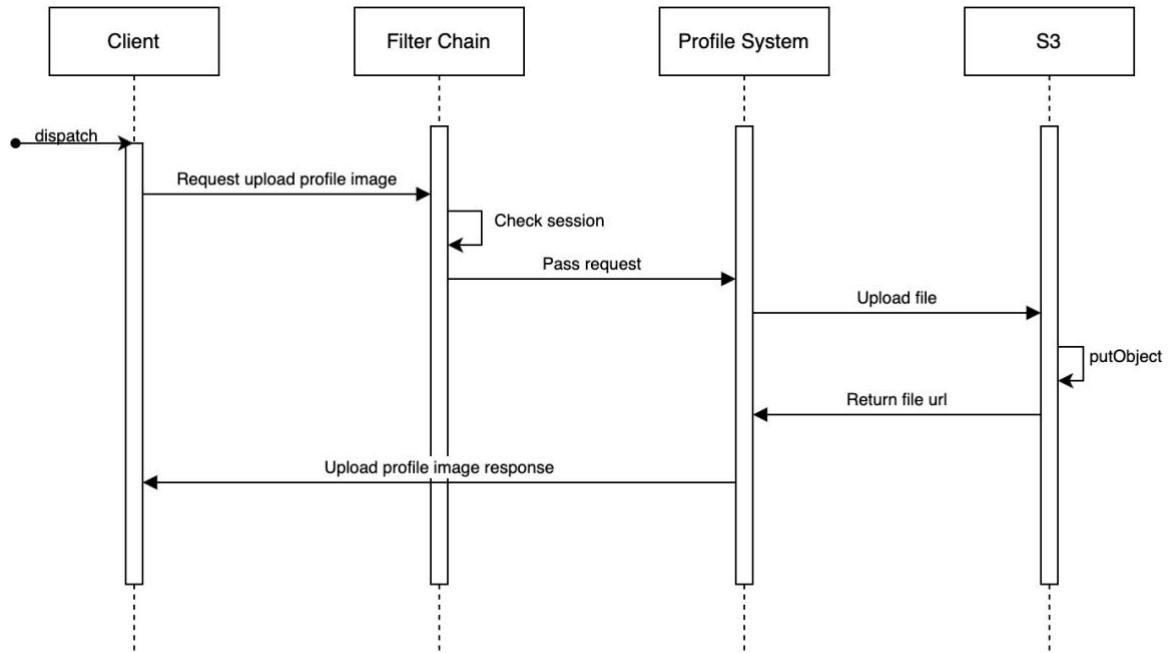
5.3.1.4 Sequence Diagram



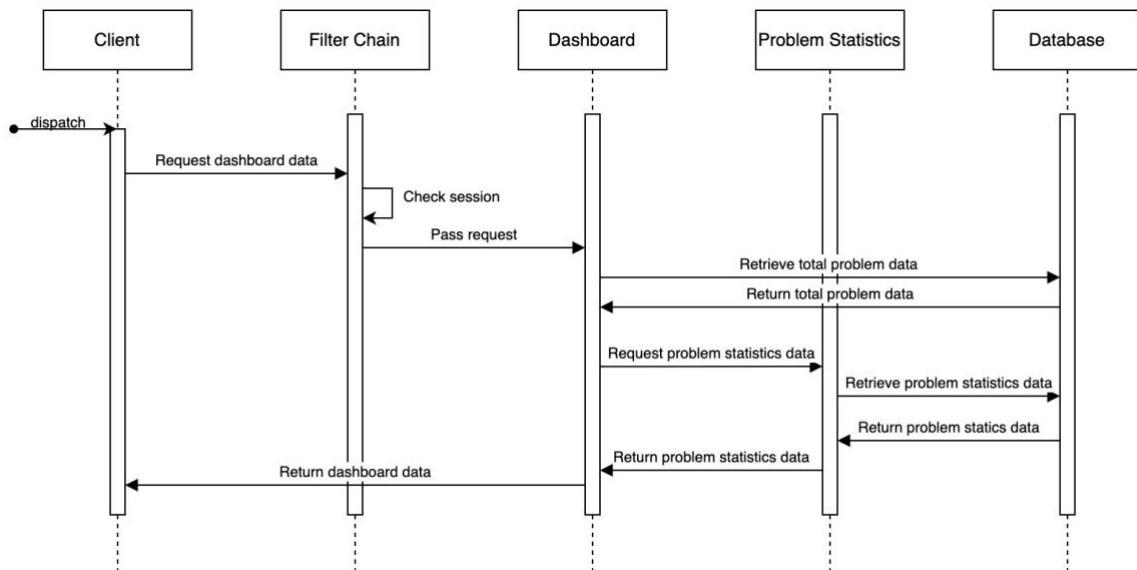
[Figure 27] Sequence Diagram – User System : Auth



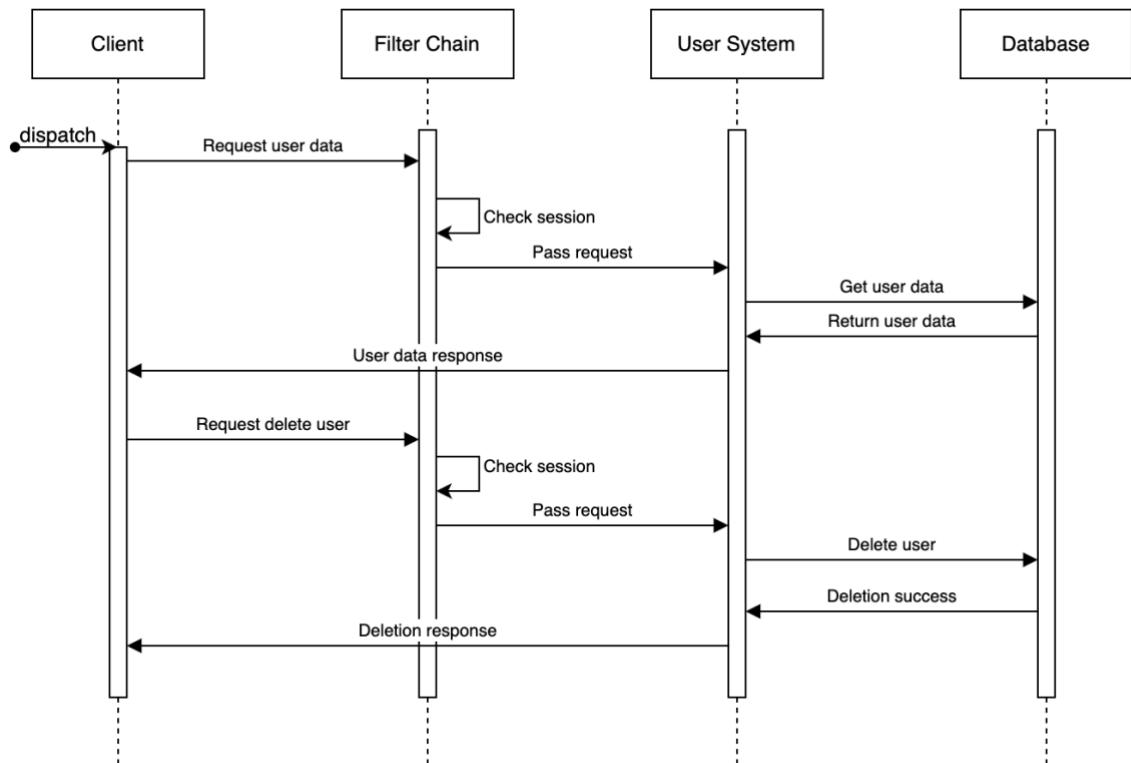
[Figure 28] Sequence Diagram – User System : change password



[Figure 29] Sequence Diagram – User System : upload profile image



[Figure 30] Sequence Diagram – User System : dashboard



[Figure 31] Sequence Diagram – User System : Admin User Management

5.3.2 Problem System

문제풀이의 전체적인 시스템이다. 사용자가 제출한 문제는 데이터베이스에 기본적으로 저장되며, 채점 서버에서 정답 여부를 판별한다. 채점 결과에 따라 제출의 결과가 업데이트된다. 성공으로 평가된 제출은 공유게시판에 공유하여 다른 유저들이 볼 수 있으며, 해당 문제에 대해서 다른 유저들 또한 동일하게 문제풀이를 진행할 수 있다.

5.3.2.1 State

Submission

- submission_id: 제출 식별 ID
 - submission_date: 제출 날짜
 - result: 채점 결과

Problem

- problem_id: 문제 식별 ID
 - problem_title: 문제 제목
 - problem_context: 문제 설명
 - problem_requirement: 문제 요구사항
 - problem_level: 문제 난이도

Judgement

- judgement_id: 판별 고유 ID
- input_code: 제출된 코드
- correct_code: 정답 코드
- result: 판별 결과

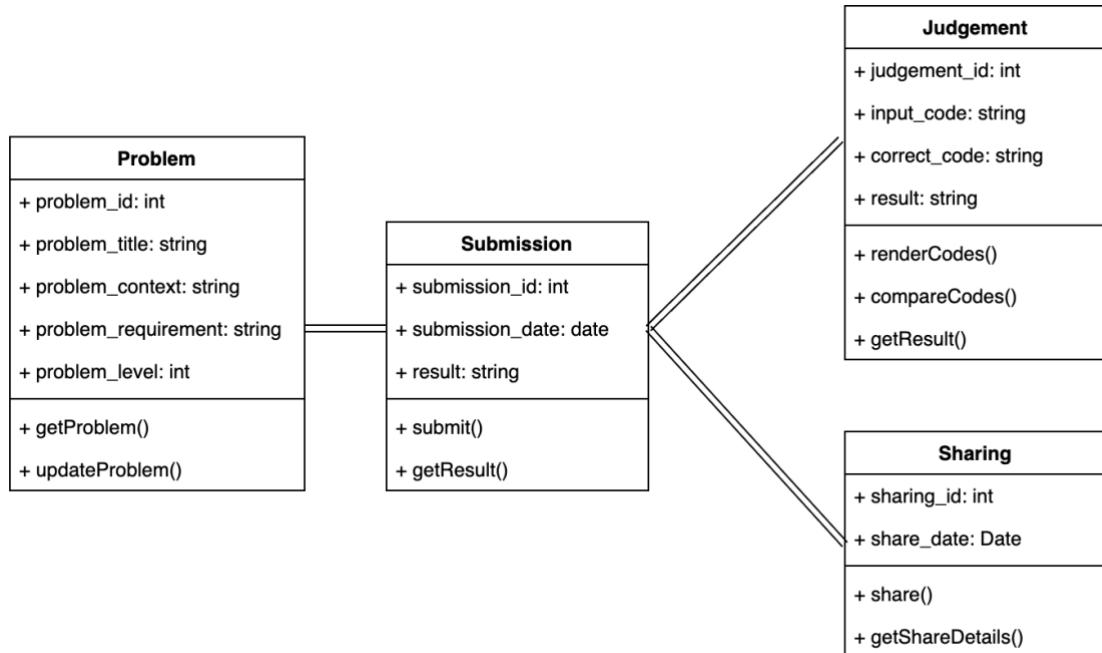
Sharing

- sharing_id: 공유 고유 ID
- share_date: 공유 날짜

5.3.2.2 Methods

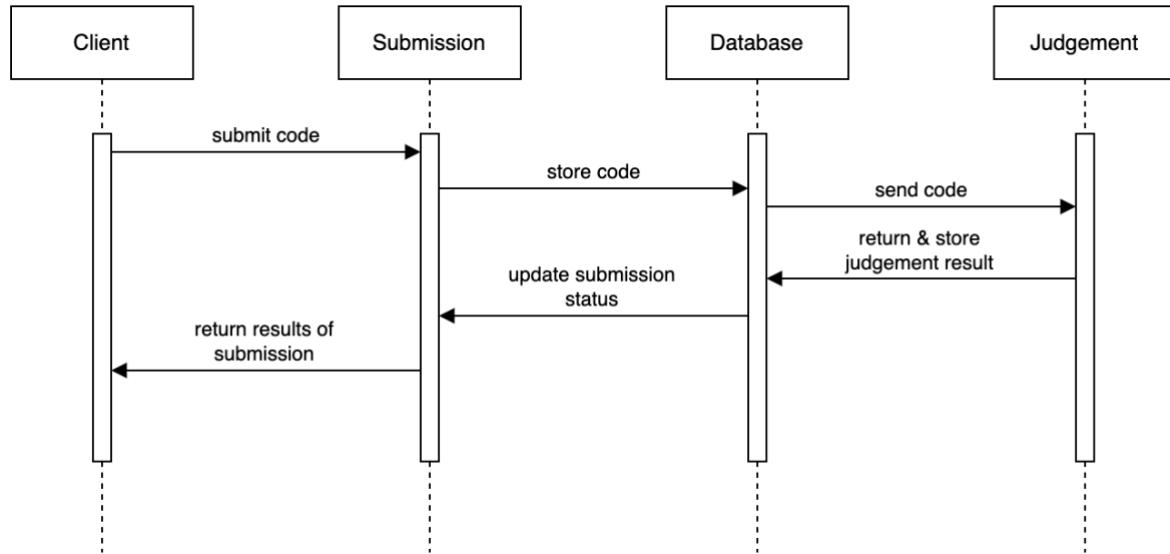
- submit(): 사용자 제출 데이터 기록
- getResult(): 제출 결과 반환
- getProblem(): 문제 데이터 반환
- updateProblem(): 문제 데이터 수정
- renderCodes(): 제출된 코드와 정답 코드 시각화
- compareCodes(): 코드 비교
- getResult(): 판별 결과 반환
- share(): 문제 공유
- getShareDetails(): 공유된 문제 정보 반환

