Design Specification

소프트웨어공학개론 5조

제출일	22.05.15	그룹	5조
과목	소프트웨어공학개론	담당교수	이은석 교수님
이름	김영현	학번	2016311033
이름	김진성	학번	2016311902
이름	안주현	학번	2014312840
이름	이규민	학번	2017314833
이름	이동현	학번	2017311710
이름	정지연	학번	2018311543

목차

1. Preface

1.1 Objective

Preface에서는 본 문서의 독자층을 정의하고 문서가 담는 내용과 각 내용에 대한 자세한 설명을 기술한다. 각 문서의 목차마다 objective를 기술한다.

1.2 Readership

본 문서의 독자층은 개/고양이 classifier 코딩 교육 시스템을 개발 및 유지보수하는 소프트웨어 엔지니어, 시스템의 전반적인 구조를 설계하는 시스템 아키텍터, 사용자의 편의성을 위한 지원팀 등으로 상정한다.

1.3 Document structure

A. Preface

Preface에서는 본 문서의 독자층을 정의하고 문서가 담는 내용과 각 내용에 대한 자세한 설명을 기술한다. 각 문서의 목차마다 objective를 기술한다.

B. Introduction

Introduction에서는 시스템의 설계에 사용한 UML(Unified Modeling Language)와 데이터베이스의 구조를 시각화하기 위해 사용한 개발 툴을 소개한다.

C. System Architecture

System Architecture에서는 전반적인 시스템의 구조에 대하여 시각적인 자료를 이용하여 자세하게 설명한다. 전체 시스템의 구조는 Block diagram을 이용하여 표현한다. 서브 시스템의 상호작용관계와 실제 배포 형태는 Deployment Diagrm을 이용하여 표현한다.

D. Web Server System

Web Server Ssytem의 구조를 표현하기 위해 Class diagram, Sequence diagram, 그리고 State diagram을 이용하여 시각화하고 상세 내용을 기술한다.

E. Admin Server System

Admin Server Ssytem의 구조를 표현하기 위해 Class diagram, Sequence diagram, 그리고 State diagram을 이용하여 시각화하고 상세 내용을 기술한다.

F. Grading Server System

Grading Server System 은 주어진 문제를 관리 및 채점하며 학습을 진행하는 IDE 시스템이다. Sub-System으로는 Problem, Judge, Test 3개로 나뉘어진다. 이러한 Grading Server System을 Class Diagram, Sequence Diagram, State Diagram을 이용하여 시각화한다.

G. Protocol Design

Protocol Design에서는 sub-system간의 상호작용 과정에서 필수적으로 준수해야하는 프로토콜에 관해 설명하며, sub-system간의 통신 과정에서 전달되는 메시지의 형식 및 용도를 설명한다.

H. Database Design

본 시스템의 데이터베이스를 설계한다. ER Diagram을 사용하여 객체-데이터 관계를 표현하며 Relational Schema를 표를 이용하여 시각화한다.

I. Testing Plan

Testing Plan에서는 요구사항 명세서에서 기술한, 사용자 및 어드민의 시나리오를 바탕으로 전체 시스템이 그에 맞춰 잘 실행되는지 확인한다. 그리고 또한 테스팅의 과정을 통해 시스템의 실행 도중 발생할 수 있는 오류들을 미리 찾아낸다. 이러한 테스트를 잘 수행할 수 있는 Testing Policy

를 아래에서 기술한다. 각 Plan에서는 Testing Policy와 각 test case를 설명한다.

J. Development Environment

Development Environment에서는 서비스 개발을 위해 필요한 시스템 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. Reference

문서 작성에 참고한 참고 문헌 목록을 기술한다.

1.4 Version of document

A. Version Format

버전 넘버는 Major.minor[.maintenance] 포맷으로 표현한다. 본 문서는 버전 0.1부터 시작한다.

B. Version Management Policy

본 명세서에 수정 사항이 생길 경우 버전을 업데이트한다. 업데이트 간격은 1시간으로 설정하여

1시간 이내에 발생한 모든 추가사항에 대하여 같은 버전으로 간주한다. 새로운 부분이 추가되거나 문서의 구조가 바뀌는 커다란 변화에 대하여 Major 번호를 변경한다. 이미 작성된 부분에 대한 변화가 생길 경우 minor 번호를 변경한다. 도표나 표, 그림과 같은 자료에 대한 수정이나 오타에 대한 수정이 생기는 경우 maintenance 번호를 변경한다.

C. Version Update History

버전 업데이트에 대한 기록을 표로 나타낸다.

0.0	2022-05-08	Preface, Introduction 추가
1.0	2022-05-09	System Architecture 추가
2.0	2022-05-11	Sub-system, Protocol Design, Database Design 추가
3.0	2022-05-12	Testing Plan, Development Environment, Development Plan 추가
4.0	2022-05-13	Index, Reference 추가
4.0.1	2022-05-14	Sub-system Diagram 수정
4.0.2	2022-05-14	Index 순서 수정, Reference 추가

2. Introduction

2.1 Objective

Introduction에서는 시스템의 설계에 사용한 UML(Unified Modeling Language)와 데이터베이스의 구조를 시각화하기 위해 사용한 개발 툴을 소개한다.

2.2 Applied Diagram

A. UML

Figure 1 Logo of UML

UML(Unified Modeling Language, 통합 모델링 언어)이란 소프트웨어공학에서 사용하는 표준화된 범용 모델링 언어이다. 이 표준은 UML을 고안한 객체 관리 그룹, OMG(Object Management Group, 객체 관리 그룹)에서 하고 있다. UML은 소프트웨어 집약 시스템의 시각적 모델을 만들기 위한 도안 표기법을 포함한다.

UML은 객체 지향 프로그래밍 소프트웨어 집약 시스템을 개발할 때 산출물을 명세화, 시각화, 문서화할 때 사용한다. UML은 데이터 모델링(개체-관계 모델)과 비즈니스 모델링(업무 흐름), 객체 모델링, 부품 모델링의 최선의 기술을 조합한다.

UML을 기반으로 하는 UML Diagram이 현재 13종류가 존재하며, 이중 본 문서에서는 sub-system을 묘사할때 Deployment Diagram, Class Diagram, Sequence Diagram, State Diagram 4가지를 사용하며, 데이터베이스를 묘사할때는 ER Diagram을 사용한다.

B. Deployment Diagram

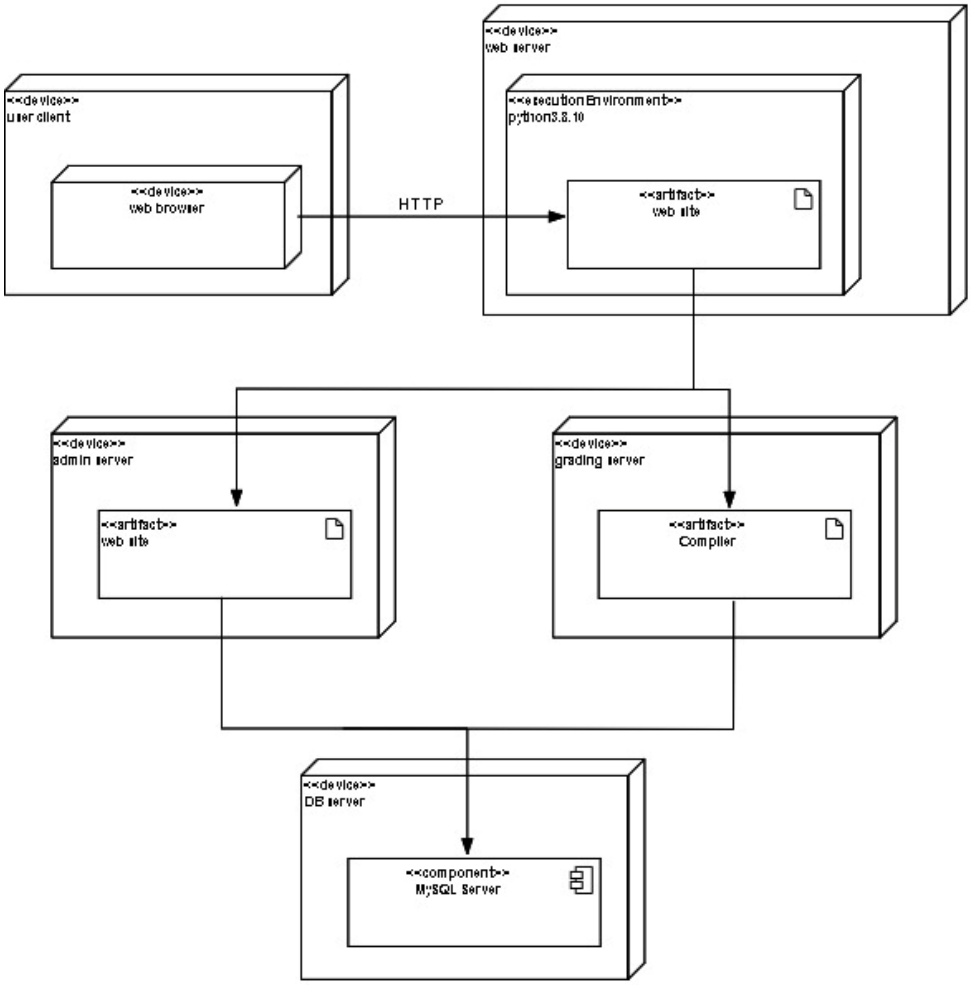


Figure 2 Deployment Diagram example

Deployment diagram은 시스템을 구성하는 하드웨어 자원 간의 연결 관계를 표현하고, 각 하드웨어 자원에 대해 소프트웨어 컴포넌트의 배치 상태를 표현한다. 시스템을 구성하는 소프트웨어적 요소가 위치하는 하드웨어 자원을 정의한다. Deployment diagram의 구성요소로는 Node, Component, Artifact 그리고 관계 등이 있다.

C. Class Diagram

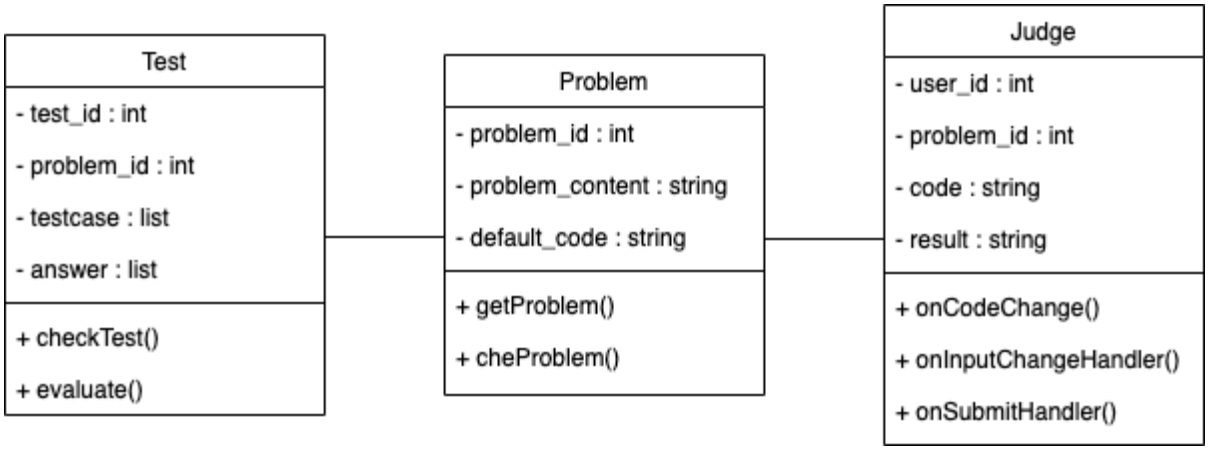


Figure 3 Class Diagram example

Class Diagram은 객체 지향 설계에서 사용하는 기본적인 Diagram이다. 시스템의 정적인 구조를 설명하며 논리적으로 행하는 기능들에 대해 묘사한다. 하나의 클래스는 이름(name), 속성(Attribute), 기능이나 함수(method, operation)을 가지고 있으며 위의 예시처럼 직사각형으로 표현되어 위에서부터 이름, 속성, 함수를 나타낸다. 각 클래스간의 상속 관계와 상호 연결 관계등을 표현한다.

