

Design Specification

QURIOUS

소프트웨어공학개론 4 조



제출일	19.05.19	그룹	4조 Qurious
과목	소프트웨어공학개론	담당교수	이은석 교수님
이름	조상연	학번	2013313217
이름	김연재	학번	2014311060
이름	최지원	학번	2015311016
이름	손현	학번	2013310546
이름	정광현	학번	2013313119

목차

1 Preface.....	5
1.1 Objective.....	5
1.2 Readership.....	5
1.3 Document Structure.....	5
A. Preface	5
B. Introduction.....	5
C. System Architecture	5
D. User Management System	5
E. Problem System.....	5
F. Recommendation System	5
G. Protocol Design	6
H. Database Design	6
I. Testing Plan.....	6
J. Development Environment.....	6
K. Development Plan	6
L. Index	6
M. References.....	6
1.4 Version of the document	7
A. Version Format.....	7
B. Version Management Policy	7
C. Version Update History.....	7
2 Introduction	8
2.1 Objective.....	8
2.2 Applied Diagram.....	8
A. UML.....	8
B. Deployment Diagram.....	9
C. Class Diagram.....	10
D. State Diagram.....	11
E. Sequence Diagram.....	12
F. ER Diagram	13
2.3 Applied Tool.....	14

A.	Visual Studio Code	14
B.	Enterprise Architect	14
C.	AQuery Tool	15
D.	Mermaid Live Editor	16
E.	Draw.io	16
2.4	Project Scope.....	17
A.	Overview	17
B.	User Management System	17
C.	Problem System.....	18
D.	Recommendation System	19
3	System Architecture	20
3.1	Objective.....	20
3.2	System Organization.....	20
A.	User Management System	21
B.	Problem System.....	22
C.	Recommendation System	24
3.3	Deployment Diagram.....	25
4	User Management System.....	25
4.1	Objective.....	25
4.2	Class Diagram.....	26
A.	DB_Handler	26
B.	Account.....	26
4.3	Sequence Diagram	27
A.	회원가입 및 로그인	27
4.4	State Diagram.....	28
A.	회원가입	28
B.	로그인	29
5	Problem System	29
5.1	Objective.....	29
5.2	Class Diagram.....	30

A.	Problem	30
B.	Test.....	31
C.	Judge	31
5.3	Sequence Diagram	33
5.4	State Diagram.....	35
6	<i>Recommendation System</i>.....	36
6.1	Objective.....	36
6.2	Class Diagram.....	37
A.	Recommender	37
B.	Model	37
6.3	Sequence Diagram	38
6.4	State Diagram.....	39
7	<i>Protocol Design</i>.....	40
7.1	Overview.....	40
7.2	JSON	40
7.3	Protocol Description	40
A.	Overview	40
B.	Login Protocol.....	40
C.	Registration Protocol.....	41
D.	Level Test Protocol	41
E.	Show Problem Protocol	42
F.	Submission Protocol.....	42
G.	Get Recommendation Protocol.....	42
H.	Create Problem Protocol	43
8	<i>Database Design</i>	43
8.1	Objective.....	43
8.2	ER Diagram	44
8.3	Relational Schema	44
9	<i>Testing Plan.....</i>	49

9.1	Objective.....	49
9.2	Testing Policy.....	49
A.	Developing testing	49
B.	Release testing	49
C.	User testing	49
9.3	Test Case.....	50
A.	User Management System	50
B.	Problem System.....	53
C.	Recommendation System	57
10	<i>Development Environment.....</i>	59
10.1	Objective	59
10.2	Bootstrap	59
10.3	Vue.js	59
10.4	Django	60
10.5	MySQL	60
10.6	Github.....	61
11	<i>Development Plan</i>	62
11.1	Objective	62
11.2	Gantt Chart.....	62
12	<i>Index.....</i>	63
12.1	Objective	63
12.2	Figure Indexx	63
12.3	Diagram Index	63
12.4	Table Index	64
13	<i>References.....</i>	65
13.1	Objectives	65
13.2	Reference	65

1 Preface

1.1 Objective

Preface에서는 본 문서의 독자를 정의하고, 문서의 전반적인 구조와 각 부분의 내용에 대해 기술한다. 문서 구조를 소개할 때는 각 목차의 목적과 개요를 서술한다.

1.2 Readership

본 문서에서는 독자를 Qurious를 개발 및 유지보수하는 소프트웨어 엔지니어, 시스템 구조를 설계하는 시스템 아키텍처, 고객 기술 지원을 위한 팀 모두 본 문서의 독자로 설정한다.

1.3 Document Structure

A. Preface

Preface에서는 본 문서의 독자를 정의하고, 문서의 전반적인 구조와 각 부분의 내용에 대해 기술한다. 문서 구조를 소개할 때는 각 목차의 목적과 개요를 서술한다.

B. Introduction

Introduction에서는 시스템의 설계에 사용한 UML(Unified Modeling Language)과 데이터베이스 모델의 다양한 다이어그램 작성을 위해 사용한 개발 툴을 소개한다.

C. System Architecture

System Architecture에서는 타겟 시스템에 대한 높은 수준의 개요와 시스템 기능이 전체적인 분포를 서술한다. 전체 시스템의 구조는 Block Diagram을 통해 도식화하였다. 서브 시스템의 관계와 실제 배포 형태는 Block Diagram, Deployment Diagram을 사용하여 표현하였다.

D. User Management System

사용자 관리 시스템에서는 사용자 데이터베이스와 연동되어, 회원가입, 로그인, 사용자의 개인정보 확인, 푼 문제를 관리한다. 회원가입, 로그인, 마이페이지와 같은 3개의 Sub-system으로 나뉜다. 이 절에서는 Class Diagram, Sequence Diagram, State Diagram을 통해 User Management System의 구조와 사용자 및 데이터베이스와의 상호 작용을 설계한다.

E. Problem System

Problem system이 있다. Problem system은 문제를 관리 및 채점하고 진단고사(*level test*)와 관련된 시스템이다. Sub-system으로는 Problem, Judge, Test 3개로 나뉜다. 이러한 Problem system에 대하여 Class diagram, Sequence diagram, 그리고 State diagram을 통해 Problem system의 구조를 표현하고 상세 내용을 기술한다.

F. Recommendation System

Recommendation System은 문제를 추천 받기 위한 서비스로 사용자와 관련된다. Recommendation System은 Recommender와 Model과 같은 2개의 Sub-system으로 나뉜다. 이 절에서는 Class Diagram, Sequence Diagram, State Diagram을 통해 Recommendation System의 구조와 사용자 및 데이터베이스와의 상호 작용을 설계한다.

G. Protocol Design

Protocol Design에서는 sub-system들의 상호작용에서 필수적으로 준수해야 하는 프로토콜에 대하여 설명한다. 통신하는 메시지의 형식과 용도, 의미를 설명한다.

H. Database Design

Database Design에서는 요구사항 명세서에서 기술하였던 요구사항을 기반으로 데이터베이스를 설계하고 이에 대해 기술한다. 요구사항에 기반한 데이터베이스의 ER Diagram을 설명하고, 이에 대한 Relation Schema를 기술한다. Normalization 과정을 통하여 데이터베이스 간에 존재할 수 있는 중복을 방지한다.

I. Testing Plan

Testing Plan에서는 요구사항 명세서에서 기술한, 사용자 및 관리자 시나리오를 바탕으로 한 전체 시스템이 구상한대로 잘 실행되는지 확인한다. 또한, 발생할 수 있는 오류를 사전에 발견하기 위해 진행한다. 관련 테스트를 수행하기 위한 테스팅 정책을 기술하고, 이를 설계한다. 각 testing plan에서는 testing policy와 test case를 설명한다.

J. Development Environment

Development Environment에서는 실제 시스템 개발을 위해 필요한 개발 환경과 코딩 규칙에 대해 기술한다. 개발 과정에서의 버전 관리 도구를 설명한다. 프로그래밍 과정에서 기반이 되는 규칙들에 대해 서술한다.

K. Development Plan

Development Plan에서는 프로젝트 개발 일정을 기술한다. Gantt Chart로 전체적인 개발 순서와 시기를 알아보기 쉽게 표현하고 현재까지의 개발 현황을 설명한다.

L. Index

Index에서는 문서의 인덱스들을 정리한다. 다이어그램 인덱스 및 기능 인덱스가 포함된다.

M. References

문서 작성에 참고한 참고문헌 목록을 기술한다.

1.4 Version of the document

A. Version Format

버전 번호는 major.minor[.maintenance] 형태로 표현하며, 본 문서의 버전은 0.1부터 시작한다.

B. Version Management Policy

본 명세서가 수정될 때마다 버전을 업데이트한다. 다만 변경 간의 간격이 1시간 이내일 때 버전 번호를 추가적으로 업데이트하는 대신, 하나의 업데이트로 통합하여 간주한다. 새로운 부분이 추가되거나 문제의 구성이 과거 문서에 비해 커다란 변화가 있는 경우 major number를 변경한다. 이미 완성되거나 작성된 부분에 대한 변경일 경우, minor number를 변경한다. 이미 작성된 부분에 대하여 오타를 수정하거나 문서의 구조를 단순히 변경하는 경우 maintenance number를 변경한다.

C. Version Update History

버전이 업데이트된 기록을 기술한다.

0.0	2019-05-07	Preface, Introduction, System Architecture 작성
1.0	2019-05-08	Sub-System 초안 작성 및 표지 작성
2.0	2019-05-11	Testing Plan 작성, Protocol Design 작성, Database Design 작성
2.1	2019-05-14	Test Case 수정
3.0	2015-05-15	Development Environment 작성, Sub-system 작성
4.0	2019-05-19	다이어그램 추가 및 Index, Reference 작성

2 Introduction

2.1 Objective

Introduction에서는 시스템의 설계에 사용한 UML(Unified Modeling Language)과 데이터베이스 모델의 다양한 다이어그램 작성을 위해 사용한 개발 툴을 소개한다

2.2 Applied Diagram

A. UML



Figure 1 Logo of UML

UML(Unified Modeling Language, 통합 모델링 언어)는 소프트웨어 공학에서 사용하는 표준화된 범용 모델링 언어이다. 이는 시스템의 시각적인 모델을 표현하며, 객체 지향 소프트웨어 시스템을 개발할 때, 산출물을 가시화, 명세화, 문서화할 때 사용한다.

UML의 관리는 처음 고안한 OMG(Object Management Group)에서 진행하며 소프트웨어 산업 내 시각화의 실질적인 표준 역할을 하고 있다. 개발자의 의사소통을 원활하게 하고 stakeholder들에게 시스템을 공유하여 의견을 얻는 데 도움을 준다. UML을 기반으로 UML Diagram이 존재하며, 현재 13개의 종류의 Diagram이 존재한다.

Qurious에서 사용할 UML Diagram은 Deployment Diagram, Class Diagram, Sequence Diagram, State Diagram 4가지와, 데이터베이스를 표현할 ER Diagram를 활용할 예정이다.

B. Deployment Diagram

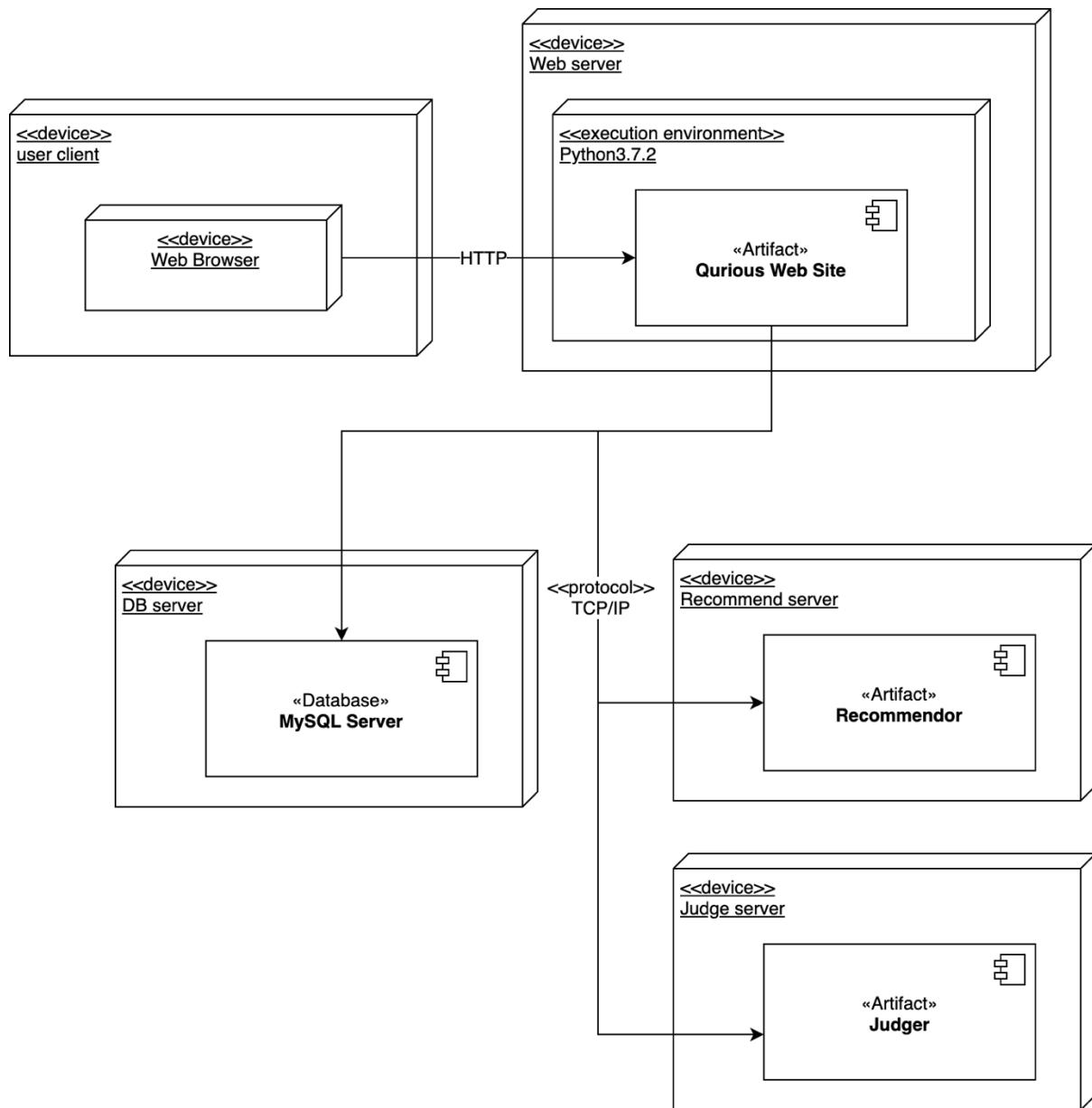


Figure 2 Example of Deployment Diagram

Deployment diagram은 시스템을 구성하는 하드웨어 자원 간의 연결 관계를 표현하고, 하드웨어 자원에 대한 소프트웨어 컴포넌트의 배치 상태를 표현한 것이다. 소프트웨어 시스템이 배치되고 실행될 하드웨어 자원을 정의한다. 소프트웨어 컴포넌트가 어떤 하드웨어 자원에 탑재되어 실행될 지도 정의하며, 하드웨어 자원의 물리적 구성을 정의한다. Deployment diagram의 구성요소로는 노드 혹은 컴포넌트 그리고 관계가 있다.

C. Class Diagram

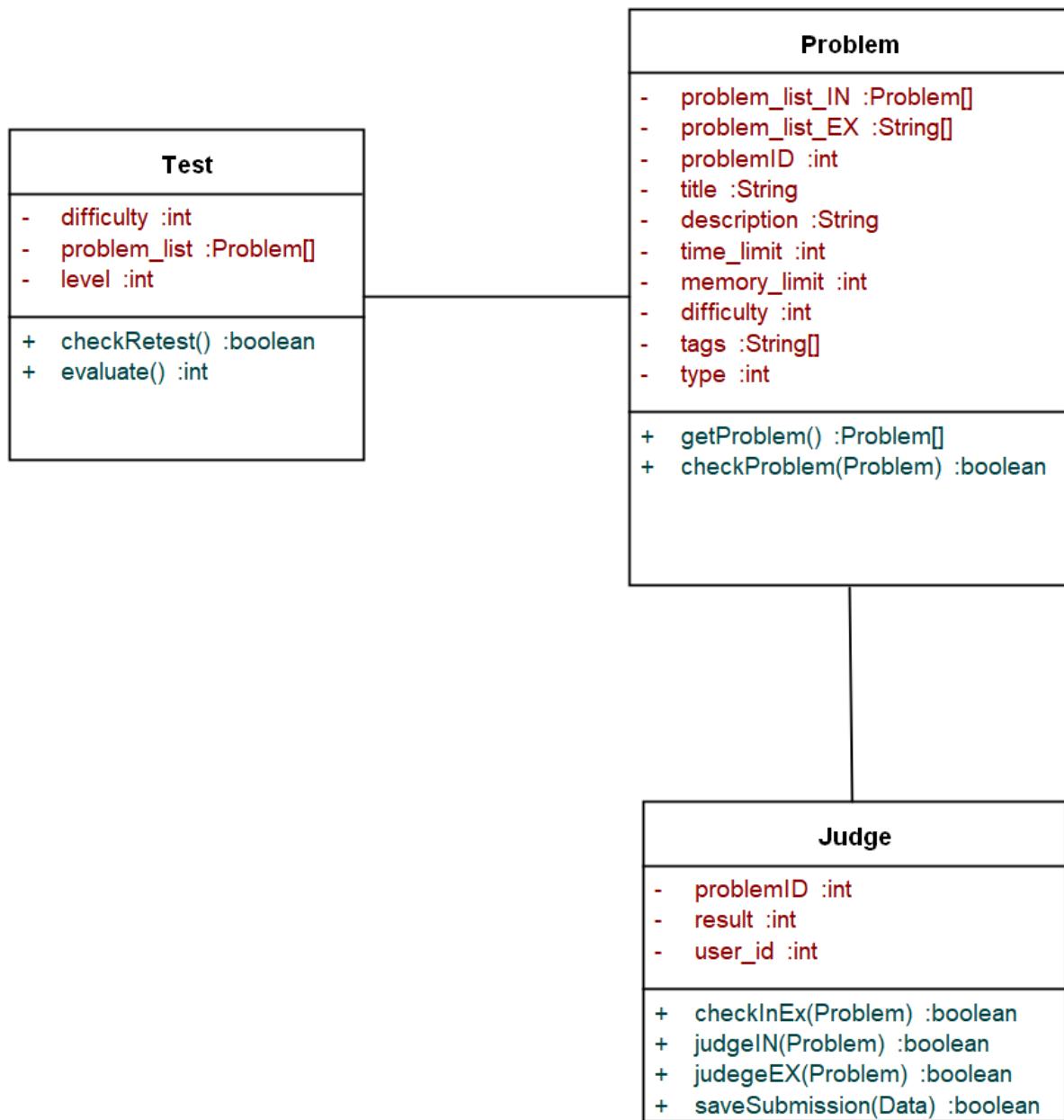


Figure 3 Example of Class Diagram

Class Diagram은 거의 모든 객체 지향 설계에서 사용하는 기본적인 Diagram이다. 시스템의 정적인 구조를 설명한다. 시스템의 논리적 구조 또한 표현한다. 하나의 클래스는 직사각형의 모양이다. 위에서부터 순서대로 클래스 이름, 속성(attributes), 그리고 함수(method, operation)이 포함된다. 각 클래스 간 상속뿐 아니라 다양한 관계를 표현할 수 있으며, Implementation 단계에서 직접적인 도움이 된다.