5.3.2.3 Class Diagram

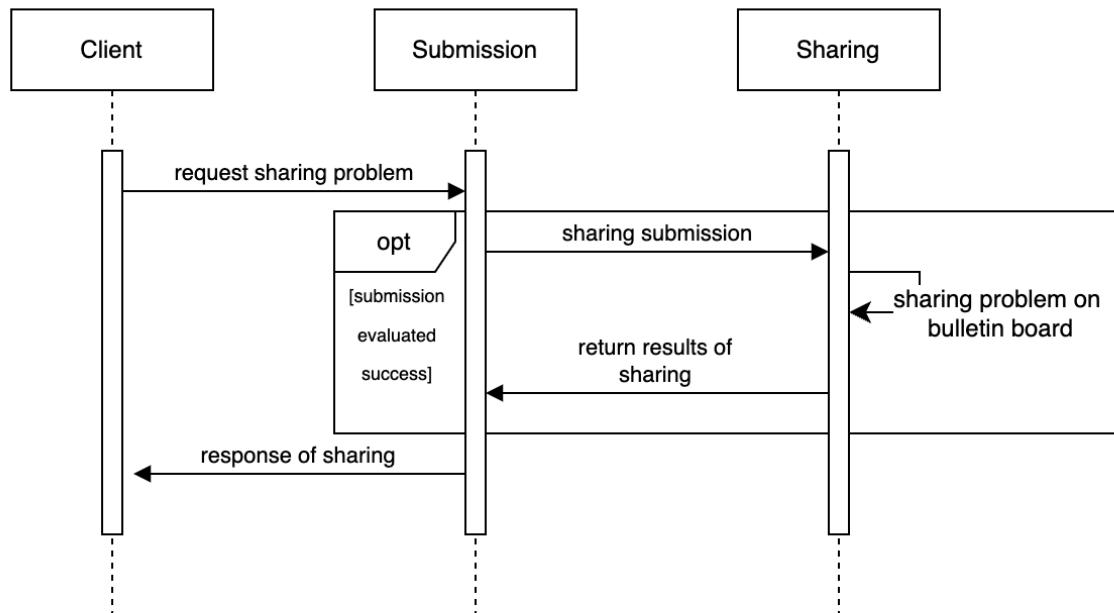


[Figure 32] Class Diagram – Problem System

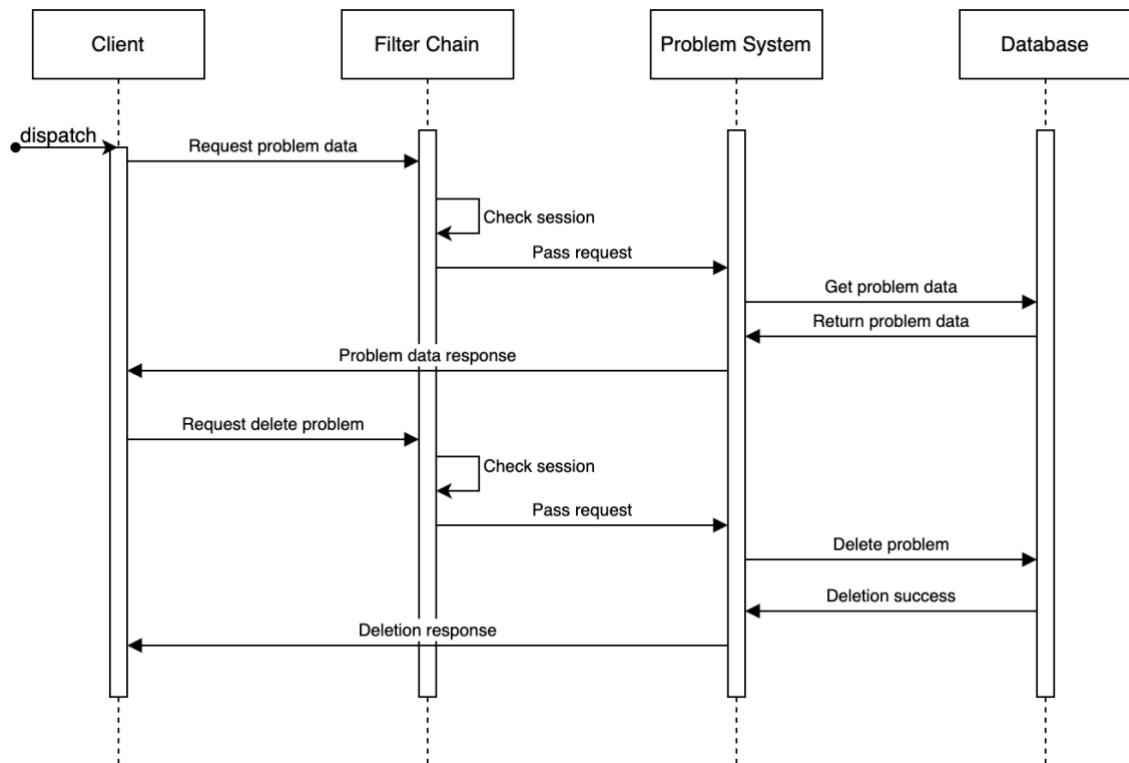
5.3.2.4 Sequence Diagram



[Figure 33] Sequence Diagram – Problem System : Solving problem



[Figure 34] Sequence Diagram – Problem System : Sharing Problem

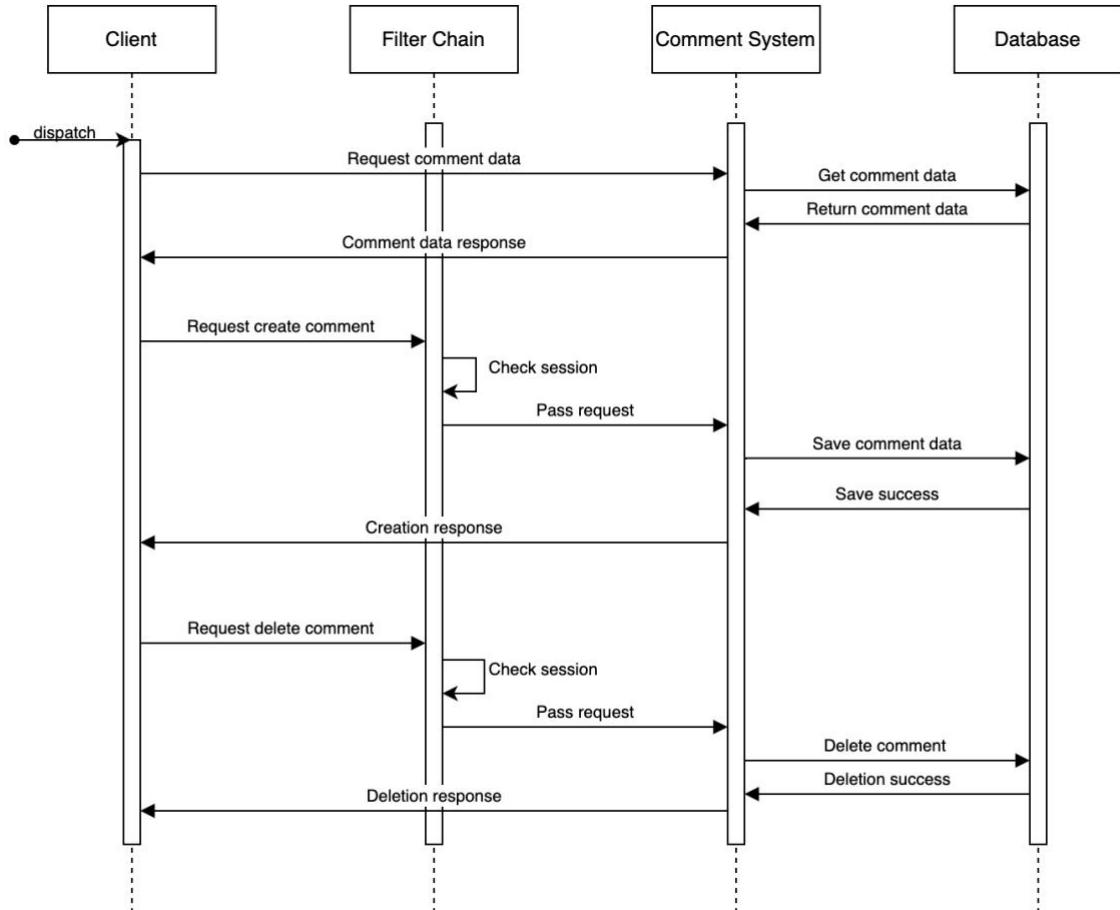


[Figure 35] Sequence Diagram – Problem System : Admin Problem Management

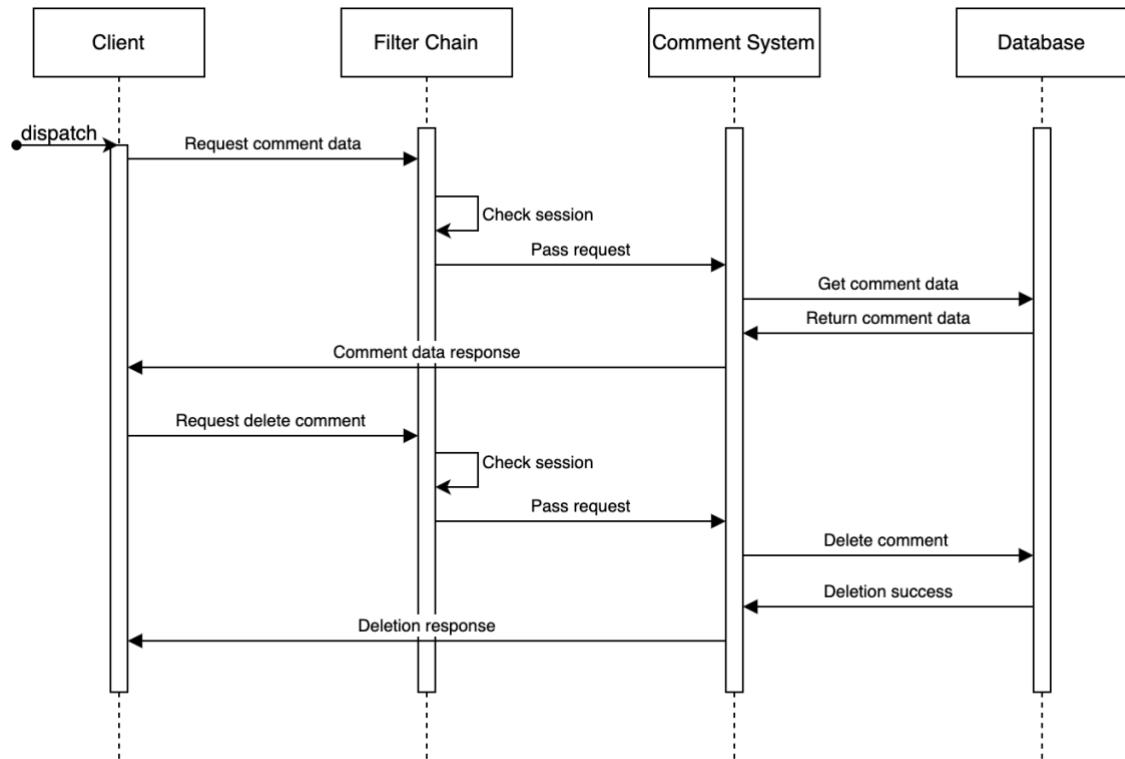
5.3.3 Comment System

댓글 시스템이다. 유저는 댓글을 남길 수 있고, 운영자는 이를 관리할 수 있다.

5.3.3.1 Sequence Diagram



[Figure 36] Sequence Diagram – Comment System : User Comment

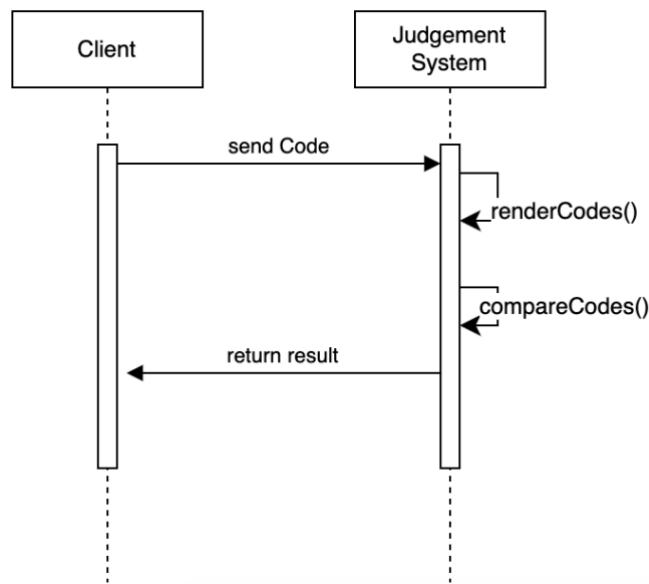


[Figure 37] Sequence Diagram – Comment System : Admin Comment Management

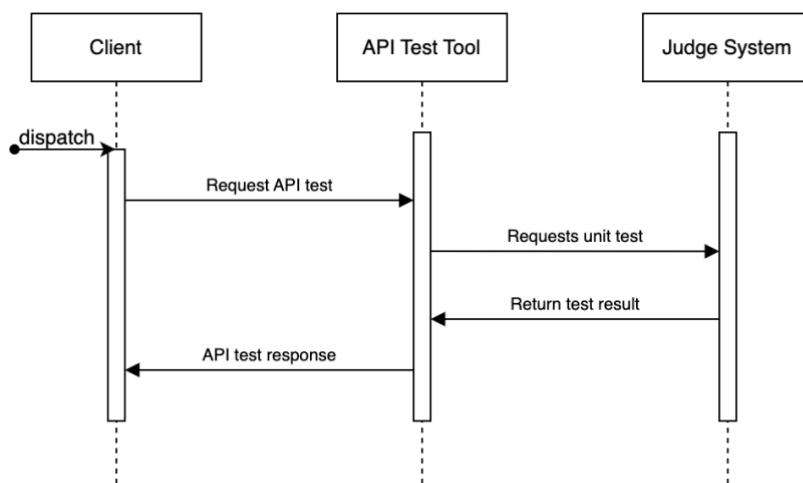
5.3.4 Judge System

채점 시스템은 크게 프론트 문제 채점과 백엔드 문제 채점으로 나뉜다. 프론트 연습문제에 대한 채점은, 문제 생성 시 문제와 정답 테스트 케이스가 함께 생성되므로, 이 둘을 렌더링 한 후 비교하여 정답 여부를 판별한다. 백엔드 연습문제에 대한 채점은, API Test Tool로 이루어지며, API가 정상적으로 작성되었는지를 판별하기 위해 자체적으로 unit test를 하여 코드의 정상 작동 여부를 판단한다.