D. State Diagram

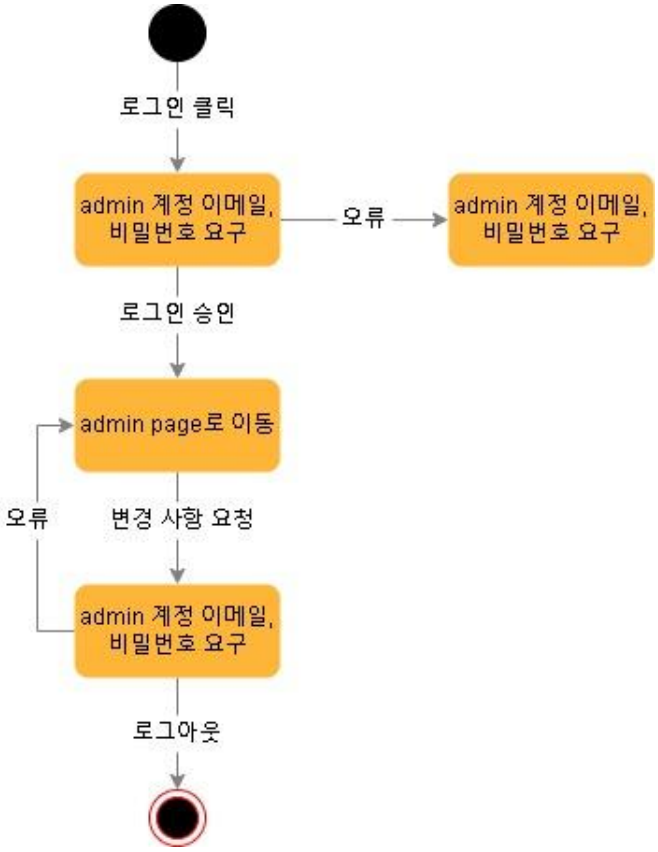


Figure 4 State Diagram example

State Diagram은 시스템의 동작을 설명하는데 사용되는 다이어그램으로, 유한한 수의 상태에 대해 시스템의 외부 자극이나 변화에 따라 system의 행동을 표현한다. 각 객체 혹은 클래스의 동적인 기능과 흐름, 상태를 표현한다. 각 다이어그램은 대게 하나의 클래스를 표현하며 시스템을 통해 해당 클래스의 여러 상태를 파악 및 추적할 수 있다.

E. Sequence Diagram

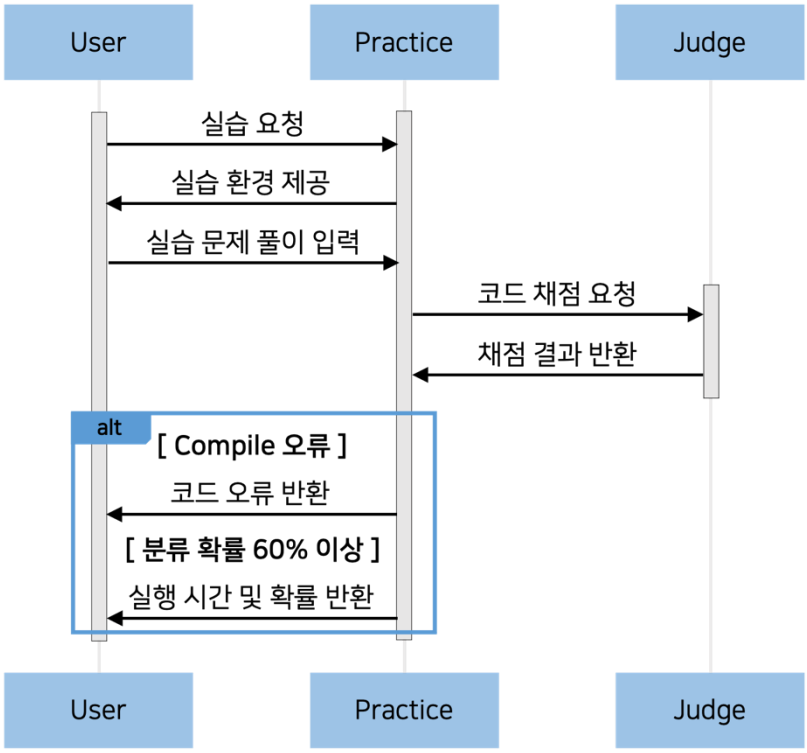


Figure 5 Sequence Diagram example

Sequence Diagram은 시간 순서로 정렬된 객체 상호작용을 보여준다. 시나리오 기능을 수행하는데 필수적인 객체들 간에 교환되는 일련의 메시지들과 시나리오에 수반되는 객체와 클래스를 표현한다. Sequence Diagram은 일반적으로 개발 중인 시스템의 논리적 뷰의 use case 실현화와 관련된다.

F. ER Diagram

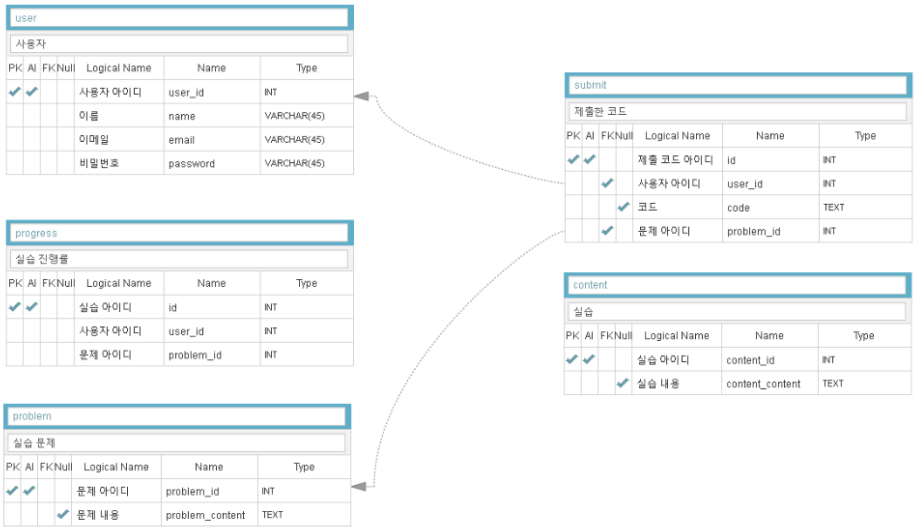


Figure 6 ER Diagram example

ER Diagram이란 ER-Modeling(Entity-Relationship Modeling)의 산출물로 구조화된 데이터를 저장하기 위한 제약조건과 구조, 엔티티와 관계등을 표현한다.

본문에서는 데이터베이스에 저장되어야 하는 정보의 타입과 제약사항을 기술한다. 객체(Entity)는 사각형의 표형태로 표현하며 각 표에 객체가 갖는 속성(Attribute)가 기술된다. 이때 객체를 고유하게 식별할 수 있게끔 유일성과 최소성을 만족하는 속성의 집합을 기본 키(Primary Key, PK)라 한다.

2.3 Applied Tool

A. Visual Studio Code

Figure 7 Logo of Visual Studio Code

Visual Studio Code는 마이크로소프트에서 개발한 텍스트 에디터로, Electron 프레임워크를 기반으로 만들어졌으며 크로스 플랫폼을 지원하는 에디터이다. Windows, macOS, Linux를 모두 지원한다. 프론트엔드의 개발에 사용하였다.

B. Visual Studio

Figure 8 Logo of Visual Studio

Visual Studio는 마이크로소프트 윈도우, macOS에서 작동하며, 다양한 언어로 프로그래밍할 수 있는 마이크로소프트의 통합 개발 환경이다. 프로그램, 웹 사이트, 웹 프로그램 등을 개발할 수 있

다.

C. Visual Paradigm

Figure 9 Logo of Visual Paradigm

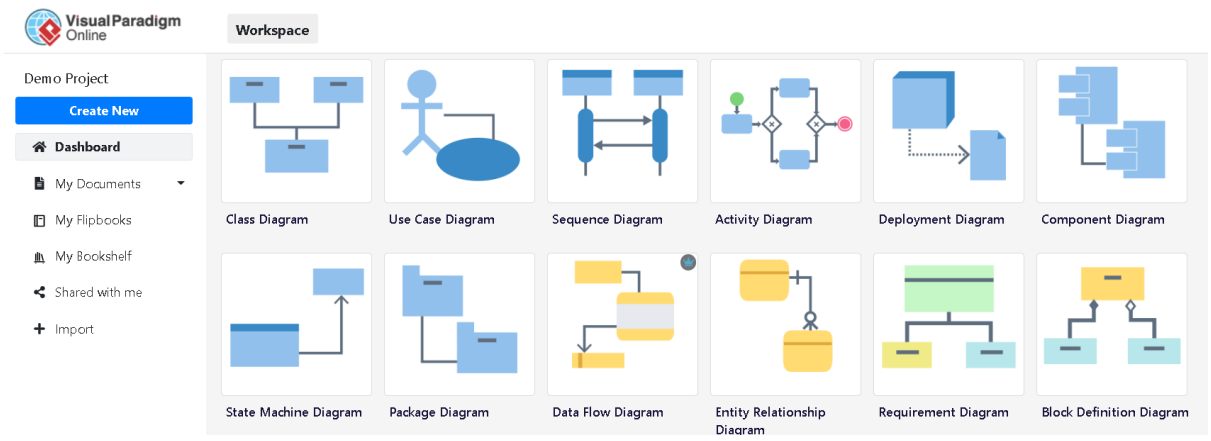


Figure 10 Interface of Visual Paradigm

VIsual Paradigm은 웹 기반의 UML CASE Tool을 지원하는 서비스로, 13가지 UML Diagram을 작성할 수 있으며, 간단한 코드를 통해 Diagram을 생성하는 기능도 가지고 있다.

D. Draw.io

Figure 11 Logo of draw.io

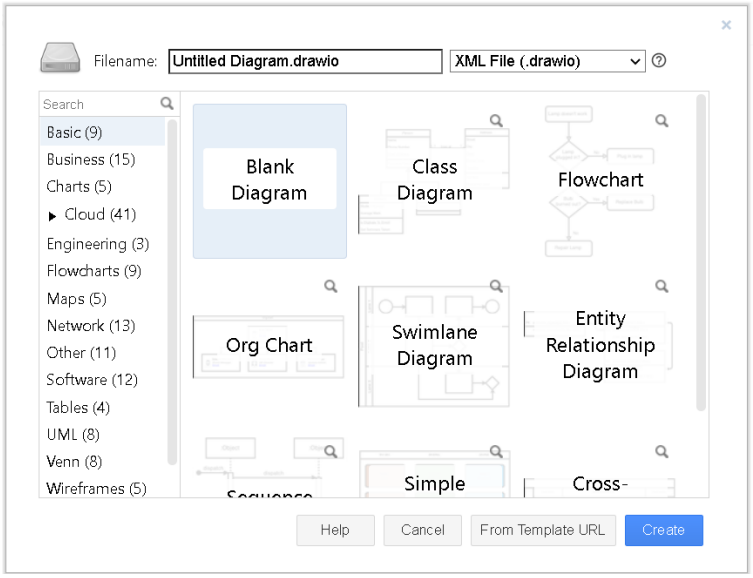


Figure 12 Interface of draw.io

Draw.io는 오픈소스 기반 cross-platform으로 그래프와 다이어그램을 쉽게 작성할 수 있는 웹-기반 서비스이다. Draw.io는 HTML5와 JavaScript로 개발되었으며, flowchart, wireframes, UML diagram 등을 작성할 수 있다. Draw.io는 web application으로도 배포가 되어있으며, Windows, macOS, Linux를 지원한다.