D. State Diagram

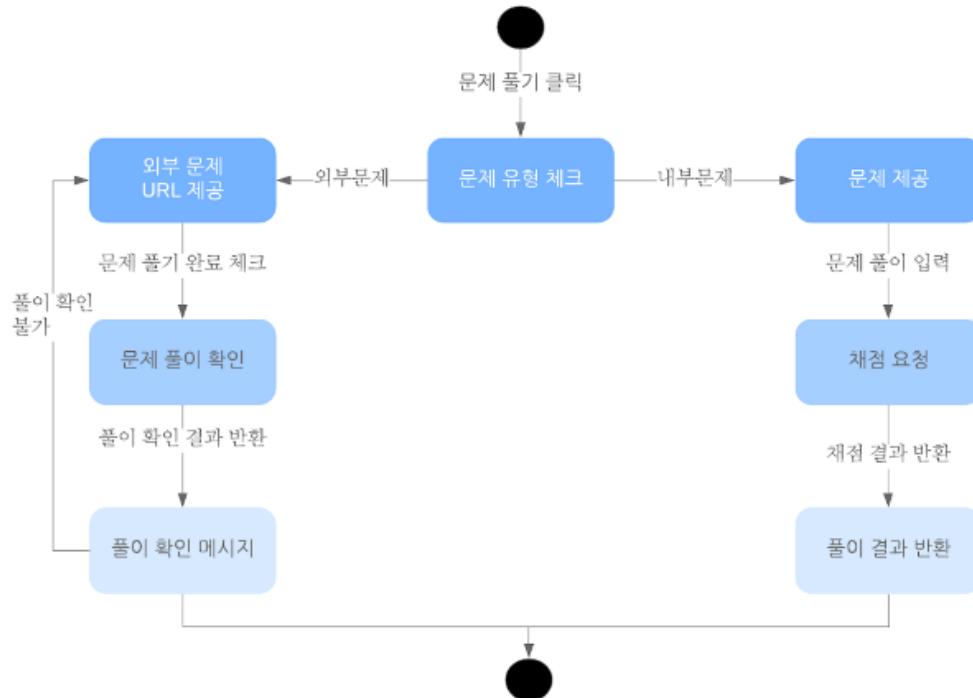


Figure 4 Example of State Diagram

State Diagram은 시스템의 상태를 표현하거나 특정 순간의 시스템의 부분을 기술한다. 이는 System을 Behavioral 관점에서 표현하는 것이다. 외부 자극의 변화나 시간에 따라 각 객체 혹은 클래스의 동적인 행동과 상태를 기술한다.

각 다이어그램은 일반적으로 단일 클래스의 객체를 표현하고 시스템을 통해 해당 객체의 여러 상태를 추적할 수 있다.

E. Sequence Diagram

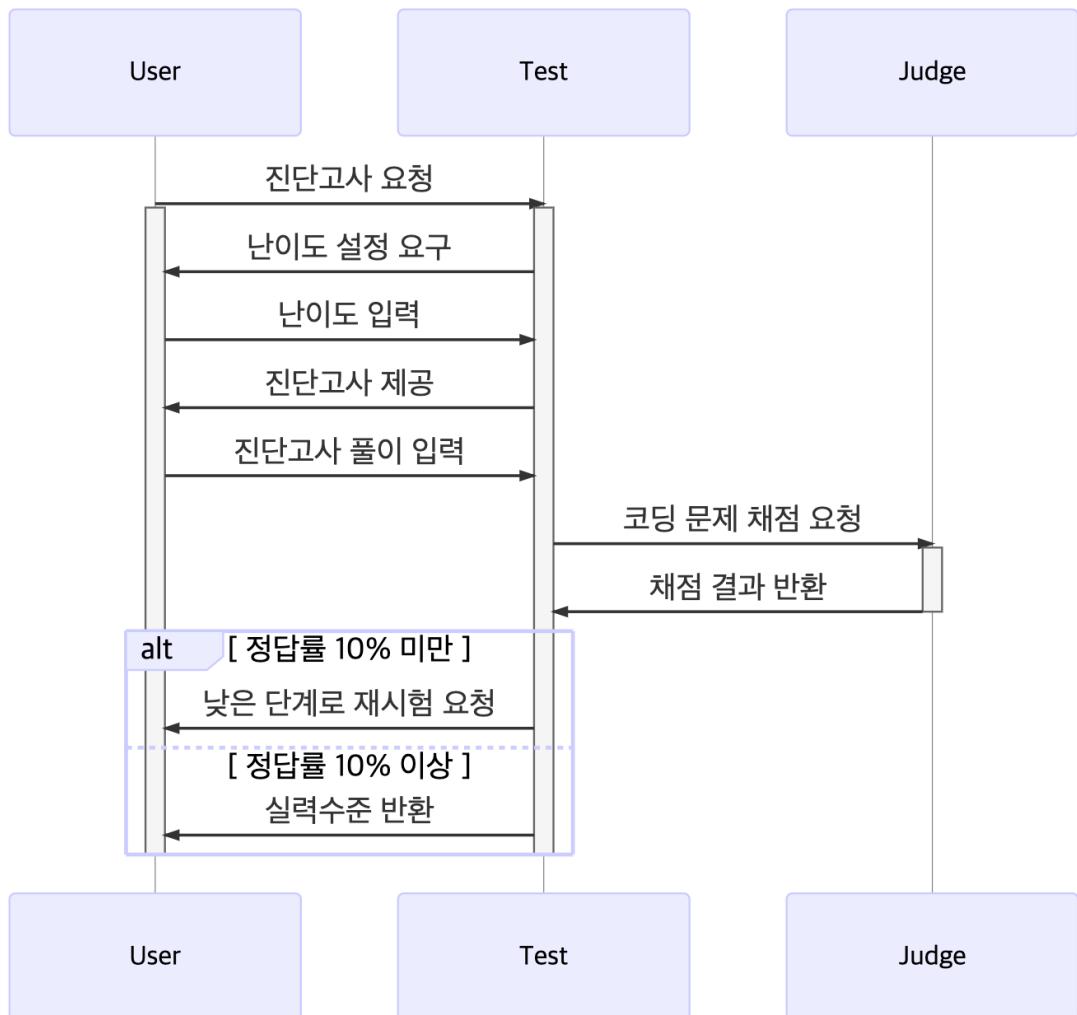


Figure 5 Example of Sequence Diagram

Sequence Diagram은 시간 흐름에 따라, 시스템, 객체, 그리고 클래스 간의 상호작용을 기술한다. 시나리오와 관련된 객체와 클래스를 설명하고 시나리오의 functionality가 진행될 때 객체간 교환하는 메시지의 결과들을 기술한다. Diagram은 개발 과정에서 시스템의 논리적 구조 내에서 use case들과 관련된다.

Sequence Diagram은 lifeline이라고 불리는 평행의 수직 직선 내에, 동시에 진행되는 다양한 프로세스나 객체를 표현한다. 가로로 표현되는 화살표는 각 객체간 교환하는 메시지를 표현한다.

F. ER Diagram

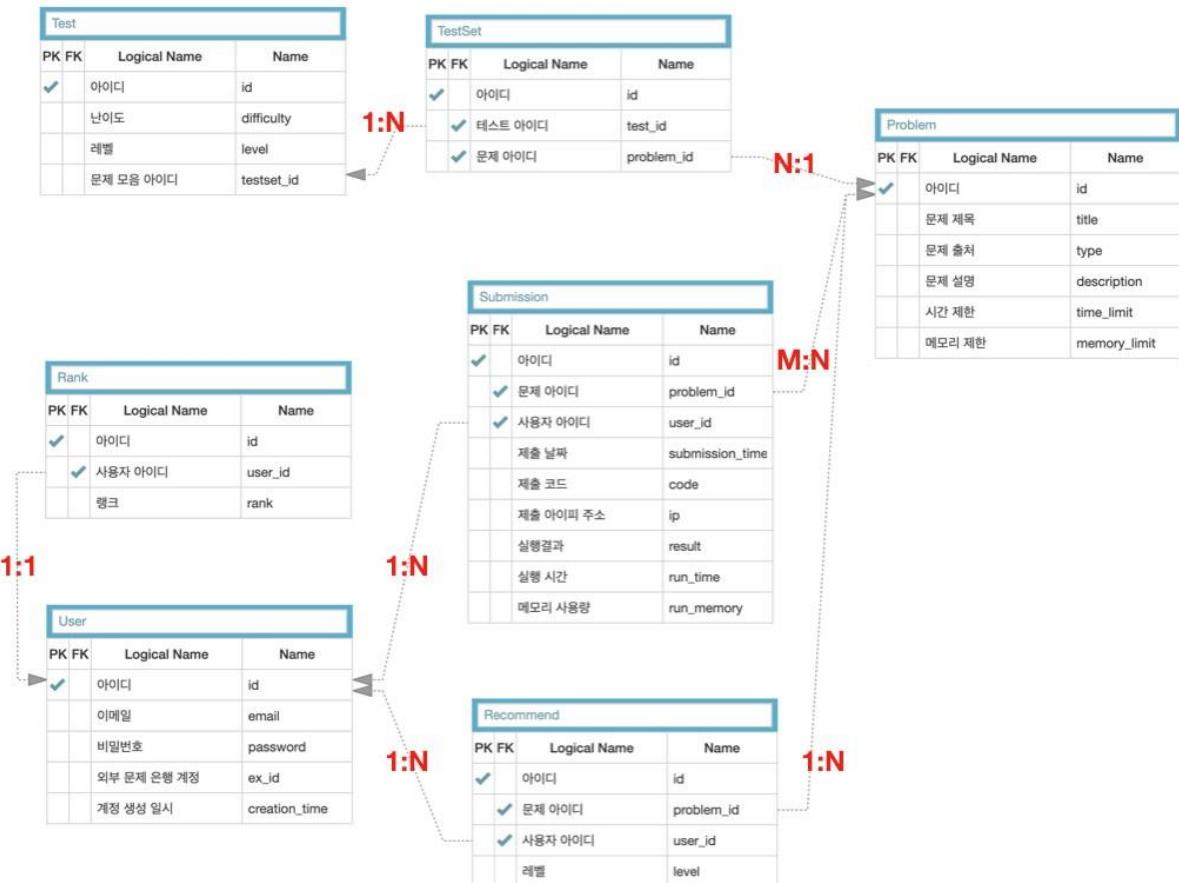


Figure 6 Example of ER Diagram

ER(Entity-Relationship)모델링의 산출물을 가리켜 ER Diagram이라 한다. ER 모델이란 구조화된 데이터를 저장하기 위한 다양한 제약조건과 구조를 표현하는 것이고, 이를 이용한 기법 중 하나가 ER모델링이다.

데이터베이스에 저장되어야 할 정보의 타입과 제약사항을 기술한다. 개체(Entity)는 사각형 표로 표시되고, 개체가 갖는 속성(attribute)는 개체 안에 작성된다. 각 개체간 관계는 실선으로 표현된다. 개체를 고유하게 식별할 수 있는 최소 개수의 속성 집합은 기본 키(Primary Key)라 불린다.

본 문서에서의 ER diagram은 테이블을 표현하고 참조하는 외래키를 화살표로 직접 표시하는 방식으로 기술하였다.

2.3 Applied Tool

A. Visual Studio Code



Figure 7 Logo of Visual Studio Code

Front-end 개발의 경우 Visual Studio Code를 사용하였다. Visual Studio Code는 마이크로소프트가 마이크로소프트 윈도우, 맥OS, 리눅스용으로 개발한 소스 코드 편집기이다.

B. Enterprise Architect



Figure 8 Logo of Enterprise Architect

Class diagram을 그릴 때 사용한 도구는 Enterprise Architect이다. 클래스 간 상속 관계 등 많은 관계를 표현하고 클래스 속성과 함수(method)를 표현하기 용이하기 때문에 사용하였다.

C. AQuery Tool

The screenshot shows the AQuery Tool interface with the following components:

- Top Navigation:** AQUERY TOOL, HELP, DEMO BASE (selected), DEMO DIC, COMMUNITY.
- Demo Base:** Demo Blog(MySQL) - 테이블 선택 (Table Selection).
- ERD View:** ERD icon.
- Toolbars:** SQL / Menu, Sort, Filter, Undo, Redo.
- Table 1: blog_author**

PK	AI	FK Null	Name	Type
✓	✓	+	id	INT
		+	display_name	VARCHAR(45)
		+	first_name	VARCHAR(45)
		+	last_name	VARCHAR(45)
- Table 2: blog_category**

PK	AI	FK Null	Name	Type
✓	✓	+	id	INT
		+	name	VARCHAR(45)
		+	name_clean	VARCHAR(45)
		+	enabled	TINYINT
		+	date_created	TIMESTAMP
- Help Box:** 도움말 (Help)

HELP 메뉴를 클릭해서 도움말을 읽어보시는 것을 추천 드립니다.
- Logic Model Box:** 논리모델 (Logic Model)

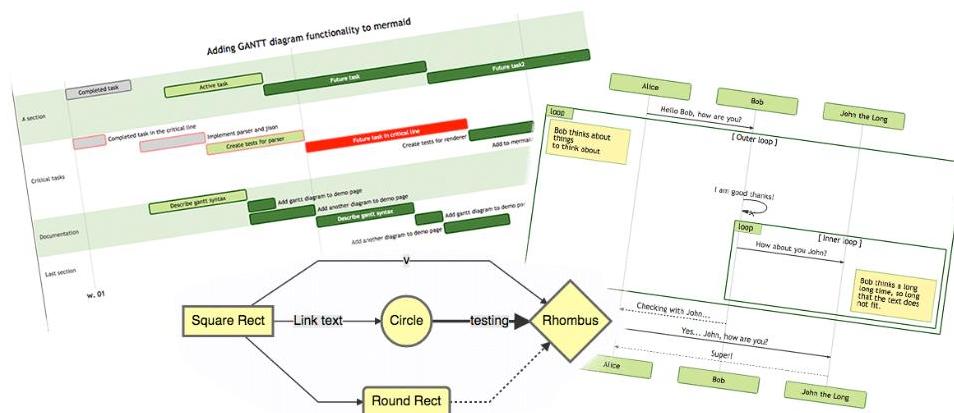
논리모델 형태도 확인하세요.
상단 도구 모음 [ERD] - [논리 모델 형태] 메뉴를 실행하세요
- Relationship:** A dotted line connects the 'id' column of the blog_author table to the 'category_id' column of the blog_category table, indicating a foreign key relationship.

Figure 9 Web page of AQuery Tool

Database를 표현하기 위해 ER diagram을 그릴 때 사용한 도구는 AQuery Tool이다. AQuery Tool은 웹 기반 ER Diagram 도구로 SQL 자동 생성 프로그램도 제공한다.

D. Mermaid Live Editor

mermaid



Generation of diagrams and flowcharts from text in a similar manner as markdown.

Ever wanted to simplify documentation and avoid heavy tools like Visio when explaining your code?

This is why mermaid was born, a simple markdown-like script language for generating charts from text via javascript. [Try it using our editor.](#)

Figure 10 Web page of Mermaid Live Editor

Mermaid Live Editor는 온라인으로 Sequence diagram을 비롯한 다양한 diagram을 작성할 수 있는 도구이다. 간단한 코드를 작성하면 그에 맞는 diagram을 생성하여 주는 것이 장점이다.