5.3.4.1 Sequence Diagram



[Figure 38] Sequence Diagram – Judge System : Judge Problem



[Figure 39] Sequence Diagram – Judge System : API Test Tool

5.3.5 LLM System

5.3.5.1 State

LLM Provision(Prompt)

- provision_id: 프롬프트 고유 식별 ID
- input_data: 문제의 대한 정보
- output_data: 문제의 정답 테스트케이스

LLM Feedback

- feedback_id: 피드백 고유 식별 ID
- feedback_text: 채점 결과에 대한 상세 피드백
- improvement_suggestions: 채점 결과에 대한 개선점

LLM Provision

- provision_id: 리소스 할당 고유 ID
- allocated_resources: LLM 요청을 처리하기 위해 할당된 자원 정보 (예: 메모리, CPU)
- request_status: LLM 요청의 현재 상태 (Pending/Success/Failed)

LLM Q&A

- qa_id: 질의응답 고유 ID
- question: 사용자가 제출한 질문
- answer: LLM 이 생성한 답변

5.3.5.2 Methods

LLM system

- generateProblem(): 사용자의 관심 언어와 분야에 맞는 문제 생성
- provideFeedback(): 사용자 제출 코드에 대한 피드백 요청
- provideQandA(): 사용자가 제출한 질문에 대한 실시간 질의응답 처리

LLM Feedback

- provideFeedback(): 채점 결과를 기반으로 사용자가 해결한 문제에 대한 피드백 생성
- calculateScore(): 문제 해결 정확도를 점수로 반환

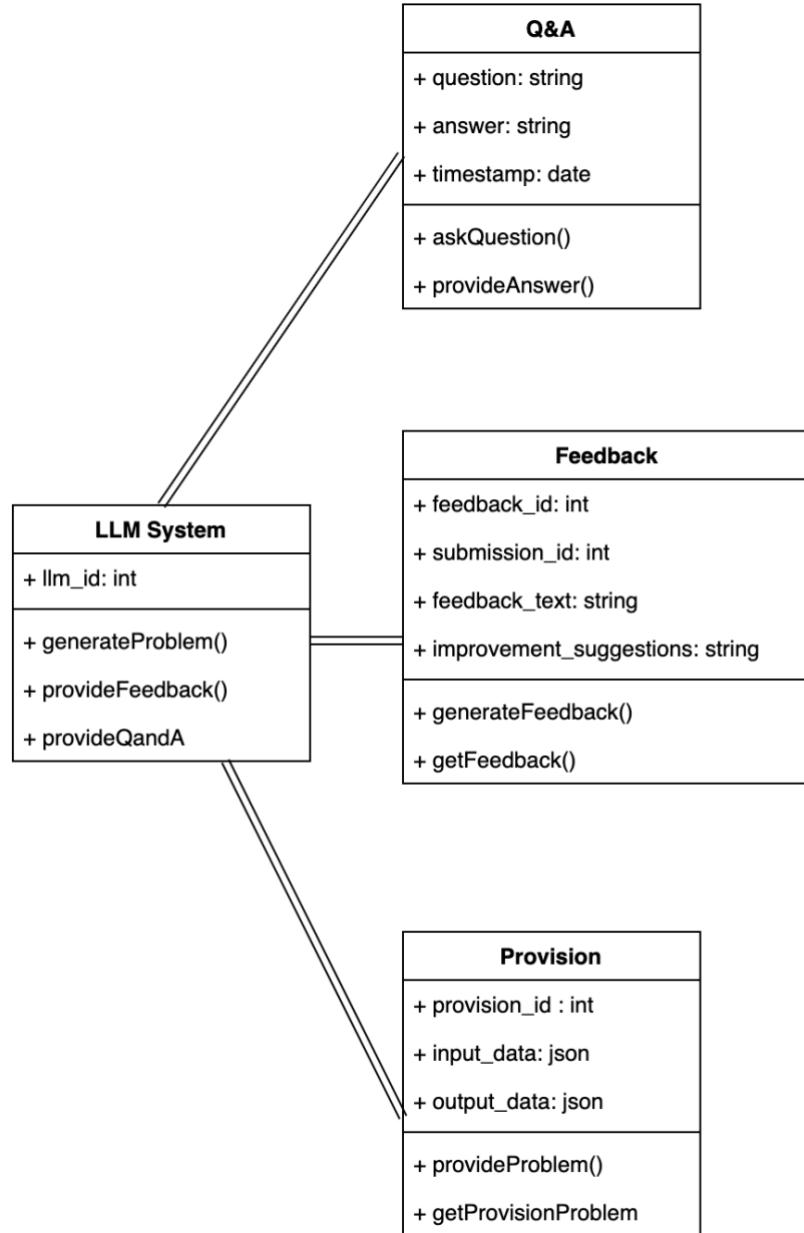
LLM Provision

- provideProblem(): 문제 생성
- getProvisionProblem(): 제공된 문제 반환

LLM Q&A

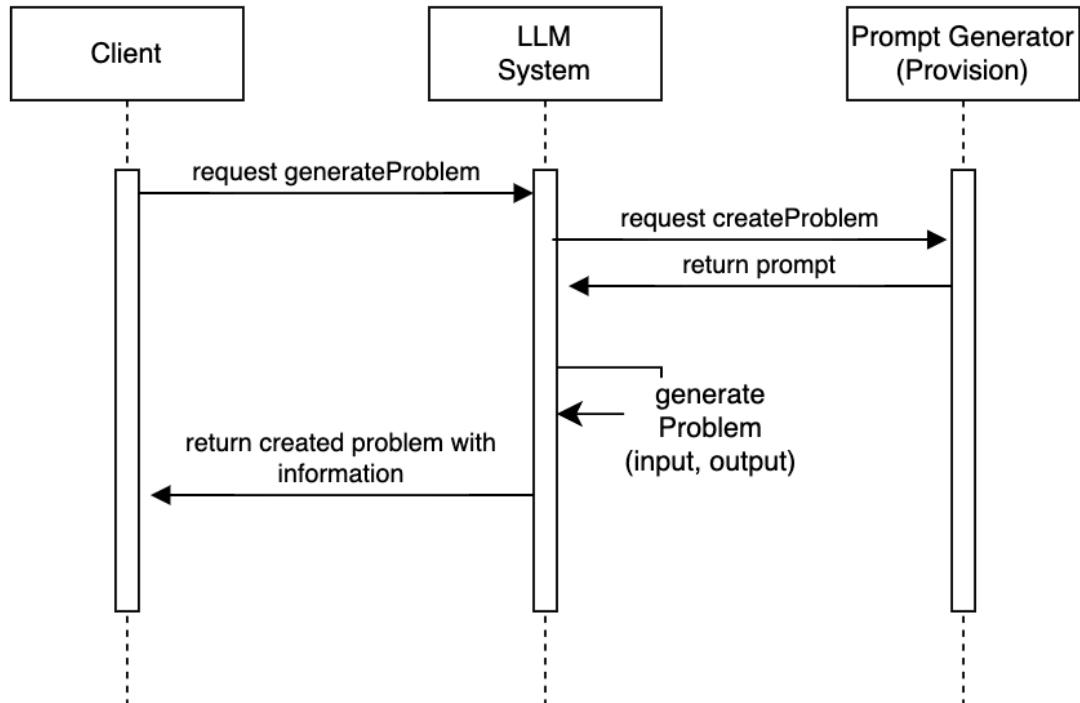
- AskQuestion(): 사용자가 제출한 질문에 대해 요청 생성
- provideAnswer(): 사용자가 제출한 질문에 대한 답변

5.3.5.3 Class Diagram

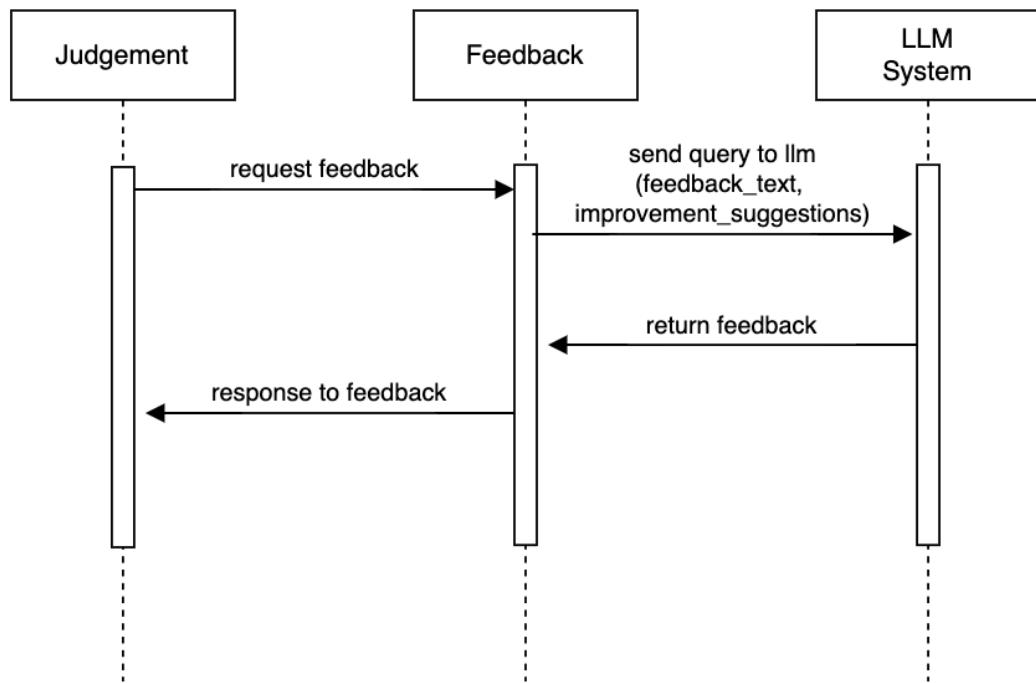


[Figure 40] Class Diagram – LLM System

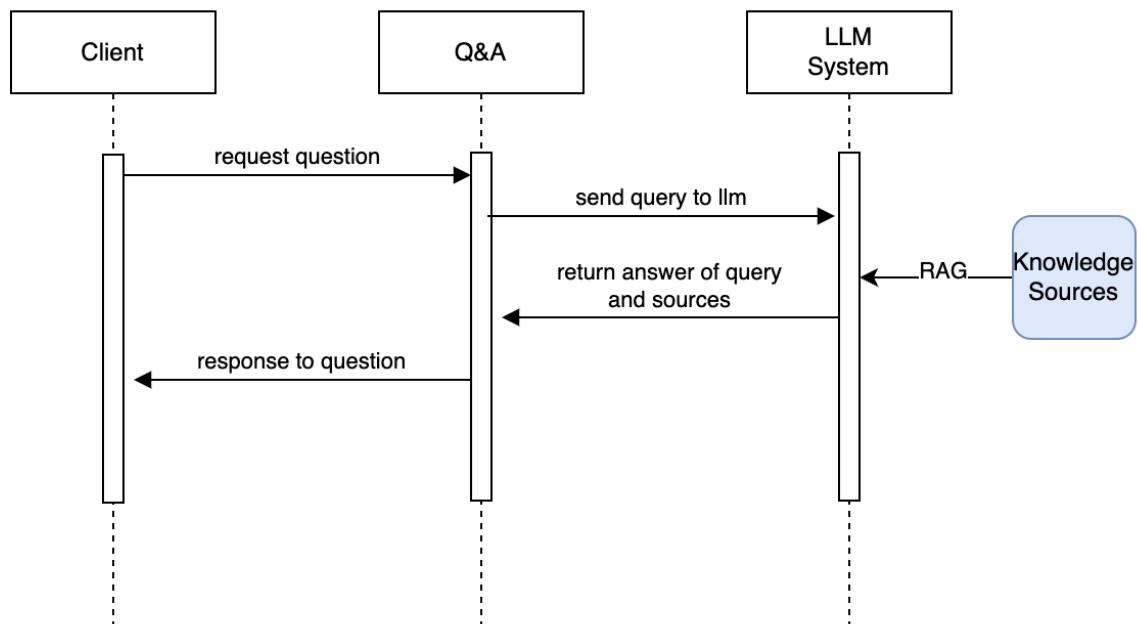
5.3.5.4 Sequence Diagram



[Figure 41] Sequence Diagram – LLM System : Provide Problem



[Figure 42] Sequence Diagram – LLM System : Problem Feedback



[Figure 43] Sequence Diagram – LLM System : User Q&A

6. Protocol Design

6.1 Objective

이번 챕터는 클라이언트와 서버 간의 안전하고 효율적인 데이터 통신을 보장하기 위해 프로토콜 설계 원칙과 구현 방식을 설명하며, JSON 데이터 형식, HTTPS 보안, CSRF 방지 및 세션 관리를 통해 신뢰성과 확장성을 제공하는 데 목적이 있다.

6.2 HTTPS (Hypertext Transfer Protocol Secure)

HTTPS는 웹 브라우저와 서버 간 통신을 암호화하여 데이터를 안전하게 전송하는 프로토콜입니다. HTTP의 확장 버전으로, TLS(Transport Layer Security) 또는 SSL(Secure Sockets Layer)을 사용하여 통신 내용을 암호화합니다.

이를 통해 데이터 도청, 변조, 중간자 공격(Man-in-the-middle attack)과 같은 보안 문제를 방지할 수 있습니다.

6.3 CSRF (Cross-Site Request Forgery) Token

CSRF 토큰은 웹 애플리케이션의 사용자가 의도하지 않은 명령을 실행하지 않도록 보호하는 보안 메커니즘입니다. 이 토큰은 주로 사용자가 로그인한 상태에서 악성 웹사이트가 사용자의 권한을 도용하는 공격을 방지하기 위해 사용됩니다.

6.4 Session Handling

세션 관리는 사용자가 서버에 접속한 후 일정 시간 동안 인증 상태를 유지하도록 관리하는 기술입니다. 세션을 통해 사용자는 로그인 상태를 유지하며 서버와 상호작용할 수 있습니다.

