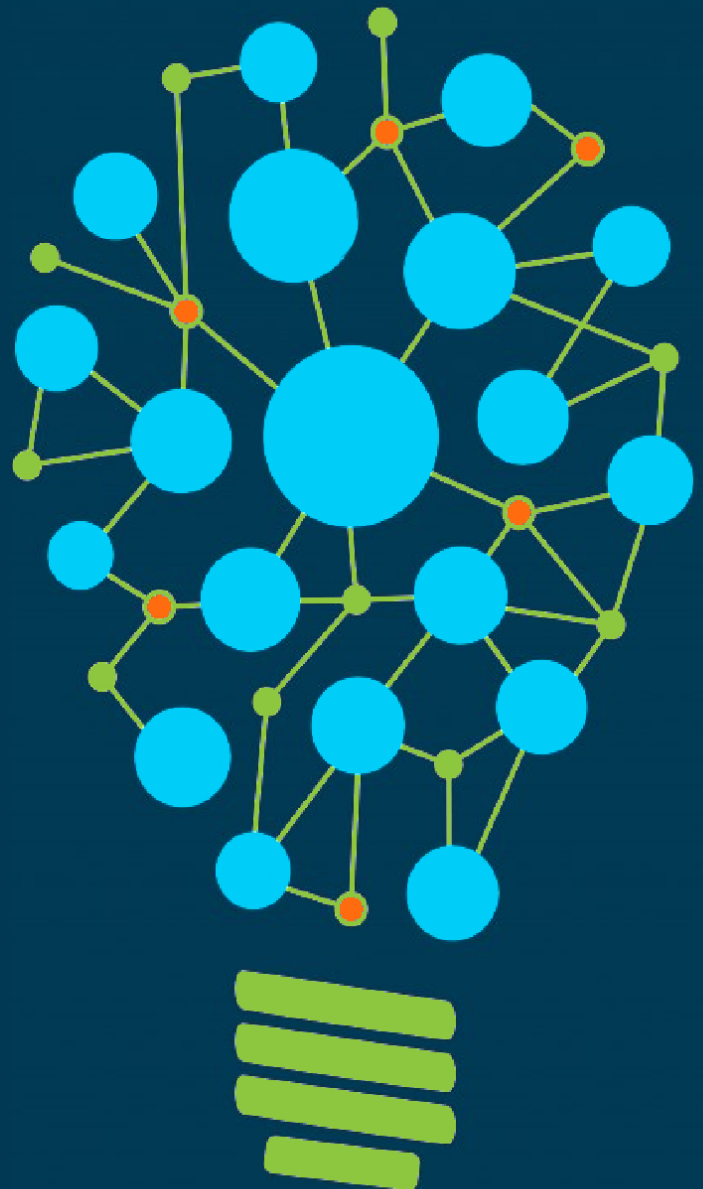


디자인 명세서

DOG/CAT MACHINE LEARNING ONLINE CLASS

SWE3002-41: Prof. 이은석

김기윤, 강영호, 강동준, 박상민, 이선종, 조민구



디자인 명세서

DOG/CAT MACHINE LEARNING ONLINE CLASS

by

김기윤, 강영호, 강동준, 박상민, 이선종,
조민구

TEAM 1

Instructor:	이은석
Teaching Assistant:	김기철, 김영경, 김진영, 허진석
Document Date:	09 April, 2022
Faculty:	SungKyunKwan University

Contents

1 Purpose	1
1.1 Readership	1
1.2 Scope	1
1.3 Objective	1
1.4 Document Structure	1
2 Introduction	2
2.1 Objectives	2
2.2 Applied Diagrams	2
2.2.1 Used Tools	2
2.2.2 Use Case Diagram	2
2.2.3 Sequence Diagram	2
2.2.4 Class Diagram	2
2.2.5 Context Diagram	2
2.2.6 Entity Relationship Diagram	2
2.2.7 Project Scope	3
2.2.8 References	3
3 System Architecture - Overall	4
3.1 Objectives	4
3.2 System Organization	4
3.2.1 System Diagram	5
3.3 Use Case Diagram	6
4 System Architecture - Frontend	7
4.1 Objectives	7
4.1.1 Profile	7
4.1.2 Classes	8
5 System Architecture - Backend	10
5.1 Objectives	10
5.2 Objectives	10
5.3 Subcomponents	11
5.3.1 User Management System	11
5.3.2 Code Test System	12
6 Protocol Design	13
6.1 Objectives	13
6.2 Ajax	13
6.3 TLS and HTTPS	13
6.4 CSRF and CSRF Token	13
6.5 Authentication	14
6.5.1 Register	14
6.5.2 Log In	15
6.5.3 Email Duplication Check	16
6.6 Progress Update	17
6.7 Progress Solving	18
7 Database Design	19
7.1 Objectives	19
7.2 ER Diagram	19
7.2.1 user	20

7.2.2	Problem	20
7.2.3	Test Data	21
7.3	Relational Schema	21
7.4	SQL DDL	22
7.4.1	User	22
7.4.2	Problem	22
7.4.3	TestData	22
8	Testing Plan	23
8.1	Objectives	23
8.2	Testing Policy	23
8.2.1	Development Testing	23
8.2.2	Release Testing	24
8.2.3	User Testing	24
8.2.4	Testing Case	24
9	Development Plan	25
9.1	Objectives	25
9.2	Frontend Environment	25
9.2.1	JavaScript	25
9.2.2	HyperText Markup Language (HTML)	25
9.2.3	Cascading Style Sheets (CSS)	26
9.3	Backend Environment	26
9.3.1	Github	26
9.3.2	Django	26
9.3.3	SQLite3	27
9.4	Constraints	27
9.5	Assumptions and Dependencies	27
10	Supporting Information	28
10.1	Software Design Specification	28
10.2	Document History	28

List of Figures

3.1	Overall System Architecture	4
3.2	Context Diagram-Overall	5
3.3	Context Diagram-Overall	6
4.1	Class Diagram - Profile	7
4.2	Sequence Diagram - Profile	8
4.3	Class Diagram - Classes	8
4.4	Sequence Diagram - Classes	9
5.1	Overall Architecture	10
5.2	Overall Architecture	11
5.3	Sequence Diagram - User Management System	11
5.4	Class Diagram - Code Test System	12
5.5	Sequence Diagram - Code Test System	12
6.1	Table of Register Request	14
6.2	Table of Register Response	14
6.3	Table of Log in Request	15
6.4	Table of Log in Response	15
6.5	Table of Email Duplication Check Request	16
6.6	Table of Email Duplication Check Response	16
6.7	Table of Progress Update Request	17
6.8	Table of Progress Update Response	17
6.9	Table of Problem Solving Request	18
6.10	Table of Problem Solving Response	18
7.1	ER Diagram, Entity, User	19
7.2	ER Diagram	20
7.3	ER Diagram, Entity, Problem	20
7.4	ER Diagram, Entity, TestData	21
7.5	Relational Schema	21
7.6	user sql schema	22
7.7	problem sql schema	22
7.8	TestData sql schema	22
9.1	JavaScript Logo	25
9.2	HTML Logo	25
9.3	CSS Logo	26
9.4	Github Logo	26
9.5	Django Logo	26
9.6	SQLite Logo	27

1

Purpose

본 문서가 예상하는 독자들, 문서의 구조, 그리고 각 단원에 대해 설명한다. 그리고 문서의 버전과 각 버전에서 만들어진 변경 사항에 대해 요약한다.

1.1. Readership

본 문서는 다음과 같은 독자들을 위해 만들어졌다. 본 시스템의 개발자들(**Team 1**)이다. 시스템 개발자는 크게 **Front-end**와 **Back-end** 개발자로 나뉘고 교육 **Contents**를 만드는 개발자 또한 포함한다. 그리고 소프트웨어 공학 개론 수업의 교수, 조교, 참여 학생이다.

1.2. Scope

머신러닝과 파이썬에 대한 이해 + (pytorch를 이용한 CNN을 이용해 Cat and Dog 뿐만 아니라 다른 data 에도 적용하여 ex) MNIST data 또한 classification 할 수 있다.)

1.3. Objective

이 소프트웨어 설계 문서의 주요 목적은 **Cat Dog Classifier** 실습 프로그램의 기술적 설계(**design**)에 대한 설명이다. 이 문서는 실습 프로그램의 구현의 기반이 되는 소프트웨어 **Front-End**, **Back-End**, **Database** 측면에서의 설계를 정의한다. 모든 설계는 앞서 제작된 **Software Requirements Specification** 문서의 요구 사항을 기반으로 작성되었다.