E. Draw.io



Figure 11 Logo of Draw.io

Draw.io는 브라우저 기반의 무료 diagramming 도구이다. UML diagram을 비롯하여 많은 diagram을 그릴 수 있는 기본 툴을 제공한다.

2.4 Project Scope

A. Overview

Qurious 시스템은 확산되는 소프트웨어 교육의 흐름에 따라, 소프트웨어 교육을 필요로 하는 학습자와 학부모의 부담을 감소시키고, 학습자의 흥미를 고취시키기 위한 시스템이다. Qurious 시스템은 User Management System, Problem System, Recommendation System, 크게 세 가지로 나눌 수 있다.

B. User Management System

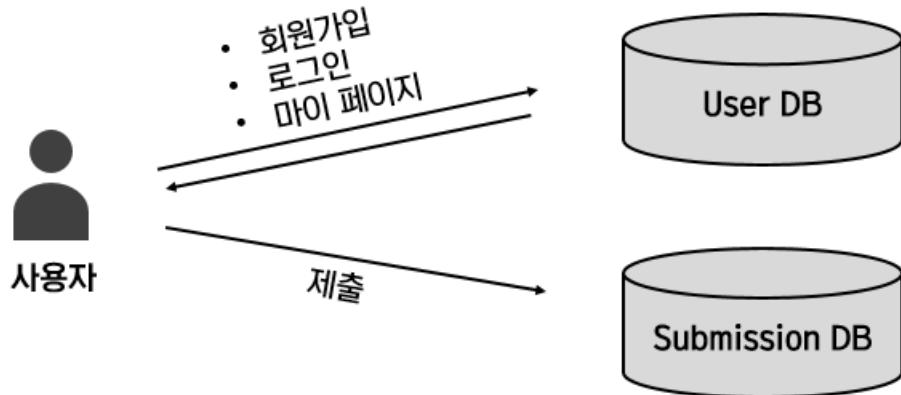


Diagram 1 Product Scope: User Management System

User Management System이다. User Management System은 사용자의 회원 정보를 관리한다. 회원가입, 로그인, 마이페이지와 같은 3개의 Sub-system으로 나뉜다. 사용자는 회원가입, 로그인, 마이페이지 기능을 통해 User DB와 상호작용한다. 또한 문제를 풀고 제출할 때, Submission DB와 상호작용한다.

C. Problem System

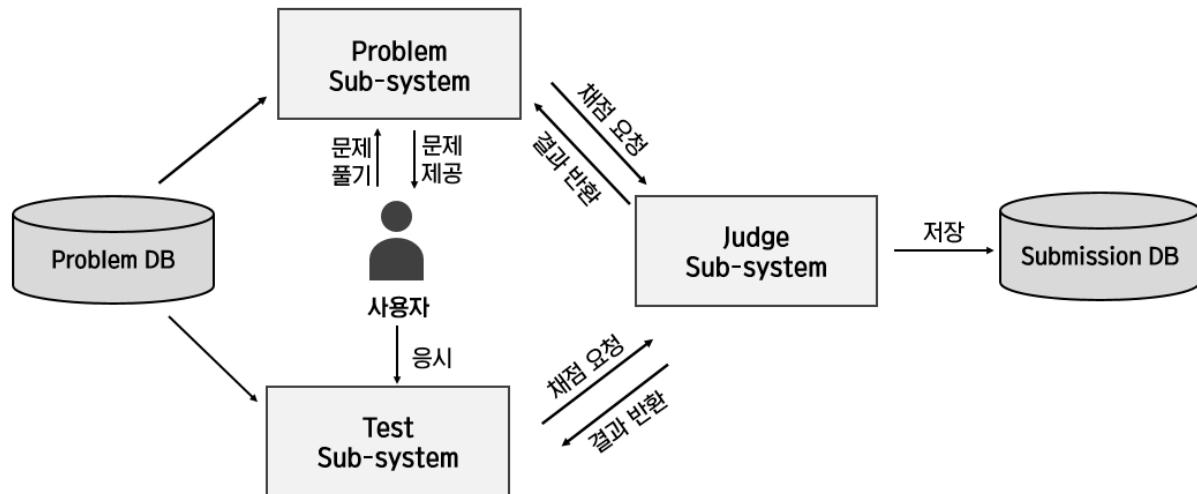


Diagram 2 Product Scope: Problem System

Problem system이 있다. Problem system은 문제를 관리, 채점하고 진단고사와 관련된 시스템이다. Sub-system으로는 Problem, Judge, Test 3개로 나뉜다. Problem DB로부터 문제를 받아, Problem sub-system이 문제를 제공하고, 사용자가 문제를 풀면 사용자의 응답을 Judge sub-system으로 보낸다. Judge sub-system은 사용자의 응답과 채점 결과를 Submission DB에 저장하고, 채점한 결과를 Problem sub-system에 반환한다. 또는 사용자는 진단고사를 보기를 요청할 수 있으며, 이를 위해 Test sub-system은 진단고사를 사용자에게 제공한다. 사용자의 응답은 Judge sub-system으로 전달된다. Judge sub-system은 위와 같이 사용자의 응답과 채점 결과를 Submission DB에 저장하고, 결과를 다시 반환한다.

D. Recommendation System

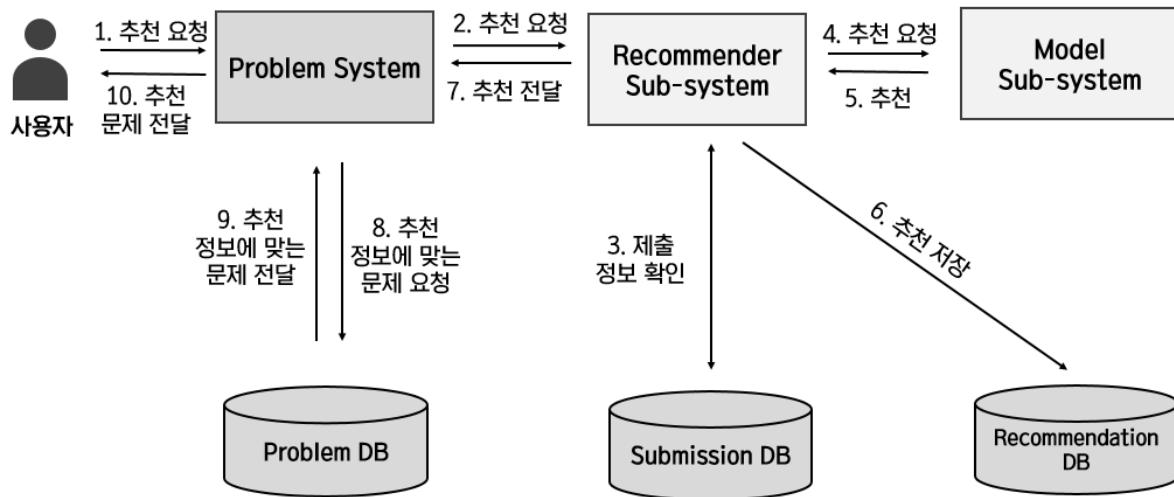


Diagram 3 Product Scope: Recommendation System

Recommendation system이 있다. Recommendation system은 사용자의 실력수준에 맞게 문제를 추천한다. Recommendation와 Model과 같은 2개의 Sub-system으로 나뉜다. Recommendation system은 Problem system과 연관되어 있다. 사용자가 문제 추천 요청을 하면, Problem system이 Recommender sub-system에게 추천 요청을 보낸다. Recommender sub-system은 Submission DB로 가서 사용자의 응답과 결과값을 확인한 후, 이를 Model sub-system에 전달하여 추천을 요청한다. Model sub-system은 문제를 추천하여 이 문제들의 정보를 반환하며, Recommender sub-system은 이를 Recommendation DB에 저장한다. 추천 받은 문제들의 정보를 Problem system에 전달하면, Problem system은 Problem DB에 가서 추천받은 문제들을 찾아서 사용자에게 전달한다.

3 System Architecture

3.1 Objective

System Architecture에서는 타겟 시스템에 대한 높은 수준의 개요와 시스템 기능이 전체적인 분포를 서술한다. 전체 시스템의 구조는 Block Diagram을 통해 도식화하였다. 서브 시스템의 관계와 실제 배포 형태는 Block Diagram, Deployment Diagram을 사용하여 표현하였다.

3.2 System Organization

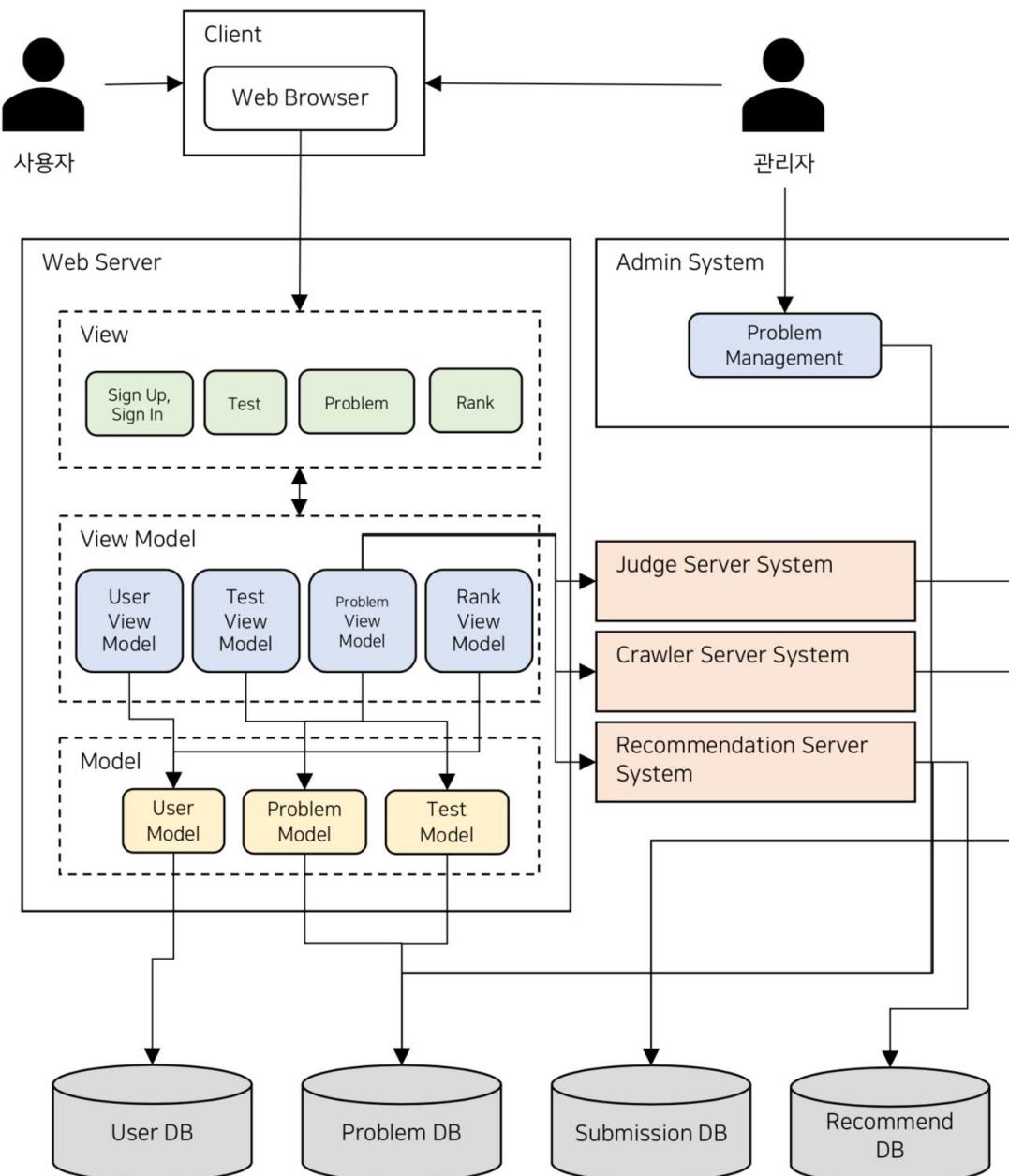


Diagram 4 System Architecture: Qurious 전체

Qurious의 전체 시스템 구조는 Web Application 구조로 Web Browser로 접속하며 크게 Web Server, Judge Server, Crawler Server, Recommendation Server 시스템으로 구성된다.

A. User Management System

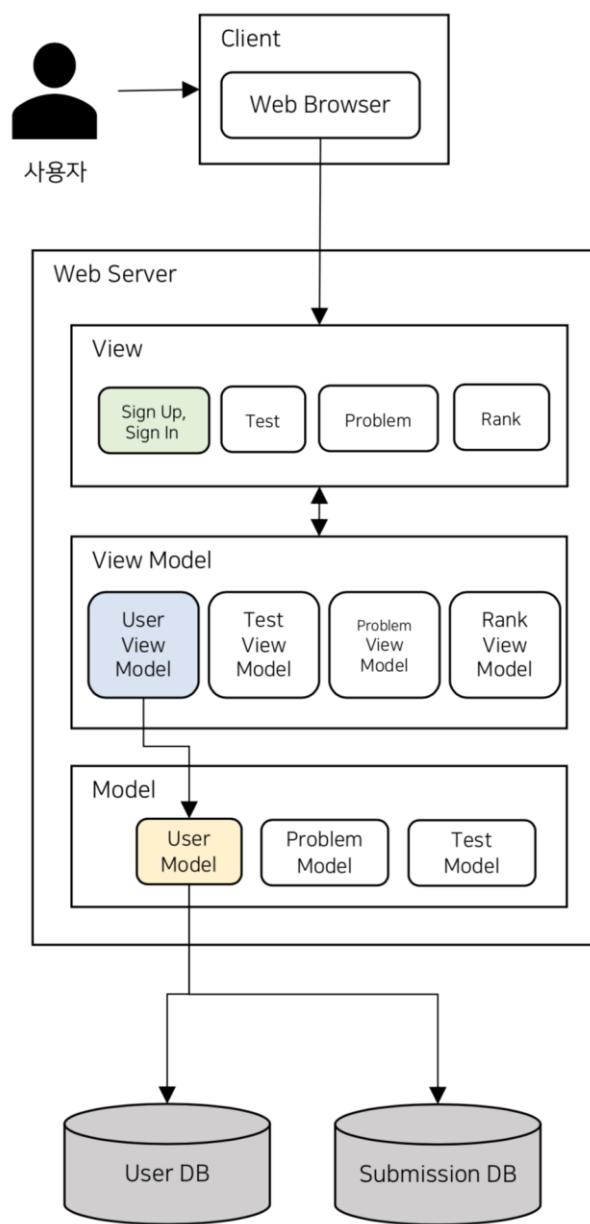


Diagram 5 System Architecture: 사용자 시스템

사용자 시스템은 사용자가 회원가입부터, 로그인, 로그아웃 등 전반적인 사용자 관련 요구사항을 담당하는 시스템이다.

B. Problem System

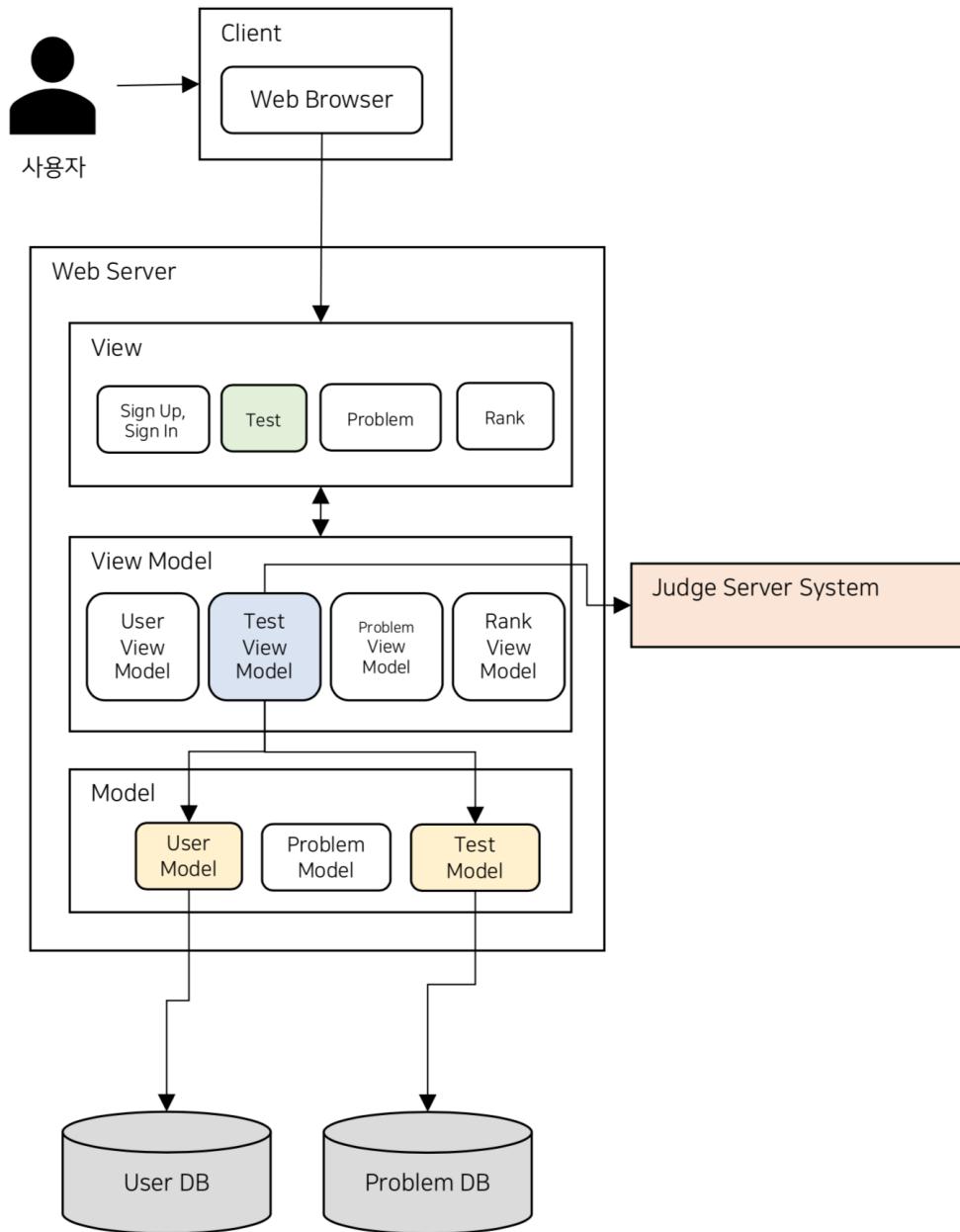


Diagram 6 System Architecture: 진단고사 시스템

진단고사 시스템은 Test View를 통해 표시되며 여기서 Test Viewmodel이 보여질 문제를 선정하여 전달한다. 최종 제출 이후 체점은 Judge Server System을 통해 이루어지고 결과를 반환받아 계산하여 최종 실력 점수를 UserDB에 저장한다.