6.5 JSON (JavaScript Object Notation)

JSON은 데이터를 저장하거나 교환할 때 사용하는 경량의 데이터 포맷입니다. 텍스트 형식으로 작성되며, 사람이 읽기 쉽고 기계가 쉽게 분석하고 생성할 수 있는 구조를 가집니다.

6.6 Authentication

6.6.1 Register

Request

[Table 1] Table of Register Request

Attribute	Detail	
Protocol	HTTPS	
Method	POST	
Request Body	Email	사용자 아이디 (이메일)
	Username	사용자 이름
	Password	사용자 암호
	Interest_language	사용자 관심 언어
	Interest_field	사용자 관심 분야

Response

[Table 2] Table of Register Response

Attribute	Detail	
Protocol	HTTPS	
Success code	201 Created	
Failure code	400 Bad Request	필수 필드가 누락되었을 때
	409 Conflict	중복된 email 또는 username이 있을 때.
Success Response Body	Message	Register 성공 메세지
Failure Response Body	Message	Register 실패 메세지

6.6.2 Login & Logout

6.6.2.1 Login

Request

[Table 3] Table of Login Request

Attribute	Detail	
Protocol	HTTPS	
Method	POST	
Request Body	Email	사용자 아이디 (이메일)
	Password	사용자 암호
	CSRF token	요청 위조 방지를 위한 인증용 토큰

Response

[Table 4] Table of Login Response

Attribute	Detail	
Protocol	HTTPS	
Success code	200 OK	
Failure code	401 Unauthorized	일반적인 인증 실패 (예: 잘못된 비밀번호).
	400 Bad Request	요청이 잘못된 형식일 때 (예: 이메일이나 비밀번호 필드가 누락됨).
Success Response Body	Message	Login 성공 메세지
	CSRF token	요청 위조 방지를 위한 인증용 토큰
	Session ID	사용자 세션을 식별하기 위한 고유 식별자
Failure Response Body	Message	Login 실패 메세지

6.6.2.2 Logout

Request

[Table 5] Table of Logout Request

Attribute	Detail	
Protocol	HTTPS	
Method	POST	
Request Body	CSRF token	요청 위조 방지를 위한 인증용 토큰

Response

[Table 6] Table of Logout Response

Attribute	Detail	
Protocol	HTTPS	
Success code	200 OK	
Failure code	401 Unauthorized	토큰 인증 실패
Success Response Body	Message	Logout 성공 메세지
	CSRF token	요청 위조 방지를 위한 인증용 토큰
Failure Response Body	Message	Logout 실패 메세지 (유효하지 않은 토큰)

6.6.3 Reset Password (send email)

Request

[Table 7] Table of Reset Password(send email) Request

Attribute	Detail	
Protocol	HTTPS	
Method	POST	
Request Body	email	사용자 아이디 (이메일)

Response

[Table 8] Table of Reset Password(send email) Response

Attribute	Detail	
Protocol	HTTPS	
Success code	200 OK	
Failure code	400 Bad Request	이메일 존재하지 않음
Success Response Body	Message	Reset password 요청 성공 메세지
Failure Response Body	Message	Reset password 요청 실패 메세지

6.6.4 Reset Password (verification code)

Request

[Table 9] Table of Reset Password(verification code) Request

Attribute	Detail	
Protocol	HTTPS	
Method	POST	
Request Body	email	사용자 아이디 (이메일)
	Verification code	사용자가 이메일로 전송 받은 verification 입력

Response

[Table 10] Table of Reset Password(verification code) Response

Attribute	Detail	
Protocol	HTTPS	
Success code	200 OK	
Failure code	400 Bad Request	Verification failed
Success Response Body	Message	Reset Password(verification code) 요청 성공 메세지
	CSRF token	요청 위조 방지를 위한 인증용 토큰
Failure Response Body	Message	Reset Password(verification code) 요청 실패 메세지

6.6.5 Reset Password (send new password)

Request

[Table 11] Table of Reset Password(send new password) Request

Attribute	Detail	
Protocol	HTTPS	
Method	POST	
Request Body	email	사용자 아이디 (이메일)
	CSRF token	요청 위조 방지를 위한 인증용 토큰
	New password	사용자가 재설정하고자 하는 새로운 비밀번호

Response

[Table 12] Table of Reset Password(send new password) Response

Attribute	Detail	
Protocol	HTTPS	
Success code	200 OK	
Failure code	400 Bad Request	Reset password 요청 실패
Success Response Body	Message	Table of Reset Password(send new password) 요청 성공 메세지
Failure Response Body	Message	Table of Reset Password(send new password) 요청 실패 메세지

6.6.6 Submit Solution

Request

[Table 13] Table of Submit Solution Request

Attribute	Detail	
Protocol	HTTPS	
Method	POST	
Request Body	Email	사용자 아이디 (이메일)
	Problem ID	문제 고유번호
	Solution Code	풀이 코드
	CSRF Token	요청 위조 방지를 위한 인증용 토큰

Response

[Table 14] Table of Submit Solution Response

Attribute	Detail	
Protocol	HTTPS	
Success code	200 OK	
Failure code	400 Bad Request	요청 형식 오류
Success Response Body	Message	Submit Solution 요청 성공 메세지
	Submission ID	제출한 문제 고유번호
Failure Response Body	Message	Submit Solution 요청 실패 메세지

6.6.7 Provide Problem

Request

[Table 15] Table of Provide Problem Request

Attribute	Detail	
Protocol	HTTPS	
Method	GET	
Request Body	Email	사용자 아이디 (이메일)
	Problem ID	문제 고유번호
	CSRF Token	요청 위조 방지를 위한 인증용 토큰

Response

[Table 16] Table of Provide Problem Response

Attribute	Detail	
Protocol	HTTPS	
Success code	200 OK	
Success Response Body	Failure code	400 Bad Request 요청 형식 오류
	Message	Provide Problem 요청 성공 메세지
	Problem ID	제출한 문제 고유번호
	Problem metadata	문제 meta data (title, context, level, keyword, ...)
Failure Response Body	Problem Code	문제 코드
	Message	Provide Problem 요청 실패 메세지

6.6.8 Judge Submission

Request

[Table 17] Table of Judge Submission Request

Attribute	Detail	
Protocol	HTTPS	
Method	POST	
Request Body	Email	사용자 아이디 (이메일)
	Submission ID	제출 고유번호
	Problem ID	문제 고유번호
	CSRF Token	요청 위조 방지를 위한 인증용 토큰

Response

[Table 18] Table of Judge Submission Response

Attribute	Detail	
Protocol	HTTPS	
Success code	200 OK	
Failure code	400 Bad Request	요청 형식 오류
Success Response Body	Message	Judge Submission 요청 성공 메세지
	Submission ID	제출한 문제 고유번호
	Result	문제 풀이 결과
	Feedback data	피드백 정보
Failure Response Body	Message	Judge Submission 요청 실패 메세지

6.6.9 Chatbot Interaction

Request

[Table 19] Table of Chatbot Interaction Request

Attribute	Detail	
Protocol	HTTPS	
Method	POST	
Request Body	Email	사용자 아이디 (이메일)
	Message	사용자 입력 메세지
	Session ID	사용자 세션을 식별하기 위한 고유 식별자
	CSRF Token	요청 위조 방지를 위한 인증용 토큰

Response

[Table 20] Table of Chatbot Interaction Response

Attribute	Detail	
Protocol	HTTPS	
Success code	200 OK	
Failure code	400 Bad Request	요청 형식 오류
Success Response Body	Bot Response	챗봇 응답
	Timestamp	응답 시간
Failure Response Body	Message	Chatbot Interaction 요청 실패 메세지

6.6.10 Provide Problem Comments

Request

[Table 21] Table of Problem Comments Request

Attribute	Detail	
Protocol	HTTPS	
Method	GET	
Request Body	Email	사용자 아이디 (이메일)
	Problem ID	문제 고유번호
	CSRF Token	요청 위조 방지를 위한 인증용 토큰

Response

[Table 22] Table of Problem Comments Response

Attribute	Detail	
Protocol	HTTPS	
Success code	200 OK	
Failure code	400 Bad Request	요청 형식 오류
Success Response Body	Message	Provide Problem Comment 요청 성공 메세지
	Comment IDs	제출한 문제의 Comment 고유번호 List
	Comment	Comment content
Failure Response Body	Message	Provide Problem Comment 요청 실패 메세지

6.6.11 Provide User Profile

Request

[Table 23] Table of Provide User Profile Request

Attribute	Detail	
Protocol	HTTPS	
Method	GET	
Request Body	Email	사용자 아이디 (이메일)
	Profile ID	프로필 아이디
	CSRF Token	요청 위조 방지를 위한 인증용 토큰

Response

[Table 24] Table of Provide User Profile Response

Attribute	Detail	
Protocol	HTTPS	
Success code	200 OK	
Failure code	400 Bad Request	요청 형식 오류
Success Response Body	Message	Provide Comment 요청 성공 메세지
	Original Img	원본 이미지
	Uploaded Img	처리 후 업로드된 이미지
Failure Response Body	Message	Provide User Profile 요청 실패 메세지

6.6.12 Provide Learning Progress

Request

[Table 25] Table of Provide Learning Progress Request

Attribute	Detail	
Protocol	HTTPS	
Method	GET	
Request Body	Email	사용자 아이디 (이메일)
	CSRF Token	요청 위조 방지를 위한 인증용 토큰

Response

[Table 26] Table of Provide Learning Progress Request

Attribute	Detail	
Protocol	HTTPS	
Success code	200 OK	
Failure code	400 Bad Request	요청 형식 오류
Success Response Body	Message	Provide Learning Progress 요청 성공 메세지
	Dashboard id	사용자 대시보드 아이디
	Dashboard content	Json 형태의 대시보드 content
	Stat id	Problem statistics 아이디
	Stat content	Json 형태의 Problem statistics content
Failure Response Body	Message	Provide Learning Progress 요청 실패 메세지

6.6.13 Provide User Information

Request

[Table 27] Table of Provide User Information Request

Attribute	Detail	
Protocol	HTTPS	
Method	GET	
Request Body	Email	사용자 아이디 (이메일)
	CSRF Token	요청 위조 방지를 위한 인증용 토큰

Response

[Table 28] Table of Provide User Information Response

Attribute	Detail	
Protocol	HTTPS	
Success code	200 OK	
Failure code	400 Bad Request	요청 형식 오류
Success Response Body	Message	Provide User Information 요청 성공 메세지
	User id	사용자 아이디
	User role	유저 타입 (일반, 관리자)
	User Content	Json 형태의 User class 정보
Failure Response Body	Message	Provide User Information 요청 실패 메세지

6.6.14 Provide Problem Comments

Request

[Table 29] Table of User Comments Response

Attribute	Detail	
Protocol	HTTPS	
Method	GET	
Request Body	Email	사용자 아이디 (이메일)
	CSRF Token	요청 위조 방지를 위한 인증용 토큰

Response

[Table 30] Table of User Comments Response

Attribute	Detail	
Protocol	HTTPS	
Success code	200 OK	
Failure code	400 Bad Request	요청 형식 오류
Success Response Body	Message	Provide User Comment 요청 성공 메세지
	Problem IDs	User의 모든 Problem IDs
	Comment IDs	User의 모든 Comment IDs
	Comments	Comment contents
Failure Response Body	Message	Provide User Comment 요청 실패 메세지

7. Database Design

7.1 Objectives

이 챕터는 시스템의 데이터 구조와 이러한 구조가 데이터베이스로 어떻게 구현되었는지를 설명하는 것을 목적으로 한다. 이를 위해 먼저 ER 다이어그램(Entity-Relationship Diagram)을 사용하여 주요 엔티티와 그 관계를 식별한다. 이후, 이를 바탕으로 관계형 스키마를 정의하고, SQL DDL(Data Definition Language)을 통해 데이터베이스 구조를 구현하는 과정을 서술한다.

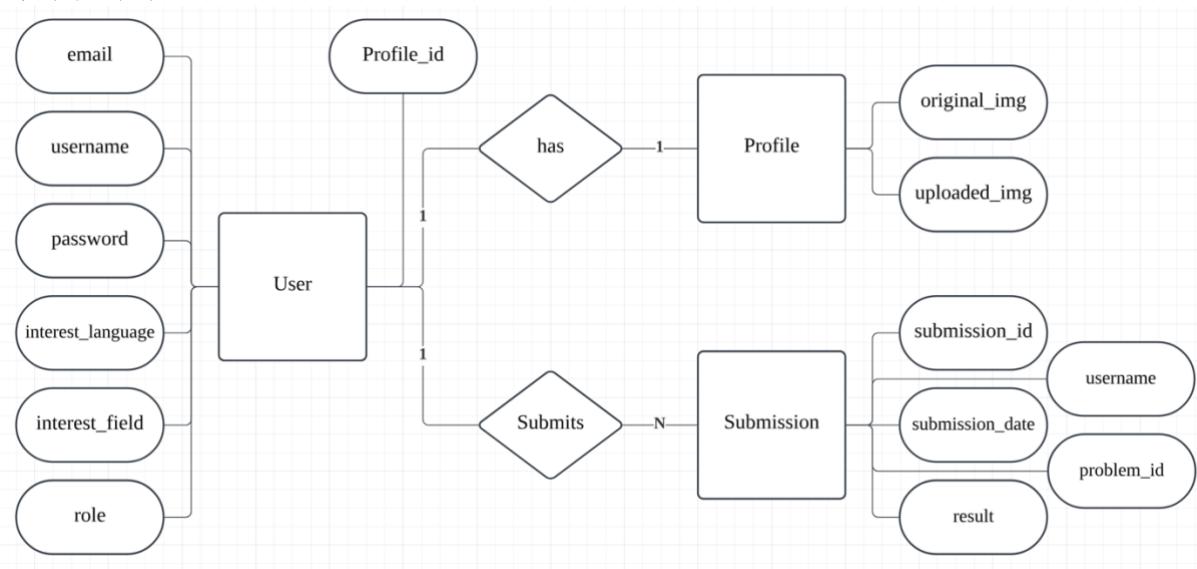
7.2 ER Diagram

본 어플리케이션 시스템은 총 7 가지 entity로 이루어져 있다. User, Profile, Submission, Problem, Comment, Dashboard, Problem_statistics.