1.4. Document Structure

- 1) **Preface**: 본 문서의 목적, 예상 독자 및 문서의 구조에 대해 설명한다.
- 2) **Introduction**: 본 문서를 작성하는데 사용된 도구들과 다이어그램들, 참고 자료들에 대해 설명한다.
- 3) **Overall System Architecture**: 시스템의 전체적인 구조를 **Context Diagram**, **Use-case Diagram**, **Sequence Diagram**을 이용하여 서술한다.
- 4) **System Architecture - Frontend**: Frontend 시스템의 구조를 **Class Diagram**, **Sequence Diagram**을 이용하여 서술한다.
- 5) **System Architecture - Backend**: Backend 시스템의 구조를 **Sequence Diagram**, **Class Diagram**을 이용하여 서술한다.
- 6) **Protocol Design**: 클라이언트와 서버의 커뮤니케이션을 프로토콜 디자인을 서술한다.
- 7) **Database Design**: 시스템의 **Database Requirements**를 기반으로 **ER Diagram**, **Relation Schema**를 이용하여 시스템의 데이터베이스 디자인을 서술한다.
- 8) **Testing Plan**: 시스템을 위한 테스트 계획을 서술한다.
- 9) **Development Plan**: 시스템의 구현 계획 및 구현을 위한 개발 도구, 라이브러리 등의 개발 환경을 설명한다.
- 10) **Supporting Information**: 본 문서의 작성 및 수정 기록을 기술한다.

2

Introduction

design document는 프로젝트 구현에 있어서 기반이 될 수 있는 설계(design)를 제공한다. 또한, 설계는 앞서 제작된 Software Requirements Specification 문서에서 명시된 요구 사항을 따른다.

2.1. Objectives

이번 챕터에서는 제안한 시스템의 설계에 사용된 다이어그램, 틀에 대해 설명하고 개발 범위에 대해 설명한다.

2.2. Applied Diagrams

2.2.1. Used Tools

Microsoft Powerpoint 발표용 자료 제작 프로그램인 Powerpoint를 통해 기본적인 도형이나 아이콘을 생성할 수 있는 도구로 다이어그램을 그릴 수 있다.

2.2.2. Use Case Diagram

Use case Diagram은 시스템에서 제공해야 하는 기능이나 서비스를 명세화한 다이어그램이다. 사용자와 use cases 간의 관계를 보여주며, 사용자 - 시스템 간 상호작용을 표현한다. User를 Unregistered User, Registered User로 나누어서 Registered User에게만 System이 서비스를 제공하며 Learning(Lesson Quiz), Coding을 할 수 있도록 설계하였다.

2.2.3. Sequence Diagram

Sequence Diagram은 시간순서로 행동별로 어떤 객체와 어떻게 상호작용을 하는지 표현하는 Diagram이다. Event Diagram이라고도 부르는데, 시나리오와 관련된 객체와 시나리오의 기능 수행에 필요한 객체 간에 교환되는 메시지를 순서대로 표현한다. 시나리오와 관련된 객체는 사용자 클라이언트 서버가 있다.

2.2.4. Class Diagram

Class Diagram은 시스템을 구성하는 클래스, 그 속성, 기능 및 객체들 간의 관계를 표현하여 시스템의 정적인 부분을 보여준다. 이 시스템에서는 크게 사용자(student), 클라이언트, 서버가 있다고 봤다. 이 다이어그램은 실제로 구현될 소스코드와는 다를 수 있으며 의미나 해석 또한 경우에 따라 달라질 수 있다.

2.2.5. Context Diagram

Context Diagram은 Data Flow Diagram에서 가장 높은 수준의 다이어그램으로, 시스템 전체와 외부 요인의 입력 및 출력을 보여준다. 시스템과 해당 시스템과 상호작용할 수 있는 모든 외부의 엔티티들을 나타내며, 그 사이의 정보의 흐름을 표현한다. 엔지니어 또는 개발자가 사용하기보다는 프로젝트의 이해 관계자가 사용한다.

2.2.6. Entity Relationship Diagram

ER Diagram은 구조화된 데이터와 그들 간의 관계를 사람이 이해할 수 있는 형태로 표현하는 다이어그램이며, 현실에서의 요구사항들을 이용한 데이터베이스 설계과정에서 활용된다. 해당 다이어그램은 Entity,

Attribute, Relationship으로 구성된다. 이 다이어그램을 통해 데이터베이스의 논리적 구조를 설명한다. 왼쪽은 이 시스템에서 사용하는 User 정보를 관리하기 위한 database를 나타낸 Diagram이다. 오른쪽은 Test Data와 Problem Data 간의 관계의 구조를 나타낸 Diagram이다.

2.2.7. Project Scope

본 문서에서 설계하는 Cat Dog Classifier 실습 프로그램은 사용자들로 하여금 CNN을 만들어보고 구현함으로써 Deep Learning을 이해할 수 있도록 도와주기 위해 만들어졌다. 파이썬의 기초 문법과 지식을 습득한 학생이 이 서비스를 통해 산업에서 많이 쓰일 예정인 Deep-Learning 방법론에 대한 이해와 실습을 처음부터 끝까지 한 번 해보면서 CNN을 다른 문제에도 적용할 수 있도록 다양하게 학습을 돕고자 한다.

2.2.8. References

- IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements
- Specifications, In IEEEExplore Digital Library
- <https://github.com/skkuse/2021spring41classteam9>
- <https://github.com/skkuse/2020spring41classteam1/blob/master/docs/SDDTEAM1.pdf>

System Architecture - Overall

3.1. Objectives

이 챕터에서는 프론트 엔드 설계에서 백 엔드 설계에 이르는 프로젝트 어플리케이션의 시스템 구성에 대해 설명한다.

3.2. System Organization

이 서비스는 클라이언트 - 서버 모델을 적용하여 설계되었으며, 프론트 엔드 어플리케이션은 사용자와의 모든 상호작용을 담당한다. 프론트 엔드 어플리케이션과 백 엔드 어플리케이션은 JSON 기반의 HTTP 통신을 통해 데이터를 주고받는다. 백 엔드 어플리케이션은 설계 사양 요청을 프론트 엔드로부터 컨트롤러로 배포하고, SQLite3 데이터베이스에서 필요한 객체 정보를 얻어서 데이터베이스에서 처리한 후 JSON 형식으로 전달한다. 백 엔드 어플리케이션은 사용자가 작성한 머신러닝 코드를 전달받고 이를 실행하여 얻은 결과를 다시 내보낸다. 결과를 내보냄과 동시에 데이터베이스에 문제 및 코드의 정오, accuracy를 저장하고 사용자가 결과에 관한 정보를 요청하면 업데이트된 정보가 전달된다. 사용자가 서버를 통해 머신러닝을 이론, 문제, 코드와 같은 콘텐츠로 학습하기 위해서는 개개인의 계정을 생성하여야 하며, 계정을 생성하기 위해서는 주어진 절차에 따라 기존 SQLite3 데이터베이스와 대조 작업을 거친 중복적이지 않은 새로운 계정을 생성하여야 한다.