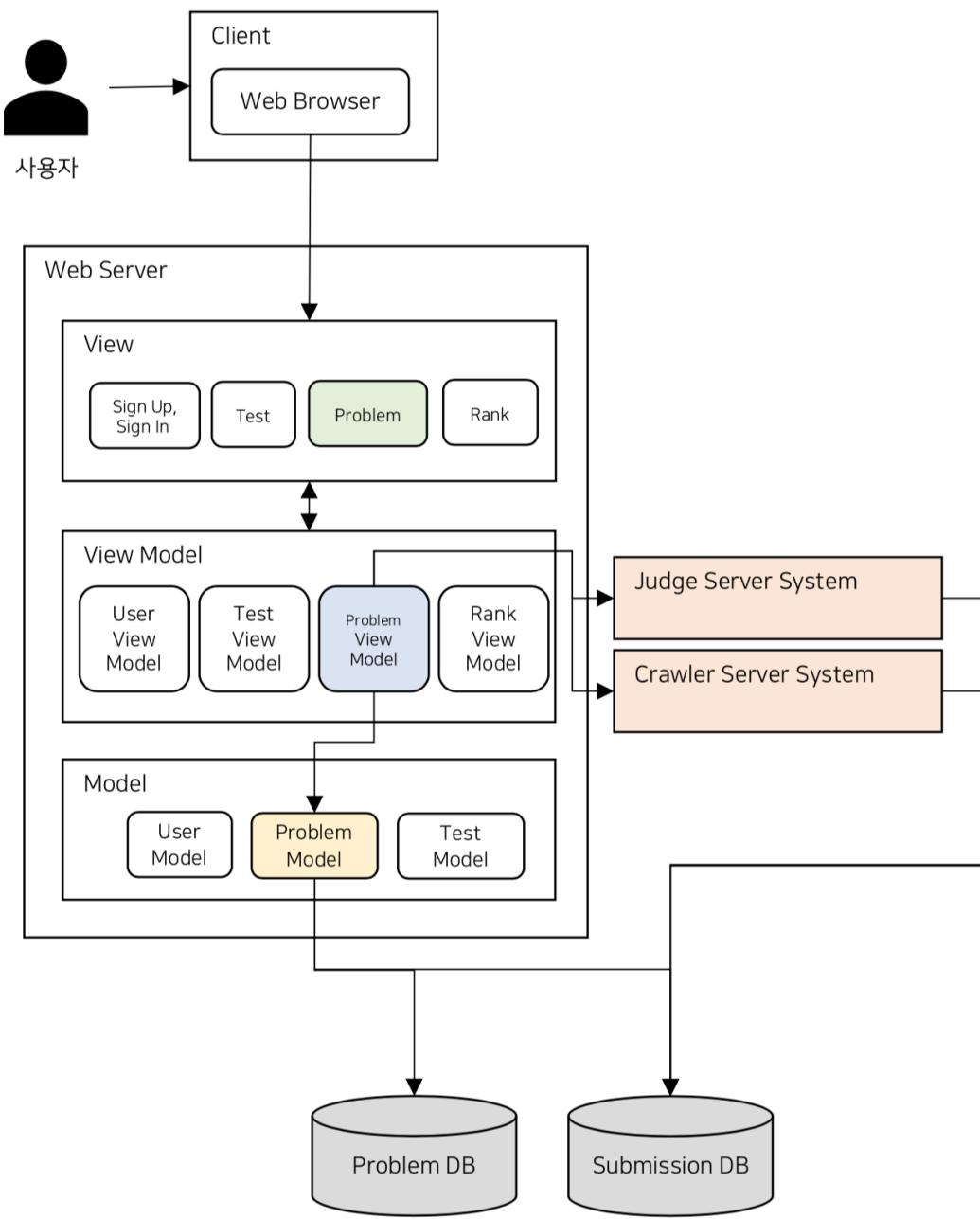


Diagram 7 System Architecture: 문제 풀이 시스템

문제 풀이 시스템은 Problem View를 통해 표시되며 Problem View Model을 통해 문제 풀이 기능을 구현한다. 내부 문제의 경우 Judge Server System을 통해, 외부 문제의 경우 Crawler Server System을 통해 문제를 채점하고 이를 Submission DB에 저장한다.

C. Recommendation System

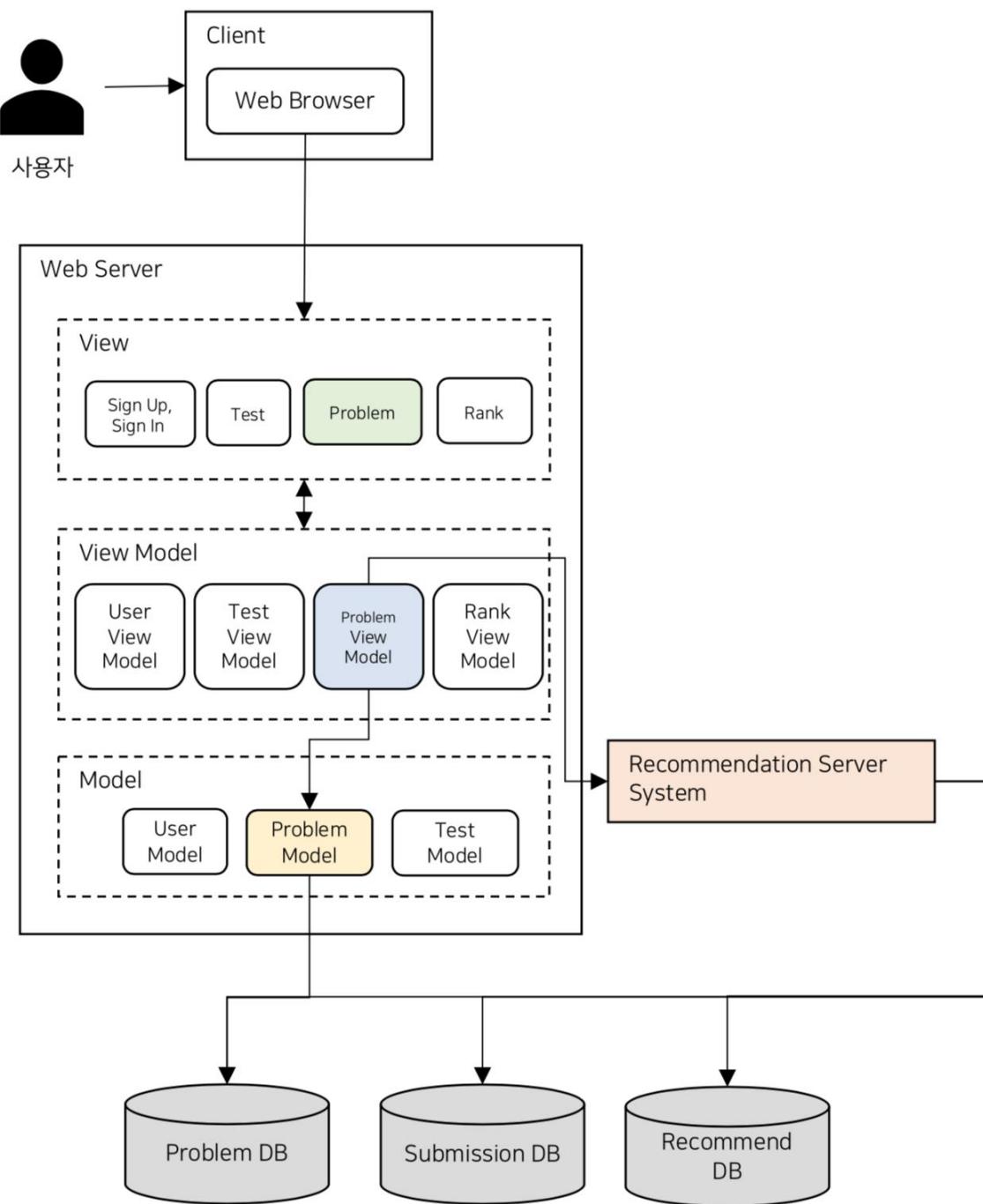


Diagram 8 System Architecture: 문제 추천 시스템

문제 추천 시스템은 Problem View 와 ViewModel을 통해 이루어지며 추천하기 버튼 클릭시 Recommendation Server System으로 요청을 보낸다. 시스템 내 협업 필터링 모델에서 사용자의 Submission데이터를 입력받아 문제를 추천하고 이를 문제 ID로 반환한다.

3.3 Deployment Diagram

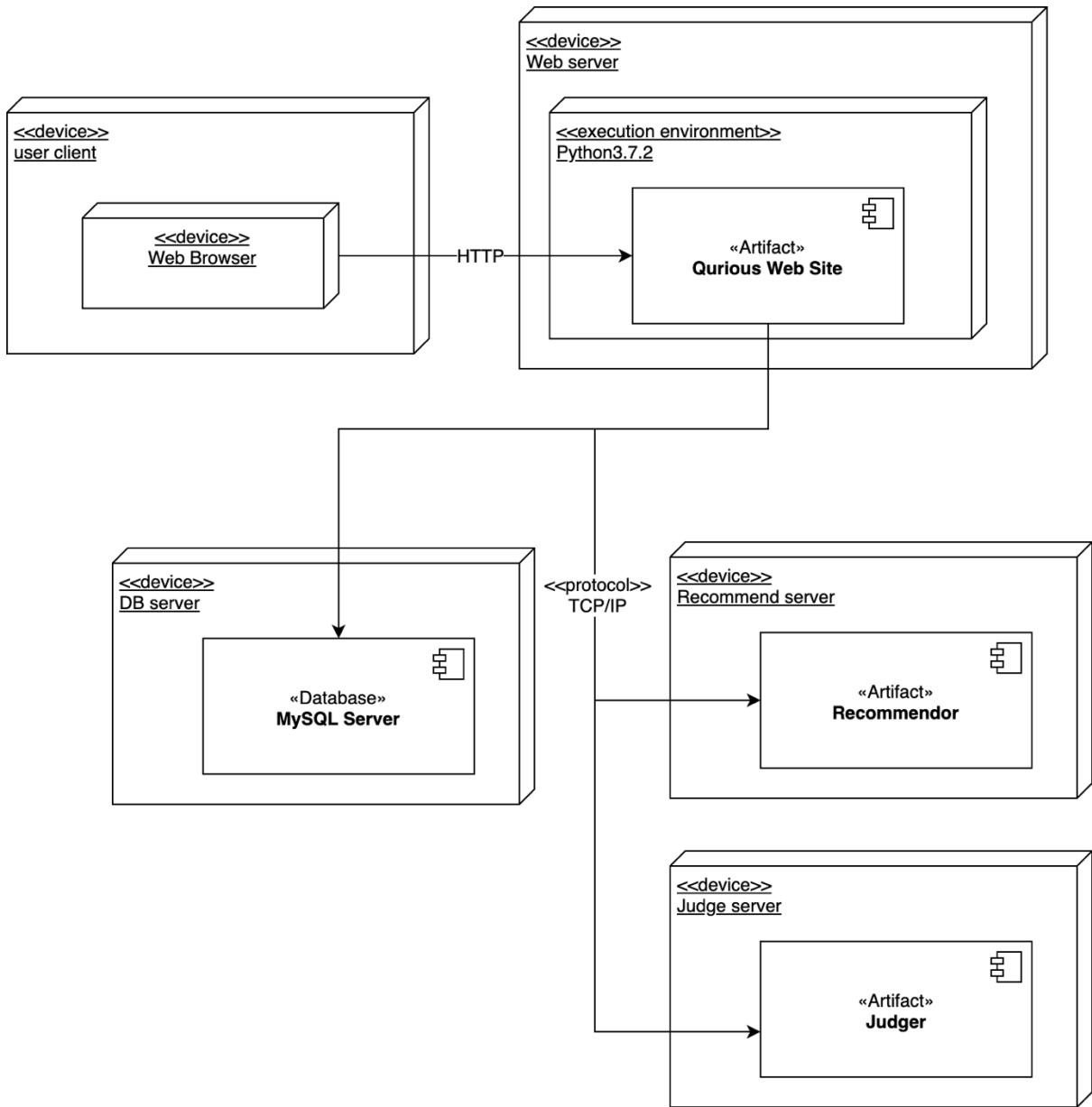


Diagram 9 Deployment Diagram

4 User Management System

4.1 Objective

Qurious의 회원 가입과 로그인 기능을 이용 시에 시스템에서 발생하는 데이터를 처리하는 사용자 관리 시스템의 설계에 대해 설명한다. Class Diagram, Sequence Diagram, State Diagram 등 다양한 도식을 활용하여 User Management System의 구조를 표현하고 서술한다.

4.2 Class Diagram

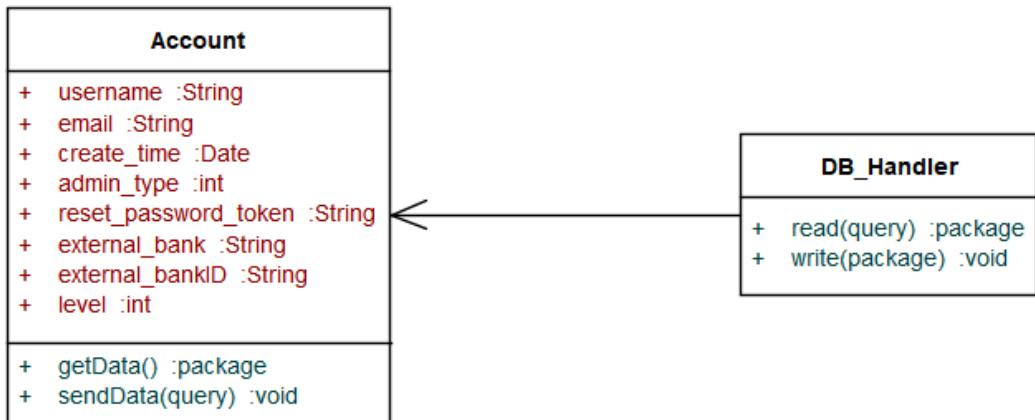


Diagram 10 User Management System: Class Diagram

A. DB_Handler

A.1 Attributes

해당 사항 없음

A.2 Methods

- + package read(query): 해당 데이터베이스에서 원하는 데이터를 읽어온다.
- + void write(package): 해당 데이터베이스에 데이터를 저장한다.

B. Account

B.1 Attributes

- + username : 해당 계정의 ID
- + email : 해당 계정의 이메일 주소
- + create_time : 해당 계정이 생성된 시간
- + admin_type : 해당 계정의 타입 (사용자/관리자)
- + reset_password_token : 해당 계정의 패스워드 토큰
- + external_bank : 해당 계정의 외부 문제은행 이름 (백준/해커랭크)
- + external_bankID : 해당 계정의 외부 문제은행 ID
- + level : 해당 계정의 코딩 수준

B.2 Methods

- + package getData(): 해당 데이터베이스에서 원하는 데이터를 얻어온다.
- + void sendData(Package): 해당 데이터베이스에 데이터를 보낸다.