ER-Diagram은 entity 간의 관계, 그리고 entity와 attribute의 관계를 다이어그램으로 설명한다. 각 entity마다 대응되는 개수는 entity를 연결하는 선 주변에 표기되어 있어 확인 가능하다.

7.2.1 User-Profile, User-Submission

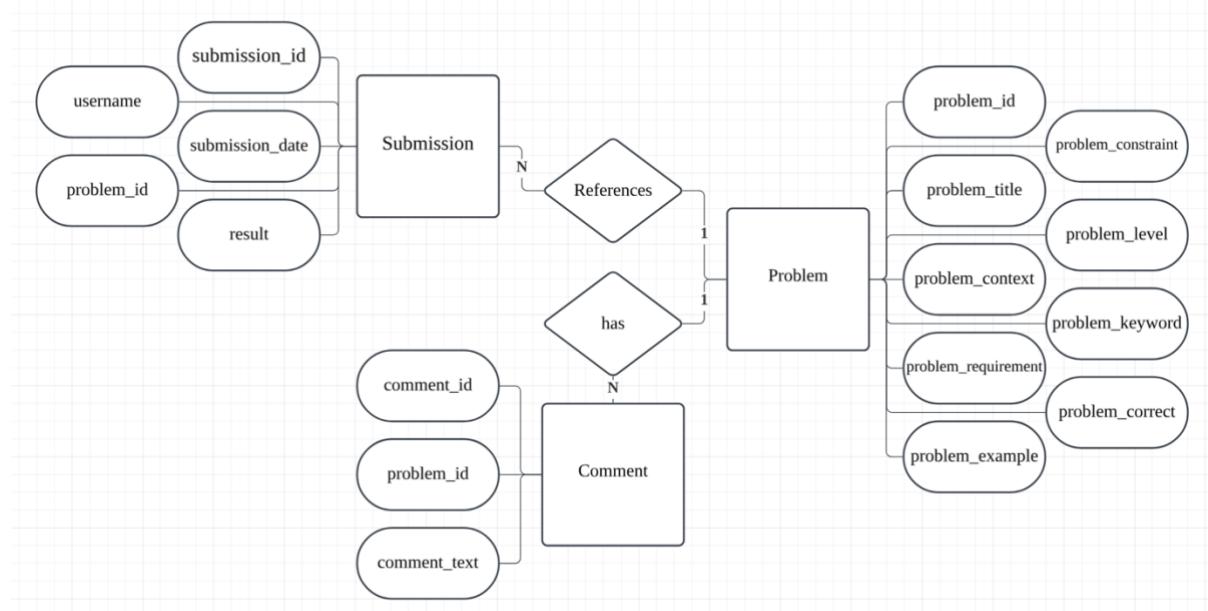
이 다이어그램은 사용자의 데이터와 관련된 구조를 설명합니다. 사용자는 이메일, 사용자 이름, 비밀번호, 관심 언어와 관심 분야, 역할 등의 정보를 가지며, 각 사용자는 하나의 프로필과 연결됩니다. 프로필에는 원본 이미지와 업로드된 이미지 정보가 포함됩니다. 또한 사용자는 여러 제출 기록을 가질 수 있으며, 각 제출은 제출 ID, 제출 날짜, 결과 등의 정보를 포함하고, 문제 ID와 연결됩니다. 이 구조는 사용자의 프로필과 제출 데이터를 체계적으로 관리하기 위한 시스템을 나타냅니다.



[Figure 44] User-Profile, User-Submission

7.2.2 Submission-Problem, Problem-Comment

제출(Submission)은 제출 ID, 사용자 이름, 문제 ID, 제출 날짜, 결과 등의 정보를 포함하며, 하나의 문제(Problem)와 연결됩니다. 문제는 문제 ID, 제목, 설명, 요구사항, 예제, 제약 조건, 난이도, 키워드, 정답 여부 등의 정보를 가집니다. 각 문제는 여러 댓글(Comment)을 가질 수 있으며, 댓글은 댓글 ID, 문제 ID, 댓글 내용을 포함합니다. 이 구조는 문제, 제출, 댓글 데이터를 효과적으로 관리하기 위해 설계되었습니다.

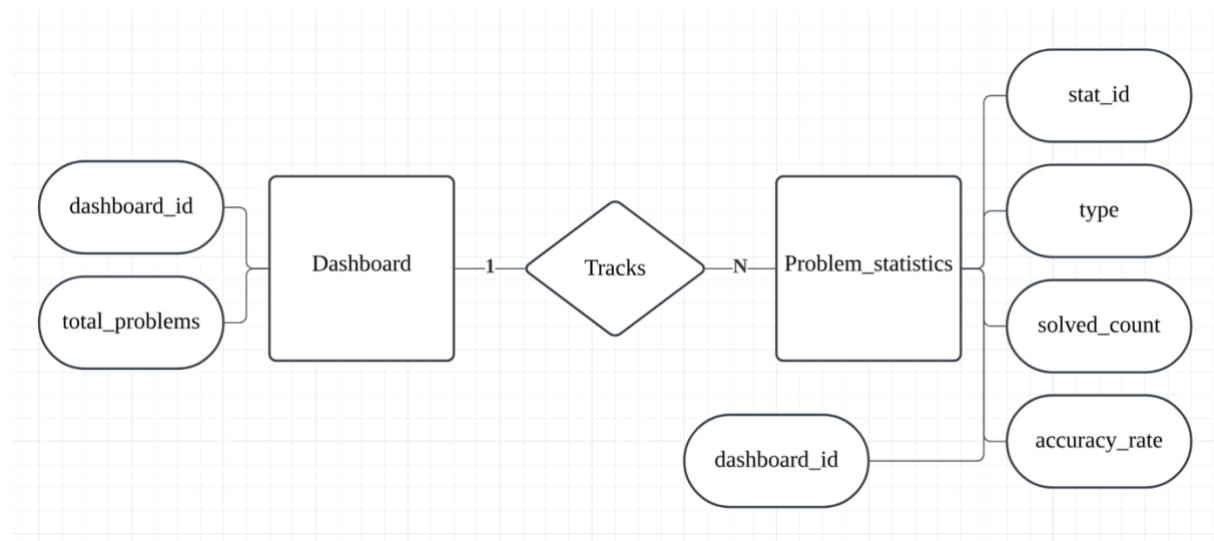


[Figure 45] Submission-Problem, Problem-Comment

7.2.3 Dashboard-Problem_statistics

이 다이어그램은 대시보드와 문제 통계 데이터를 설명합니다. 대시보드(Dashboard)는 대시보드 ID 와 총 문제 수(total_problems)를 포함하며, 하나의 대시보드는 여러 문제 통계(Problem_statistics)와 연결됩니다. 문제 통계는 통계 ID(stat_id), 문제 유형(type), 해결된 문제 수(solved_count), 정답률(accuracy_rate) 등의 정보를 포함하며, 각 통계는 대시보드 ID 를 참조합니다. 이러한 구조는 대시보드별로 문제 풀이 통계를 효과적으로 추적하고 관리하기 위해 설계되었습니다.

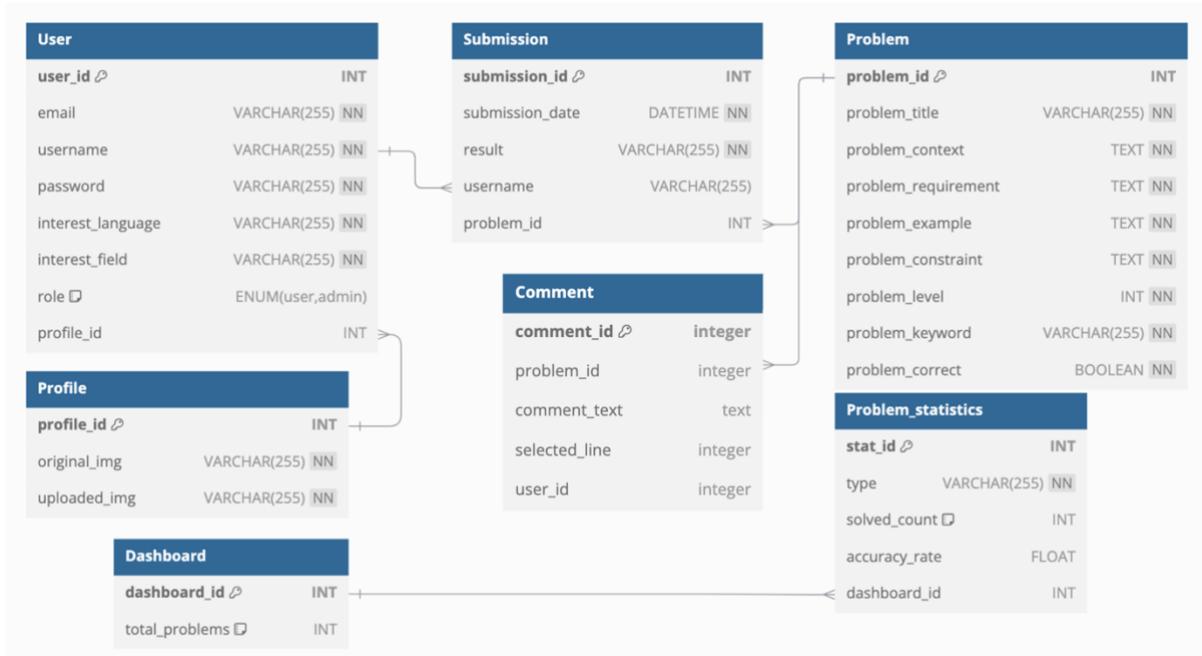
[Figure 46] Dashboard-Problem_statistics



7.3 Relational Schema

CSRF 토큰은 웹 애플리케이션의 사용자가 의도하지 않은 명령을 실행하지 않도록 보호하는 보안 메커니즘입니다. 이 토큰은 주로 사용자가 로그인한 상태에서 악성 웹사이트가 사용자의 권한을 도용하는 공격을 방지하기 위해 사용됩니다.

[Figure 47] Entity Relational Schema



7.4 SQL DDL

7.4.1 User

[Figure 48] User SQL Schema

```

CREATE TABLE User (
    user_id INT PRIMARY KEY AUTO_INCREMENT, -- 사용자 아이디
    email VARCHAR(255) NOT NULL UNIQUE, -- 사용자 이메일
    username VARCHAR(255) NOT NULL UNIQUE, -- 사용자 이름
    password VARCHAR(255) NOT NULL, -- 사용자 암호
    interest_language VARCHAR(255) NOT NULL, -- 관심 언어
    interest_field VARCHAR(255) NOT NULL, -- 관심 분야
    role ENUM('user', 'admin') DEFAULT 'user', -- 일반/관리자 구분
    profile_id INT, -- 프로필 사진 FK
    FOREIGN KEY (profile_id) REFERENCES Profile(profile_id)
);

```

7.4.2 Profile

[Figure 49] Profile SQL Schema

```
-- Profile Table
CREATE TABLE Profile (
    profile_id INT PRIMARY KEY AUTO_INCREMENT, -- 프로필 사진 ID
    original_img VARCHAR(255) NOT NULL, -- 원본 파일명
    uploaded_img VARCHAR(255) NOT NULL -- S3 업로드명
);
```

7.4.3 Problem

[Figure 50] Problem SQL Schema

```
-- Problem Table
CREATE TABLE Problem (
    problem_id INT PRIMARY KEY AUTO_INCREMENT, -- 문제 아이디
    problem_title VARCHAR(255) NOT NULL, -- 문제 제목
    problem_context TEXT NOT NULL, -- 문제 설명
    problem_requirement TEXT NOT NULL, -- 문제 요구사항
    problem_example TEXT NOT NULL, -- 문제 화면 예시
    problem_constraint TEXT NOT NULL, -- 문제 채점 기준
    problem_level INT NOT NULL CHECK (problem_level BETWEEN 1 AND 5),
    problem_keyword VARCHAR(255) NOT NULL, -- 문제 키워드
    problem_correct BOOLEAN NOT NULL -- 문제 정답 여부
);
```

7.4.4 Submission

[Figure 51] Submission SQL Schema

```
-- Submission Table
CREATE TABLE Submission (
    submission_id INT PRIMARY KEY AUTO_INCREMENT, -- 제출 아이디
    submission_date DATETIME NOT NULL, -- 제출 날짜
    result VARCHAR(255) NOT NULL, -- 제출 결과
    username VARCHAR(255) NOT NULL, -- 제출한 유저 (FK)
    problem_id INT NOT NULL, -- 제출된 문제 ID (FK)
    FOREIGN KEY (username) REFERENCES User(username),
    FOREIGN KEY (problem_id) REFERENCES Problem(problem_id)
);
```

7.4.5 Comment

[Figure 52] Comment SQL Schema

```
-- Comment Table
CREATE TABLE Comment (
    comment_id INT PRIMARY KEY AUTO_INCREMENT, -- 댓글 ID
    problem_id INT NOT NULL, -- 해당 문의의 ID (FK)
    comment_text TEXT NOT NULL, -- 댓글 내용
    selected_line INT, -- 선택된 코드 라인
    user_id INT, -- 댓글을 단 유저 ID
    FOREIGN KEY (problem_id) REFERENCES Problem(problem_id),
    FOREIGN KEY (user_id) REFERENCES User(user_id)
);
```

7.4.6 Dashboard

[Figure 53] Dashboard SQL Schema

```
-- Dashboard Table
CREATE TABLE Dashboard (
    dashboard_id INT PRIMARY KEY AUTO_INCREMENT, -- 대시보드 ID
    total_problems INT DEFAULT 0 -- 생성한 전체 문제 수
);
```

7.4.7 Problem Statistic

[Figure 54] Problem Statistic SQL Schema

```
-- Problem Statistics Table
CREATE TABLE Problem_statistics (
    stat_id INT PRIMARY KEY AUTO_INCREMENT, -- 통계 ID
    type VARCHAR(255) NOT NULL, -- 문제 유형
    solved_count INT DEFAULT 0, -- 푼 문제 수
    accuracy_rate FLOAT CHECK (accuracy_rate BETWEEN 0 AND 100), -
    dashboard_id INT NOT NULL, -- 대시보드와 연결 (FK)
    FOREIGN KEY (dashboard_id) REFERENCES Dashboard(dashboard_id)
);
```

8. Testing Plan

8.1. Objective

WebShooter 시스템의 테스트는 플랫폼의 안정성과 신뢰성을 보장하기 위한 필수적인 절차로 정의된다. 본 계획은 기능적 및 비기능적 요구사항을 충족하는지 확인함으로써 시스템의 오류를 사전에 예방하고, 사용자 경험을 최적화하는 것을 목표로 한다. 이를 통해 시스템이 기대한 대로 작동하며, 사용자에게 원활한 학습 환경을 제공할 수 있도록 한다.