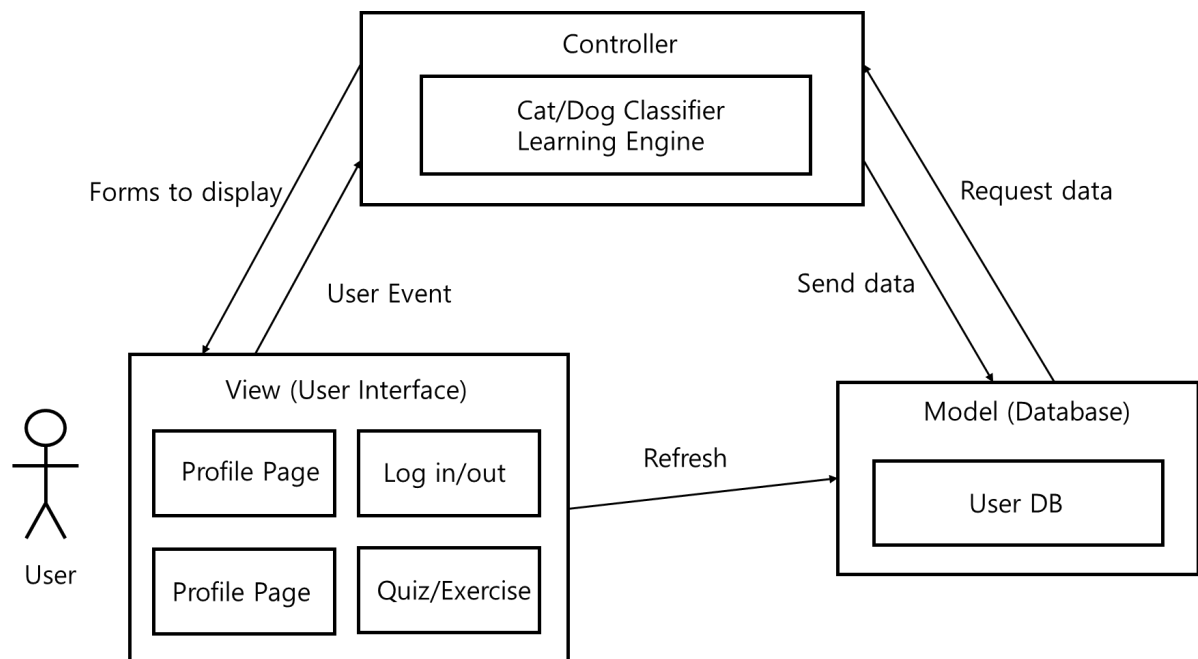


Figure 3.1: Overall System Architecture

3.2.1. System Diagram

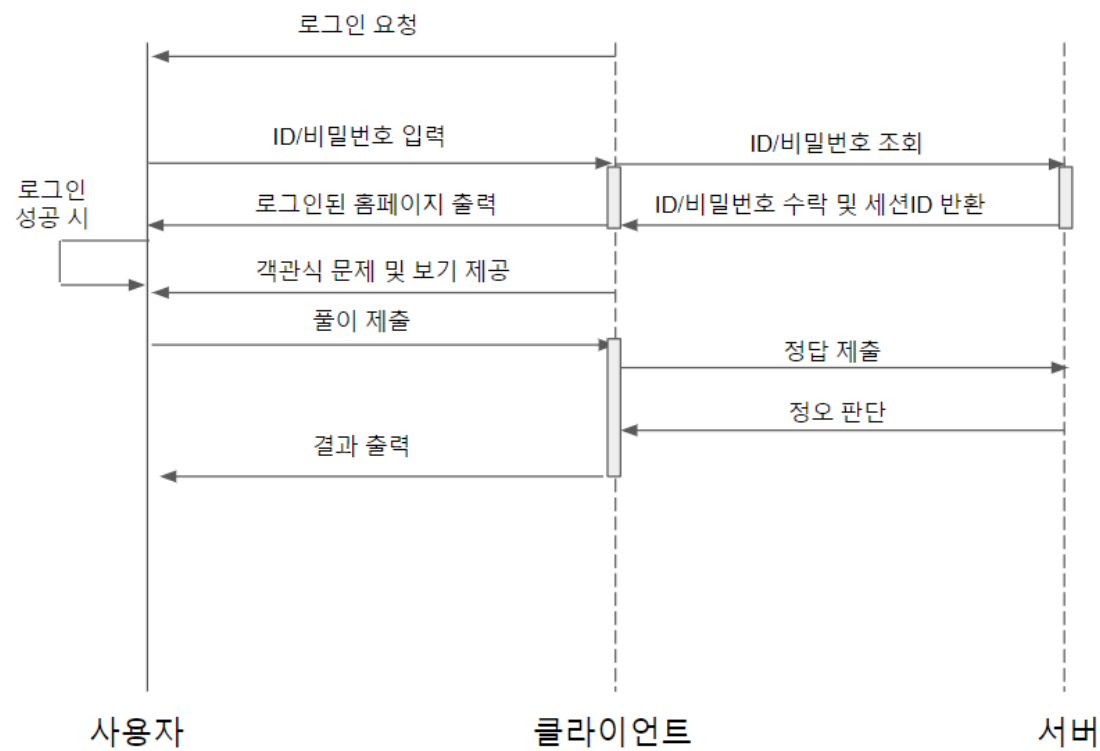


Figure 3.2: Context Diagram-Overall

3.3. Use Case Diagram

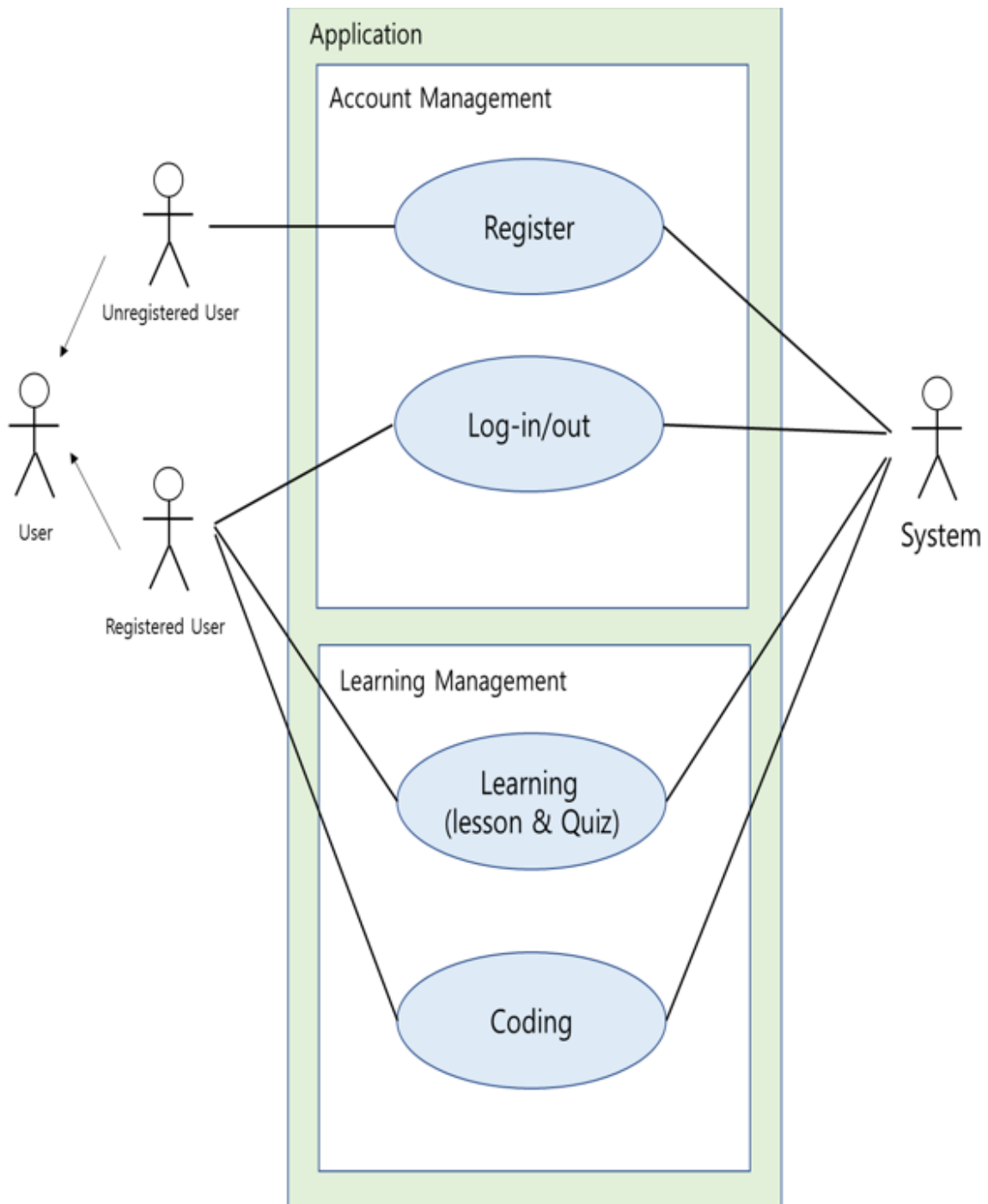


Figure 3.3: Context Diagram-Overall

System Architecture - Frontend

4.1. Objectives

이 장에서는 프론트엔드 시스템의 구조, 속성 및 기능을 설명하고 CAT/DOG 분류기에서 각 구성 요소의 관계를 설명합니다

4.1.1. Profile

기본 사용자 정보가 포함된 프로필 클래스 세부 정보입니다. 사용자가 등록함에 따라 사용자는 프로필에 기본 정보를 입력해야 합니다. 사용자는 등록 후 사용자 프로필을 수정할 수 있습니다(primaryID 제외).

Objectives

email: 사용자의 이메일

name: 사용자의 이름

password: 사용자의 비밀번호

Methods

createID()

getID()

Class Diagram

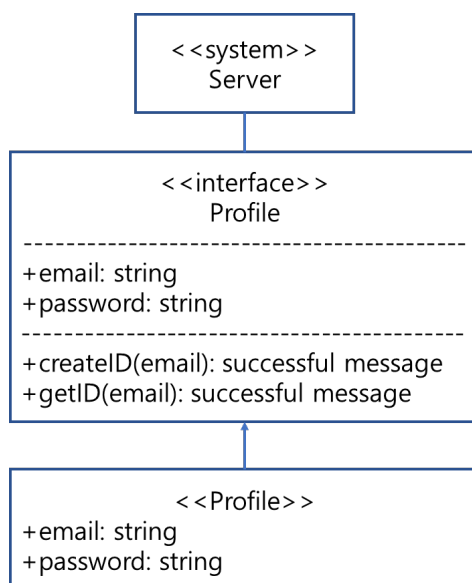


Figure 4.1: Class Diagram - Profile

Sequence Diagram

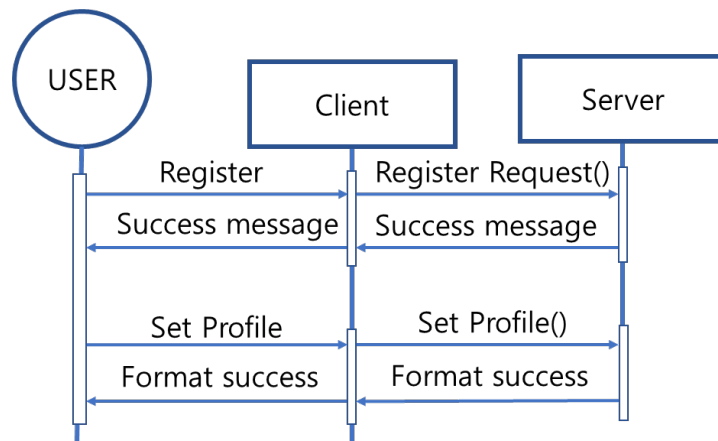


Figure 4.2: Sequence Diagram - Profile

4.1.2. Classes

사용자의 코드 학습 정보가 포함된 클래스 클래스 세부 정보. 사용자가 우리 서비스에서 배울 때 학습자는 다음 단계로 넘어가기 위해 필요한 교훈을 배워야 합니다. 각 수업 후 시스템은 진행 상황을 추적합니다.