E. AQuery Tool

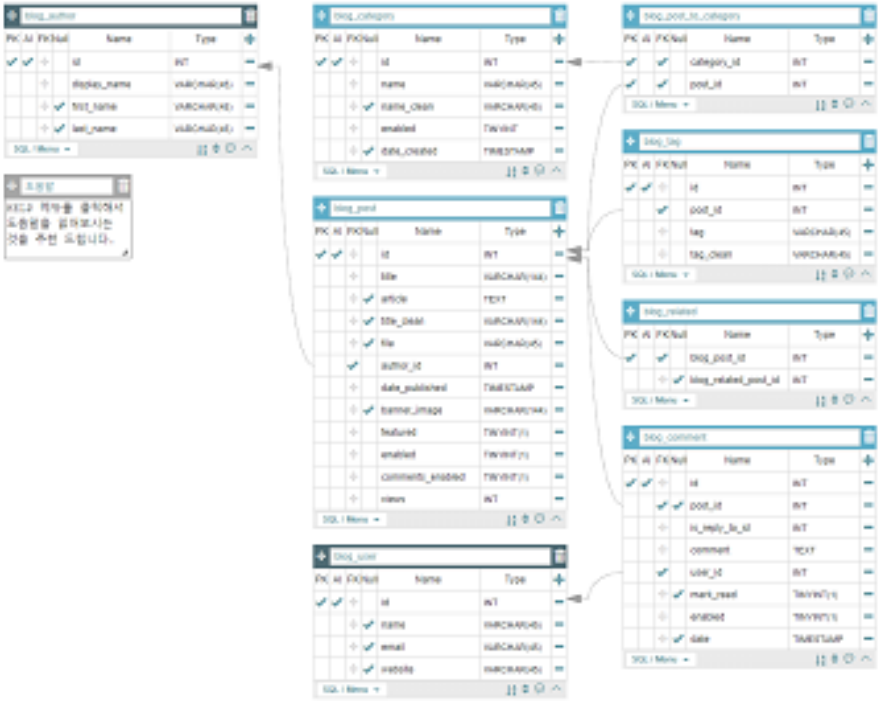


Figure 13 AQuery Tool Interface

AQeury Tool은 웹기반의 ERD (Entitiy-Relation Diagram) 툴로 SQL을 자동으로 생성해주는 프로그램이다. 본 문서의 ER diagram을 작성할 때 사용하였다.

2.4 Project Scope

A. Overview

본 문에서 소개하는 시스템은 확산되는 소프트웨어 교육과 인공지능의 가능성에 대한 높은 관심을 토대로 머신러닝 학습을 필요로 하는 학습자에게 실습을 통한 코딩 역량과 관련 지식에 대한 습득을 용이하게 해준다. 이 시스템은 크게 Web Server System, Admin Server System, Grading Server System으로 구성되어 있다.

B. Web Server System

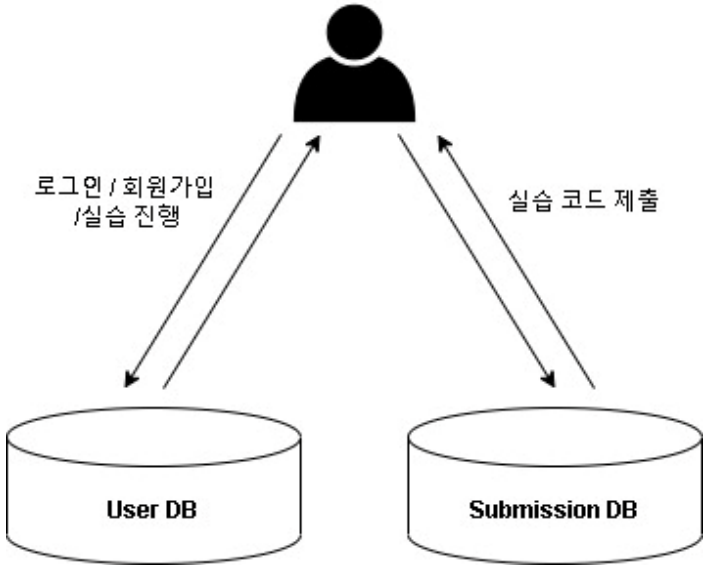


Diagram 1 Product Scope : Web Server System

Web Server System은 Web-based 시스템으로 Web Browser를 통한 사용자의 접속을 장려한다. 사용자에게 user interface를 제공하며 사용자의 프로필, 사용자 정보와 학습 컨텐츠의 목차와 실습을 다루는 sub-system으로 이루어져있다. 사용자의 회원가입, 로그인, 학습 진행에 대해 User DB와 상호작용한다. 사용자가 코드를 제출할때 Submission DB와 상호작용한다.

C. Admin Server System

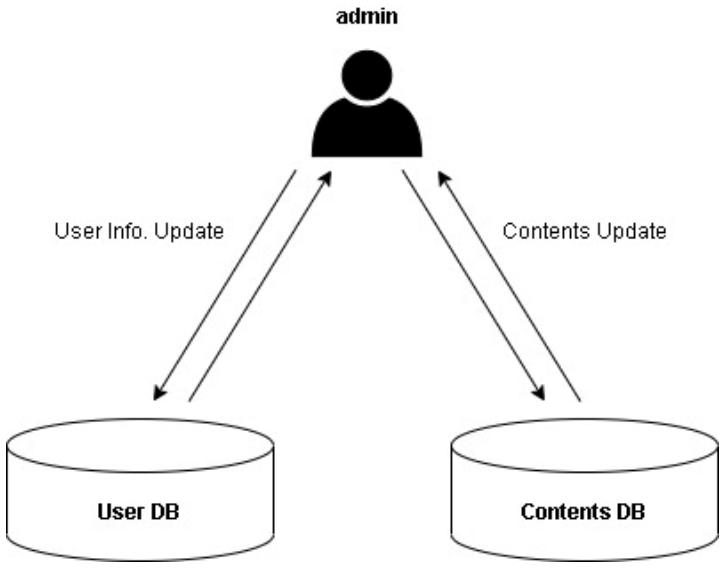


Diagram 2 Product Scope : Admin Server System

Admin Server System은 사용자에게 제공하는 contents와 사용자의 정보에 접근하여 수정사항이 필요한 경우에 한해서 데이터를 변경할 수 있다. 사용자의 정보에 업데이트가 필요한 경우 User DB와 상호작용한다. 실습 컨텐츠에 업데이트가 필요한 경우 Contents DB와 상호작용한다.

D. Grading Server System

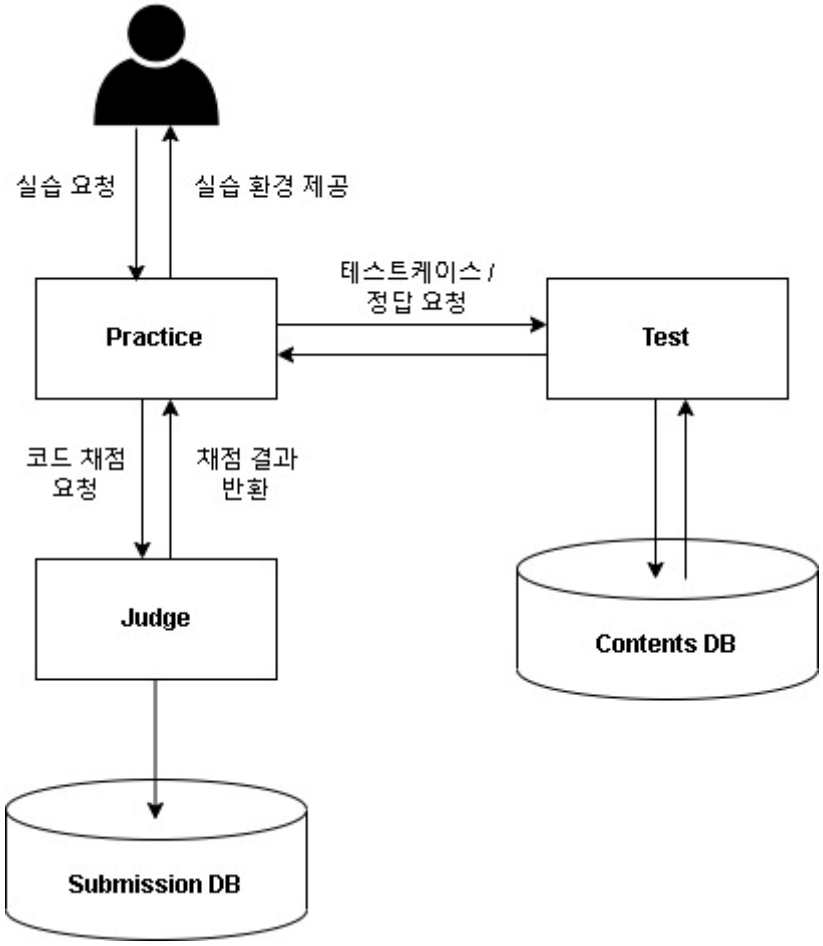


Diagram 3 Product Scope : Grading Server System

Grading Server System은 주어진 문제를 관리 및 채점하여 학습을 진행하는 IDE 시스템이다. Practice, Judge, Test 3개의 sub-system을 가지고 있다. Practice sub-system은 사용자에게 학습에 대한 문제를 제공하며, 사용자가 실습 코드를 제출하면 Judge sub-system으로 코드를 보내 채점 결과를 반환한다. Judge sub-system은 입력받은 사용자의 실습 코드에 대하여 Contents DB에 저장되어 있는 정답 데이터를 통해 채점을 진행하며, 채점 결과를 반환한다. Test System은 사용자가 학습하는 목차의 인덱스와 실습 문제에 대한 id, 테스트케이스, 정답을 Contents DB로 부터 받아오는 역할을 한다.

3. System Architecture

3.1 Objective

System Architecture에서는 전반적인 시스템의 구조에 대하여 시각적인 자료를 이용하여 자세하게 설명한다. 전체 시스템의 구조는 Block diagram을 이용하여 표현한다. 서브 시스템의 상호작용관계와 실제 배포 형태는 Block diagram과 Deployment Diagram을 이용하여 표현한다.