8.2. Testing Policy

8.2.1. Release Testing

Release Testing은 WebShooter 시스템의 주요 성능 요구사항을 충족하고, 실제 운영 환경에서 안정적으로 작동하는지 확인하기 위해 설계되었다. 아래는 각 항목에 대한 상세 설명 및 테스트 케이스이다.

8.2.1.1. End-to-End Testing

End-to-End Testing은 WebShooter의 주요 기능이 사용자 관점에서 명세서에 정의된 대로 작동하는지를 확인한다.

각 테스트 케이스는 수동 및 자동화된 방법으로 수행되며, 테스트 도구를 활용하여 반복성과 정확성을 보장한다.

Authentication Page:

- Objective: 회원가입, 로그인, 비밀번호 재설정 등 인증 기능의 정상 동작 확인
- Method:
 - Tool: Selenium (UI 자동화), Postman (API 테스트), Chrome Developer Tools (Network monitoring)
 - Automated Test:
 - Selenium을 활용하여 회원가입/로그인 플로우를 스크립트로 작성
 - 잘못된 입력값 처리 테스트: Postman으로 API 호출 및 오류 메시지 검증
 - Partition Testing:
 - 비밀번호 길이:

- 정상 값: 8자, 128자
- 경계값: 7자, 129자
- 비정상 값: 공백, 특수문자만 포함
- 이메일 형식:
 - 정상 값: user@example.com
 - 비정상 값: user@com, user.com, 공백
- Manual Test:
 - 세션 유지 및 동시 로그인 기능을 직접 검증
 - 개발자 도구에서 로그인 상태가 쿠키나 세션 스토리지에 정상적으로 저장되었는지 확인
- Verification Point:
 - 회원가입 및 로그인 처리 시간이 5초 이내에 완료되는지 확인
 - 비밀번호 재설정 요청 시 이메일이 정상 전송되고 링크가 작동
 - 잘못된 입력값에 대해 오류 메시지가 명확히 표시

Problem List:

- Objective: 사용자가 요청한 주제와 난이도에 맞는 문제 생성 및 데이터 동기화 검증
- Method:
 - Tool: Selenium, Database Query Tools (SQLAlchemy, pgAdmin)
 - Automated Test:
 - 문제 생성 요청 후 목록이 업데이트 되는지 Selenium으로 확인
 - Database Query로 문제 정보의 정확성 검증
 - Partition Testing:
 - 난이도:
 - 정상 값: Easy, Medium, Hard.
 - 경계값: NULL
 - 비정상 값: 숫자 입력(예: 123), 빈 문자열
 - 주제:
 - 정상 값: "Frontend", "Backend"

- 비정상 값: "12345", 공백, 특수문자만 포함
- Manual Test:
 - 여러 난이도와 주제로 문제 생성 요청 후 목록 반영 여부 확인
- Verification Point:
 - 사용자가 요청한 주제와 난이도에 맞는 문제가 2분 이내에 생성
 - 목록에 표시된 문제 정보가 데이터베이스와 일치

Prompt Engineering Workflow:

- Objective: 프롬프트 실행 결과가 요구사항을 충족하고 예상대로 작동하는지 확인
- Method:
 - Tool: Python + pytest (검증 스크립트), Postman (API 호출)
 - Automated Test:
 - Python 스크립트로 프롬프트 실행 후 생성된 문제의 입력/출력 구조 검증
 - 실패 시 대체 프롬프트로 자동 재생성 테스트
 - Manual Test:
 - 생성된 문제를 UI 상에서 검토하여 검증 기준 충족 여부 확인
- Verification Point:
 - 문제 생성 및 검증 완료가 2분 이내에 완료되는지 확인
 - 실패 시 대체 프롬프트로 재생성 후 동일 과정 반복

Code Editor 및 Live Preview:

- Objective: 실시간 편집 및 미리보기 기능의 정상 작동 확인
- Method:
 - Tool: Cypress (UI 반응 테스트), Network Sniffer (실시간 통신 확인)
 - Automated Test:
 - Cypress로 사용자가 질문 입력 후 챗봇 응답 시간을 측정
 - 비정상 입력값 처리 및 오류 메시지 검증
 - Manual Test:
 - 사용자가 질문을 제출하고 챗봇의 응답을 확인하는 여부

- 에디터의 실시간 미리보기 기능 동작 여부 확인
- Verification Point:
 - 챗봇의 응답이 2분 이내에 반환되는지 확인
 - 비정상 입력에 대해 적절한 오류 메시지 반환

AI Chatbot

- Objective: 챗봇의 핫 리로딩 및 자동 저장 기능 확인
- Method:
 - Tool: Browser Developer Tools (LocalStorage 확인), Cypress
 - Automated Test:
 - Cypress로 코드 저장 후 새로고침 시 복원 여부 확인
 - Manual Test:
 - 핫 리로딩 및 코드 하이라이팅 기능 직접 테스트
- Verification Point:
 - 자동 저장된 코드 복원 및 실시간 동작 여부

Grading Panel:

- Objective: 코드 제출 후 피드백 및 채점 기능 검증
- Method:
 - Tool: Selenium, pytest (결과 검증 스크립트)
 - Automated Test:
 - 제출 후 5초 내에 피드백이 반환되는지 검증
 - 피드백 내용이 코드의 기능 및 UI 검증 여부 포함
 - Manual Test:
 - 복잡한 코드를 제출하여 다양한 피드백 시나리오 점검
- Verification Point:
 - 채점 및 피드백 제공이 5초 이내에 완료되는지 확인

Dashboard Update:

- Objective: 대시보드 업데이트 및 시각화 데이터의 정확성 검증
- Method:
 - Tool: Selenium (UI 테스트), SQLAlchemy (DB와 대시보드 데이터

비교)

- Automated Test:
 - Selenium으로 대시보드 그래프와 통계 정보가 업데이트되는지 확인
 - 데이터베이스와 대시보드 데이터의 일치 여부 검증
- Manual Test:
 - 대시보드 데이터를 시각적으로 점검
- Verification Point:
 - 데이터 업데이트가 5초 이내에 완료되는지 확인
 - 시각화된 정보가 데이터베이스 기록과 정확히 일치하는지 확인

8.2.1.2. Performance Testing

Performance Testing은 WebShooter의 성능이 비기능적 요구사항과 예상된 성능 목표를 충족하는지 검증한다. 각 테스트 케이스는 수동 및 자동화된 방법으로 수행되며, 테스트 도구를 활용하여 반복성과 정확성을 보장한다.

동시 사용자 테스트:

- Objective: 동시 사용자 접속 및 요청 처리 성능 확인
- Method:
 - Tool: Apache JMeter, Locust, AWS CloudWatch.
 - Automated Test:
 - JMeter로 200명의 동시 사용자 접속을 시뮬레이션
 - Locust로 50명 이상의 사용자 요청(챗봇 질문 및 문제 생성) 처리 테스트
 - Manual Test:
 - 실제 다수의 테스트 계정을 사용하여, 각기 다른 디바이스와 네트워크 환경에서 동시 작업 수행
 - AWS CloudWatch 대시보드를 통해 요청 처리 상태 및 서버 부하 확인
- Verification Points:
 - 서버 응답 시간이 2초 이내로 유지되는지 확인
 - 동시 요청 처리에서 병목 현상 없는지 확인

문제 생성 및 검증 시간 테스트:

- Objective: 문제 생성과 검증 속도 확인
- Method:
 - Tool: Python + pytest, Selenium
 - Automated Test:
 - GPT API 호출을 통해 문제 생성 요청 후 처리 시간 측정
 - 자동 input/output 테스트와 검증 실패 시 대체 프롬프트 처리 확인
 - Manual Test:
 - UI에서 직접 문제를 생성하고, 처리 완료 시간을 측정
 - UI와 데이터베이스 간 동기화를 확인하기 위해 데이터베이스 조회
- Verification Points:
 - 문제 생성이 2분 이내에 완료되는지 확인
 - 검증 실패 시 적절한 오류 메시지 및 대체 프롬프트 처리

챗봇 응답 시간:

- Objective: GPT API를 활용한 챗봇 응답 시간 확인
- Method:
 - Tool: Postman, Selenium
 - Automated Test:
 - Postman으로 다양한 질문을 전송하여 응답 시간을 기록
 - Manual Test:
 - UI에서 챗봇과 직접 상호작용하며 다양한 질문 시나리오를 테스트
 - 네트워크 상태를 변경(저속 연결, 연결 차단 등)하여 챗봇의 응답 처리 확인
- Verification Points:
 - 응답 시간이 2분 이내인지 확인
 - 네트워크 상태에 따라 시간 지연 발생 시에도 요청 처리 정상 완료

호스트 서버-클라이언트 통신 테스트:

- Objective: 클라이언트와 서버 간 데이터 통신의 정확성과 신뢰성 확인
- Method:
 - Tool: Wireshark, Browser Developer Tools (Network Tab).
 - Automated Test:
 - Selenium으로 클라이언트 요청 전송 후 서버 응답의 일관성 확인
 - WebSocket 종료 상황 시 데이터 유실 여부 검증
 - Manual Test:
 - 개발자 도구(Network 탭)를 통해 각 요청/응답 데이터를 직접 분석
 - 데이터 유실 및 전송 속도 직접 확인
 - Verification Points:
 - 데이터 전송 속도가 10Mbps 이상인지 확인
 - 소켓 종료 시 데이터 손실 없음

모니터링과 서버 성능 확인:

- Objective: 실시간 동기화 및 서버 처리 성능 확인
- Method:
 - Tool: AWS CloudWatch, Grafana
 - Automated Test:
 - Selenium으로 UI 데이터 동기화 상태를 확인
 - Manual Test:
 - Grafana 대시보드에서 실시간 서버 부하 및 데이터 동기화 상태를 수동 검토
 - Verification Points:
 - 동기화 지연이 100ms 이내인지 확인
 - 데이터 갱신 주기가 1초 이내인지 확인

페이지 로딩 시간:

- Objective: 페이지 로딩 시간이 성능 요구사항을 충족하는지 확인
- Method:

- Tool: Google Lighthouse, Selenium
- Automated Test:
 - Lighthouse로 페이지 로딩 시간 측정
- Manual Test:
 - 다양한 브라우저와 디바이스 환경에서 수동으로 로딩 시간 확인
 - 느린 네트워크 환경에서 동작 테스트
- Verification Points:
 - 페이지 로딩 시간이 3초이내인지 확인

데이터 저장 및 로드 시간:

- Objective: 학습 데이터 저장 및 불러오기 속도 확인
- Method:
 - Tool: Selenium, SQLAlchemy
 - Automated Test:
 - Selenium으로 데이터 저장 후 대시보드 갱신 시간을 기록
 - Manual Test:
 - 데이터를 저장한 후 UI에서 직접 데이터 로드를 확인
 - 저장된 데이터를 데이터베이스에서 직접 조회하여 정확성 검토
- Verification Points:
 - 저장 및 불러오기 작업 완료가 5초이내인지 확인

피드백 및 채점 응답 속도:

- Objective: 채점과 피드백 제공 속도 검증
- Method:
 - Tool: Locust (부하 테스트), Selenium
 - Automated Test:
 - Locust로 동시 코드 제출 요청을 시뮬레이션
 - 각 요청에 대한 채점 결과와 피드백 제공 시간 기록
- Verification Points:
 - 채점 및 피드백 제공 시간이 5초 이내에 완료

- 여러 요청에도 병목현상 발생 없음

실시간 코드 실행 응답성:

- Objective: 코드 실행 결과가 실시간으로 반영되는지 확인
- Method:
 - Tool: Selenium, Browser Developer Tools
 - Automated Test:
 - Live Preview 창에서 실행 결과 반영 시간 기록
 - Manual Test:
 - 핫 리로딩 기능을 직접 테스트하여 결과를 시각적으로 확인
- Verification Points:
 - 실행 결과 반영이 5초 이내에 이루어지는지 확인

시스템 관리자 페이지:

- Objective: 관리자 페이지의 데이터 처리 성능을 검증
- Method:
 - Tool: Selenium, Database Query Tools (pgAdmin)
 - Automated Test:
 - 사용자 정보 조회, 수정, 삭제 작업 처리 시간 측정
 - 게시판 수정/삭제 작업이 데이터베이스에 반영되는 시간 기록
- Verification Points:
 - 모든 작업이 5초 이내로 처리되고 반영
 - 플랫폼 운영 현황 데이터가 정확히 시각화

8.2.1.3. Compatibility Testing

WebShooter의 호환성을 다양한 환경에서 점검한다.