4.3 Sequence Diagram

A. 회원가입 및 로그인

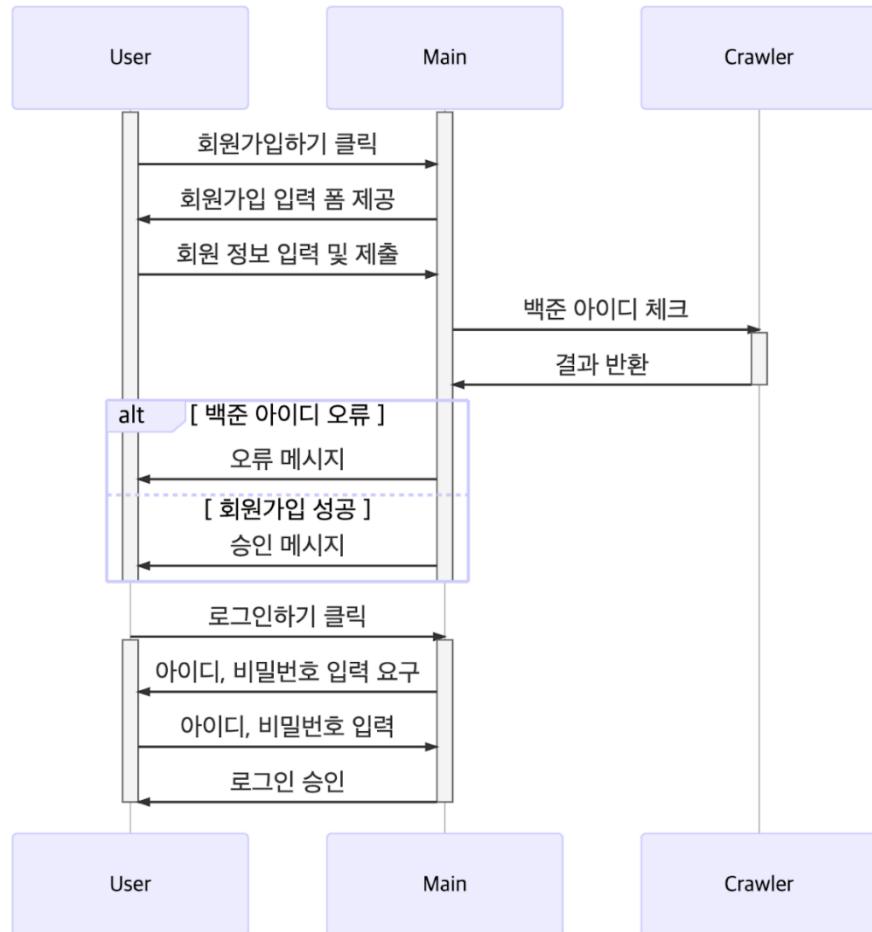


Diagram 11 User Management System: Sequence Diagram

4.4 State Diagram

A. 회원가입

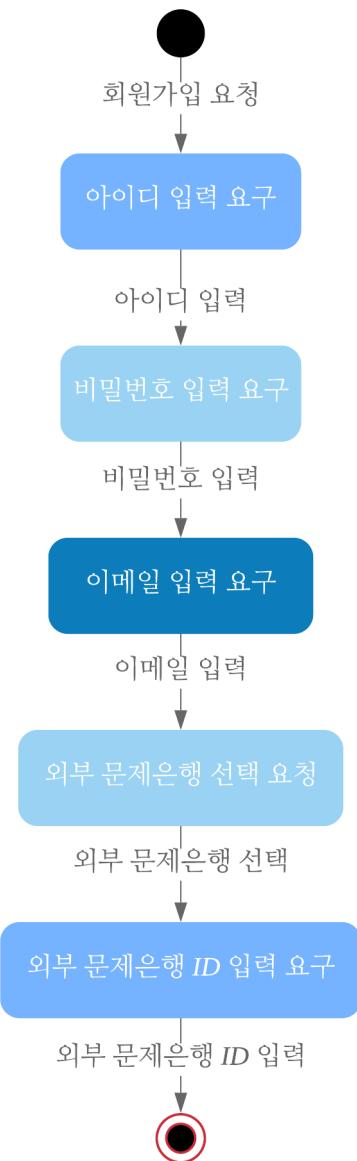


Diagram 12 User Management System: State Diagram 회원가입

B. 로그인

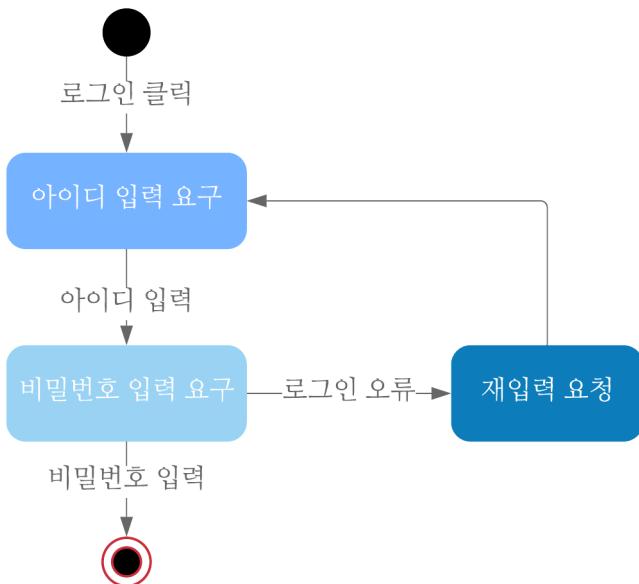


Diagram 13 User Management System: State Diagram 로그인

5 Problem System

5.1 Objective

Problem system이 있다. Problem system은 문제를 관리 및 채점하고 진단고사(level test)와 관련된 시스템이다. Sub-system으로는 Problem, Judge, Test 3개로 나뉜다. 이러한 Problem system에 대하여 Class diagram, Sequence diagram, 그리고 State diagram을 통해 Problem system의 구조를 표현하고 상세 내용을 기술한다

5.2 Class Diagram

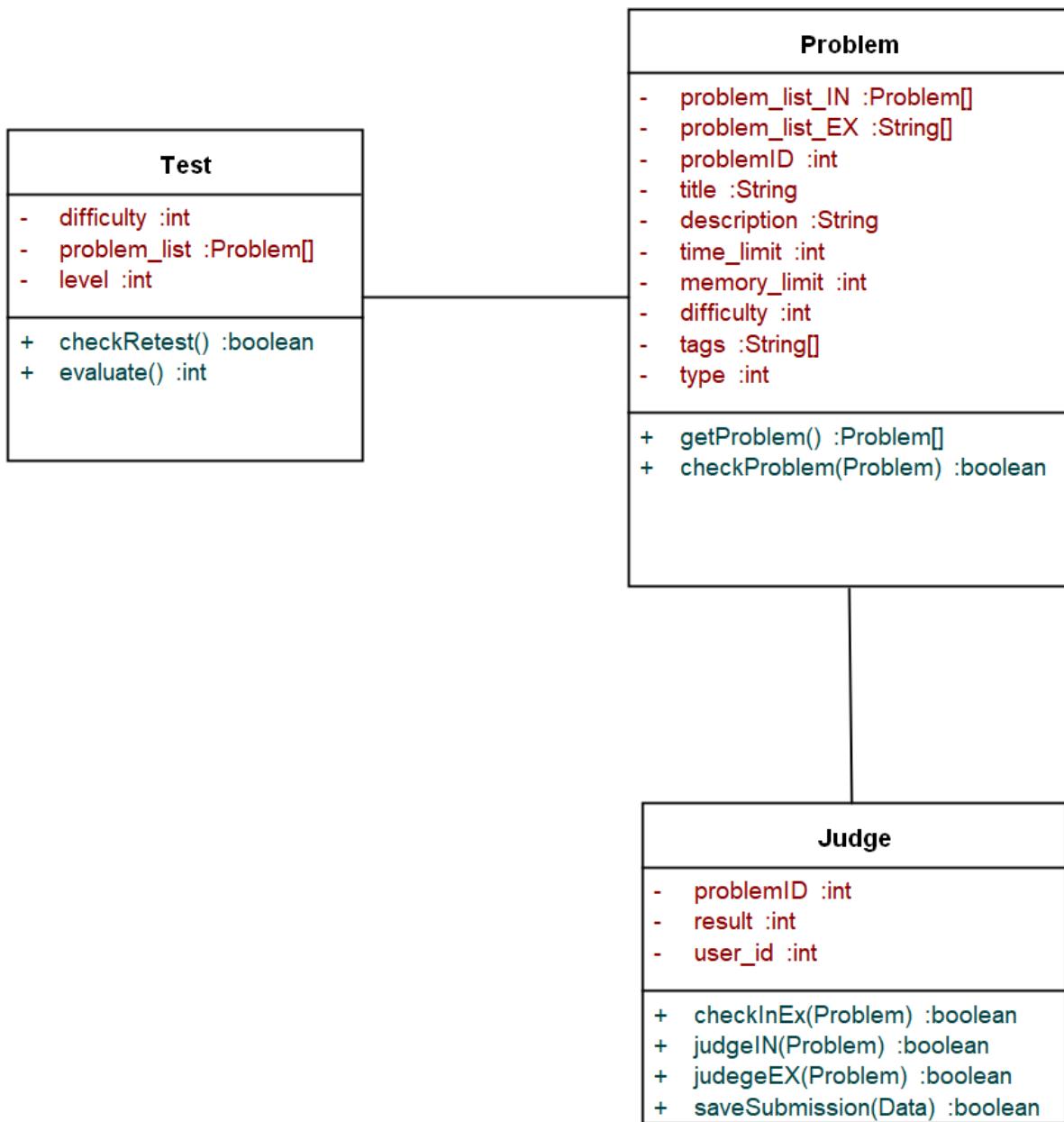


Diagram 14 Problem System: Class Diagram

A. Problem

A.1. Attributes

+problem_list_IN:Problem[]: 내부 문제 배열

+problem_list_EX:String[]: 외부 문제 배열

- +problemID:int: 문제 식별번호
- +title:string: 문제 제목
- +description:string: 문제 설명
- +time_limit:int: 문제의 시간 제한(ms)
- +memory_limit:int: 문제의 메모리 제한(kb)
- +difficulty:int: 난이도
- +tags:string[]: 태그
- +type:int: 문제 유형(0, 1 - n, ex)

A.2 Method

- +getProblem():Problem[]: 문제를 호출한다
- +checkProblem(Problem):Boolean: 문제를 채점한다.