3.2 System Organization

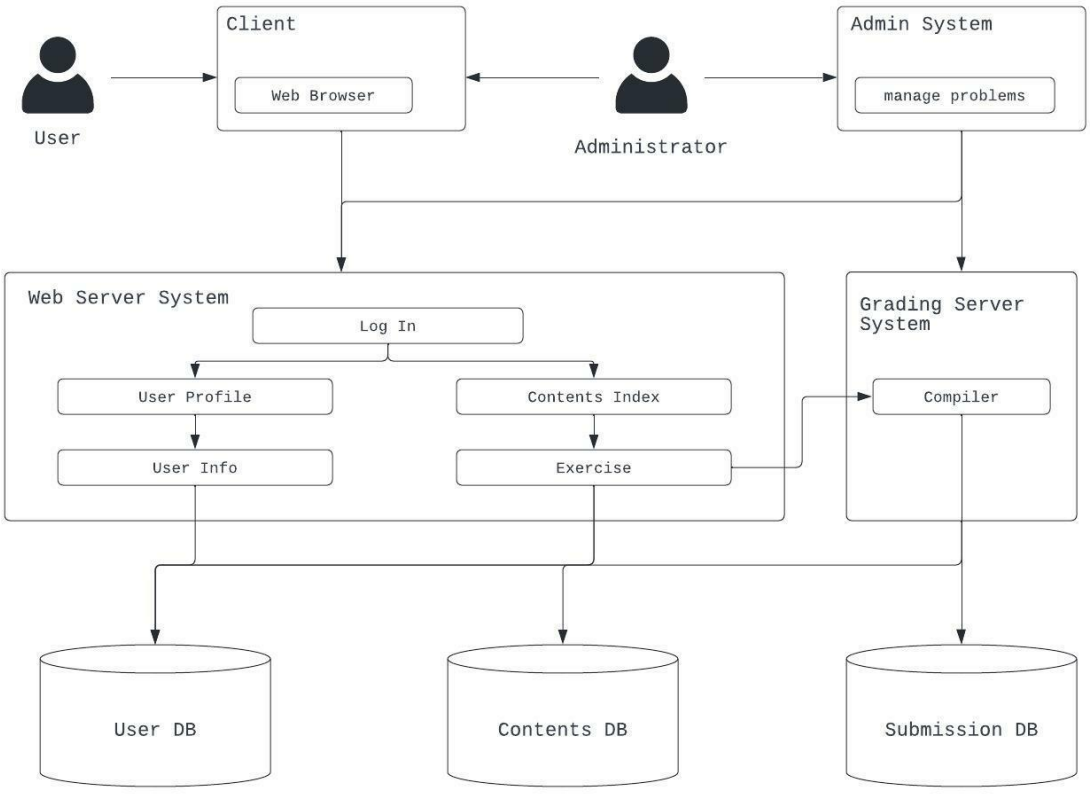


Diagram 4 전체 System Architecture

본 시스템은 Web-based 시스템으로 Web Browser을 이용하여 접속한다. Web Server System, Admin Server System, Grading Server System으로 구성된다. Web Server System은 Web의 모든 interface를 총괄한다. Admin Server System은 실습 컨텐츠에 대한 management를 담당한다. Grading Server System은 사용자의 실습 내용에 대한 데이터를 관리한다.

A. Web Server System

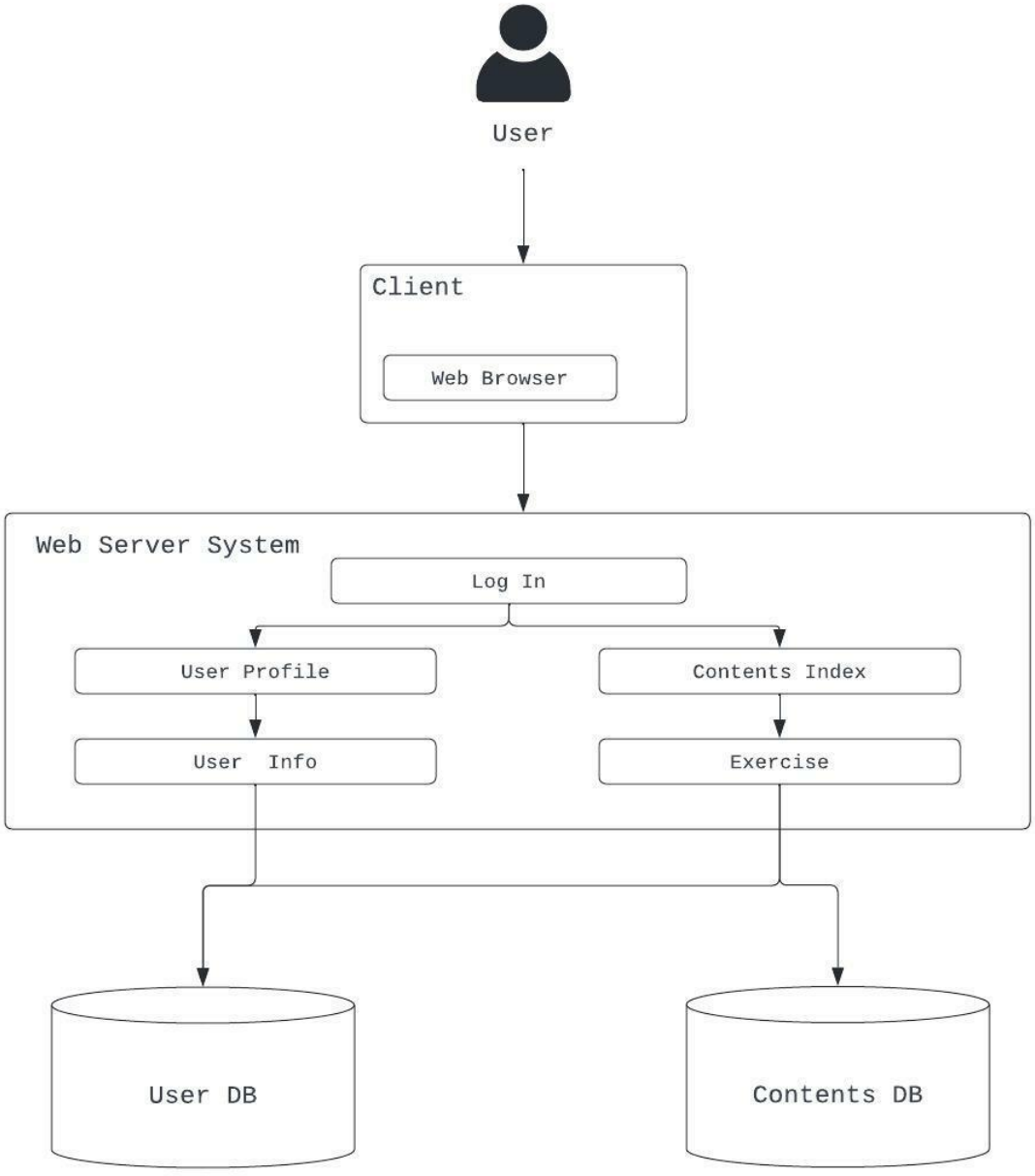


Diagram 5 System Architecture : Web Server System

Web Server는 User에게 User Interface를 제공하며 Log-In interface, User Profile interface, User Info interface, Contents Index interface, Exercise interface를 가진다. 각 interface는 로그인, 사용자 프로필, 사용자 정보, 학습 컨텐츠 목차와 실습을 담당하며 각 페이지에서 발생한 event의 데이터는 해당 database에 저장한다.

B. Admin Server System

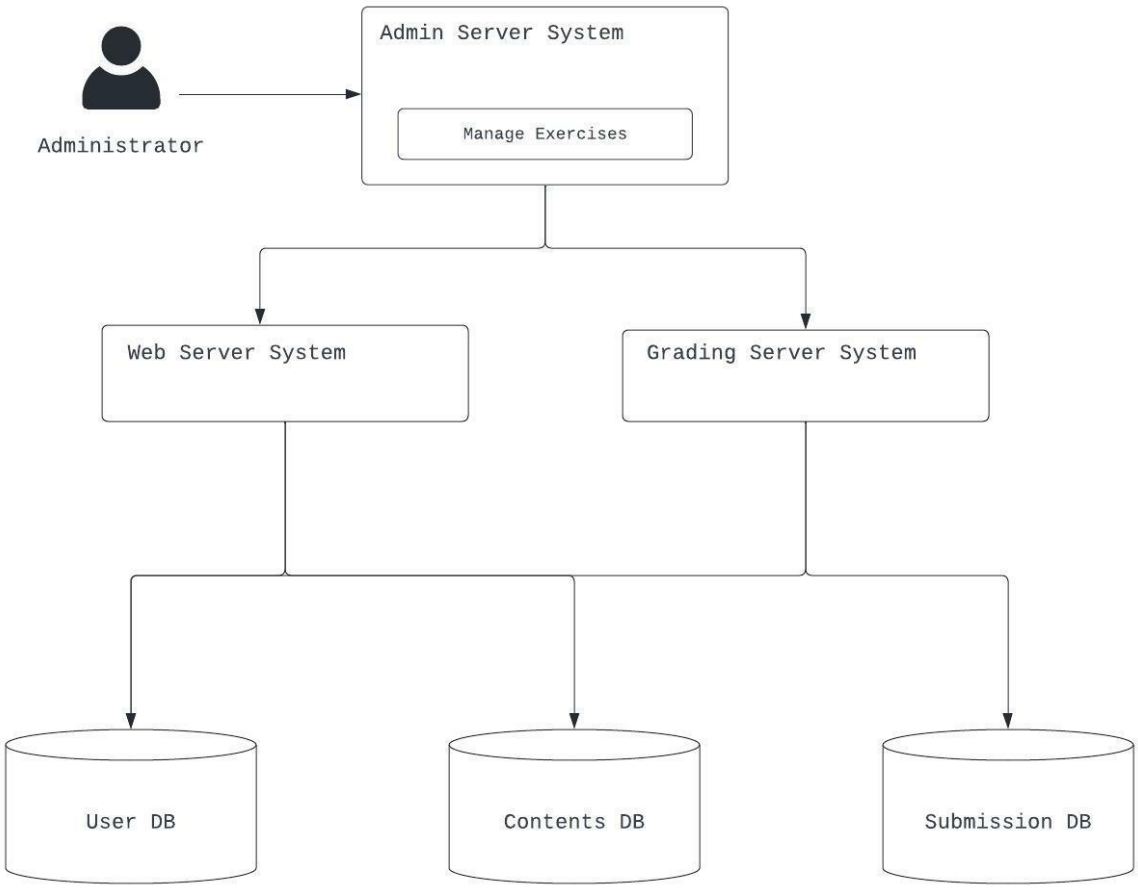


Diagram 6 System Architecture : Admin Server System

Admin Server system은 Web server와 Grading Server System과 상호운용하며 하나의 컨텐츠에 대한 추가/삭제/업데이트 및 수정을 담당한다. 관리자의 권한으로 접속하여 동작하며, 업데이트 된 컨텐츠의 내용에 대해 database에 접속하여 전체적인 실습 컨텐츠를 관리한다.

Web Server에서 추가할 실습이 존재하는 경우 관리자 권한으로 Admin Server에 접속한 뒤 실습을 추가한다. 그 후 Contents DB에 수정한 데이터를 저장하고, User DB에서 사용자의 학습 진도와 관련된 데이터를 수정한다.

Grading Server에 추가된 실습의 정답과 test case에 대한 데이터의 접근 경로를 생성하여 Grading Server와 Contents DB와의 데이터 유동 환경을 제공한다.