브라우저 호환성:

1. Chrome, Edge, Firefox 기능 검증

- Tools:
 - BrowserStack: 다양한 브라우저와 버전에서의 호환성 테스트
 - Selenium WebDriver: 주요 브라우저에서 테스트 스크립트를 실행하여 기능 검증

- Lighthouse (Chrome Developer Tools): 성능, 접근성, SEO, 및 브라우저 렌더링 검토
- Method:
 - BrowserStack을 사용해 다양한 브라우저(Chrome, Edge, Firefox)에서 주요 기능(로그인, 문제 생성, 대시보드 업데이트 등)을 실행
 - Selenium WebDriver를 통해 자동화 테스트 스크립트를 작성하여 기능 검증
 - Lighthouse를 사용해 렌더링 문제 및 성능 병목 현상 확인
- Validation Points:
 - 모든 브라우저에서 UI가 동일하게 렌더링되는지 확인
 - 브라우저 간 기능 차이 없음

2. API 호출 및 오류 메세지 확인

- Tools:
 - Postman: API 요청 테스트
 - Selenium: 브라우저 자동화 및 네트워크 에러 확인
- Method:
 - Postman으로 브라우저별 API 요청 동작 검증
 - Selenium 스크립트를 작성하여 네트워크 요청 실패 상황을 시뮬레이션하고 오류 메시지 확인
- Validation Points:
 - API 요청 결과가 브라우저 간 동일
 - 네트워크 실패 시 사용자 친화적인 오류 메시지가 표시

3. JavaScript 비활성화 테스트

- Tools:
 - Browser Developer Tools: JavaScript 비활성화 기능
 - Selenium: JavaScript가 비활성화된 상태에서 자동화 테스트
- Method:
 - 브라우저의 개발자 도구를 사용하여 JavaScript 비활성화 후, 제한된 기능 안내 메시지가 표시되는지 수동 확인
 - Selenium으로 JavaScript 비활성화 상태를 시뮬레이션하여 주요 경고 메시지 확인

- Validation Points:

- JavaScript 비활성화 시 명확한 경고 메시지가 표시
- 제한된 기능 안내가 정확히 제공

디바이스 호환성:

1. 최소 사양 환경 검증

- Tools:

- Real Device Testing: 실제 저사양 기기를 사용해 테스트
- BrowserStack: 저사양 환경을 시뮬레이션

- Method:

- BrowserStack을 활용해 최소 사양(1.0GHz 프로세서, 2GB RAM)에서의 브라우저 성능 테스트
- 실제 저사양 기기를 사용해 성능 및 UI 반응성 확인

- Validation Points:

- 저사양 기기에서 주요 기능이 정상 작동

2. 고해상도 디스플레이 검증

- Tools:

- Figma Mirror: UI 디자인과 실제 구현된 UI 비교
- BrowserStack: 고해상도 환경 시뮬레이션

- Method:

- BrowserStack을 사용하여 4K/QHD 환경에서 UI 확인
- Figma Mirror로 디자인 파일과 구현된 UI의 픽셀 정확도 비교

- Validation Points:

- 고해상도 환경에서도 UI가 정확히 배치
- 텍스트 및 이미지가 스케일링 문제 없이 표시

3. 반응형 설계 검증

- Tools:

- Google Chrome Developer Tools: 다양한 화면 비율 시뮬레이션
- Selenium Grid: 여러 디바이스와 해상도를 동시에 테스트

- Method:

- Chrome Developer Tools를 사용하여 다양한 화면 비율(16:9, 4:3, 3:2)을 테스트
- Selenium Grid로 여러 디바이스 해상도에서 반응형 UI 테스트 스크립트를 실행
- Validation Points:
 - 모든 화면 비율에서 UI가 깨지지 않고 정확히 표시
 - 레이아웃이 각 해상도에 맞게 조정

8.2.1.4. Security Testing

WebShooter 시스템의 보안성을 평가하며, 사용자 데이터 보호와 데이터 접근 권한 능력을 검증한다.

사용자 인증 보안:

1. HTTPS 프로토콜 검증

- Tools: Browser Developer Tools, OpenSSL.
- Method:
 - 개발자 도구에서 HTTPS 활성화 여부 확인.
 - OpenSSL을 사용해 인증서의 유효성 및 암호화 알고리즘 확인.
- Verification Points:
 - 모든 데이터 전송이 HTTPS를 통해 암호화.
 - SSL/TLS 인증서가 유효하며 최신 암호화 프로토콜(예: TLS 1.2/1.3)을 사용.

2. 비밀번호 정책 검증

- Tools: Selenium (UI 자동화 테스트)
- Method:
 - Selenium 스크립트를 통해 다양한 비밀번호(짧은 비밀번호, 단순 문자) 입력 테스트
 - 규칙에 부합하지 않는 비밀번호에 대해 적절한 오류 메시지 확인
- Verification Points:
 - 비밀번호 최소 길이(예: 8자 이상) 및 문자 조합(대소문자, 숫자, 특수문자) 요구
 - 오류 메시지가 명확하고 사용자에게 가이드 제공

3. 비밀번호 재설정 검증

- Tools: Postman (API 테스트), Selenium
- Method:
 - Postman을 통해 비밀번호 재설정 요청 후, 이메일에 포함된 링크의 만료 시간 및 보안 토큰 검증
 - Selenium을 사용해 잘못된 토큰으로 접근 시 적절한 오류 처리 확인
- Verification Points:
 - 이메일 링크가 설정된 만료 시간 내에서만 작동
 - 보안 토큰이 올바르게 생성되고, 누락 또는 변조 시 접근 차단

4. 세션 만료 및 차단

- Tools: Selenium, Browser Developer Tools
- Method:
 - 사용자가 일정 시간(예: 30분) 동안 활동하지 않았을 때 세션 만료 여부 확인
 - 만료된 세션으로 다시 요청을 보낼 경우 서버에서 적절히 차단되는지 확인
- Verification Points:
 - 세션 만료 후 새로고침 시 자동 로그아웃
 - 비정상적인 세션 재활용 시 오류 메시지 제공

데이터 접근 권한

1. 접근 권한 구분

- Tools: Selenium, Postman
- Method:
 - 사용자, 관리자 계정으로 로그인하여 각 역할에 맞는 권한만 허용되는지 확인
 - Postman으로 API 호출을 통해 권한이 없는 요청 차단 확인
- Verification Points:
 - 일반 사용자가 관리 기능에 접근할 수 없음
 - 관리자만 민감한 데이터(사용자 정보 등)에 접근 가능

2. 관리자 페이지 보안

- Tools: OWASP ZAP, Log Monitoring Tools (Splunk)
- Method:
 - 비정상적인 접근 시도를 시뮬레이션하여 로그 기록 여부 확인
 - Splunk 또는 로그 모니터링 도구를 통해 경고 메시지 생성 여부 점검
- Verification Points:
 - 모든 관리자 페이지 접근이 로그로 기록
 - 비정상적인 접근 시 관리자에게 알림 발송

8.2.1.5. Failover and Recovery Testing

WebShooter 시스템이 장애 상황에서 데이터 무결성을 유지하고, 서비스 중단 시간을 최소화하며, 사용자 경험에 미치는 영향을 줄이는 복구 능력을 검증한다.

서버 장애 복구:

1. AWS 자동 스케일링 복구 테스트

- Tools: AWS CloudWatch, AWS Elastic Load Balancer (ELB)
- Method:
 - AWS CloudWatch에서 특정 서버 인스턴스를 종료하여 장애 상황을 시뮬레이션
 - Elastic Load Balancer와 Auto Scaling 설정을 통해 새로운 서버 인스턴스가 자동으로 추가되는지 확인
- Verification Points:
 - 장애 발생 후 새로운 서버 인스턴스 생성 및 정상화 ≤ 5분
 - 서비스 중단 시간 동안 모든 요청이 적절히 처리(또는 오류 메시지 제공)

2. 알림 시스템 검증

- Tools: AWS SNS (Simple Notification Service), Email Notification Tools
- Method:
 - 서버 다운타임 시 AWS SNS를 사용해 관리자에게 알림 메시지 발송 테스트
 - 장애 상태 및 복구 진행 상황이 포함된 알림 메시지 내용 확인
- Verification Points:

- 장애 발생 시 관리자 알림 전송
- 알림 메시지에 복구 예상 시간, 상태 정보 포함

3. 비상 복구 절차 검증

- Tools: Manual Testing, Incident Response Documentation
- Method:
 - 문서화된 복구 절차를 따라 데이터베이스 및 서버 복구를 실행
 - 복구 과정에서 발생하는 로그를 기록하고 분석
- Verification Points:
 - 복구 절차가 문서화된 대로 실행 가능
 - 복구 이후 모든 서비스가 정상 작동

데이터 무결성

1. 데이터 손실 방지 검증

- Tools: Database Backup Tools (AWS RDS Snapshots), SQLAlchemy
- Method:
 - 장애 상황에서 사용자 학습 기록 및 데이터베이스 백업 파일 복원
 - 복원된 데이터와 백업 파일의 데이터 비교
- Verification Points:
 - 데이터 손실 없이 완전 복구
 - 장애 전 학습 기록이 데이터베이스에 정확히 저장

2. 대시보드 데이터 반영 검증

- Tools: Selenium, Database Query Tools (pgAdmin)
- Method:
 - 데이터 복구 후 대시보드에 학습 진행 정보가 올바르게 표시되는지 확인
 - 데이터베이스 값과 대시보드 UI 값 일치 여부 검증
- Verification Points:
 - 복구 후 대시보드 데이터가 최신 상태로 정확히 반영

중단 없는 사용자 경험

1. 오류 메시지 및 세션 복원 검증

- Tools: Selenium, Browser Developer Tools
- Method:
 - 서버 다운타임 동안 UI에 적절한 오류 메시지가 표시되는지 확인
 - 서버 복구 후 사용자 세션이 자동 복원되는지 테스트
- Verification Points:
 - 장애 중 명확한 오류 메시지 제공
 - 복구 후 사용자 세션 복원

2. 데이터 유실 방지 검증

- Tools: Selenium, Manual Testing
- Method:
 - 문제 등록, 학습 기록 저장 도중 장애 상황을 시뮬레이션
 - 복구 후 UI와 데이터베이스에서 모든 데이터가 정상적으로 반영되는지 확인
- Verification Points:
 - 저장된 데이터가 유실되지 않고 복구 후에도 그대로 반영
 - 사용자 인터페이스에서 이전에 저장된 데이터를 정확히 로드

8.2.2. User Testing

User Testing은 WebShooter의 사용성을 실제 사용자 환경에서 검증하고, 개선 사항을 도출하기 위해 설계되었다.

8.2.2.1. Beta Testing

- 테스트 그룹:
 - 초보자부터 중급자까지 다양한 웹 개발 경험을 가진 사용자 그룹을 대상으로 진행
 - 테스트 인원은 최소 20명 이상을 대상으로, 사용성 피드백을 수집
- 테스트 시나리오:
 - 회원가입 및 로그인, 문제 생성, 실습 환경에서의 코드 작성 및 실행, 문제 풀이 후 피드백 확인, 대시보드까지의 전체 사용자 플로우를 점검

- 결과 분석:
 - 사용자 피드백과 오류 보고서를 분석하여 UI/UX 및 기능 개선 방안을 도출

8.2.2.2. Error Tolerance Testing

- Objective:
 - 예외적인 상황에서 시스템 안정성을 검증
- Test Scenario:
 1. 서버 연결 오류
 - Tools: Network Simulation Tools (Charles Proxy, Browser Throttling)
 - Method:
 - 네트워크 연결 끊김 상태를 시뮬레이션
 - 작업이 중단되지 않고 데이터가 저장되는지 확인
 - Validation Points:
 - 네트워크 오류 발생 시 사용자 작업이 중단되지 않으며, 데이터가 유실되지 않음

8.2.2.3. Interaction Testing

- Objective:
 - 사용자의 입력(키보드, 마우스)이 시스템에 정확히 반영되는지 확인
- 테스트 항목:
 1. 입력 처리 속도
 - Tools: Selenium, Performance Monitoring Tools
 - Method:
 - 키보드 입력과 마우스 클릭, 드래그, 드롭 동작의 반응 속도 측정
 - 반응 시간(100ms 이내)을 기록
 - Partition Testing:
 - 키보드 입력:
 - 정상 입력: 문자, 숫자, 특수문자
 - 비정상 입력: 빈 필드, 길이 초과
- Validation Points:

- 모든 입력이 100ms 이내에 반영

2. 단축키 충돌

- Tools: Manual Testing, Keyboard Event Logging Tools
- Method:
 - 브라우저 단축키와 WebShooter 시스템 지정 단축키 동작 확인
 - 충돌 발생 시 시스템 단축키의 우선 순위 유지 여부 검증
- Validation Points:
 - 단축키 충돌 없이 시스템 단축키가 정상 작동

3. 드래그 정확도

- Tools: Manual Testing
- Method:
 - 드래그 앤 드롭 동작을 여러 번 실행하여 정확도를 측정
 - 드래그 위치 오차 범위 (5px 이내) 확인
 - Partition Testing:
 - 드래그 위치:
 - 정상 값: 지정된 영역 내.
 - 비정상 값: 화면 경계 바깥
- Validation Points:
 - 드래그 앤 드롭 동작이 정확히 수행

8.3 Manual Test Case Document example

Element	Description
Summary	사용자가 로그인 요청 시, 처리 시간이 5초 이내에 완료되는지 확인
Pre-Conditions	<ol style="list-style-type: none">사용자는 이미 등록된 계정 정보를 보유인터넷 연결 상태가 양호해야 함로그인 페이지에 접근 가능한 브라우저를 사용해야 함
Test Steps	<ol style="list-style-type: none">로그인 페이지로 이동이메일과 비밀번호를 입력"로그인" 버튼을 클릭서버는 사용자가 입력한 정보를 토대로 사용자를 인증한 값을 반환인증 성공 시, 세션 토큰을 생성하고 클라이언트에 반환성공적으로 대시보드로 리다이렉션 되는지 확인
Expected results	<ol style="list-style-type: none">올바른 입력값으로 로그인 요청 시, 5초 이내에 대시보드로 리다이렉션잘못된 입력값일 경우, 적절한 오류 메시지 표시

[Figure 55] testcase1 : 로그인 처리 시간 확인

Element	Description
Summary	비밀번호 재설정 및 이메일 인증 동작 확인
Pre-Conditions	1. 사용자는 이미 등록된 계정 정보를 보유 2. 인터넷 연결 상태가 양호해야 함 3. 로그인 페이지에 접근 가능한 브라우저를 사용해야 함
Test Steps	1. 로그인 페이지로 이동 2. "비밀번호 재설정" 클릭 후, 재설정 페이지로 이동 3. 이메일 주소 입력 후 "재설정 요청" 버튼 클릭 4. 서버는 사용자가 입력한 정보를 토대로 사용자를 인증한 값을 반환 5. 인증 성공 시, 인증 메일 전송 6. 이메일로 전송된 링크를 클릭하여 새 비밀번호를 설정
Expected results	1. 올바른 이메일로 재설정 요청시, 재설정 링크 도착 2. 잘못된 입력값일 경우, 적절한 오류 메시지 표시