B. Test

B.1. Attributes

- +difficulty:int: 난이도
- +problem_list:Problem[]: 문제 인스턴스 배열
- +level:int: 실력수준

B.2 Method

- +evaluate():int: 평가하기
- +checkRetest():Boolean: 재시험 여부 판단

C. Judge

C.1. Attributes

+ problemId:int: 문제 식별번호

+result:int: 채점 결과

+user_id:int: 사용자 식별번호

C.2 Method

+checkINEx(Problem):Boolean: 내/외부 문제 여부 판단

+judgeIN(Problem):boolean: 내부문제 채점

+judgeEX(Problem):Boolean: 외부문제 채점

+saveSubmission(Data):Boolean: 제출정보 Submission DB에 저장

5.3 Sequence Diagram

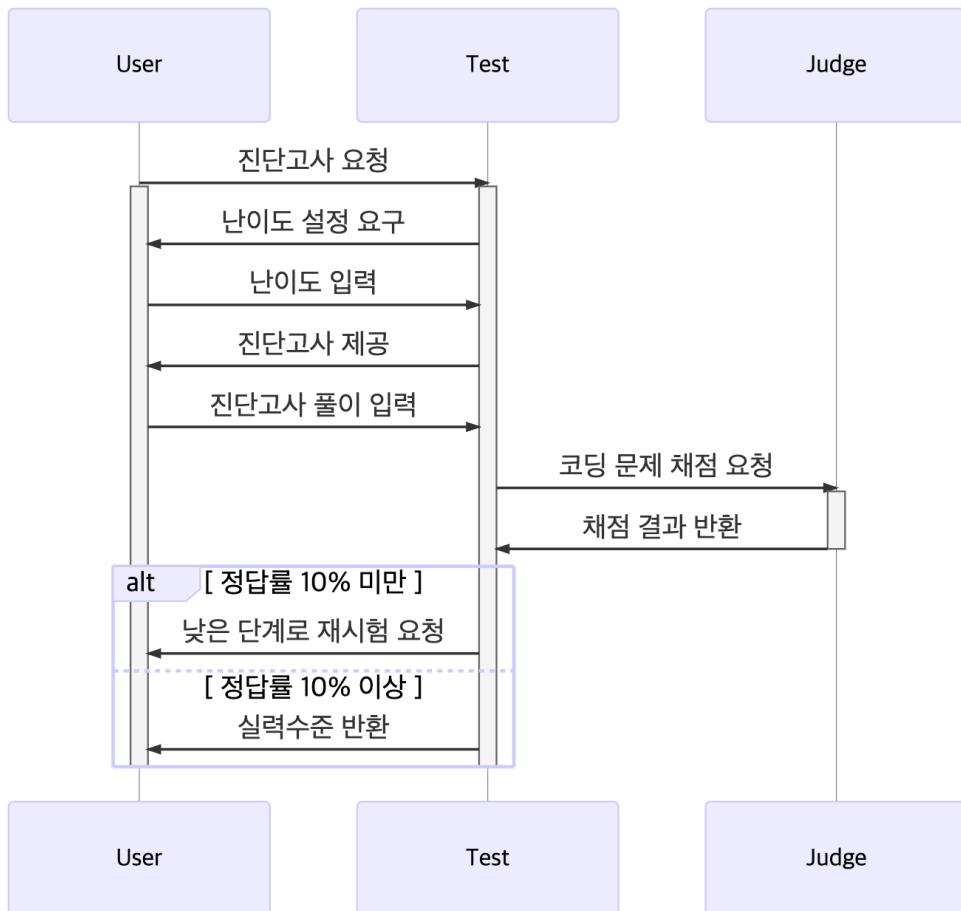


Diagram 15 Problem System: Sequence Diagram 진단고사

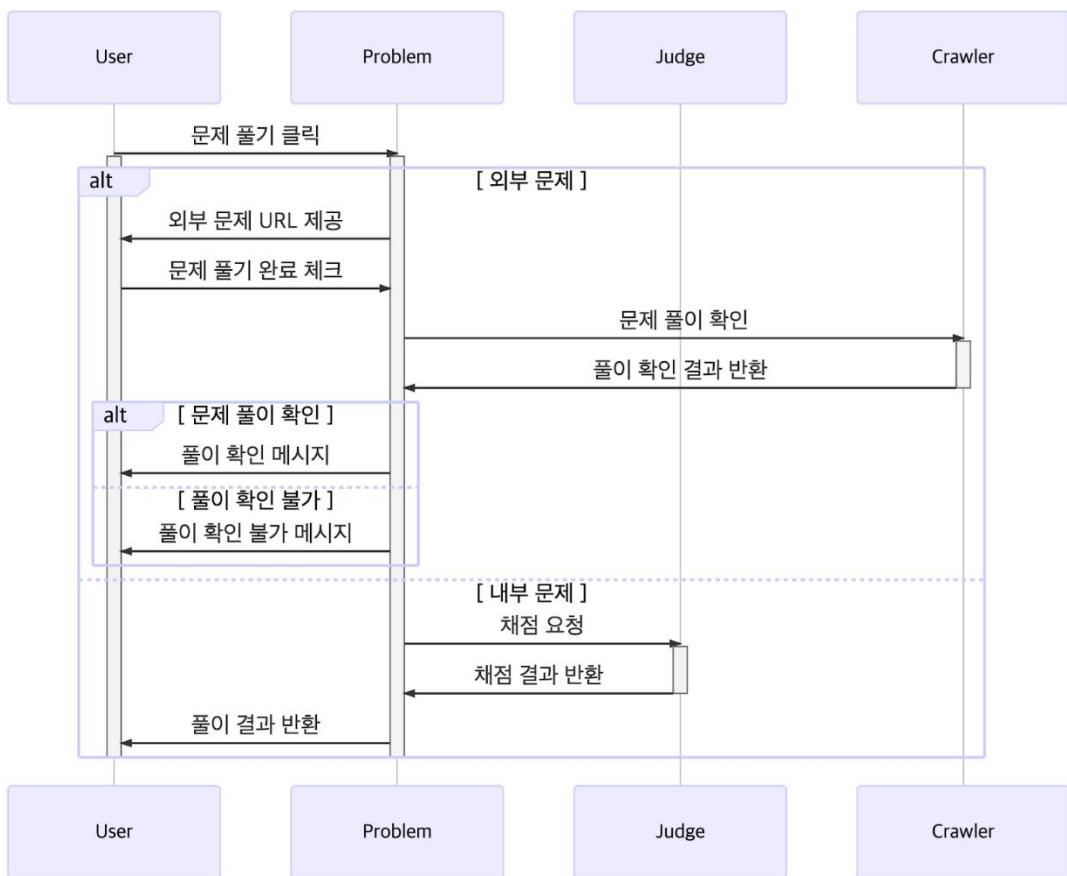


Diagram 16 Problem System: Sequence Diagram 외부 문제 풀이

5.4 State Diagram

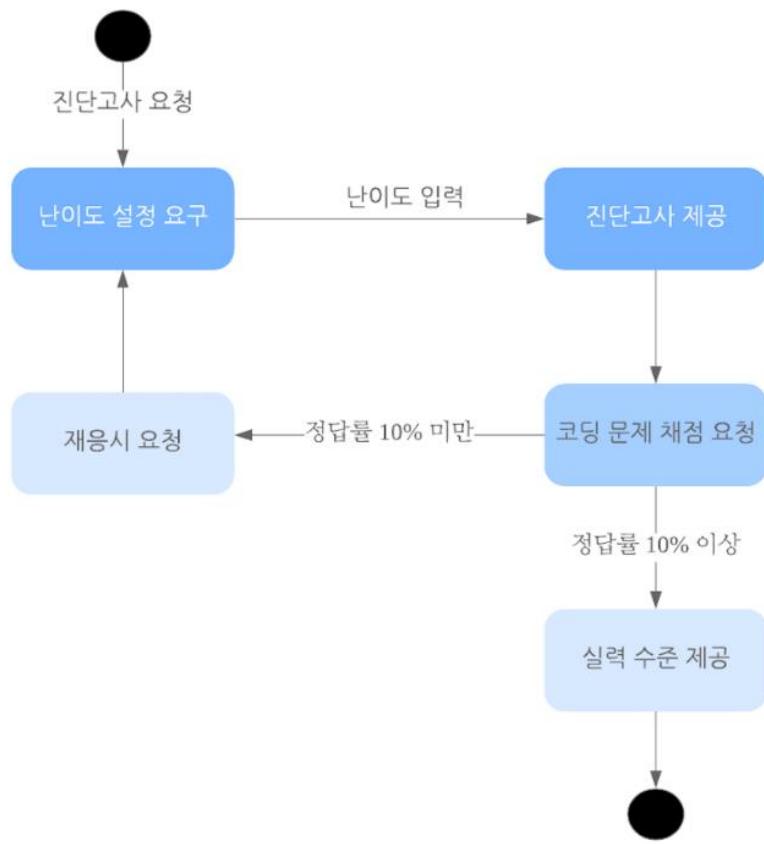


Diagram 17 Problem System: Event Driven Diagram 진단고사

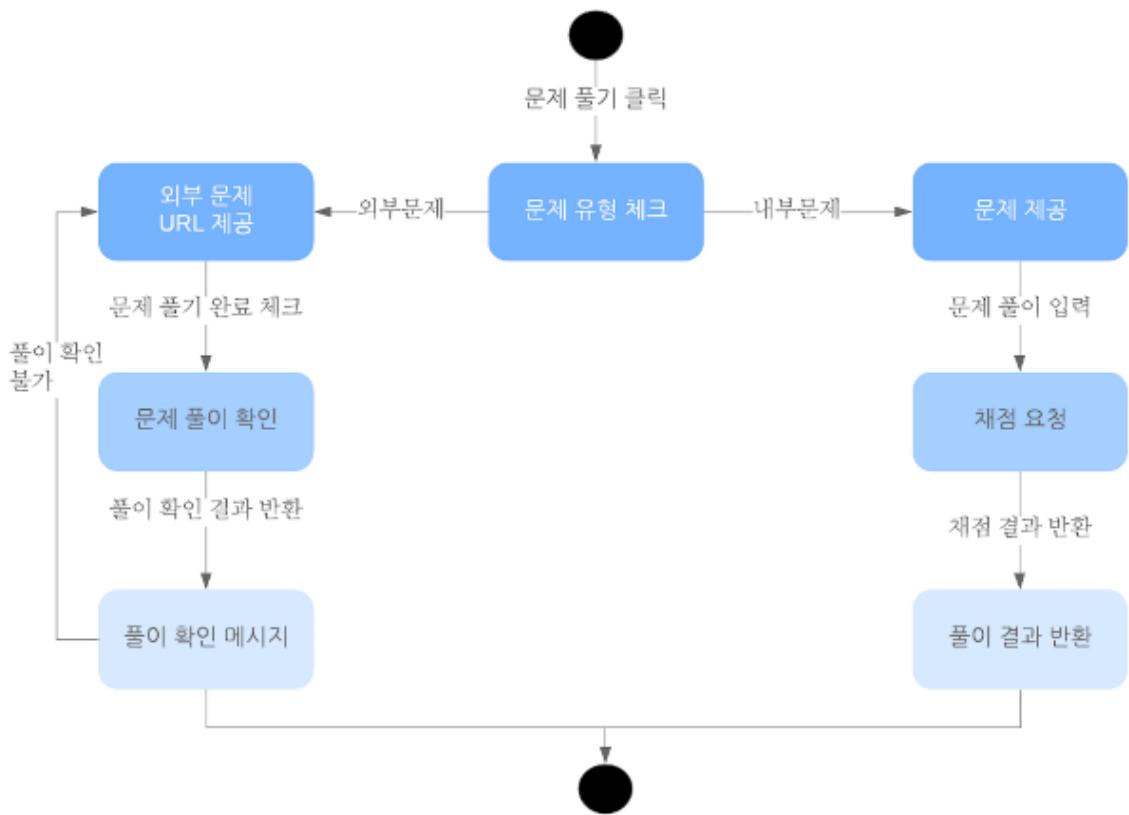


Diagram 18 Problem System: Event Driven Diagram 문제 풀이

6 Recommendation System

6.1 Objective

Recommendation System은 문제를 추천 받기 위한 서비스로 사용자와 관련된다. Recommendation System은 Recommender와 Model와 같은 2개의 Sub-system으로 나뉜다. 이 절에서는 Class Diagram, Sequence Diagram, State Diagram을 통해 Recommendation System의 구조와 사용자 및 데이터베이스와의 상호 작용을 설계한다.

6.2 Class Diagram

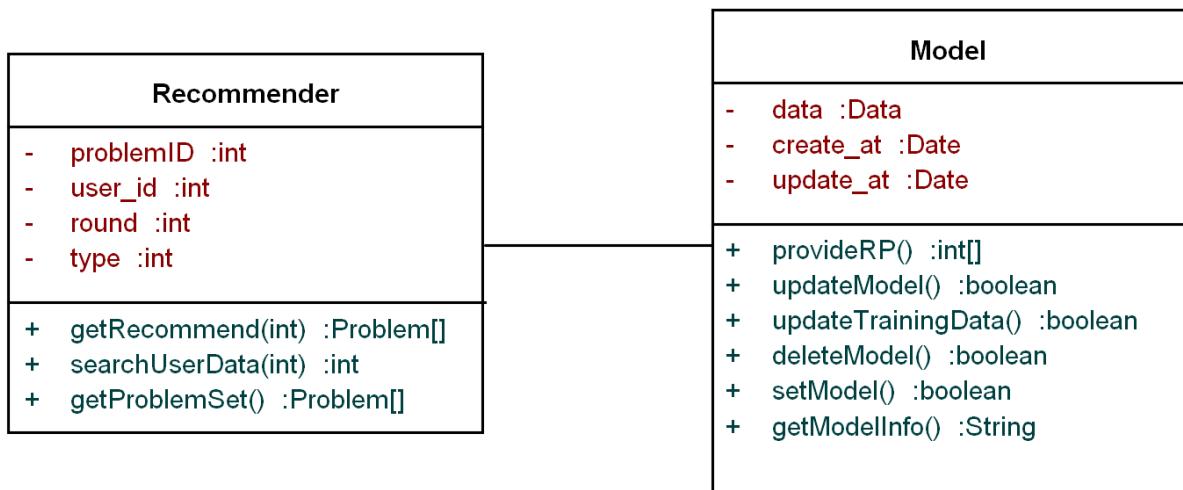


Diagram 19 Recommendation System: Class Diagram

A. Recommender

A.1. Attributes

+problemID:int: 문제 식별번호

+user_id:int: 사용자 식별번호

+round:int: 추천 회차

+type:int: 추천 유형(진단고사, 문제)

A.2 Method

+getRecommend():Problem[]: 추천 ID 받기

+searchUserData():int: 사용자 데이터 가져오기

+getProblemSet():Problem[]: Problem 정보 가져오기

B. Model

B.1. Attributes

+data:Bytes: 추천 모델 데이터

+create_at:datetime 생성 시간:

+update_at:datetime 수정 시간

B.2 Method

+provideRP():int[]: 추천 문제 제공
+updateModel():Boolean: 추천 모델 업데이트하기
+updateTrainingData():boolean: 학습 데이터 업데이트하기
+deleteModel():boolean: 모델 삭제하기
+setModel():Boolean: 추천 모델로 설정하기
+getModellInfo():String: 모델 정보 가져오기

6.3 Sequence Diagram

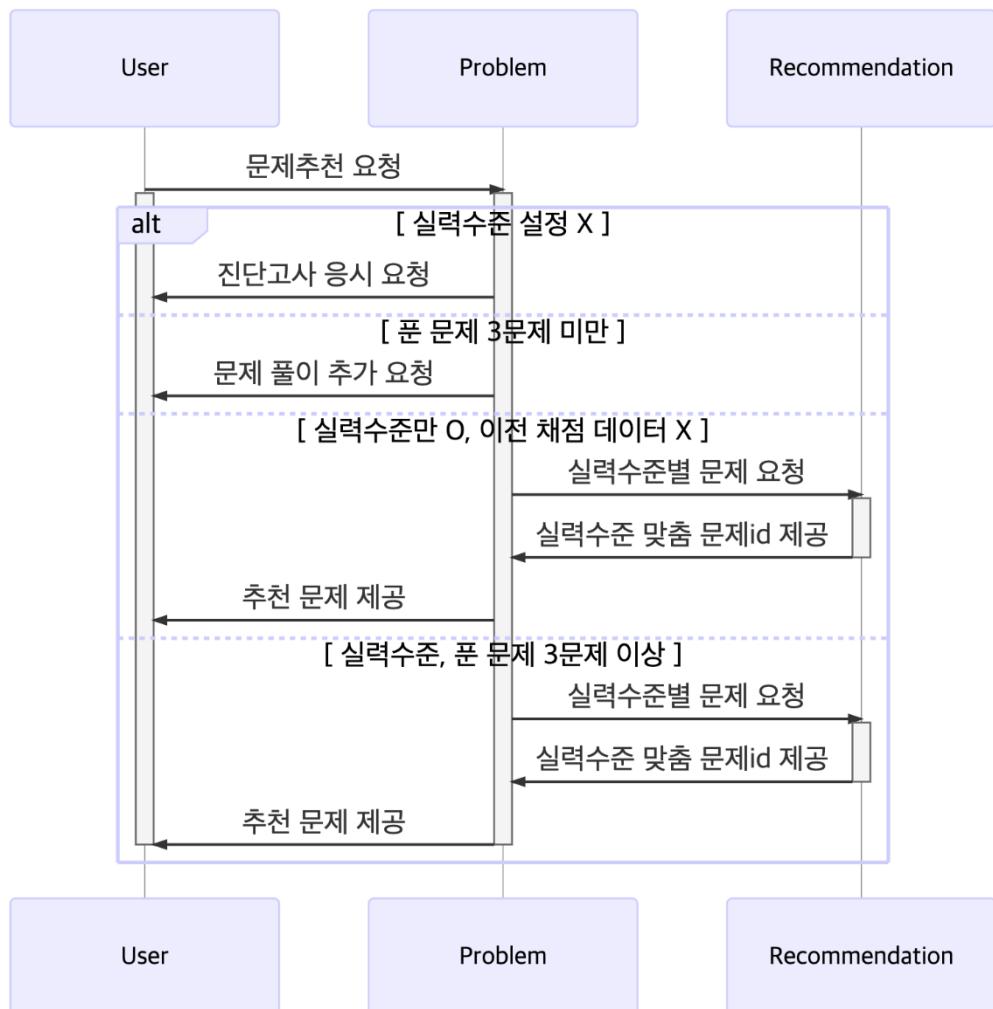


Diagram 20 Recommendation System: Sequence Diagram 추천

6.4 State Diagram

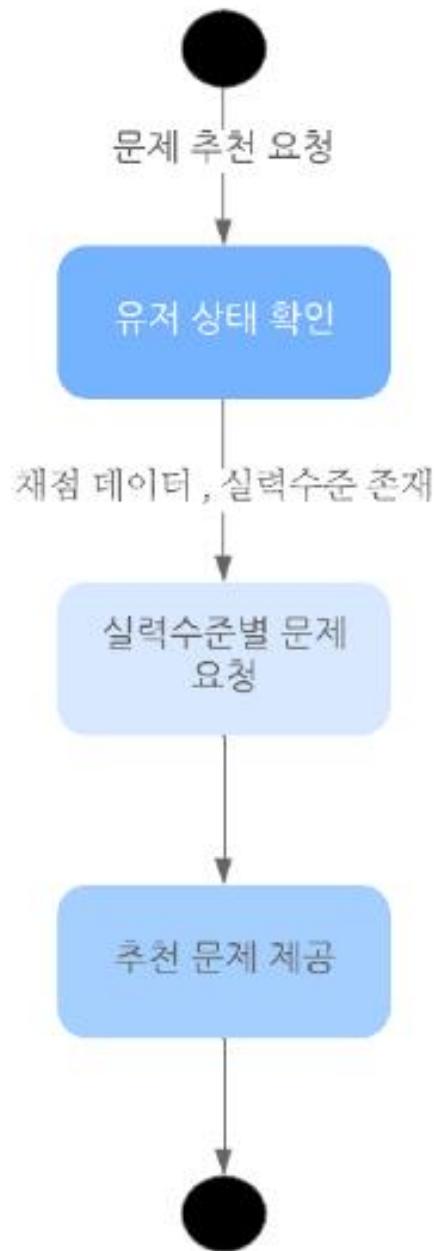


Diagram 21 Recommendation System: Event Driven Diagram 문제 추천

7 Protocol Design

7.1 Overview

Protocol Design에서는 sub-system들의 상호작용에서 필수적으로 준수해야 하는 프로토콜에 대하여 설명한다. 통신하는 메시지의 형식과 용도, 의미를 설명한다.

7.2 JSON

JSON (JavaScript Object Notation)은 속성-값 쌍 또는 키-값 쌍으로 이루어진 데이터 오브젝트를 전달하기 위해 인간이 읽을 수 있는 텍스트를 사용하는 개방형 표준 포맷이다. 기본 자료형으로는 수, 문자열, 참/거짓, 배열, 객체, null이 있다. 인터넷에서 자료를 주고 받을 때 자료를 표현하는 방법으로, 자료 종류의 제한이 없으며 언어 및 플랫폼 독립적인 장점이 있다.

7.3 Protocol Description

A. Overview

Client 와 Server 사이의 요청-응답의 메시지를 protocol 의 구현 기능마다 구분하여 설명하였다. 해당 절의 표는 캡션을 생략한다.

B. Login Protocol

GET	/user/login		
사용자가 로그인할 때 요청하는 주소			
분류	항목명	설명 및 제약사항	예시
Parameter	Id	사용자 아이디	
	password	비밀번호	
Response	Message	"성공", "실패"	
	Status code	상태 코드	
Status Code	200	로그인 성공	
	400	로그인 실패	

C. Registration Protocol

POST	/user		
사용자가 회원가입을 요청할 때 사용하는 주소			
분류	항목명	설명 및 제약사항	예시
Parameter	Id	사용자 아이디	
	Password	비밀번호	
	Password_confirmation	비밀번호 확인, 위 비밀번호와 동일 해야함	
	Hackerrank_id	해커랭크 아이디	
	Baekjoon_id	백준 아이디	
Response	Message	"성공", "실패: 사유"	
	Status code		
Status Code	201	회원가입 성공	
	400	비밀번호 확인이 비밀번호와 동일하지 않음, 백준/해커랭크 아이디가 틀림	
	409	사용자 id 중복일 경우	

D. Level Test Protocol

GET	/problems/test		
사용자가 진단고사를 요청하는 주소			
분류	항목명	설명 및 제약사항	예시
Parameter	level	사용자 설정 난이도	1, 2, 3
Response	Problem_id_list	진단고사 문제 id 리스트	
	Status code	상태 코드	
Status Code	200	테스트 요청 성공	
	400	Level 파라미터가 숫자형이 아님	
	404	해당 난이도가 존재하지 않음	

E. Show Problem Protocol

GET	/problems		
사용자가 문제를 요청하는 주소			
분류	항목명	설명 및 제약사항	예시
Parameter	Id	문제 ID	
Response	Title	문제 제목	
	Type	문제 유형: 진단고사, 내부문제, 외부문제	"level", "external", "internal"
	Data	문제 데이터: 문제 설명, 문제 URL 등 json 형식으로 제공	
	Status code	상태 코드	
Status Code	200	문제 요청 성공	
	400	Id 파라미터가 숫자형이 아님	
	404	해당 문제 존재하지 않음	

F. Submission Protocol

GET	/problems/submission		
사용자가 문제의 정답을 제출하여 채점을 요청하는 주소			
분류	항목명	설명 및 제약사항	예시
Parameter	answer	정답 + 코드 리스트,	
Response	Redirect_url	제출 결과 보여주는 URL 반환	
	Status code	상태 코드	
Status Code	200	문제 제출 성공	
	400	코드와 정답이 형식에 맞지 않음	

G. Get Recommendation Protocol

GET	/recommendations		
사용자가 추천을 요청하는 주소			
분류	항목명	설명 및 제약사항	예시
Parameter	User_id	사용자 ID	
Response	Problem_id_list	추천 문제 ID 리스트	
	Status code	상태 코드	
Status Code	200	문제 추천 성공	
	400	사용자의 Submission수가 부족함	