C. Grading Server System

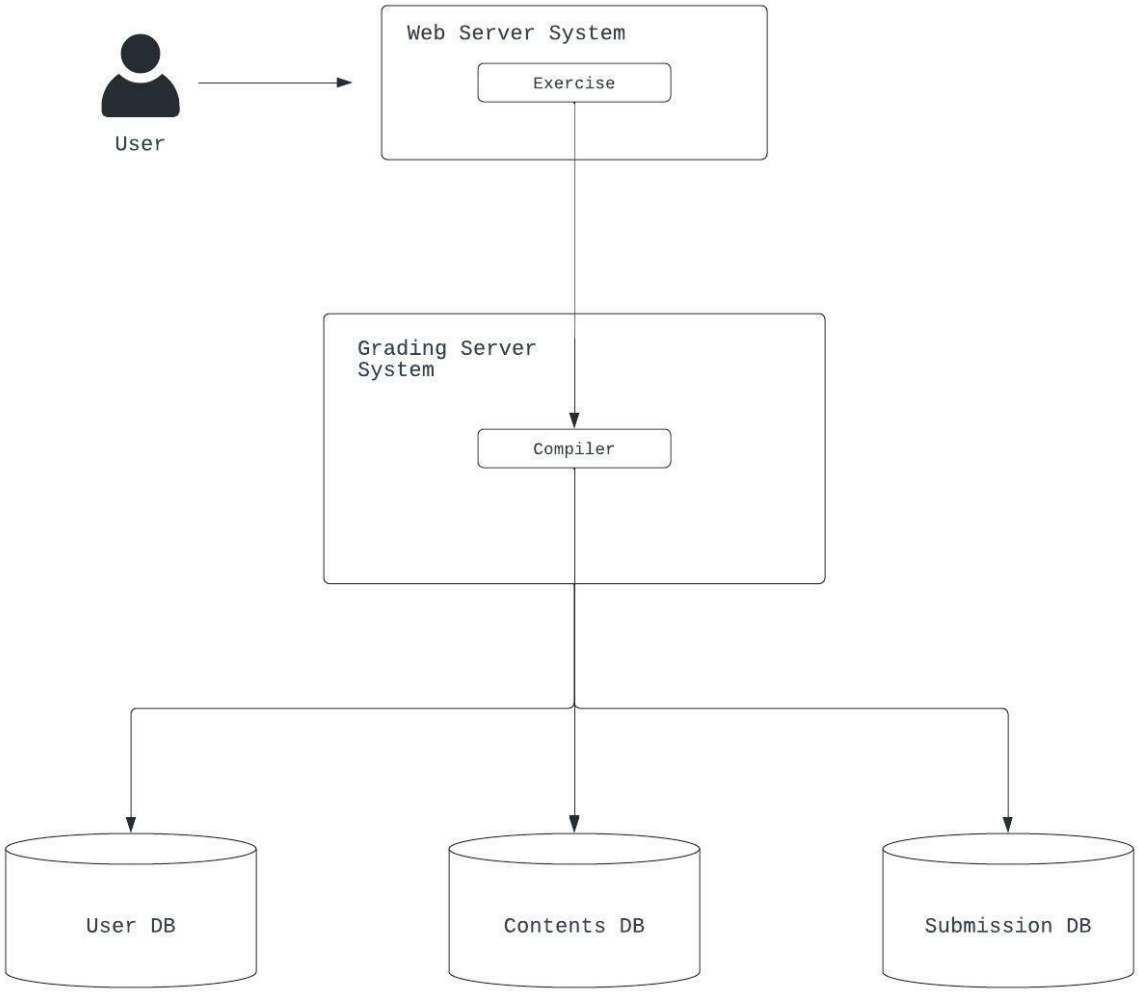


Diagram 7 System Architecture : Grading Server System

Grading Server System은 Web Server에서 제출한 사용자의 실습 코드에 대해 Compile을 진행하며 해당 코드의 정확도를 판단한다. Contents DB로부터 해당 실습에 대한 test case를 받아오며 사용자의 제출 코드에 대해 test를 진행하여 정답률을 계산한다. 이후 User DB와 Submission DB에 각각 사용자의 최근 실습 log와 제출한 코드를 저장한다.

3.3 Deployment Diagram

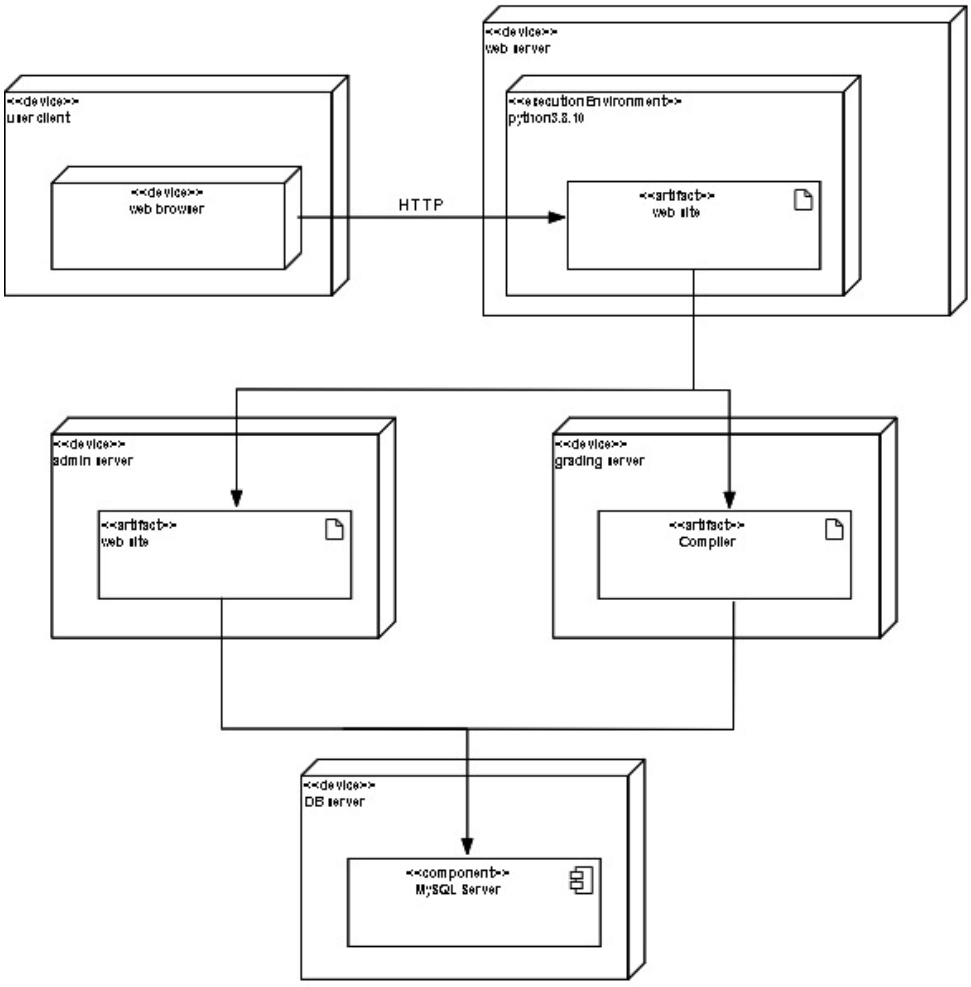


Diagram 8 Deployment Diagram

4. Web Server System

4.1 Objective

Web Server System의 구조를 표현하기 위해 Class diagram, Sequence diagram, 그리고 State diagram을 이용하여 시각화하고 상세 내용을 기술한다.

4.2 Class diagram

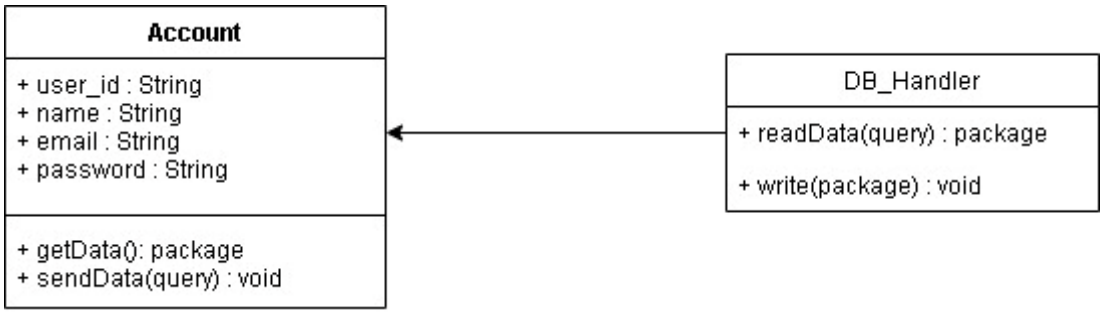


Diagram 9 Web Server System : Class Diagram

A. DB_Handler

A.1 Methods

+ package read(query) : query가 요구한 data를 데이터베이스에 접근하여 읽어온다.

+ void write(package) : 해당 데이터를 데이터베이스에 저장한다.

B. Account

B.1 Attributes

+ user_id : 해당 계정 id

+ name : 해당 계정의 이름

+ email : 해당 계정의 이메일

+ password : 해당 계정의 비밀번호

B.2 Methods

+ package getData() : 데이터베이스에서 데이터를 받아온다.

+ void sendData(query) : 데이터베이스에 데이터를 전송한다.