Attributes

ID(Primary Key): 사용자 이메일

password: 사용자의 비밀번호

학습내용: 학생들이 서비스에서 배우는 정보

문제내용: 학생이 해야 하는 답변

Answer set: 사용자의 답변을 비교하여 정답인지 판단하는 데 사용되는 데이터

Class Diagram

Objects identification: 학습자가 ML 학습을 하는데 객관식 퀴즈에 대해 풀이하여 제출하면 서버가 정답인지 판단하며 결과를 돌려준다

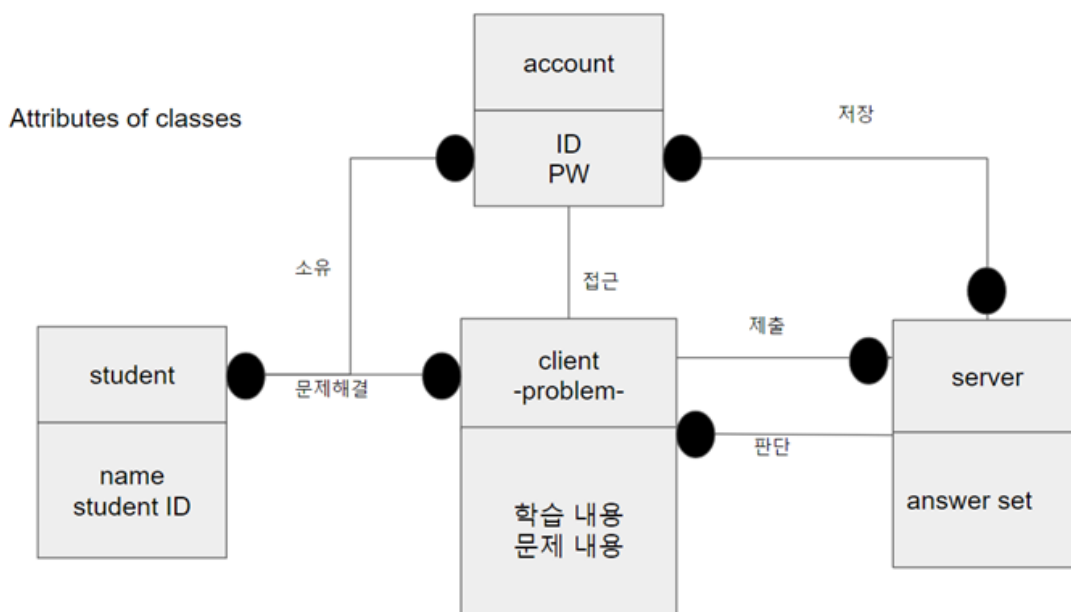


Figure 4.3: Class Diagram - Classes

Sequence Diagram

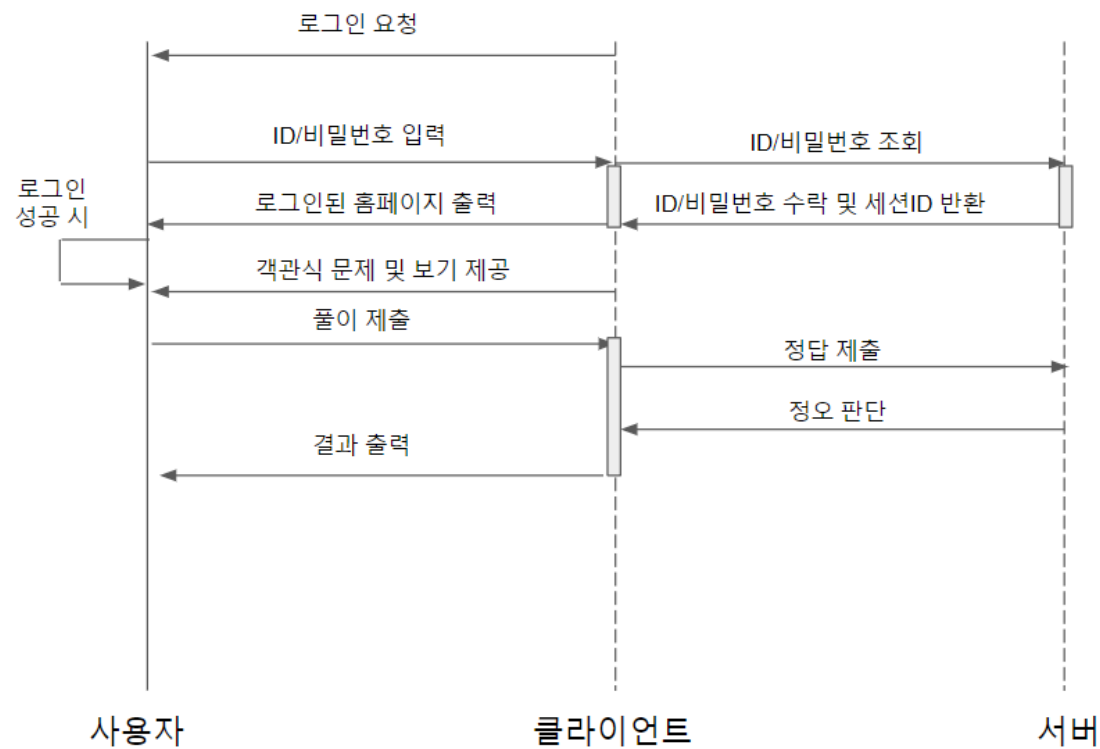


Figure 4.4: Sequence Diagram - Classes

5

System Architecture - Backend

5.1. Objectives

이 챕터는 back-end 시스템의 구조에 대해서 기술한다.

5.2. Objectives

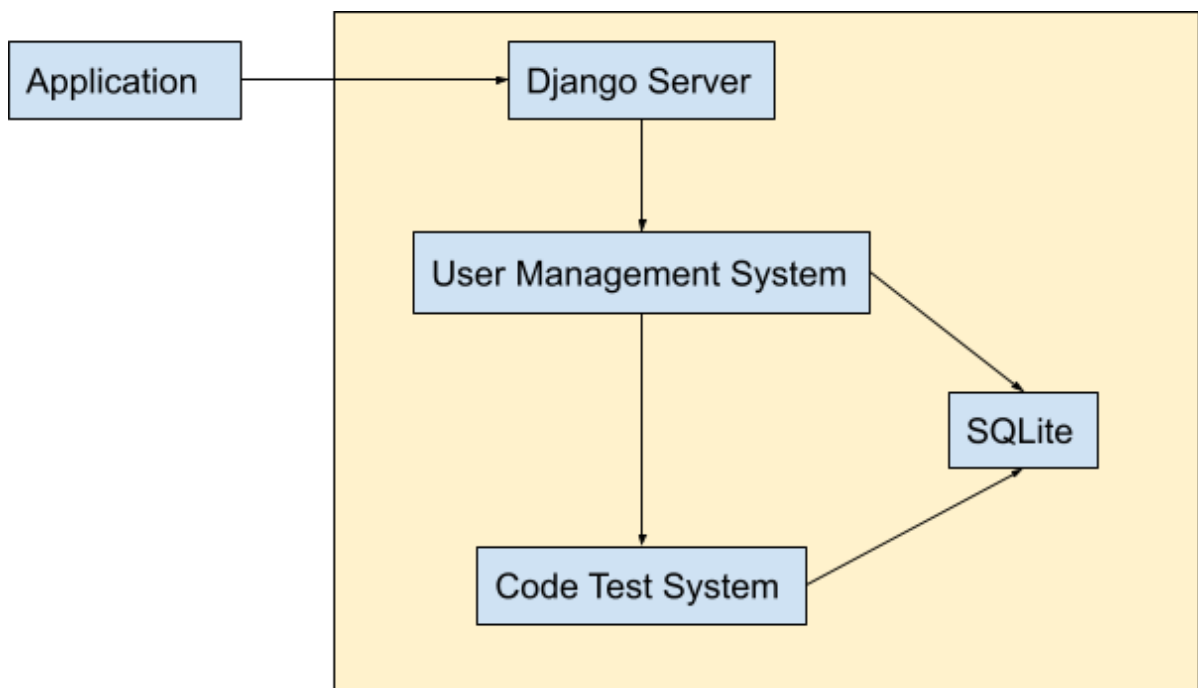


Figure 5.1: Overall Architecture

사용자로 하여금 Python을 학습하고 결과적으로 Cat Dog Classifier 프로그램을 작성할 수 있도록 유도하는 본 시스템의 전반적인 구조는 위와 같다. front-end로부터 요청이 들어오면 Django server에서 해당 요청을 처리하는 function을 실행한다. back-end에서 처리하는 요청으로는 로그인 처리, Python 코드 테스트 등이 있으며, Django에서 기본적으로 제공하는 데이터베이스인 SQLite를 사용한다.

