

# 디자인 명세서

이정혁 김남현 김태영 이상현 한준호

TEAM 6

Instructor: 이은석

Teaching Assistant: 김영경, 김진영, 최동욱, 허진석

Document Date: 24, 11, 2024

Faculty: SungKyunKwan University

# Index

<b>1. Purpose .....</b>	<b>5</b>
1.1 Readership.....	5
1.2 Scope .....	5
1.3 Objectives .....	5
1.4 Document Structure.....	6
 <b>2. Introduction .....</b>	 <b>7</b>
2.1 Objectives .....	7
2.2 Applied Diagrams.....	7
2.2.1 Used Tool.....	7
2.2.2 Use Case Diagram .....	7
2.2.3 Sequence Diagram .....	8
2.2.4 Class Diagram .....	8
2.2.5 Context Diagram .....	8
2.2.6 Entity Relationship Diagram.....	8
2.2.7 Project Diagram .....	8
2.2.8 References .....	9
 <b>3. System Architecture - Overall .....</b>	 <b>10</b>
3.1 Objectives .....	10

3.2 System Organization.....	10
3.2.1 System Diagram.....	11
3.3 Use Case Diagram.....	12
<b>4. System Architecture - Fronted .....</b>	<b>13</b>
4.1 Objectives.....	13
4.1.1 Profile .....	13
4.1.2 LLM Q&A.....	15
4.1.3 Dialogue Save.....	17
<b>5. System Architecture - Backend .....</b>	<b>20</b>
5.1 Objectives.....	20
5.2 Objectives.....	20
5.3 Subcomponents.....	21
5.3.1 Account Management System .....	21
5.3.2 Learning Management System .....	22
5.3.3 LLM Interaction System.....	25
<b>6. Protocol Design .....</b>	<b>27</b>
6.1 Objectives.....	27
6.2 Ajax.....	27
6.3 TLS & HTTPS .....	27
6.4 CSRF & CSRF Token .....	27
6.5 Authentication .....	28
6.5.1 Register.....	28

6.5.2 Login .....	29
6.5.3 Logout .....	30
6.5.4 Email Duplication Check .....	32
6.5.5 Request Keyword Lists .....	33
6.5.6 Save Keyword Lists .....	34
6.5.7 Progress Update .....	35
6.5.8 LLM Interaction .....	36
6.5.9 Study Code Update .....	37
6.5.10 Study Note Update .....	38
6.5.11 Study Note Update (Training Log) .....	39
 <b>7. Database Design .....</b>	 <b>40</b>
7.1 Objectives .....	40
7.2 ER Diagram .....	40
7.2.1 Account Table .....	41
7.2.2 Curriculum Table .....	42
7.2.3 Module .....	42
7.2.4 Curriculum Progress .....	43
7.2.5 Training Log .....	43
7.2.6 User Code .....	44
7.2.7 LLM Data .....	44
7.3 SQL DDL .....	45
7.3.1 Account .....	45
7.3.2 Curriculum .....	45

7.3.3 Module.....	40
7.3.4 Curriculum Progress.....	41
7.3.5 Training Log.....	41
7.3.6 User Code.....	41
7.3.7 LLM Data.....	42
<b>8. Testing Plan.....</b>	<b>49</b>
8.1 Objectives.....	49
8.2 Testing Policy.....	49
8.2.1 Development Testing.....	49
8.2.2 Release Testing.....	69
8.2.3 User Testing.....	69

# 1

# Purpose

## 1.1 Readership

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

## 1.2 Scope

본 프로젝트는 코딩교육에 LLM 을 접합해 사용자에게 서비스를 제공하는 웹 코딩 교육 플랫폼이다. 따라서 LLM 과 코딩교육에 대한 이해가 필요하다.

## 1.3 Objective

이 소프트웨어 설계 문서의 주요 목적은 CodeSunbi 코딩 교육 프로그램의 기술적 설계(design)에 대한 설명이다. 이 문서는 실습 프로그램의 구현의 기반이 되는 소프트웨어 Front-end, Back-end, Database, LLM 측면에서의 설계를 정의한다. 모든 설계는 앞서 제작된 Software Requirements Specification 문서의 요구사항을 기반으로 작성되었다.