H. Create Problem Protocol

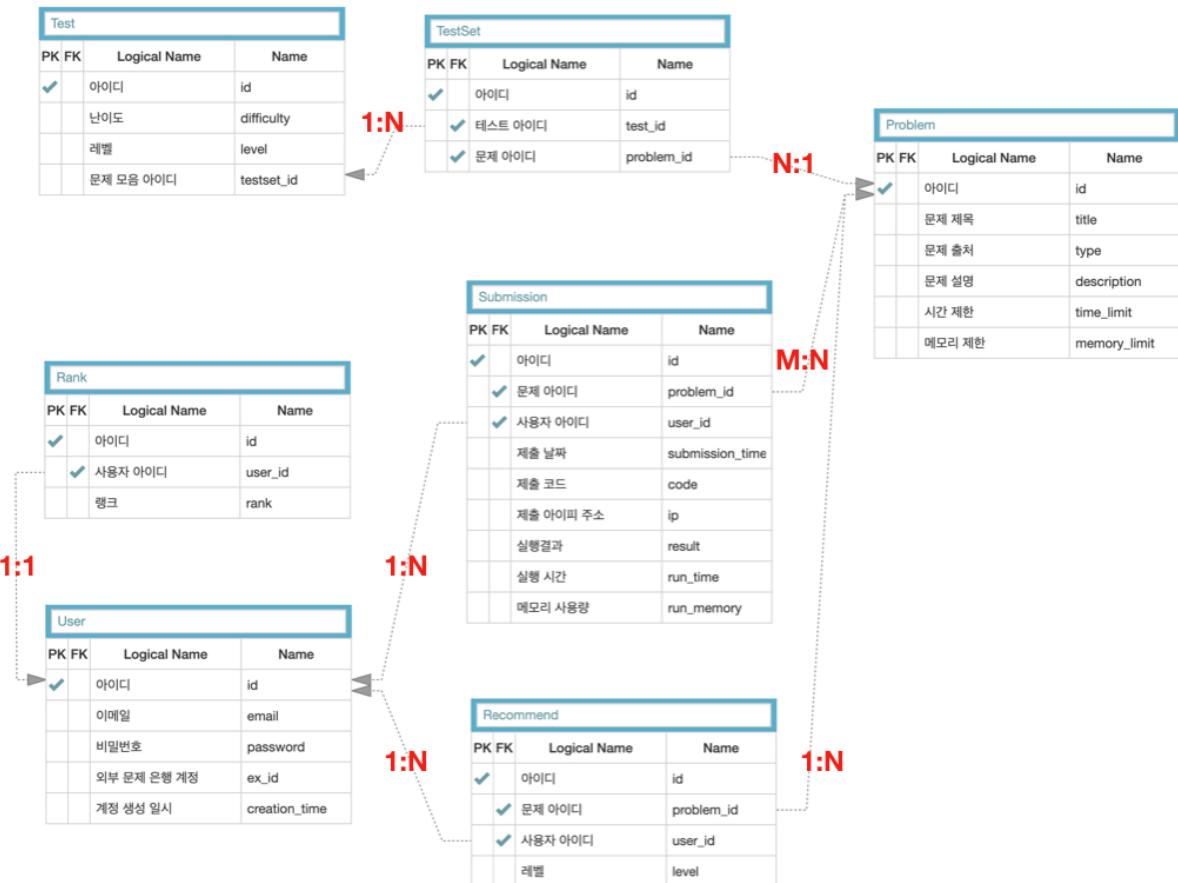
POST	/problems		
관리자가 문제를 추가하는 주소			
분류	항목명	설명 및 제약사항	예시
Parameter	pid	문제 ID	
	Type	문제 유형	
	data	문제 데이터 (문제 설명, 문제 제목, 문제 URL, 테스트 케이스)	
Response	Message	"성공", "실패"	
	Status code	상태 코드	
Status Code	200	문제 추가 성공	
	400	파라미터가 제대로 채워지지 않음	

8 Database Design

8.1 Objective

요구사항 명세서를 기반으로 데이터베이스를 설계한다. 데이터베이스 내의 객체-데이터 관계를 표현하는 ER Diagram을 제시하고, Relational Schema를 도식화하여 서술한다. 또한 실제 테이블을 작성할 수 있는 SQL의 DDL 쿼리를 작성한다.

8.2 ER Diagram



8.3 Relational Schema

A. Problem

테이블 이름	Problem							
테이블 설명	문제							
PRIMARY KEY	id							
FOREIGN KEY								
INDEX								
NO	PK	AI	FK	NULL	컬럼 이름	TYPE	설명	참조 테이블
1	Y	Y			id	INT	아이디	
2					title	VARCHAR(45)	문제 제목	
3					type	INT	0: 내부, 1: 백준	
4					description	VARCHAR(45)	문제 설명	
5				Y	time_limit	DECIMAL(18, 0)	시간 제한	
6				Y	memory_limit	INT	kb 단위	

```
-- Problem Table Create SQL
CREATE TABLE Problem
(
    `id`          INT      NOT NULL  AUTO_INCREMENT COMMENT '아이디',
    `title`        VARCHAR(45) NOT NULL  COMMENT '문제 제목',
    `type`         INT      NOT NULL  COMMENT '0: 내부, 1: 백준',
    `description` VARCHAR(45) NOT NULL  COMMENT '문제 설명',
    `time_limit`   DECIMAL(18, 0) NULL    COMMENT '시간 제한',
    `memory_limit` INT      NULL     COMMENT 'kb단위',
    PRIMARY KEY (id)
);
```

B. Test

테이블 이름		Test						
테이블 설명		진단고사						
PRIMARY KEY		id						
FOREIGN KEY								
INDEX								
NO	PK	AI	FK	NULL	컬럼 이름	TYPE	설명	참조 테이블
1	Y	Y			id	INT	아이디	
2					difficulty	INT	난이도	
3					level	INT	레벨	
4					testset_id	INT	문제 모음 아이디	

```
-- Test Table Create SQL
```

```
CREATE TABLE Test
(
    `id`          INT      NOT NULL  AUTO_INCREMENT COMMENT '아이디',
    `difficulty`  INT      NOT NULL  COMMENT '난이도',
    `level`       INT      NOT NULL  COMMENT '레벨',
    `testset_id`   INT      NOT NULL  COMMENT '문제 모음 아이디',
    PRIMARY KEY (id)
);
```

C. TestSet

테이블 이름		TestSet						
테이블 설명		문제를 바탕으로 생성된 진단고사 문제 집합						
PRIMARY KEY		id						
FOREIGN KEY		test_id, problem_id						
INDEX								
NO	PK	AI	FK	NULL	컬럼 이름	TYPE	설명	참조 테이블
1	Y	Y			id	INT	아이디	
2			Y		test_id	INT	테스트 아이디	Test
3			Y		problem_id	INT	문제 아이디	Problem

```
-- TestSet Table Create SQL
CREATE TABLE TestSet
(
    `id`          INT      NOT NULL      AUTO_INCREMENT COMMENT '아이디',
    `test_id`     INT      NOT NULL      COMMENT '테스트 아이디',
    `problem_id`  INT      NOT NULL      COMMENT '문제 아이디',
    PRIMARY KEY (id)
);

ALTER TABLE TestSet
ADD CONSTRAINT FK_TestSet_test_id FOREIGN KEY (test_id)
    REFERENCES Test (testset_id) ON DELETE RESTRICT ON UPDATE RESTRICT;

ALTER TABLE TestSet
ADD CONSTRAINT FK_TestSet_problem_id FOREIGN KEY (problem_id)
    REFERENCES Problem (id) ON DELETE RESTRICT ON UPDATE RESTRICT;
```

D. Recommend

테이블 이름	Recommend							
테이블 설명	추천 문제							
PRIMARY KEY	id							
FOREIGN KEY	problem_id, user_id							
INDEX								
NO	PK	AI	FK	NULL	컬럼 이름	TYPE	설명	참조 테이블
1	Y	Y			id	INT	아이디	
2			Y		problem_id	INT	문제 아이디	Problem
3			Y		user_id	INT	사용자 아이디	User
4				Y	level	INT	레벨	

```
-- Recommend Table Create SQL
CREATE TABLE Recommend
(
    `id`          INT      NOT NULL      AUTO_INCREMENT COMMENT '아이디',
    `problem_id`  INT      NOT NULL      COMMENT '문제 아이디',
    `user_id`    INT      NOT NULL      COMMENT '사용자 아이디',
    `level`       INT      NULL        COMMENT '레벨',
    PRIMARY KEY (id)
);

ALTER TABLE Recommend
ADD CONSTRAINT FK_Recommend_problem_id FOREIGN KEY (problem_id)
    REFERENCES Problem (id) ON DELETE RESTRICT ON UPDATE RESTRICT;

ALTER TABLE Recommend
ADD CONSTRAINT FK_Recommend_user_id FOREIGN KEY (user_id)
    REFERENCES User (id) ON DELETE RESTRICT ON UPDATE RESTRICT;
```

E. User

테이블 이름	User							
테이블 설명								
PRIMARY KEY	id							
FOREIGN KEY								
INDEX								
NO	PK	AI	FK	NULL	컬럼 이름	TYPE	설명	참조 테이블
1	Y	Y			id	INT	아이디	
2					email	VARCHAR(45)	이메일	
3					password	VARCHAR(45)	비밀번호	
4					ex_id	VARCHAR(45)	외부 문제 응행 계정	
5					creation_time	DATETIME	계정 생성 일시	

```
-- User Table Create SQL
CREATE TABLE User
(
    `id` INT NOT NULL AUTO_INCREMENT COMMENT '아이디',
    `email` VARCHAR(45) NOT NULL COMMENT '이메일',
    `password` VARCHAR(45) NOT NULL COMMENT '비밀번호',
    `ex_id` VARCHAR(45) NOT NULL COMMENT '외부 문제 응행 계정',
    `creation_time` DATETIME NOT NULL COMMENT '계정 생성 일시',
    PRIMARY KEY (id)
);
```

F. Submission

테이블 이름	Submission							
테이블 설명	사용자가 제출한 내용							
PRIMARY KEY	id							
FOREIGN KEY	problem_id, user_id							
INDEX								
NO	PK	AI	FK	NULL	컬럼 이름	TYPE	설명	참조 테이블
1	Y	Y			id	INT	아이디	
2			Y		problem_id	INT	문제 아이디	Problem
3			Y		user_id	INT	사용자 아이디	User
4					submission_time	DATETIME	제출 날짜	
5					code	LONGTEXT	제출 코드	
6					ip	VARCHAR(45)	제출 아이피 주소	
7					result	VARCHAR(45)	정답 / 시간 초과 / 메모리 초과	
8					run_time	DECIMAL(18, 0)	실행 시간	
9					run_memory	DECIMAL(18, 0)	메모리 사용량	

```
-- Submission Table Create SQL
CREATE TABLE Submission
(
    `id`          INT          NOT NULL  AUTO_INCREMENT COMMENT '아이디',
    `problem_id`  INT          NOT NULL  COMMENT '문제 아이디',
    `user_id`     INT          NOT NULL  COMMENT '사용자 아이디',
    `submission_time` DATETIME   NOT NULL  COMMENT '제출 날짜',
    `code`        LONGTEXT    NOT NULL  COMMENT '제출 코드',
    `ip`          VARCHAR(45) NOT NULL  COMMENT '제출 아이피 주소',
    `result`      VARCHAR(45) NOT NULL  COMMENT '정답 / 시간 초과 / 메모리 초
    과 ....,
    `run_time`    DECIMAL(18, 0) NOT NULL  COMMENT '실행 시간',
    `run_memory`  DECIMAL(18, 0) NOT NULL  COMMENT '메모리 사용량',
    PRIMARY KEY (id)
);

ALTER TABLE Submission
ADD CONSTRAINT FK_Submission_problem_id_Problem_id FOREIGN KEY (problem_id)
REFERENCES Problem (id) ON DELETE RESTRICT ON UPDATE RESTRICT;

ALTER TABLE Submission
ADD CONSTRAINT FK_Submission_user_id_User_id FOREIGN KEY (user_id)
REFERENCES User (id) ON DELETE RESTRICT ON UPDATE RESTRICT;
```

G. Rank

테이블 이름		Rank						
테이블 설명								
PRIMARY KEY		id						
FOREIGN KEY		user_id						
INDEX								
NO	PK	AI	FK	NULL	컬럼 이름	TYPE	설명	참조 테이블
1	Y	Y			id	INT	아이디	
2			Y	Y	user_id	INT	사용자 아이디	User
3				Y	rank	VARCHAR(45)	랭크	

```
-- Rank Table Create SQL
CREATE TABLE Rank
(
    `id`          INT          NOT NULL  AUTO_INCREMENT COMMENT '아이디',
    `user_id`     INT          NULL      COMMENT '사용자 아이디',
    `rank`        VARCHAR(45) NULL      COMMENT '랭크',
    PRIMARY KEY (id)
);

ALTER TABLE Rank
ADD CONSTRAINT FK_Rank_user_id_User_id FOREIGN KEY (user_id)
REFERENCES User (id) ON DELETE RESTRICT ON UPDATE RESTRICT;
```

9 Testing Plan

9.1 Objective

Testing Plan에서는 요구사항 명세서에서 기술한, 사용자 및 관리자 시나리오를 바탕으로 한 전체 시스템이 구상한대로 잘 실행되는지 확인한다. 또한, 발생할 수 있는 오류를 사전에 발견하기 위해 진행한다. 관련 테스트를 수행하기 위한 테스팅 정책을 기술하고, 이를 설계한다. 각 testing plan에서는 testing policy와 test case를 설명한다.

9.2 Testing Policy

본 서비스의 Testing은 Development testing, Release testing, User testing 세 단계로 구성된다. 그 중 Development testing은 component testing, system testing, acceptance testing을 순차적으로 진행한다. 다음은 각각의 testing의 정의와 과제들을 서술하고 있다.

A. Developing testing

시스템을 개발하는 과정에서 진행하는 테스트 단계이다. component testing은 개별 component들을 독립적으로 테스트하며, component란 기능, 객체, 혹은 entity 모음들이다. system testing은 component들이 모여 하나의 전체 system을 테스트한다. 기능적인 요소뿐만 아니라 비기능적인 요구사항을 만족하는지 확인한다. Acceptance testing에서는 실제 고객의 데이터를 사용하여 시스템이 고객의 니즈를 만족하는지 확인한다.

B. Release testing

최종적으로 완성된 시스템을 deploy하기 전에 실시하는 테스트이다. 시스템이 사용자의 요구사항을 모두 충족하는지, 서비스를 개발의도에 맞게 제공하는지, 정상적으로 기능이 작동하는지 등을 확인한다.

C. User testing

이 단계는 사용자의 실제 사용 환경에서 시스템을 테스트한다. 실제 사용자들을 통해 product가 직관적인지, 사용하기 쉬운지 등을 평가하고 product를 바로 출시할 것인지 결정한다.

9.3 Test Case

A. User Management System

A.1 회원가입

구성요소	설명
테스트 이름 (Test Title)	회원가입 테스트
사전 조건 (Precondition)	<ol style="list-style-type: none"> 1. 이메일: ' 아이디 문자열 @ 도메인 문자열 ' 형식이며, 중복되어서는 안된다. 2. 비밀번호는 8~16자의 문자열 형식이다. 3. 외부 알고리즘 서비스는 백준, Hackerrank로 한정한다.
수행 절차 (Test Step)	<ol style="list-style-type: none"> 1. 이메일 형식 체크 2. 이메일 중복 체크 3. 비밀번호 형식 체크 4. 외부 문제 은행 아이디 존재 여부 체크 5. 모든 양식 작성했는지 체크
기대되는 결과 (Expected Result)	회원 가입 후 데이터베이스에 회원 정보 저장 로그인 화면으로 전환

Table 1 Test Case: 회원가입

A.2 로그인

구성요소	설명
테스트 이름 (Test Title)	로그인 테스트
사전 조건 (Precondition)	<ol style="list-style-type: none"> 1. 이메일은 '아이디 문자열 @ 도메인 문자열' 형식이다. 2. 비밀번호는 4~12자의 문자열 형식이다. 3. 여러번 로그인 시도할 경우, 최초 한 번의 시도만 서버에 요청한다.
수행 절차 (Test Step)	<ol style="list-style-type: none"> 1. 이메일 형식 체크 2. 비밀번호 형식 체크 3. 모든 양식 기입했는지 체크 4. 이메일 존재여부 체크 5. 이메일과 비밀번호 일치 여부 체크
기대되는 결과 (Expected Result)	사용자 로그인 상태로 기록, 메인 화면으로 전환

Table 2 Test Case: 로그인

A.3 마이페이지 비밀번호 변경

구성요소	설명
테스트 이름 (Test Title)	마이페이지 비밀번호 변경 테스트
사전 조건 (Precondition)	사용자는 로그인한 상태여야 한다.

수행 절차 (Test Step)	<ol style="list-style-type: none"> 1. 사용자가 마이페이지 버튼을 누른다. 2. 사용자가 비밀번호 변경 버튼을 누른다. 3. 새로운 비밀번호를 입력할 화면이 뜬다 4. 새로운 비밀번호를 입력한다. 5. 비밀번호 확인 칸에 다시 입력한다. 6. 비밀번호 변경 확인 버튼을 누른다.
기대되는 결과 (Expected Result)	새로운 비밀번호를 사용자DB에 저장한다.

Table 3 Test Case: 마이페이지 비밀번호 변경

A.4 마이페이지 푼 문제 확인

구성요소	설명
테스트 이름 (Test Title)	마이페이지 푼 문제 확인 테스트
사전 조건 (Precondition)	사용자는 로그인한 상태여야 한다.
수행 절차 (Test Step)	<ol style="list-style-type: none"> 1. 사용자가 마이페이지 버튼을 누른다. 2. 사용자가 푼 문제 리스트를 본다 3. 사용자가 자신이 푼 문제를 확인한다.
기대되는 결과 (Expected Result)	사용자가 자신이 푼 문제들을 확인한다.

Table 4 Test Case: 마이페이지 푼 문제 확인

B. Problem System

B.1 진단고사 응시

구성요소	설명
테스트 이름 (Test Title)	진단고사 응시 테스트
사전 조건 (Precondition)	<ol style="list-style-type: none"> 1. 사용자는 로그인한 상태여야 한다. 2. 사용자는 모든 문항에 공백이 아닌 응답을 하여야 한다.
수행 절차 (Test Step)	<ol style="list-style-type: none"> 1. 사용자가 테스트 응시하기 버튼을 누른다. 2. 난이도 초급, 중급, 고급을 선택할 수 있게 한다. 3. 사용자는 난이도를 하나 선택한다. 4. 선택한 난이도에 맞는 진단고사 화면이 뜬다. 5. 사용자는 데이터를 응답창에 입력하고 제출한다. 6. 사용자의 응답이 채점되어 정답률을 계산하여 실력수준이 리턴된다. 7. 사용자의 실력수준이 표시된다.
기대되는 결과 (Expected Result)	사용자의 실력수준이 새로운 화면에 표시되고, 사용자 데이터베이스에 실력수준이 저장된다.