5.3. Subcomponents

5.3.1. User Management System

Class Diagram

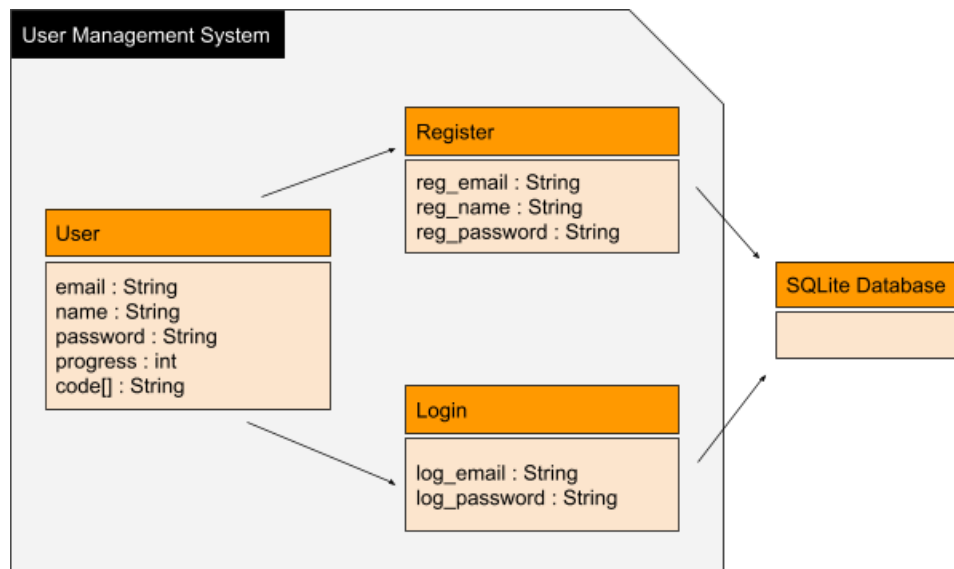


Figure 5.2: Overall Architecture

Register class: 입력받은 email이 중복되지 않았는지 확인하며, 입력받은 정보를 Database에 새로 추가한다.

Login class: 입력받은 데이터가 기존 Database에 존재하는지 확인하고, 정보가 일치한다면 해당 class의 정보를 불러온다.

Sequence Diagram

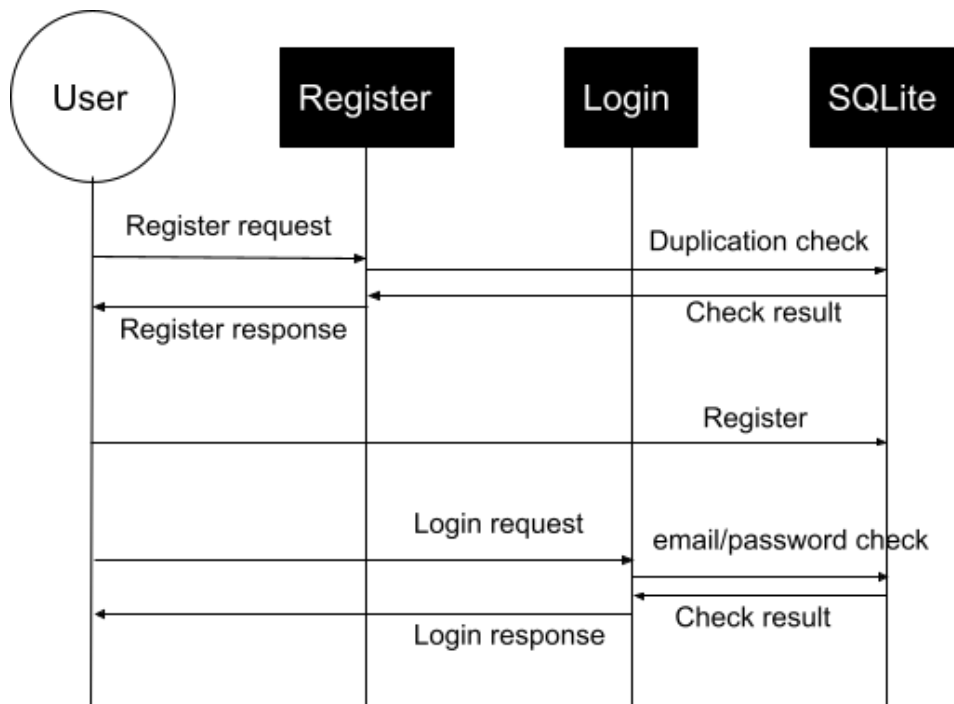


Figure 5.3: Sequence Diagram - User Management System

5.3.2. Code Test System

Class Diagram

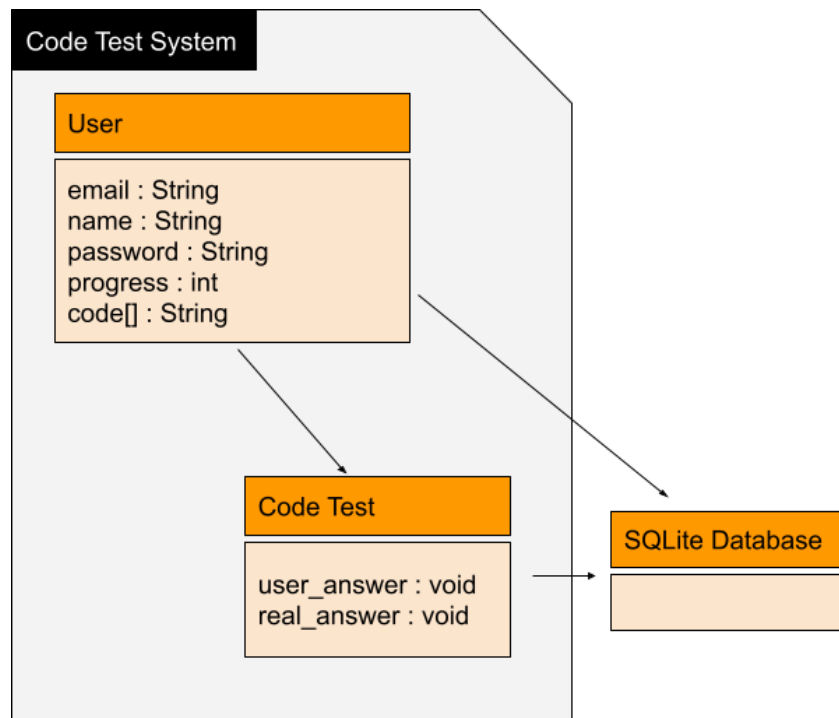


Figure 5.4: Class Diagram - Code Test System

User class: 해당 코딩 문제에서 사용자가 작성한 코드 전체를 **String**의 형태로 **Database**에 저장한다.
 Code Test class: 해당 문제에 대한 정답을 가지고 있으며, 사용자가 작성한 코드를 실행시킨 결과를 받아
 둘을 비교한 결과를 **Database**에 저장한다.

Sequence Diagram

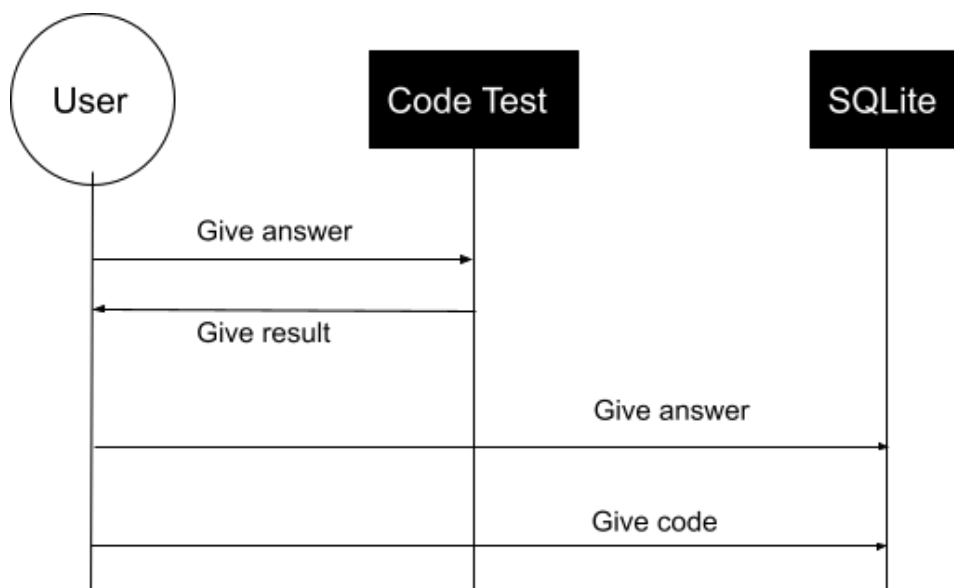


Figure 5.5: Sequence Diagram - Code Test System

6

Protocol Design

6.1. Objectives

이 챕터는 front-end 애플리케이션과 서버가 어떤 프로토콜로 상호작용하는 지를 기술한다. 또한 각 인터페이스가 어떻게 정의되어 있는지 기술한다.