4.3 Sequence Diagram

A. 회원가입 및 로그인

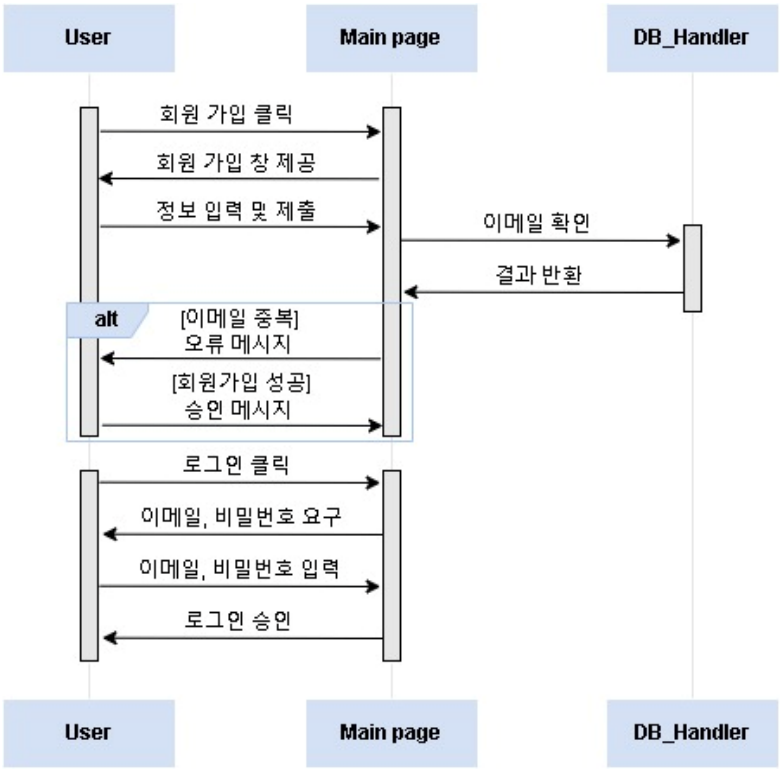


Diagram 10 Web Server System : Sequence Diagram

4.4 State Diagram

A. 회원가입

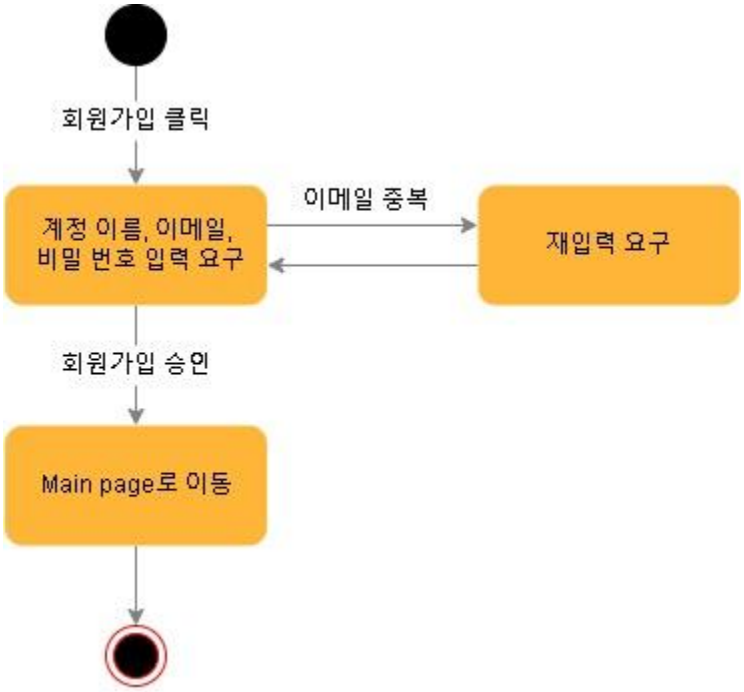


Diagram 11 Web Server System : State Diagram 회원가입

B. 로그인

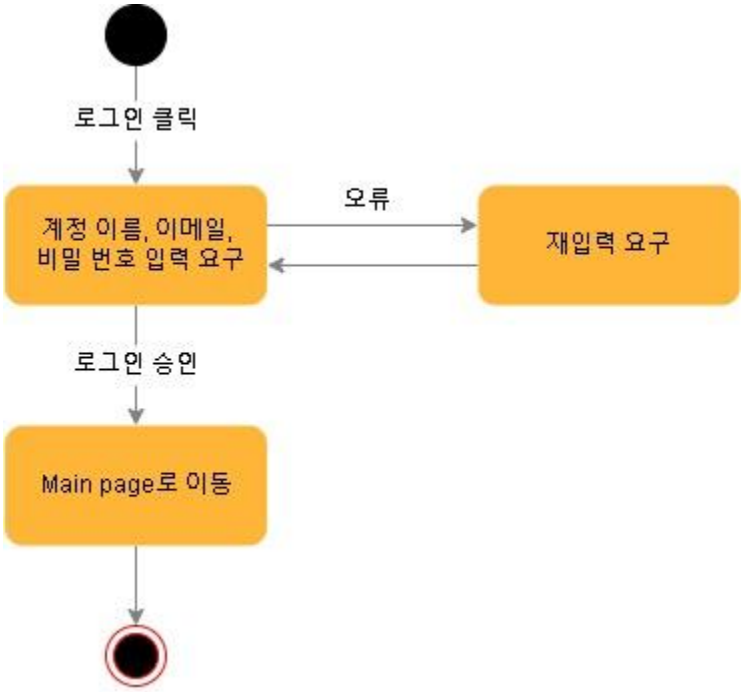


Diagram 12 Web Server System : State Diagram 로그인

5. Admin Server System

5.1 Objective

Admin Server System의 구조를 표현하기 위해 Class diagram, Sequence diagram, 그리고 State diagram을 이용하여 시각화하고 상세 내용을 기술한다.

5.2 Class Diagram

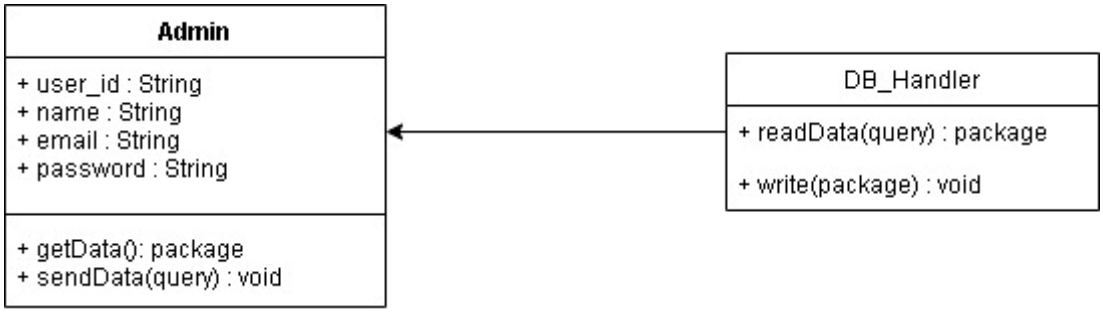


Diagram 13 Admin Server System : Class Diagram

A. DB_Handler

A.1 Methods

+ package read(query) : query가 요구한 data를 데이터베이스에 접근하여 읽어온다.

+ void write(package) : 해당 데이터를 데이터베이스에 저장한다.

B. Admin

B.1 Attributes

+ user_id : 해당 계정 id

+ name : 해당 계정의 이름

+ email : 해당 계정의 이메일

+ password : 해당 계정의 비밀번호

B.2 Methods

+ package getData() : 데이터베이스에서 데이터를 받아온다.

+ void sendData(query) : 데이터베이스에 데이터를 전송한다.

5.3 Sequence Diagram

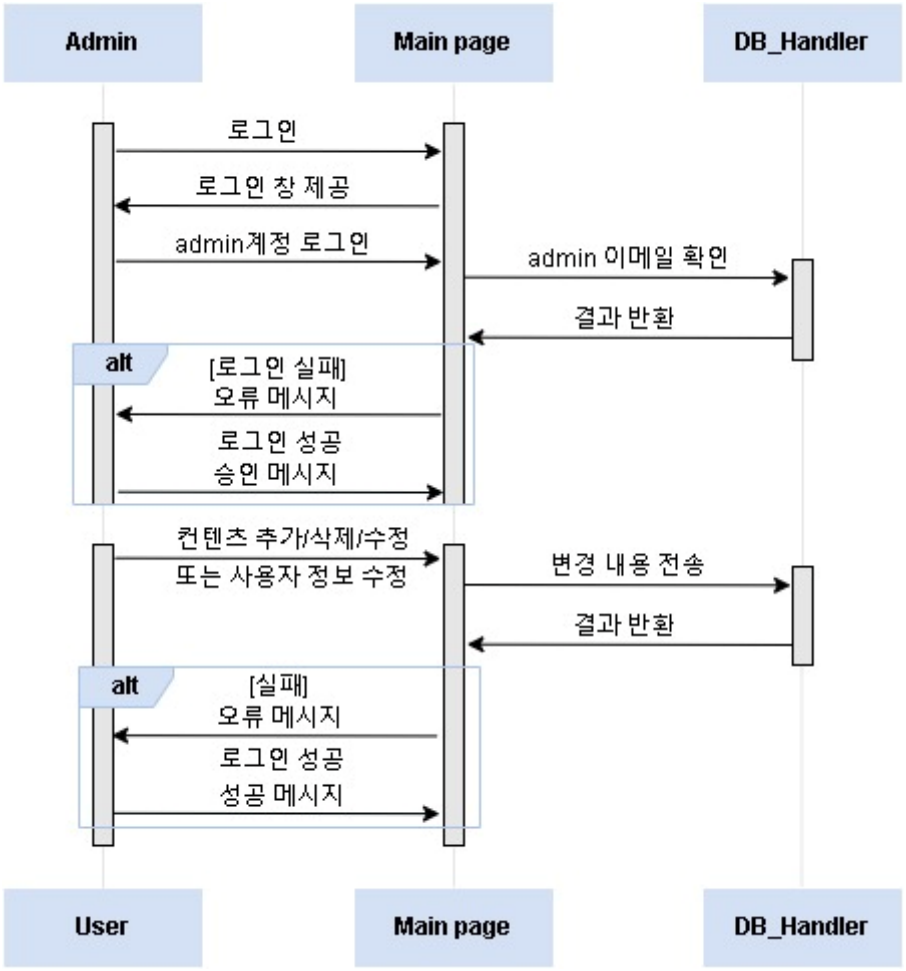


Diagram 14 Admin Server System : Sequence Diagram

5.4 State Diagram

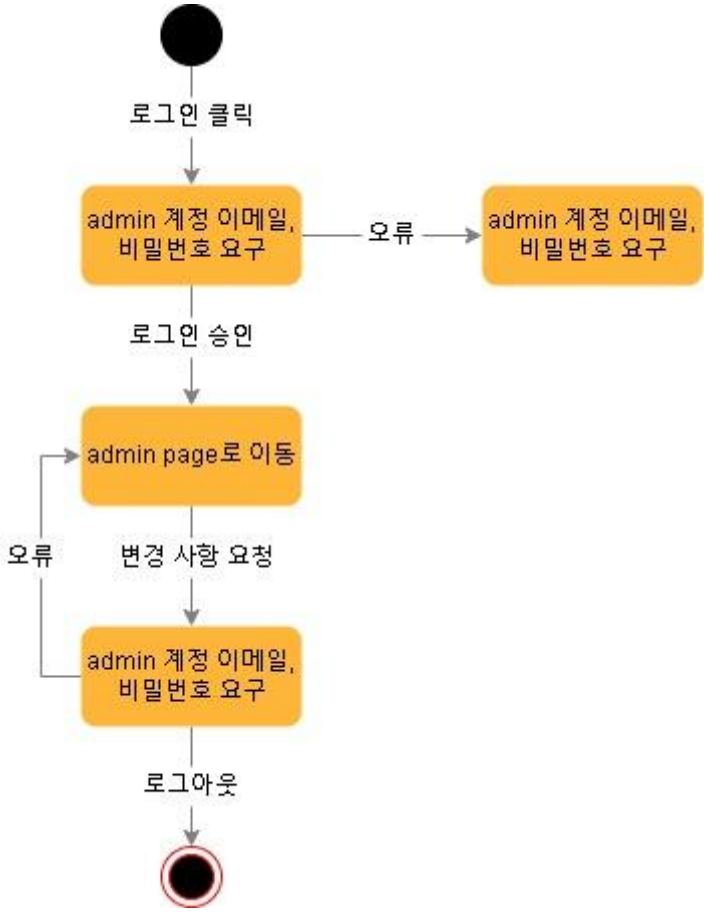


Diagram 15 Admin Server System : State Diagram

6. Grading Server System

6.1 Objective

Grading Server System 은 주어진 문제를 관리 및 채점하며 학습을 진행하는 IDE 시스템이다. Sub-System으로는 Problem, Judge, Test 3개로 나뉘어진다. 이러한 Grading Server System

6.2 Class Diagram

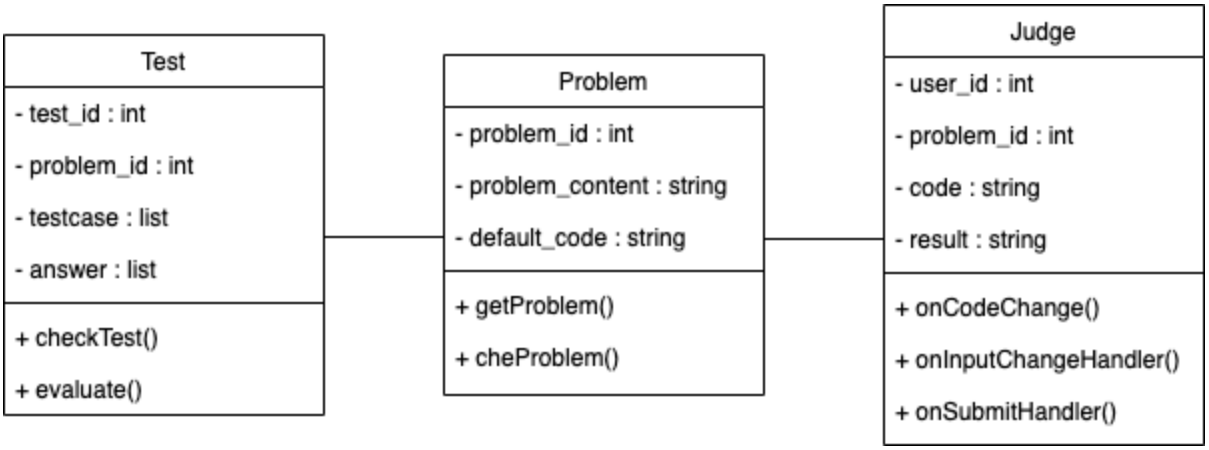


Diagram 16 Grading Server System : Class Diagram

A. Problem

A.1 Attributes

- problem_id : 문제 식별 번호
- problem_content : 문제 설명 내용
- default_code : 문제별 기본 코드

A.2 Method

+ getProblem() : 문제를 가져온다.

+ checkProblem() : 문제 채점 결과를 반환한다.

B. Test

B.1 Attributes

- test_id : 테스트 식별번호

- problem_id : 문제 식별 번호 FK
- testcase : 문제에 맞는 테스트 케이스
- answer : 테스트 케이스 별 정답

B.2 Method

+ checkTest() : 테스트케이스를 돌려보며 확인한다.
+ evaluate() : 테스트케이스 별 정답을 평가한다.