Table 5 Test Case: 진단고사 응시

B.2 내부 문제 풀기

구성요소	설명
테스트 이름 (Test Title)	내부 문제 풀기 테스트

사전 조건 (Precondition)	사용자는 로그인된 상태여야한다.
수행 절차 (Test Step)	<ol style="list-style-type: none"> 1. 사용자가 내부 문제를 클릭한다. 2. 사용자는 문제 설명을 확인한다. 3. 사용자는 내부 문제의 정답 코드를 작성한다. 4. 사용자는 제출하기 버튼을 클릭한다. 5. 사용자는 결과화면을 확인한다.
기대되는 결과 (Expected Result)	사용자의 내부 문제 채점 결과 (성공, 실패, 에러)

Table 6 Test Case: 내부 문제 풀기

B.3 외부 문제 풀기

구성요소	설명
테스트 이름 (Test Title)	외부 문제 풀기 테스트
사전 조건 (Precondition)	사용자는 로그인 된 상태여야 하고 실력 수준이 있는 상태여야 한다.
수행 절차 (Test Step)	<ol style="list-style-type: none"> 1. 사용자가 문제 추천 받기 버튼을 클릭한다 2. 추천받을 외부 문제 은행(백준, hackerrank)을 선택한다. 3. 외부 문제를 5개 추천받는다. 4. 풀고 싶은 문제 URL을 클릭한다.

	<p>5. 외부 문제 은행 사이트에서 코드를 작성한다.</p> <p>6. 문제를 풀고 Qurious 페이지로 돌아와서 체크박스에 체크한다.</p>
기대되는 결과 (Expected Result)	<ul style="list-style-type: none"> - Qurious 문제, 백준 문제, hackerrank 문제 버튼이 나온다. - 5개의 URL을 출력한다. - 해당 외부 문제 페이지로 이동한다. - 외부문제 풀이 여부를 체크한다.

Table 7 Test Case: 외부 문제 풀기

B.4 내부 문제 관리하기

구성요소	설명
테스트 이름 (Test Title)	내부 문제 관리하기 테스트
사전 조건 (Precondition)	관리자로 로그인된 상태여야한다.
수행 절차 (Test Step)	<ol style="list-style-type: none"> 1. 관리자는 내부 문제 추가하기 버튼을 누른다. 2. 관리자는 내부 문제 ID를 입력한다. (0~9999999) 3. 관리자는 문제 이름을 입력한다. 4. 관리자는 문제 설명을 입력한다. 5. 관리자는 입력 설명을 입력한다. 6. 관리자는 출력 설명을 입력한다. 7. 관리자는 입력 예시를 입력한다. 8. 관리자는 출력 예시를 입력한다.

	<p>9. 관리자는 입출력 케이스를 업로드한다.</p> <p>10. 관리자는 기타 사항을 입력한다.</p> <p>11. 관리자는 추가하기 버튼을 누른다.</p>
기대되는 결과 (Expected Result)	시스템 메시지 (성공, 에러)

Table 8 Test Case: 내부 문제 관리하기

B.5 외부 문제 관리하기

구성요소	설명
테스트 이름 (Test Title)	외부 문제 관리하기 테스트
사전 조건 (Precondition)	관리자로 로그인된 상태여야한다.
수행 절차 (Test Step)	<ol style="list-style-type: none"> 관리자는 외부 문제 추가하기 버튼을 누른다. 관리자는 외부 문제 ID를 입력한다. (0~99999999) 관리자는 문제 이름을 입력한다. 관리자는 외부 문제 응행 이름을 선택한다. 관리자는 외부 문제 URL를 입력한다. 관리자는 추가하기 버튼을 누른다.
기대되는 결과 (Expected Result)	시스템 메시지 (성공, 에러)

Table 9 Test Case: 외부 문제 관리하기

B.6 랭킹 확인

구성요소	설명
테스트 이름 (Test Title)	랭킹 확인
사전 조건 (Precondition)	사용자는 로그인 된 상태여야 한다
수행 절차 (Test Step)	사용자가 랭킹 확인 버튼을 클릭한다.
기대되는 결과 (Expected Result)	<ul style="list-style-type: none"> - 사용자의 개별 랭킹을 출력한다. - Qurious 사용자의 전체 랭킹을 출력한다.

Table 10 Test Case: 랭킹 확인

C. Recommendation System

C.1 진단고사 기반 추천

구성 요소	설명
테스트 이름 Test Title	진단고사 기반 추천 테스트
사전 조건 (Precondition)	사용자는 로그인 된 상태여야 하고, 문제 풀이 데이터가 존재하지 않고, 진단고사를 푼 이후 실력 수준만 나온 상태여야 한다.

수행 절차 (Test Step)	<ol style="list-style-type: none"> 1. 사용자는 로그인을 하고 진단고사를 보고 메인 페이지로 이동한다. 2. 메인 페이지의 '추천 받기' 버튼을 클릭한다.
기대되는 결과 (Expected Result)	사용자의 진단고사 결과에 따른 추천 문제 5개가 메인 페이지의 대시보드에 차례대로 나타난다.

Table 11 Test Case: 진단고사 기반 추천

C.2 추천 서버 기반 추천

구성 요소	설명
테스트 이름 (Test Title)	추천 서버 기반 추천 테스트
사전 조건 (Precondition)	사용자는 로그인 된 상태에서 진단고사를 푼 이후, 진단고사 기반으로 추천 받은 문제들을 최소 3개이상 풀어야 한다.
수행 절차 (Test Step)	<ol style="list-style-type: none"> 1. 사용자는 로그인을 하고 메인 페이지로 이동한다. 2. 문제를 3문제 이상 풀고 '추천 받기' 버튼을 클릭한다.
기대되는 결과 (Expected Result)	사용자의 실력 수준 및 문제 풀이 데이터에 기반한 추천 문제 5개가 메인 페이지의 대시보드에 우선순위대로 나타난다.

Table 12 Test Case: 추천 서버 기반 추천

10 Development Environment

10.1 Objective

Development Environment에서는 실제 시스템 개발을 위해 필요한 개발 환경과 코딩 규칙에 대해 기술한다. 개발 과정에서의 버전 관리 도구를 설명한다. 프로그래밍 과정에서 기반이 되는 규칙들에 대해 서술한다.

10.2 Bootstrap



Diagram 22 Logo of Bootstrap

Bootstrap은 웹사이트를 쉽게 만들 수 있게 도와주는 HTML, CSS, JS 프레임워크이다. 하나의 CSS로 휴대폰, 태블릿, 데스크탑까지 다양한 기기에서 작동한다. 다양한 기능을 제공하여 사용자가 쉽게 웹사이트를 제작, 유지, 보수할 수 있도록 도와준다. (wikipedia, 2019)

10.3 Vue.js



Diagram 23 Logo of Vue.js

Vue.js는 사용자 인터페이스 빌드를 위한 오픈 소스 프로그래시브 자바스크립트 프레임워크이다. 다른 단일형 프레임워크와 달리 Vue는 점진적으로 채택할 수 있도록 설계되어 있다. 핵심 라이브러리는 뷰 레이어만 초점을 맞추어 다른 라이브러리나 기존 프로젝트와의 통합이 쉽다.

10.4 Django



Diagram 24 Logo of Django

Django는 파이썬으로 작성된 오픈 소스 웹 애플리케이션 프레임워크로, 모델-뷰-컨트롤러(MVC) 패턴을 따르고 있다. 좀 더 쉽게 웹 개발을 할 수 있게 돋는 오픈소스 기술이다. 장고는 개발 과정을 단축시켜주고, 다양한 추가 기능을 포함하고 있으며 보안성이 높다. 유연하고 빠른 확장성을 제공해 CMS부터 SNS까지 다양한 플랫폼에 구축·활용되고 있다.

10.5 MySQL



Diagram 25 Logo of MySQL

MySQL은 세계에서 가장 많이 쓰이는 오픈 소스의 관계형 데이터베이스 관리 시스템이다. 다중

스레드, 다중 사용자 형식의 구조질의어 형식의 데이터베이스 관리 시스템으로서 오라클이 관리 및 지원하고 있으며, Qt처럼 이중 라이선스가 적용된다.

10.6 Github



Diagram 26 Logo of GitHub

버전관리 시스템인 git을 이용하는 프로젝트들의 버전제어 및 공동 작업을 위한 코드 호스팅 플랫폼이다. 전 세계에서 오픈소스 프로젝트 관리를 위해 가장 많이 사용되는 서비스 중 하나이다. 사용자에게 무료로 계정과 저장소를 제공하고, 부분적으로 비공개 프로젝트도 무료로 진행할 수 있다. 한국을 포함한 전세계 IT 업계에서는 GitHub 계정이 취업 시 일종의 포트폴리오로 사용되기도 한다.

11 Development Plan

11.1 Objective

Development Plan에서는 프로젝트 개발 일정을 기술한다. Gantt Chart로 전체적인 개발 순서와 시기를 알아보기 쉽게 표현하고 현재까지의 개발 현황을 설명한다.

11.2 Gantt Chart

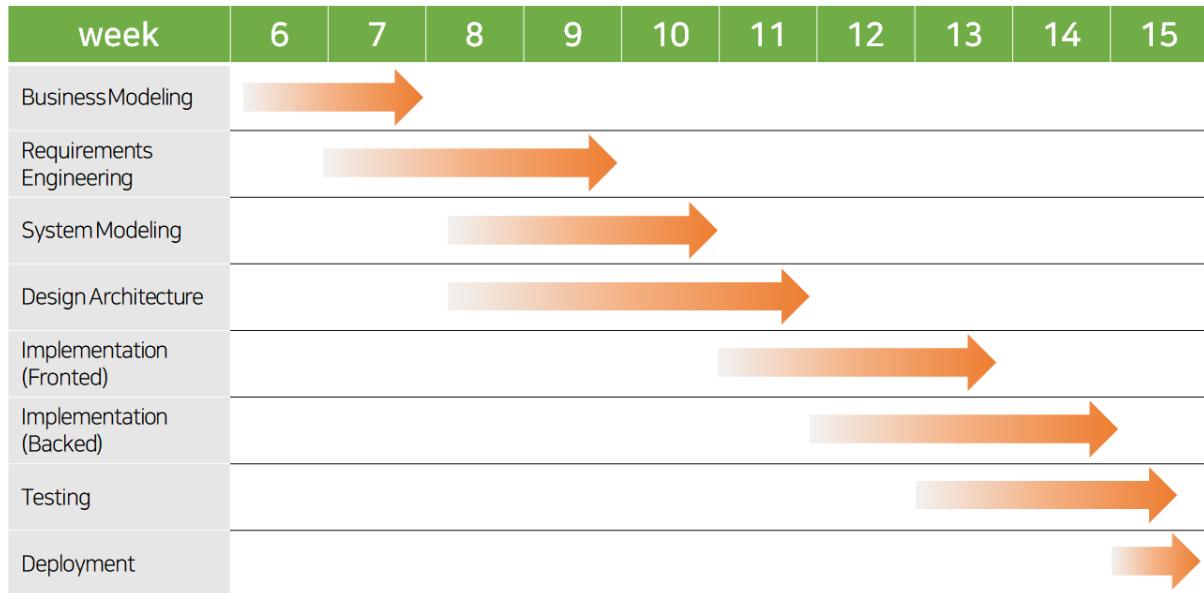


Diagram 27 Gantt Chart

지금까지 개발 상황은 다음과 같다. 점점 중요해지는 소프트웨어 교육을 사용자 맞춤형으로 제공하는 서비스와 필요한 기능들을 정의하였다. 또 사용자 요구사항을 구체적으로 명세화 하였다. 요구사항을 functional, non-functional로 나누어 분석하고, 동시에 system modeling을 다양한 diagram으로 표현하였다.

12 Index

12.1 Objective

Index에서는 문서의 인덱스들을 정리한다. 다이어그램 인덱스 및 기능 인덱스가 포함된다.

12.2 Figure Indexx

Figure 1 Logo of UML	8
Figure 2 Example of Deployment Diagram	9
Figure 3 Example of Class Diagram	10
Figure 4 Example of State Diagram	11
Figure 5 Example of Sequence Diagram	12
Figure 6 Example of ER Diagram	13
Figure 7 Logo of Visual Studio Code	14
Figure 8 Logo of Enterprise Architect	14
Figure 9 Web page of AQuery Tool.....	15
Figure 10 Web page of Mermaid Live Editor	16
Figure 11 Logo of Draw.io	16

12.3 Diagram Index

Diagram 1 Product Scope: User Management System.....	17
Diagram 2 Product Scope: Problem System	18
Diagram 3 Product Scope: Recommendation System.....	19
Diagram 4 System Architecture: Qurious 전체.....	20
Diagram 5 System Architecture: 사용자 시스템	21
Diagram 6 System Architecture: 진단고사 시스템	22
Diagram 7 System Architecture: 문제 풀이 시스템	23
Diagram 8 System Architecture: 문제 추천 시스템	24

Diagram 9 Deployment Diagram	25
Diagram 10 User Management System: Class Diagram.....	26
Diagram 11 User Management System: Sequence Diagram	27
Diagram 12 User Management System: State Diagram 회원가입	28
Diagram 13 User Management System: State Diagram 로그인	29
Diagram 14 Problem System: Class Diagram	30
Diagram 15 Problem System: Sequence Diagram 진단고사.....	33
Diagram 16 Problem System: Sequence Diagram 외부 문제 풀이.....	34
Diagram 17 Problem System: Event Driven Diagram 진단고사	35
Diagram 18 Problem System: Event Driven Diagram 문제 풀이	36
Diagram 19 Recommendation System: Class Diagram.....	37
Diagram 20 Recommendation System: Sequence Diagram 추천	38
Diagram 21 Recommendation System: Event Driven Diagram 문제 추천.....	39
Diagram 22 Logo of Bootstrap	59
Diagram 23 Logo of Vue.js.....	59
Diagram 24 Logo of Django	60
Diagram 25 Logo of MySQL.....	60
Diagram 26 Logo of GitHub.....	61
Diagram 27 Gantt Chart.....	62

12.4 Table Index

Table 1 Test Case: 회원가입.....	50
Table 2 Test Case: 로그인	51
Table 3 Test Case: 마이페이지 비밀번호 변경	52
Table 4 Test Case: 마이페이지 푼 문제 확인	52

Table 5 Test Case: 진단고사 응시	53
Table 6 Test Case: 내부 문제 풀기	54
Table 7 Test Case: 외부 문제 풀기	55
Table 8 Test Case: 내부 문제 관리하기	56
Table 9 Test Case: 외부 문제 관리하기	57
Table 10 Test Case: 랭킹 확인	57
Table 11 Test Case: 진단고사 기반 추천	58
Table 12 Test Case: 추천 서버 기반 추천	58

13 References

13.1 Objectives

문서 작성에 참고한 참고문헌 목록을 기술한다.

13.2 Reference

- Robert A. Maksimchuk, 운명적 존재를 위한 UML, 지앤선, 2005.9.20, (2019.05.06)
- State Diagram, (2019.05.06)

<https://www.geeksforgeeks.org/unified-modeling-language-uml-state-diagrams/>
- 위키피디아 ER모델, (2019.05.06)

https://ko.wikipedia.org/wiki/%EA%B0%9C%EC%B2%B4-%EA%B4%80%EA%B3%84_%EB%AA%A8%EB%8D%B8#%EC%82%AC%EC%9A%A9%EC%B2%98
- 위키피디아 Bootstrap, (2019.05.12)

[https://ko.wikipedia.org/wiki/%EB%B6%80%ED%8A%B8%EC%8A%A4%ED%8A%B8%EB%9E%A9_\(%ED%94%84%EB%A1%A0%ED%8A%B8%EC%97%94%EB%93%9C_%ED%94%84%EB%A0%88%EC%9E%84%EC%9B%8C%ED%81%AC\)](https://ko.wikipedia.org/wiki/%EB%B6%80%ED%8A%B8%EC%8A%A4%ED%8A%B8%EB%9E%A9_(%ED%94%84%EB%A1%A0%ED%8A%B8%EC%97%94%EB%93%9C_%ED%94%84%EB%A0%88%EC%9E%84%EC%9B%8C%ED%81%AC))
- 위키피디아 Vue.js (2019.05.13)

<https://en.wikipedia.org/wiki/Vue.js>

<https://ko.wikipedia.org/wiki/Vue.js>

- 위키피디아 Django (2019.05.11)

[https://ko.wikipedia.org/wiki/%EC%9E%A5%EA%B3%A0_\(%EC%9B%B9_%ED%94%84%EB%A0%88%EC%9E%84%EC%9B%8C%ED%81%AC\)](https://ko.wikipedia.org/wiki/%EC%9E%A5%EA%B3%A0_(%EC%9B%B9_%ED%94%84%EB%A0%88%EC%9E%84%EC%9B%8C%ED%81%AC))

<https://hackernoon.com/why-i-chose-django-over-java-frameworks-for-my-recent-project-7ec85cb35756>

- 위키피디아 MySQL (2019.05.11)

<https://ko.wikipedia.org/wiki/MySQL>

- 한국정보통신기술협회 깃허브, Github

http://terms.tta.or.kr/dictionary/dictionaryView.do?word_seq=166874-3