6.2. Ajax

AJAX란 비동기 자바스크립트와 XML (Asynchronous JavaScript And XML)을 말한다. 간단히 말하면, 서버와 통신하기 위해 XMLHttpRequest 객체를 사용하는 것을 말한다. JSON, XML, HTML 그리고 일반 텍스트 형식 등을 포함한 다양한 포맷을 주고 받을 수 있다. AJAX의 강력한 특징은 페이지 전체를 리프레쉬 하지 않고서도 수행되는 비동기성이다. 이러한 비동기성을 통해 사용자의 Event가 있으면 전체 페이지가 아닌 일부분만을 업데이트 할 수 있게 해준다.

6.3. TLS and HTTPS

TLS(Transport Layer Security)는 웹사이트와 브라우저 사이(또는 두 서버 사이)에 전송되는 데이터를 암호화하여 인터넷 연결을 보호하기 위한 표준 기술이다.

HTTPS(HyperText Transfer Protocol over Transport Layer Security)는 월드 와이드 웹 통신 프로토콜인 HTTP를 TLS위에서 패킹하여 강화된 보안을 제공하는 버전이다.

6.4. CSRF and CSRF Token

CSRF 공격(Cross Site Request Forgery)은 웹 어플리케이션 취약점 중 하나로 인터넷 사용자(희생자)가 자신의 의지와는 무관하게 공격자가 의도한 행위(수정, 삭제, 등록 등)를 특정 웹사이트에 요청하게 만드는 공격이다.

CSRF Token CSRF 공격을 방지하기 위해 서버에서 난수 토큰을 생성하여 서버에 요청을 주는 페이지에 부여하여 정당하지 않은 페이지에서 오는 요청을 막는 기법이다.

6.5. Authentication

6.5.1. Register

Request

Attribute	Detail	
Protocol	HTTPS	
Method	POST	
Request Body	Email	User email (for identification)
	Name	User name
	Password	User password
	CSRF Token	CSRF Token issued by server

Figure 6.1: Table of Register Request

Response

Attribute	Detail	
Protocol	HTTPS	
Success Code	200 OK	
Failure Code	HTTP error code = 400(Bad Request, overlap)	
Success Response Body	Session ID	Session ID to access
	Message	Success message
Failure Response Body	Message	Failure message

Figure 6.2: Table of Register Response

6.5.2. Log In
Request

Attribute	Detail	
Protocol	HTTPS	
Method	POST	
Request Body	Email	User email (for identification)
	Password	User password
	CSRF Token	CSRF Token issued by server

Figure 6.3: Table of Log in Request

Response

Attribute	Detail	
Protocol	HTTPS	
Success Code	200 OK	
Failure Code	HTTP error code = 400(Bad Request, overlap)	
Success Response Body	Session Token	Token for ID session
	Message	Success message
Failure Response Body	Message	Failure message

Figure 6.4: Table of Log in Response

6.5.3. Email Duplication Check

Request

Attribute	Detail	
Protocol	HTTPS	
Method	Ajax / POST	
Request Body	Email	User email

Figure 6.5: Table of Email Duplication Check Request

Response

Attribute	Detail	
Protocol	HTTPS	
Success Code	200 OK	
Failure Code	HTTP error code = 400(Bad Request, overlap)	
Success Response Body	Message	Success message
	Result	True or false value to identify duplication of email
Failure Response Body	Message	Failure message

Figure 6.6: Table of Email Duplication Check Response

6.6. Progress Update
Request

Attribute	Detail	
Protocol	HTTPS	
Method	POST	
Request Body	Session ID	Session ID to access
	Progress	progress data for learning to update
	CSRF Token	CSRF Token issued by server

Figure 6.7: Table of Progress Update Request

Response

Attribute	Detail	
Protocol	HTTPS	
Success Code	200 OK	
Failure Code	HTTP error code = 400(Bad Request, overlap)	
Success Response Body	Message	Success message
Failure Response Body	Message	Failure message

Figure 6.8: Table of Progress Update Response

6.7. Progress Solving

Request

Attribute	Detail	
Protocol	HTTPS	
Method	POST	
Request Body	Session ID	Session ID to access
	Code	Python code written by learner

Figure 6.9: Table of Problem Solving Request

Response


Attribute	Detail	
Protocol	HTTPS	
Success Code	200 OK 	
Failure Code	HTTP error code = 400(Bad Request, overlap)	
Success Response Body	Message	Success message
	Result	True or false value to notice answer
	Log	Python interpreter logs to debug when the answer is wrong
Failure Response Body	Message	Failure message

Figure 6.10: Table of Problem Solving Response

Database Design

7.1. Objectives

7장에서는 시스템 데이터 구조와 이러한 구조가 데이터베이스로 어떻게 구현되었는지에 대해 설명한다. 먼저 ER 다이어그램(Entity Relationship diagram)을 통해 entity와 그 관계를 식별한다. 그런 다음 관계형 스키마 및 SQL DDL(Data Definition Language)을 작성한다.

7.2. ER Diagram

본 어플리케이션 시스템은 총 3가지 entity로 이루어져 있다; User, Problem, TestData,. ER-Diagram은 entity간의 관계, 그리고 entity와 attribute의 관계를 다이어그램으로 설명한다. 각 entity의 primary key는 밑줄로 표시되어 있다. 각 entity마다 대응되는 개수는 entity를 연결하는 선 주변에 표기되어 있어 확인 가능하다.

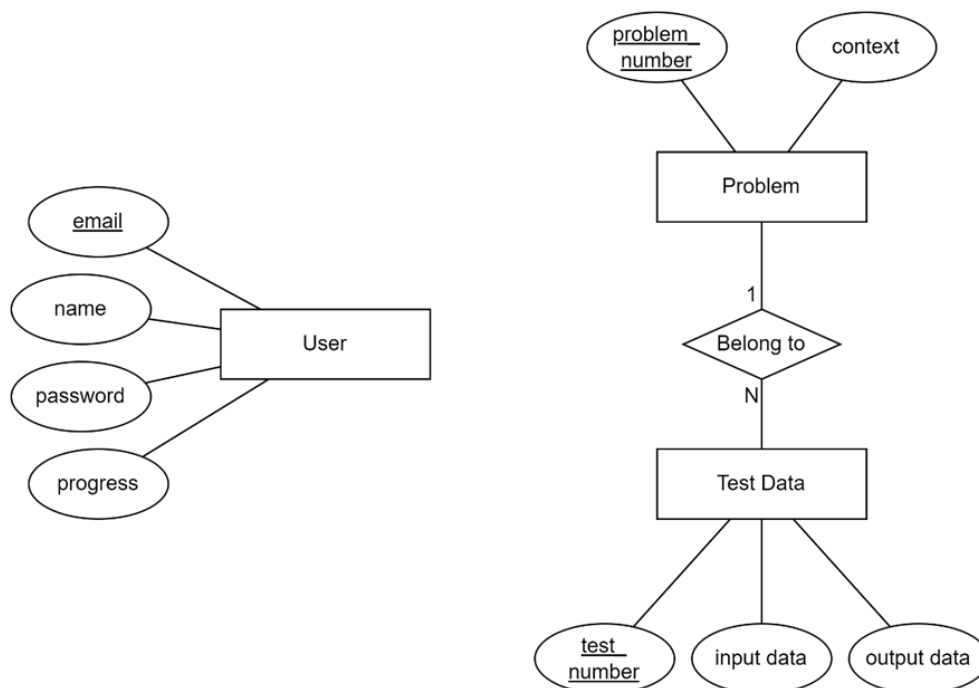


Figure 7.1: ER Diagram, Entity, User

7.2.1. user

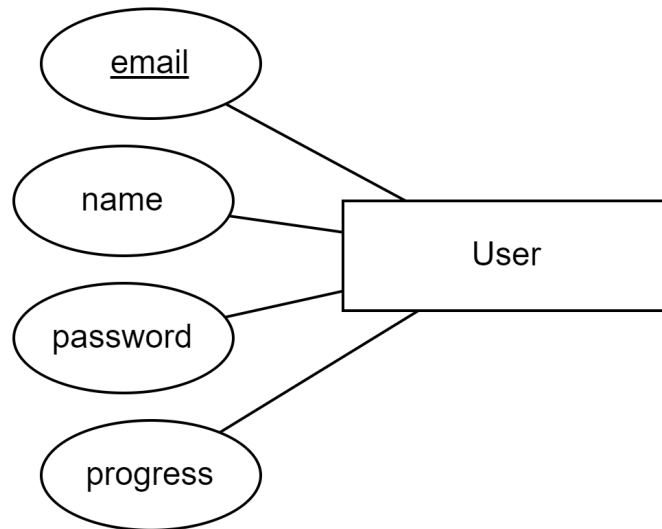


Figure 7.2: ER Diagram

User Entity는 애플리케이션 사용자에게 해당한다. User Entity는 email, name, password, progress를 attribute로 가지며, email이 primary key이다. 이 attribute들은 애플리케이션 회원 가입 때 사용자가 기입한 정보이다. progress는 최초의 0으로 초기화되고 학습진행에 따라 업데이트된다