C. Judge

C.1 Attributes

- user_id : 사용자 식별 번호 FK
- problem_id : 문제 식별 번호 FK
- code : 제출한 코드
- result : 코드 실행 결과

C.2 Method

+ onCodeChange() : 사용자가 온라인 IDE에 쓰는 코드 변화를 감지한다.

+ onInputChangeHandler() : 온라인 IDE의 input 값을 감지한다.

+ onSubmitHandler() : 코드를 서버로 보내 결과를 가져온다.

6.3 Sequence Diagram

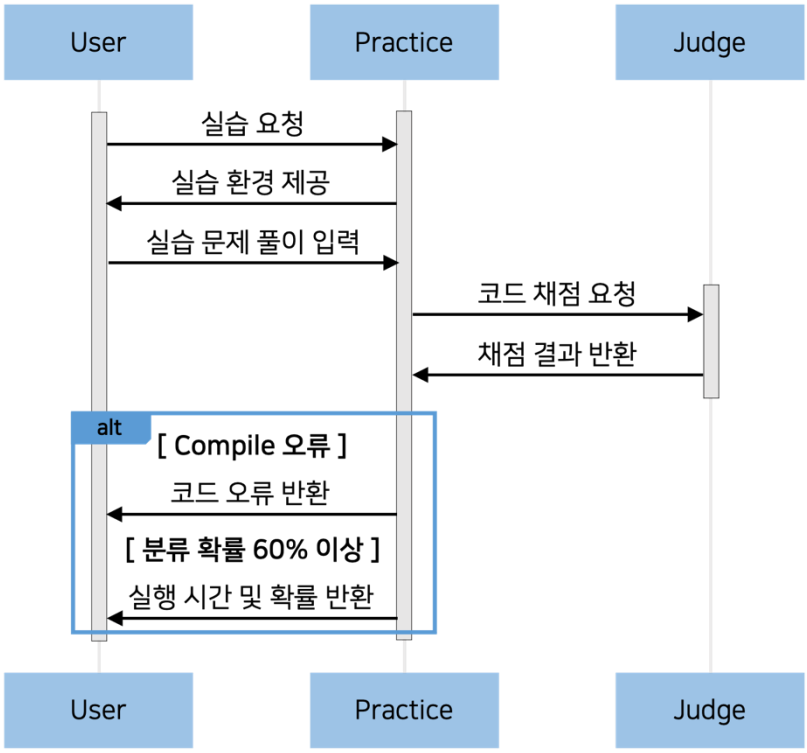


Diagram 17 Grading Server System : Sequence Diagram

6.4 State Diagram

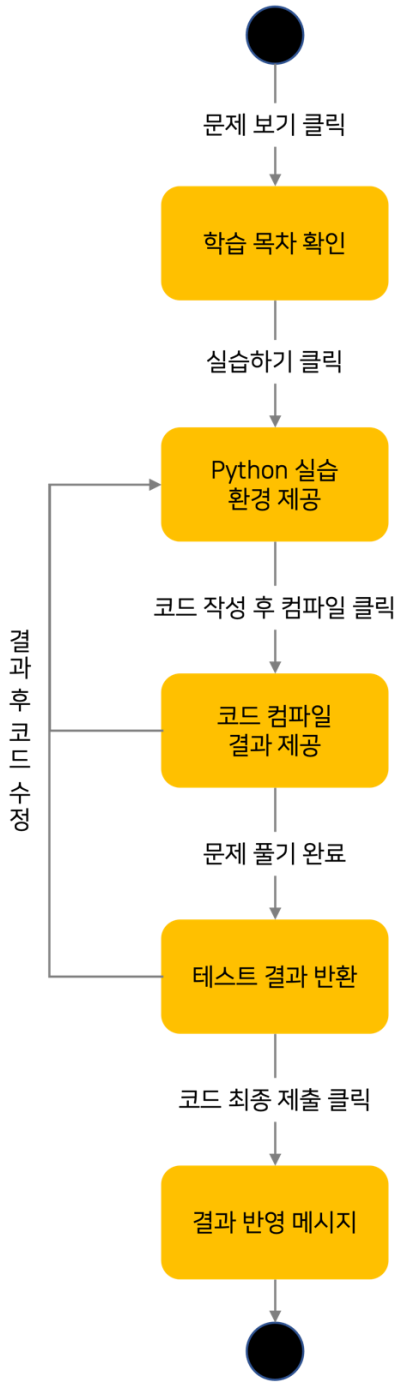


Diagram 18 Grading Server System : Event Driven Diagram

7 Protocol Design

7.1 Overview

Protocol Design에서는 sub-system간의 상호작용 과정에서 필수적으로 준수해야하는 프로토콜에 관해 설명하며, sub-system간의 통신 과정에서 전달되는 메시지의 형식 및 용도를 설명한다.

7.2 JSON

JSON (JavaScript Object Notation)은 속성-값 쌍 또는 키-값 쌍으로 이루어진 데이터오브젝트를 전달하기 위해 인간이 읽을 수 있는 텍스트를 사용하는 개방형 표준포맷이다. 인터넷에서 전달되는 자료를 표현하는 방법으로, 기본 자료형으로는 수, 문자열, 참/거짓, 배열, 객체, null이있다. 자료 종류의 제한이 없으며 프로그래밍 언어 및 플랫폼 독립적인 언어 독립형 데이터 포맷이다.

7.3 Protocol Description

A. Overview

Client와 Server간의 요청 및 응답 메시지를 프로토콜의 구현 기능별로 나누어 설명한다. 해당 절의 표는 캡션을 생략한다.

B. Login Protocol

GET	/api/user		
사용자가 로그인할 때 요청하는 주소			
분류	항목명	설명 및 제약사항	예시
Parameter	Email	사용자 이메일	
	password	사용자 비밀번호	
Response	message	"로그인 실패"	
	status code	상태 코드	
Status Code	1	로그인 성공	
	0	로그인 실패	

Table 1 Login Protocol

C. Registration Protocol

POST	/api/user		
사용자가 회원가입 할 때 요청하는 주소			
분류	항목명	설명 및 제약사항	예시
Parameter	name	사용자 이름	
	email	사용자 이메일	
	password	사용자 비밀번호	
Response	message	"회원가입 실패"	
	status code	상태 코드	
Status Code	1	로그인 성공	
	0	로그인 실패	

Table 2 Registration Protocol

D. Exercise Protocol

GET	/api/problem		
사용자의 실습을 위한 코드를 요청하는 주소			
분류	항목명	설명 및 제약사항	예시
Parameter	problem_id	실습 문제의 id값	
Response	problem_id	실습 문제의 id값	
	problem_content	실습 문제의 코드	
	Status code	상태 코드	
Status Code	200	코드 불러오기 성공	
	400	코드 불러오기 실패	

Table 3 Exercise Protocol

E. Submission Protocol

POST	/api/submit		
사용자가 작성한 실습 코드를 제출하기 위해 요청하는 주소			
분류	항목명	설명 및 제약사항	예시
Parameter	Code	사용자가 작성한 코드	
	Lang	작성한 코드의 언어	

	Input	작성한 코드에 대한 입력	
	Result	작성한 코드의 실행결과	
Response	Result	제출 결과	
	Status code	상태 코드	
Status Code	201	실습 코드 제출 성공	
	400	실습 코드 제출 실패(형식 오류 등)	

Table 4 Submission Protocol

F. Educational Content Protocol

GET	/api/content/		
사용자가 학습할 내용을 요청하는 주소			
분류	항목명	설명 및 제약사항	예시
Parameter	content_id	학습 내용의 id값	
Response	content_id	학습 내용의 id값	
	content_content	학습 내용	
	Status code	상태 코드	
Status Code	200	학습내용 불러오기 성공	
	400	Content_id가 존재하지 않음	

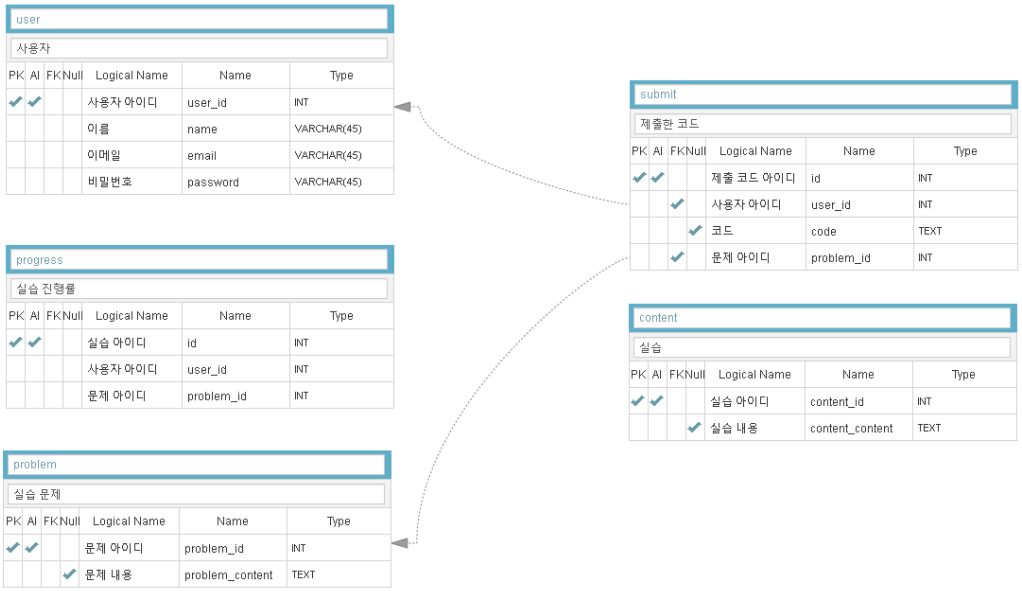
Table 5 Educational Content Protocol

8. Database Design

8.1 Objective

본 시스템의 데이터베이스를 설계한다. ER Diagram을 사용하여 객체-데이터 관계를 표현하며 Relational Schema를 표를 이용하여 시각화한다.

8.2 ER Diagram



8.3 Relational Schema

A. USER

테이블 이름				user				
테이블 설명				사용자 정보				
PRIMARY KEY				email				
FOREIGN KEY								
INDEX								
NO	PK	AI	FK	NULL	칼럼 이름	TYPE	설명	참조 테이블
1					name	VARCHAR(45)	사용자 이름	
2	Y				email	VARCHAR(45)	사용자 이메일(아이디로 사용)	
3					password	VARCHAR(45)	비밀번호	

Table 6 User Relational Schema

B. PROGRESS

테이블 이름	progress
테이블 설명	학습 진행도
PRIMARY KEY	email

FOREIGN KEY								
INDEX								
NO	PK	AI	FK	NULL	칼럼 이름	TYPE	설명	참조 테이블
1	Y				email	VARCHAR(45)	사용자 이메일	
2					problem_id_text	LONGTEXT	학습 진행도	

Table 7 Progress Relational Schema

C. SUBMIT

테이블 이름				submit				
테이블 설명				제출한 코드				
PRIMARY KEY								
FOREIGN KEY								
INDEX								
NO	PK	AI	FK	NULL	칼럼 이름	TYPE	설명	참조 테이블
1	Y	Y			id	INT	제출 아이디	
2			Y		email	VARCHAR(45)	사용자 이메일	
3					code	LONGTEXT	사용자가 제출한 코드	
4			Y		problem_id	INT	문제 아이디	

Table 8 Submit Relational Schema

D. PROBLEM

테이블 이름				problem				
테이블 설명				제출한 코드				
PRIMARY KEY								
FOREIGN KEY								
INDEX								
NO	PK	AI	FK	NULL	칼럼 이름	TYPE	설명	참조 테이블
1	Y	Y			problem_id	INT	문제 아이디	
2					problem_content	LONGTEXT	문제 내용	

Table 9 Problem Relational Schema

E. CONTENT

테이블 이름				content				
테이블 설명				제출한 코드				
PRIMARY KEY								
FOREIGN KEY								
INDEX								
NO	PK	AI	FK	NULL	칼럼 이름	TYPE	설명	참조 테이블
1	Y	Y			content_id	INT	학습 내용 아이디	
2					content_content	LONGTEXT	학습 내용	

Table 10 Content Relational Schema

9. Testing Plan

9.1 Objective

Testing Plan에서는 요구사항 명세서에서 기술한, 사용자 및 어드민의 시나리오를 바탕으로 전체 시스템이 그에 맞춰 잘 실행되는지 확인한다. 그리고 또한 테스팅의 과정을 통해 시스템의 실행 도중 발생할 수 있는 오류들을 미리 찾아낸다. 이러한 테스트를 잘 수행할 수 있는 Testing Policy를 아래에서 기술한다. 각 Plan에서는 Testing Policy와 각 test case를 설명한다.