[Figure 56] testcase2 : 비밀번호 재설정 및 이메일 인증 동작 확인

Element	Description
Summary	사용자가 지정한 주제와 난이도로 문제를 생성할 때, 생성 시간이 2분 이내로 완료되는지 확인
Pre-Conditions	1. 사용자가 로그인되어 있어야 함 2. 인터넷 연결 상태가 양호해야 함 3. 문제 생성 페이지에 접근 가능한 브라우저를 사용해야 함 4. 주제와 난이도에 대한 프롬프트 명령어들이 데이터베이스에 등록되어 있어야 함 5. GPT API와 데이터베이스가 연결되어 있어야 함
Test Steps	1. "문제 생성" 페이지로 이동 2. 주제와 난이도를 선택하고 "생성" 버튼을 클릭 3. 서버는 선택된 주제와 난이도에 대한 프롬프트 명령어를 데이터베이스에 접근 4. 프롬프트 명령어를 GPT API에 전달하여 문제 생성 요청을 수행 5. GPT API는 생성된 문제 데이터를 서버에 반환 6. 반환된 문제를 검증 후, 데이터베이스에 저장 7. 클라이언트는 데이터베이스에서 문제 데이터를 가져와 화면에 표시
Expected results	1. 문제 생성 요청 후, 지정된 주제와 난이도에 맞는 문제가 2분 이내에 화면에 표시

[Figure 57] testcase3: 문제 생성 시간 확인

Element	Description
Summary	사용자가 입력한 질문에 대해 AI 챗봇이 2분 이내에 적절한 응답을 반환하는지 확인
Pre-Conditions	1. 사용자가 로그인되어 있어야 함 2. 인터넷 연결 상태가 양호해야 함 3. 로그인 페이지에 접근 가능한 브라우저를 사용해야 함 4. GPT API와 데이터베이스가 연결되어 있어야 함 5. 문제+질문에 대한 프롬프트가 데이터베이스에 존재
Test Steps	1. 질문을 입력하고 "전송" 버튼을 클릭 2. 서버는 질문 데이터와 문제 데이터로 프롬프트를 작성 3. 프롬프트를 GPT API로 전달 4. 질문에 대한 응답 데이터를 서버에 반환 5. 반환된 데이터를 클라이언트에 전달 6. 클라이언트는 응답을 사용자 인터페이스에 표시
Expected results	1. 질문 전송 후 2분 이내에 응답이 반환 2. 질문과 관련된 내용이 정확하고 이해 가능하게 반환됩니다.

[Figure 58] testcase4 : AI 챗봇 응답 시간 및 품질 확인

Element	Description
Summary	사용자가 제출한 코드에 대해 유닛 테스트 및 스크린샷 유사도 기준으로 피드백을 제공하는 과정을 검증. 채점 결과와 피드백은 5초 이내에 반환되어야 함
Pre-Conditions	<ul style="list-style-type: none"> 1. 사용자가 로그인되어 있어야 함 2. 인터넷 연결 상태가 양호해야 함 3. 코드 제출 페이지에 접근 가능한 브라우저를 사용해야 함 4. 테스트 기준이 데이터베이스에 사전 정의되어 있어야 함
Test Steps	<ul style="list-style-type: none"> 1. 사용자가 문제 풀이 후 "코드 제출" 버튼을 클릭 2. 서버는 제출된 코드를 수신하고, 채점 서버에 요청을 보냄 3-1. 채점 서버는 코드를 실행하여 유닛 테스트를 수행하고, 결과를 서버로 반환 3-2. 채점 서버는 사용자 코드의 실행 결과와 스크린샷을 비교하여 UI 유사도를 측정하고 결과를 반환 4. 서버는 유닛 테스트 및 UI 유사도 결과를 기반으로 GPT API에 피드백 생성을 요청 5. GPT API는 입력된 데이터를 바탕으로 피드백 메시지를 생성하고, 서버로 반환합니다. 6. 클라이언트는 서버로부터 받은 피드백 메시지와 채점 결과를 화면에 표시합니다.
Expected results	<ul style="list-style-type: none"> 1. 사용자가 코드 제출 후 5초 이내에 채점 결과와 GPT 기반 피드백이 반환됩니다. 2. GPT 피드백은 유닛 테스트 실패 이유와 UI 불일치 원인을 명확히 설명하며, 구체적인 개선 방법을 제공합니다. 3. GPT 피드백이 테스트 기준과 데이터베이스의 기대값과 일치하는지 검증

[Figure 59] testcase5: 채점 결과 및 피드백 제공

Element	Description
Summary	사용자가 문제를 해결하거나 학습 데이터를 저장한 후, 대시보드가 5초 이내에 업데이트되고, 정확한 정보를 시각화하는지 확인
Pre-Conditions	<ul style="list-style-type: none"> 1. 사용자가 로그인되어 있어야 함 2. 인터넷 연결 상태가 양호해야 함 3. 대시보드 페이지에 접근 가능한 브라우저를 사용해야 함 4. 학습 기록 데이터가 데이터베이스에 저장 가능한 상태여야 함
Test Steps	<ul style="list-style-type: none"> 1. 사용자가 문제를 해결하거나 데이터를 제출합니다. 2. 서버는 제출된 데이터를 수신하고 데이터베이스에 저장 3. 저장된 데이터를 분석하고, 대시보드 시각화를 위한 통계 데이터를 생성 4. 생성된 통계 데이터를 클라이언트로 전송합니다. 5. 클라이언트는 대시보드에 새로 반영된 데이터를 시각화하여 표시합니다.
Expected results	<ul style="list-style-type: none"> 1. 문제를 해결하거나 데이터를 제출한 후, 5초 이내에 대시보드가 업데이트 됨 2. 대시보드 그래프 및 통계 정보는 데이터베이스의 기록과 일치함

[Figure 60] testcase6 : 대시보드 업데이트 및 시각화 검증

9. Development Plan

9.1 Objective

Development Testing은 WebShooter의 기능과 성능을 검증하기 위해 개발 과정에서 수행된다. 이 과정은 시스템의 모든 주요 구성 요소가 올바르게 작동하는지 확인하고, 안정성과 통합성을 보장하며, 버그를 조기에 발견하여 시스템 품질을 유지하는 것을 목표로 한다.

9.2 Frontend Environment

9.2.1 React.js

React 컴포넌트를 테스트하여 버튼 클릭, 입력 필드 상호작용, 드롭다운 선택 등 사용자의 이벤트에 대해 예상대로 작동하는지 확인한다. 이벤트 발생 시 UI가 올바르게 업데이트되고, 오류 없이 작동하는지 점검한다.

- 테스트 항목:** React 컴포넌트는 버튼 클릭, 입력 필드 상호작용, 드롭다운 및 모달 동작을 테스트한다. 각 항목에서 컴포넌트가 사용자 이벤트에 반응하고, 상태가 올바르게 업데이트되며, 예상된 화면 변화를 제공해야 한다.



[Figure 61] React.js

9.2.2 Next.js

Next.js의 라우팅 및 데이터 로딩 기능을 테스트하여 동적 라우팅, URL 변경, Query Parameter 처리 등이 예상대로 작동하는지 확인한다. 페이지 전환과 데이터 로딩 과정에서 문제가 없는지 점검한다.

- 테스트 항목:** Next.js의 동적 URL 처리, Custom Routing, Query Parameter 전달 및 데이터 로딩 과정을 테스트한다. 동적 라우팅이 URL 변경에 따라 올바른 페이지를 로드하고, 쿼리 파라미터가 올바르게 전달되며, 모든 데이터가 적절히 렌더링되는지 확인한다.



[Figure 62] Next.js

9.2.3 TypeScript

TypeScript를 사용하여 코드에서 정의된 데이터 타입이 일관되게 적용되고, 컴파일 단계에서 타입 오류가 감지되도록 테스트한다.

- **테스트 항목:** TypeScript의 타입 정의, 유효하지 않은 데이터 입력 시 경고 발생 여부, 함수의 입력 및 출력 타입 일치를 검증한다. 각 항목에서 타입 안정성이 유지되고, 예상치 못한 데이터가 사전에 차단되는지 점검한다.



[Figure 63] TypeScript

9.2.4 상태 관리 테스트

애플리케이션의 전역 상태와 개별 컴포넌트 상태를 점검하여, 상태 변경이 예상대로 반영되고 UI가 정확히 업데이트되는지 확인한다. 또한, 비동기 작업 중 상태 관리가 올바르게 이루어지는지 점검한다.

- **테스트 항목:** 상태 변경 이벤트가 발생했을 때 UI 업데이트가 정상적으로 이루어지는지 확인하며, 비동기 상태 변경 작업에서 데이터 일관성이 유지되는지 검증한다. 예를 들어, 사용자 입력에 따라 상태가 변경될 때 전역 상태와 컴포넌트 상태가 충돌 없이 동작해야 하며, 비동기 요청 중 데이터가 올바르게 업데이트되어야 한다.

9.3 Backend Environment

9.3.1 SpringBoot

SpringBoot로 개발된 RESTful API가 클라이언트 요청에 올바르게 응답하며, 정상/비정상 요청에 대해 적절한 처리가 이루어지는지 확인한다.

- **테스트 항목:** API의 GET, POST, PUT, DELETE 요청을 점검하며, 각 요청에 대해 예상된 HTTP 상태 코드(200, 400, 404, 500 등)가 반환되는지 확인한다. 잘못된 입력 값에 대해 오류 메시지가 정확히 전달되며, 비정상 요청 시 시스템이 안정성을 유지하는지도 검증한다.



[Figure 64] SpringBoot

9.3.2 MySQL

MySQL 데이터베이스와의 연결을 점검하고, 데이터 저장/수정/삭제 작업이 오류 없이 이루어지는지 확인한다.

- **테스트 항목:** 데이터베이스에 새로운 데이터를 저장하거나 기존 데이터를 수정 및 삭제할 때, 데이터 무결성이 유지되는지 점검한다. 또한, 복잡한 쿼리 실행 시 성능이 저하되지 않고, 적절한 결과를 반환하는지 확인한다.



[Figure 65] MySQL

9.3.3 GitHub

Github를 활용한 코드 변경 사항의 관리와 CI/CD 파이프라인 자동화 과정을 점검한다.

- **테스트 항목:** 코드 병합 시 충돌이 효과적으로 관리되는지 확인하며, CI/CD 파이프라인에서 빌드 및 배포 과정이 자동으로 수행되는지 검증한다. 자동화 실패 시 알림과 롤백 절차가 정상적으로 작동하는지도 확인한다.



[Figure 66] Github

9.4 Integration Testing

9.4.1 프론트엔드와 백엔드 통합 테스트

프론트엔드와 백엔드가 원활히 통합되어 작동하는지 확인한다. 특히, 데이터 흐름과 오류 처리를 중점적으로 점검하여 시스템 안정성을 확보한다.

- React에서 SpringBoot 서버로의 API 요청 및 응답 확인: React 클라이언트에서 서버로의 요청이 정확히 전달되고, 서버가 적절한 데이터를 반환하는지 검증한다. 사용자가 문제 목록을 요청했을 때, SpringBoot 서버가 MySQL 데이터베이스에서 데이터를 검색하고 이를 JSON 형식으로 반환하여 클라이언트에서 올바르게 표시되도록 한다.
- MySQL 데이터베이스와 React UI 간의 데이터 연결 확인: 데이터베이스에 저장된 학습 기록, 문제 정보 등이 React UI에서 정확히 로드되고 업데이트되는지 확인한다. 데이터 저장 및 삭제 요청에 대해 데이터베이스가 올바르게 반응하며, 변경 사항이 UI에 즉시 반영되는지도 점검한다.
- 서버 오류(500 Internal Server Error) 처리: 서버에서 예기치 않은 오류가 발생했을 경우, 클라이언트가 오류 메시지를 사용자에게 적절히 표시하는지 확인한다. "현재 서버에 문제가 발생했습니다. 잠시 후 다시 시도해 주세요"와 같은 메시지가 나타나야 한다. 오류 발생 시 로깅 및 알림이 백엔드에서 자동으로 이루어지는지도 검증한다.

9.5 Automated Testing

9.5.1 테스트 도구 활용

자동화 테스트는 개발 속도를 높이고, 코드 변경으로 인해 발생할 수 있는 잠재적 오류를 사전에 발견하기 위해 수행된다.

- **프론트엔드 유닛 테스트:** Jest와 React Testing Library를 사용해 개별 컴포넌트의 동작을 테스트한다. 버튼 클릭 시 이벤트 핸들러가 올바르게 호출되는지, 입력 필드가 사용자 입력에 따라 적절히 업데이트되는지 검증한다. 이 과정에서 모의 데이터를 사용해 API 호출을 시뮬레이션하고, 각 컴포넌트의 독립적인 동작을 점검한다.



[Figure 67] Jest

- **백엔드 유닛 테스트:** JUnit을 활용하여 API 및 비즈니스 로직을 테스트한다. 사용자가 데이터를 검색할 때 API가 적절한 HTTP 상태 코드(200, 404 등)를 반환하는지 확인하고, 잘못된 입력값이 전달되었을 때 서버가 예외를 적절히 처리하는지 점검한다.



[Figure 68] JUnit

- **통합 테스트:** Postman과 Swagger를 사용하여 API 요청 및 응답 과정을 자동화한다. 사용자 인증 API가 올바르게 작동하는지 확인하기 위해 Postman 스크립트를 작성하고, 이를 반복적으로 실행하여 인증 토큰 생성 및 검증 과정을 점검한다.



[Figure 69] Postman

10. Supporting Information

10.1 Software design specification

본 디자인 명세서는 IEEE 소프트웨어 공학 표준 권장사항에 맞추어 작성되었다. 다만, 시스템의 설계 사항을 용이하게 파악하게 할 수 있도록 본래의 양식에서 일부 추가되거나 제외된 항목이 있다.

10.2 Document history

Writer	Description
김현진	1. Preliminary, 2. Introduction, 3. System Architecture 작성
권서진	4. System Architecture (Frontend), 5. System Architecture(Backend) 작성
이유진	5. System Architecture(Backend) 작성
박경일	6. Protocol design, 7. Database design 작성
김문수	8. Testing plan, 9. Development plan, 10. Supporting Information 작성