7.2.2. Problem

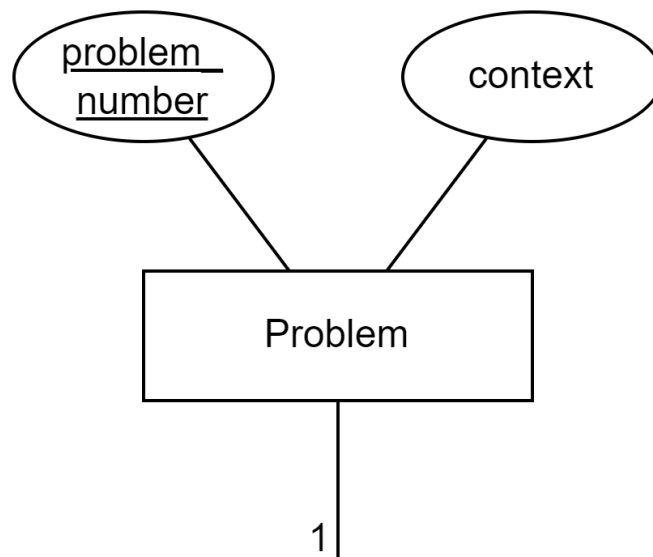


Figure 7.3: ER Diagram, Entity, Problem

Problem Entity는 사용자에게 제시할 코딩문제를 저장한다. problemnumber를 primary key로 가지고 있고 context에 문제의 내용이 기입되어있다. Problem Entity는 TestData와 1:N관계를 가지는데 하나의 문제마다 1개 이상의 테스트 데이터를 가지고 있다.

7.2.3. Test Data

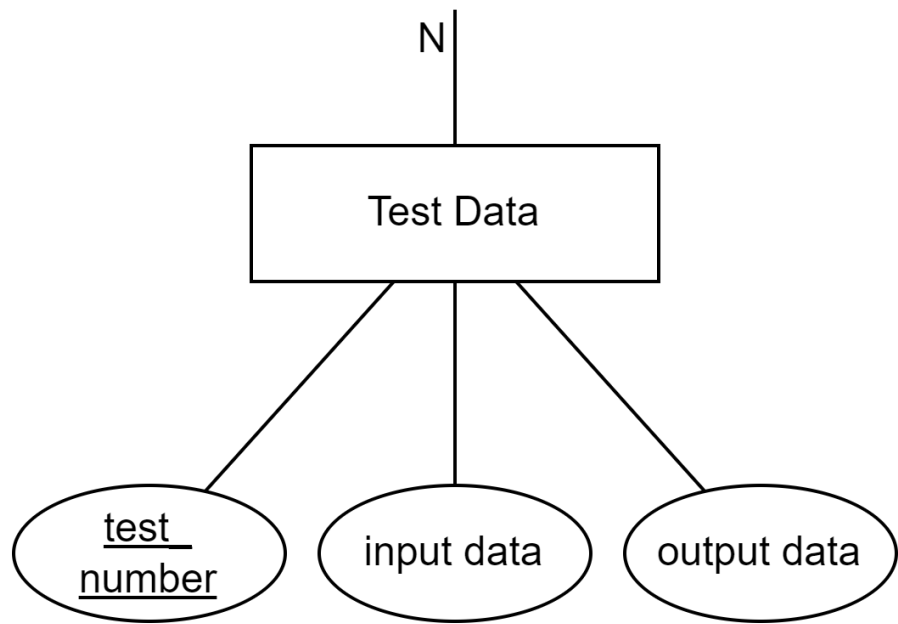


Figure 7.4: ER Diagram, Entity, TestData

TestData Entity는 특정 문제의 테스트데이터를 소유하고 있다. primary key인 testnumber와 input data, output data attribute를 가지고 있다. Problem과 TestData가 1:N 관계를 가지고 있고 TestData가 problemnumber를 foreign key로 참조하여 접근한다.

7.3. Relational Schema

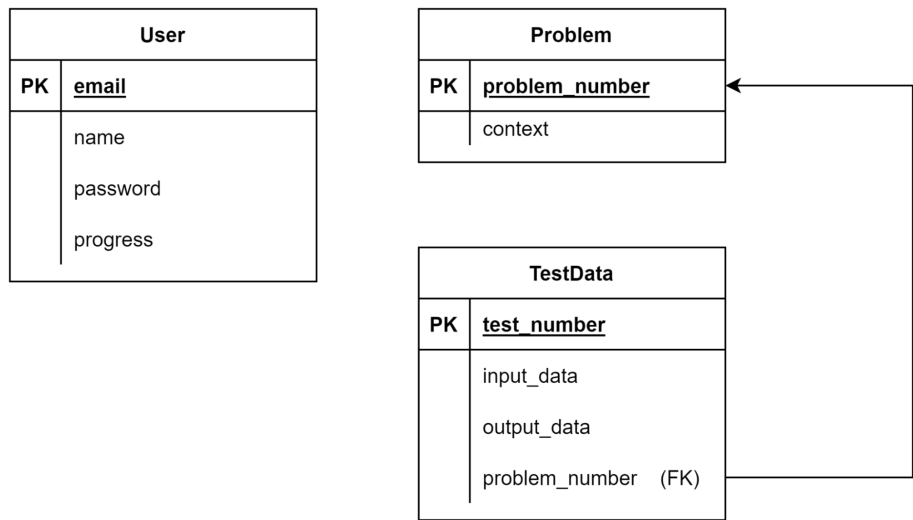


Figure 7.5: Relational Schema

7.4. SQL DDL

7.4.1. User

```
CREATE TABLE User(  
  email VARCHAR(100) NOT NULL,  
  name VARCHAR(100) NOT NULL,  
  password VARCHAR(100) NOT NULL,  
  progress INTEGER NOT NULL,  
  PRIMARY KEY(email)  
);
```

Figure 7.6: user sql schema

7.4.2. Problem

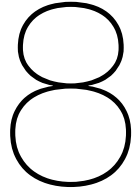
```
CREATE TABLE Problem(  
  problem_number INTEGER NOT NULL,  
  context VARCHAR(4000) NOT NULL,  
  PRIMARY KEY(problem_number)  
);
```

Figure 7.7: problem sql schema

7.4.3. TestData

```
CREATE TABLE User(  
  test_number INTEGER NOT NULL,  
  input_data VARCHAR(100) NOT NULL,  
  output_data VARCHAR(100) NOT NULL,  
  problem_number INTEGER NOT NULL,  
  PRIMARY KEY(test_number),  
  FOREIGN KEY(problem_number) REFERENCES Problem(problem_number)  
);
```

Figure 7.8: TestData sql schema



Testing Plan

8.1. Objectives

이번 장에서는 **development testing**, **release testing** 및 **user testing** 3가지 주요 하위 그룹을 포함하는 테스트에 대한 계획을 설명한다. 이러한 테스트의 목적은 애플리케이션의 잠재적인 오류와 결함을 감지하고, 애플리케이션의 완성도를 높여 안정적인 시스템을 **user**에게 제공하기 위한 것이다.

8.2. Testing Policy

8.2.1. Development Testing

Development Testing은 현재 시스템의 개발 과정 전체에서 발생하는 가능한 오류를 감지하고 감지된 오류를 수정하기 위한 것이다. 이 단계에서는 소프트웨어가 계속해서 개발 중인 단계에 있기 때문에 불안정할 수 있고, **frontend**와 **backend** (**server**와 **website**)의 충돌이 일어날 수 있다. 따라서 이 단계에서는 **data flow** 검토 및 개발 팀원간의 **code review** 등을 계획하고 이를 통해 소프트웨어의 성능과 신뢰성, 보안을 보장할 수 있도록 초점을 맞추어 테스트를 진행한다.

Performance

본 시스템의 **performance**에 관한 지표는 다음과 같다.

- 머신러닝 교육용으로 사용되는 본 시스템의 특성상 사용자의 코드에 대한 결과 처리에 소모되는 지연시간이 5분 이내로 완료되어야 한다.
- 계산된 결과는 5초 이내로 데이터베이스에 저장되고 이를 나타낼 수 있어야 한다.
- 사용자의 회원가입 및 계정정보 저장을 각각 5초 이내로 완료하여야 한다.
- 사용자의 로그인/로그아웃 요청을 5초 이내로 완료해야 한다.
- 사용자의 문제 채점 결과를 5초 이내로 데이터베이스에 저장하고 이를 사용자에게 알려주어야 한다.
- 10000명까지의 사용자의 계정 정보를 관리할 수 있어야 한다. 따라서 다음과 같은 지표를 만족할 수 있도록 실제 테스트를 다양한 환경에서 여러번 수연하여 지연시간이 위와 같은 지표를 만족하였는지 확인한다.