9.2 Testing Policy

구현하고자 하는 웹서비스의 Testing은 Development testing, Release testing, User testing 세 단계로 구성된다. 그 중 Development testing은 component testing, system testing, acceptance testing의 과정을 통해 순차적으로 진행된다. 아래는 각각의 testing의 정의와 조건들을 서술하고 있다.

A. Developing testing

시스템을 개발하는 과정에서 진행하는 테스트 단계이다. component testing은 기능 객체 등을 뜻하는 개별 component들을 독립적으로 테스트한다. 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, 프로그램을 개발 의도에 맞게 작성했는지, 서비스들의 의도된 사항대로 작동되는지, 정상적으로 기능이 작동하는지 등을 확인한다.

C. User testing

이 단계는 실제 사용 환경에서 시스템을 테스트한다. 실제 사용자들이 테스트를 진행하면서, 이 과정을 통해 product의 내용에 대해 평가하고 product의 출시 여부를 결정한다.

9.3 Test Case

A. User Management System

A.1 회원가입

구성요소	**설명**
테스트 이름	회원가입 테스트
사전 조건	1. 이메일: 중복되어서는 안된다. 2. 비밀번호와 이름은 문자열 형식이다..
수행 절차	1. 이메일 중복 체크 2. 비밀번호 형식 체크 3. 이름 형식 체크

기대되는 결과	회원 가입 후 데이터베이스에 회원 정보 저장 로그인 화면으로 전환

Table 11 Test Case: 회원가입

A.2 로그인

구성요소	**설명**
테스트 이름	로그인 테스트
사전 조건	1. 비밀번호는 문자열 형식이다. 2. 이메일과 비밀번호가 데이터베이스에 저장되어 있어야한다.
수행 절차	1. 비밀번호 형식 체크 2. 이메일 존재여부 체크 3. 이메일과 비밀번호 일치 여부 체크
기대되는 결과	사용자 로그인 상태로 기록, 메인 화면으로 전환

Table 12 Test Case: 로그인

A.3 학습 목차 확인

구성요소	**설명**
테스트 이름	사용자가 학습할 학습 목차 확인
사전 조건	사용자는 로그인한 상태여야 한다.

수행 절차	1. 사용자가 로그인을 진행한다. 2. 로그인 이후 현재 학습할 수 있는 목차가 로드된다. 3. 사용자가 자신이 학습할 내용을 확인할 수 있다.
기대되는 결과	사용자가 자신이 학습할 내용들을 확인할 수 있다.

Table 13 Test Case: 학습 목차 확인

A.4 사용자 프로필 확인

구성요소	**설명**
테스트 이름	마이페이지 푼 문제 확인 테스트
사전 조건	사용자는 로그인한 상태여야 한다.
수행 절차	4. 사용자가 마이페이지 버튼을 누른다. 5. 사용자의 이름을 확인할 수 있다. 6. 사용자가 자신이 학습한 내용을 확인할 수 있다.
기대되는 결과	사용자가 자신이 학습한 내용들을 확인할 수 있다.

Table 14 Test Case: 사용자 프로필 확인

B. Problem System

B.1 학습진행

구성요소	설명
테스트 이름	학습 진행
사전 조건	1. 사용자는 로그인한 상태여야 한다. 2. 사용자의 학습데이터가 저장되어 있어야 한다.
수행 절차	1. 사용자가 학습 페이지를 확인한다. 2. 사용자는 학습 내용에 뜬 내용을 확인한다. 3. 사용자는 학습내용에 적힌 학습을 진행한다.
기대되는 결과	사용자가 학습을 진행할 수 있다.

Table 13 Test Case: 학습진행

B.2 학습테스트

구성요소	설명
테스트 이름	학습 테스트
사전 조건	3. 사용자는 로그인한 상태여야 한다.

수행 절차	4. 사용자가 학습 페이지를 확인한다. 5. 사용자는 실습에 적힌 코드를 따라적는다. 6. 사용자는 작성한 코드를 제출한다. 7. 사용자에게 코드를 제출했음을 알려준다. 8. 사용자는 실습에 적힌 인풋값을 작성한다. 9. 사용자는 인풋값을 제출한다. 10. 사용자의 코드가 디버깅되어 실행 결과가 화면에 리턴된다.
기대되는 결과	사용자의 코드 실행 결과가 화면에 표시되고, 사용자 데이터베이스에 작성한 코드가 저장된다.

Table 16 Test Case: 학습테스트

B.2 실습 문제 풀기

구성요소	**설명**
테스트 이름	실습 문제 풀기 테스트
사전 조건	사용자는 로그인된 상태여야한다.

수행 절차	1. 사용자는 학습 내용에서 실습 문제를 확인한다. 2. 사용자는 실습 문제에 맞는 조건을 이해한다. 3. 사용자는 문제 조건에 맞는 코드를 작성한다. 4. 사용자는 코드를 제출한다. 5. 사용자에게 코드를 제출했음을 알려준다. 6. 사용자는 테스트케이스를 확인한다. 7. 사용자는 인풋값에 테스트케이스를 입력한다. 8. 사용자는 결과화면을 확인한다.
기대되는 결과	사용자의 코드의 실행결과 표시

Table 17 Test Case: 실습 문제 풀기

10. Development Environment

10.1 Objective

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

10.2 React.js

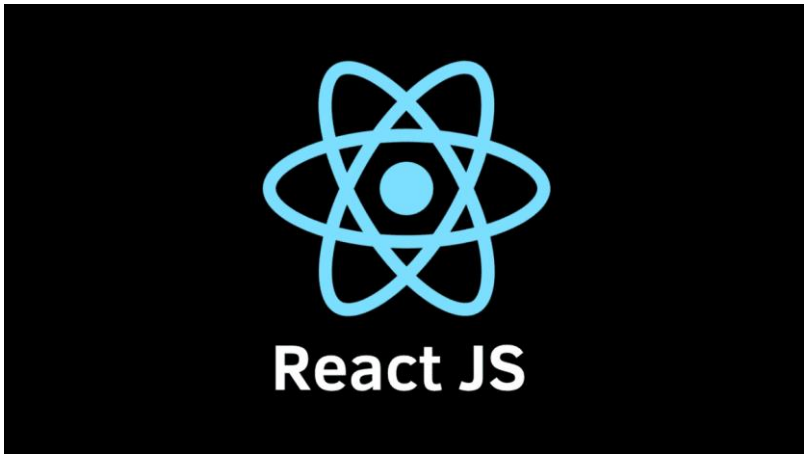


Figure 14 Logo of React.js

React.js는 프론트엔드에서 React Hook을 기반으로 상태 관리를 해주고 웹사이트를 쉽게 만들 수 게 도와주는 자바스크립트 기반 라이브러리다. Functional 프로그래밍을 통하여 Router, Redux와 함께 웹사이트 제작을 매끄럽게 해주는 기능을 제공해주며 유지, 보수할 수 있도록 도와준다.

10.3 Node.js

Figure 15 Logo of Node.js

Node.js 는 확장성 있는 네트워크 애플리케이션(특히 서버 사이드) 개발에 사용되는 소프트웨어 플랫폼이다. 작성 언어 자바스크립트 활용하며 논블로킹(Non-blocking) I/O 와 단일 스레드 이벤트 루프를 통한 높은 처리 성능을 가지고 있다.

내장 HTTP 서버 라이브러리를 포함하고 있어 웹 서버에서 아파치 등의 별도의 소프트웨어 없이 동작하는 것이 가능하며 이를 통해 웹 서버의 동작에 있어 더 많은 통제를 가능케 한다.

10.4 Django

Figure 16 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 돕는 오픈소스 기술이다. Django는 개발 과정을 단축시켜주고, 다양한 추가 기능을 포함하고 있으며 보안성이 높다. 유연하고 빠른 확장성을 제공해 CMS부터 SNS까지 다양한 플랫폼에 구축, 활용되고 있다.

10.5 MySQL

Figure 17 Logo of MySQL

MySQL 은 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, Qt 처럼 이중 라이선스가 적용된다.

10.6 GitHub

Figure 18 Logo of Github

버전관리 시스템인 git 을 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.

10.7 Docker

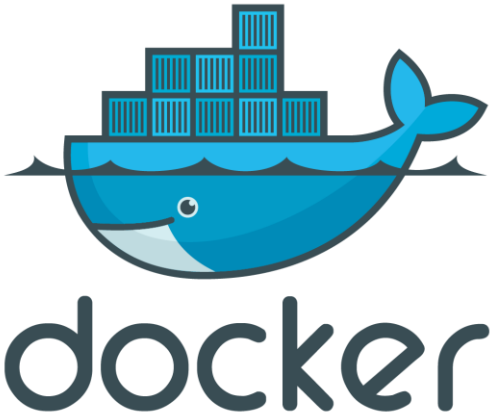


Figure 19 Logo of Docker

도커는 리눅스의 응용 프로그램들을 프로세스 격리 기술들을 사용해 컨테이너로 실행하고 관리하는 오픈 소스 프로젝트이다. 도커 웹 페이지의 기능을 인용하면 다음과 같다: 도커 컨테이너는 일종의 소프트웨어를 소프트웨어의 실행에 필요한 모든 것을 포함하는 완전한 파일 시스템 안에 감싼다.

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

Week	4	5	6	7	8	9	10	11	12	13	14
Requirements Engineering											
System Modeling											
Design Architecture											
UX/UI											
Frontend											
Backend											
Testing											
Deployment											

Table 18 Gantt Chart

현재까지의 개발 상황은 위와 같다. 온라인 IDE를 통하여 소프트웨어 교육을 원활히 제공해주기 위한 필요한 기능들을 정의하였다. 그리고 사용자 요구사항을 구체적으로 명세화하여 Functional, Nonfunctional로 나누어 분석하고, 동시에 System Modeling을 다양한 Diagram으로 표현하였다. 또한, Frontend와 Backend로 sub-system을 나눠 개발을 진행하고 통합 테스트를 거치게 되었다.

12. Index

12.1 Objective

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

12.2 Figure Index

12.3 Diagram Index

12.4 Table Index

13. References

13.1 Objectives

문서 작성에 참고한 참고 문헌 목록을 기술한다

13.2 Reference

- 위키백과 UML, (2022.02.15)
https://ko.wikipedia.org/wiki/%ED%86%B5%ED%95%A9_%EB%AA%A8%EB%8D%B8%EB%A7%81_%EC%96%B8%EC%96%B4

- 위키피디아 Visual Studio Code, (2022.05.03)

https://en.wikipedia.org/wiki/Visual_Studio_Code

- 위키피디아 Visual Paradigm, (2022.04.29)

https://en.wikipedia.org/wiki/Visual_Paradigm

- 위키피디아 draw.io, (2022.05.03)

https://en.wikipedia.org/wiki/Diagrams.net

- AQueryTool 도움말

https://aquerytool.com/help/index/