따라서 다음과 같은 지표를 만족할 수 있도록 실제 테스트를 다양한 환경에서 여러번 수연하여 지연시간이 위와 같은 지표를 만족하였는지 확인한다.

Reliability

시스템이 오류를 발생하지 않고 안전하게 작동하려면 시스템을 구성하는 각각의 하위 구성 시스템과 장치가 정상적으로 작동한 뒤, 전체 시스템으로 연결되어야 한다. 따라서 하위 시스템 개발 단계부터 개발 테스트를 순차적으로 거쳐 각 시스템이 전체 시스템에 통합되는 동안 오류를 반복적으로 확인하고 그에 따라 수정해야 한다.

Security

본 시스템에서는 금융과 관련된 어플리케이션 만큼의 높은 보안수준은 요구되지 않으나, 사용자들이 가입시 기입한 개인정보의 보안에 신경을 써야 한다. 사용자들은 자신을 제외한 사용자들의 그 어떤 정보도 열람할 수 없도록 해야 하며, 외부의 시스템이 DB에 접근하지 못하도록 해야한다. 또한 고객들의 ID/PW에 대한 암호화에 대한 검토가 요구된다.

8.2.2. Release Testing

Release Testing은 어떻게 본 시스템을 배포하는가에 대한 것을 테스트하는 단계이다. 릴리스 테스트는 소프트웨어/어플리케이션의 업데이트된 버전을 테스트하여 소프트웨어가 최신의 완전한 상태로 배포되어 작동에는 아무런 하자가 없는지 확인하여야 하며, 실제로 배포가 되기 전에 수행하여야 한다.

본 소프트웨어 배포는 일반적으로 모든 시스템의 기본 구현을 포함하는 알파 버전에서 시작하여 개발 테스트를 시작하고 사용자 및 릴리스 테스트를 포함한 추가 테스트를 위해 베타 버전을 배포할 예정이다. 베타 버전이 배포되고 난 후엔 실제 사용자로부터 피드백을 받을 수 있도록 한다.

8.2.3. User Testing

User Testing에서는 실제 사용자가 본 소프트웨어를 사용하는 과정을 통해서 테스트를 진행하게 된다. 따라서 이번 수업을 수강하는 학생들과 개발 팀원을 모아 해당 베타버전 사용자에게 배포하고 자체적인 사용 및 일어날 수 있는 여러 시나리오를 수행하며 각각의 진행과정에서 오류가 없는지 검토하는 방식으로 이루어진다.

8.2.4. Testing Case

Test case는 앞서 살펴본 performance, reliability, security에 초점을 맞추어 설계된다. 각각의 측면에서 최소 5개 이상의 test case와 scenario를 통해 본 소프트웨어에 대한 테스트를 진행하며 평가 보고서를 작성한다.

9

Development Plan

9.1. Objectives

해당 챕터에서는 시스템의 개발 환경 및 기술에 대하여 설명한다.

9.2. Frontend Environment

9.2.1. JavaScript



Figure 9.1: JavaScript Logo

JavaScript는 객체 기반의 스크립트 프로그래밍 언어이다. 주로 웹 브라우저 내에서 사용하며, 다른 응용 프로그램의 내장 객체에도 접근할 수 있는 기능을 가지고 있다. 본 소프트웨어에서는 사용자가 풀이하는 퀴즈와 관련하여 해당 기능들을 사용한다.

9.2.2. HyperText Markup Language (HTML)



Figure 9.2: HTML Logo

HTML은 웹 페이지를 위한 마크업 언어이며 W3C가 HTML과 CSS 표준의 공동 책임자이다. HTML을 통해 제목, 단락, 목록 등 본문을 위한 구조적 의미를 나타내는 것뿐만 아니라 링크, 인용과 그 밖의 항목으로 구조적 문서를 만들 수 있는 방법을 제공할 수 있다. 이는 태그로 되어있는 HTML 요소 형태로 작성되며 웹 브라우저와 같은 HTML 처리 장치의 행동에 영향을 주는 자바스크립트를 포함하거나 불러올 수 있다. 따라서 우리는 본 소프트웨어의 웹 페이지를 구현하기 위하여 HTML을 사용한다.

9.2.3. Cascading Style Sheets (CSS)



Figure 9.3: CSS Logo

CSS는 마크업 언어가 실제 표시되는 방법을 기술하는 스타일 언어로, HTML과 XHTML에 주로 쓰이며, XML에서도 사용할 수 있다. W3C의 표준이며, 레이아웃과 스타일을 정의할 때의 자유도가 높다. 마크업 언어가 웹사이트의 몸체를 담당한다면, CSS는 옷과 액세서리처럼 꾸미는 역할을 담당한다. 우리는 웹사이트의 가시성과 심미성을 높이기 위하여 이를 사용한다.

9.3. Backend Environment

9.3.1. Github



Figure 9.4: Github Logo

Github는 소프트웨어를 개발하고 Git을 통해 버전을 관리하기 위해 사용되는 툴이다. 이를 통해 여러 명의 개발자가 하나의 프로젝트를 동시에 관리하며 개발할 수 있으며, 각각의 컴포넌트들을 통합하는데 이점이 있다. 따라서 우리는 본 소프트웨어의 개발 및 버전관리를 위해 이를 사용할 것이다.

9.3.2. Django



Figure 9.5: Django Logo

Django는 파이썬으로 구성된 서버-사이드 웹 프레임워크로, 파이썬을 웹 서비스에 사용하기 위한 것이다. 머신러닝 코드를 파이썬으로 구현하고 실행하는 본 소프트웨어의 특성상, 이러한 프레임워크를 사용하여 사용자가 작성한 파이썬 코드를 웹에서 입력받아 실행할 수 있도록 한다.

9.3.3. SQLite3



Figure 9.6: SQLite Logo

SQLite는 별도의 서버 프로세스가 필요 없고 SQL 질의 언어의 비표준 변형을 사용하여 데이터베이스에 액세스할 수 있는 경량 디스크 기반 데이터베이스를 제공하는 C 라이브러리이다. 일부 응용 프로그램은 내부 데이터 저장을 위해 SQLite를 사용할 수 있으며, SQLite를 사용하여 응용 프로그램을 프로토타입한 다음 PostgreSQL 이나 Oracle과 같은 더 큰 데이터베이스로 코드를 이식할 수도 있다. 따라서 본 소프트웨어를 사용하는 사용자의 ID/PW와 같은 개인 정보를 저장하기 위해 이를 사용한다.

9.4. Constraints

본 시스템은 이 문서에서 언급된 내용들에 기반하여 디자인되고 구현될 것이다. 이를 위한 세부적인 제약사항은 다음과 같이 나타난다.

- 기존에 널리 쓰이고 있는 기술 및 언어들을 사용한다.
- 사용자의 입력 및 코드를 실행하고 결과를 저장하는 시간은 5초를 넘기면 안된다.
- 로열티를 지불하거나 **separate license**를 요구하는 기술 및 **software**의 사용을 지양한다.
- 전반적인 시스템 성능을 향상시킬 수 있도록 개발한다.
- 사용자가 편리하게 이용할 수 있도록 개발한다.
- 가능한 오픈소스 소프트웨어를 사용한다.
- 시스템 비용과 유지보수 비용을 고려하여 개발을 진행한다.
- 머신러닝 및 시스템의 확장을 고려하여 개발을 진행한다.
- 시스템 자원의 낭비를 막을 수 있도록 소스 코드를 최적화한다.
- 소스 코드 작성시 유지보수에 대하여 고려하며 필요한 부분에 대해서는 주석을 통하여 이해하기 쉽도록 한다.
- 개발은 최소 윈도우 7 이상에서 이루어져야 하며 윈도우 10, 11 환경을 타겟으로 한다.
- 윈도우 10, 11 버전에서 실행한다.

9.5. Assumptions and Dependencies

본 문서의 모든 시스템은 데스크탑 환경에 기반하여 디자인 및 구현되었다고 가정하며 작성되었다. 또한 윈도우 10, 11 기반의 OS 환경을 기반으로 하여 작성되었으며 따라서 다른 OS나 조건을 만족하지 않는 환경에서 시스템의 지원은 보장할 수 없다.

10

Supporting Information

10.1. Software Design Specification

소프트웨어 요구사항 명세서 IEEE 권장사항 (IEEE Recommend Practice for Software Requirements Specifications, IEEE-Std-830)에 따라 작성되었다.

10.2. Document History

Date	Description	Version	Writer
2022/05/12	Introduction and System Architecture	1.0	강영호
2022/05/12	System Architecture - Frontend and LATEX	1.0	조민구
2022/05/12	System Architecture – Backend and Protocol Design	1.0	박상민
2022/05/12	System Architecture – Backend and Database Design	1.0	이선종
2022/05/12	Overall architecture, Testing and Development Plan	1.0	강동훈