## 1.4 Document Structure

- 1) Purpose: 본 문서의 목적, 예상 독자 및 문서의 구조에 대해서 설명한다.

- 2) Introduction: 본 문서를 작성하는데 사용된 도구들과 다이어그램들, 참고 자료들에 대해 서술한다.
- 3) Overall System Architecture: 시스템의 전체적인 구조를 Context Diagram, Sequence Diagram, Use case 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 Context Diagram, Class Diagram, Sequence Diagram 을 이용하여 서술한다.
- 6) Protocol Design: 클라이언트와 서버의 커뮤니케이션을 프로토콜 디자인으로 서술한다.
- 7) Database Design: 시스템의 Database Requirements 를 기반으로 Entity Relationship diagram, SQL Data Definition Language 을 이용하여 시스템의 데이터베이스 디자인을 서술한다.
- 8) Testing Plan: 시스템을 위한 테스트 계획과 pseudo code 를 서술한다.

# 2

## Introduction

### 2.1 Objectives

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

### 2.2 Applied Diagrams

#### 2.2.1 Used Tolls

Microsoft PowerPoint 발표용 자료 제작 프로그램인 PowerPoint 를 통해 기본적인 도형이나 아이콘을 생성하여서 다이어그램을 그릴 수 있다.

#### 2.2.2 Use Case Diagram

Use case Diagram 은 시스템에서 제공해야 하는 기능이나 서비스를 명세화한 diagram 이다. 시용자와 use cases 간의 관계를 보여주며, 사용자와 시스템 간 상호작용을 표현한다. User 를 Unregistered User, Registered User 로 나누어서 Unregistered User 에게는 system 이 register 기능만 제공하고 Registered User 에게는 System 이 Key-word selection, Curriculum learning, LLM Q&A 의 서비스를 제공할 수 있도록 설계하였다.



### 2.2.3 Sequence Diagram

Sequence Diagram 은 시간순서로 행동별로 어떤 객체와 어떻게 상호작용을 하는지 표현하는 Diagram 이다. 해당 Diagram 은 시나리오와 관련된 객체와 시나리오의 기능수행에 필요한 객체 간에 교환되는 메시지를 순서대로 표현한다.

### 2.2.4 Class Diagram

Class Diagram 은 시스템을 구성하는 클래스, 그 속성, 기능 및 객체들 간의 관계를 표현하여 시스템의 정적인 부분을 보여준다. 이 시스템에서는 크게 User, client, server 가 있다고 보았다.

### 2.2.5 Context Diagram

Context Diagram 은 Data Flow Diagram 에서 가장 높은 수준의 Diagram 으로, 시스템 전체와 외부 요인의 입력 및 출력을 보여준다. 시스템과 해당 시스템과 상호 작용할 수 있는 모든 외부의 엔티티들을 나타내며 그 사이의 정보의 흐름을 표현한다.

### 2.2.6 Entity Relationship Diagram

ER Diagram 은 구조화된 데이터들과 그들과의 관계를 사람이 이해할 수 있는 형태로 표현하는 다이어그램이며, 현실에서의 요구사항들을 이용한 데이터베이스 설계과정에서 활용된다. 해당 다이어그램은 Entity, Attribute, Relationship 으로 구성이 되며 해당 Diagram 을 통해서 데이터베이스의 논리적 구조를 설명한다.

### 2.2.7 Project Scope

본 문서에서 설계하는 CodeSunbi 웹 코딩 교육 프로그램은 사용자들로 하여금 흥미가 가는 분야를 선택하고 해당 분야에 대한 코딩교육을 받음으로써 코딩 교육에 흥미를 쉽게 가질 수 있도록 만들어졌다. 교육을 통하여 자신만의 프로젝트를 진행한 사용자가 더욱 심화된 과정이나 다른 다양한 분야에 관심을 가질 수 있도록 돕고자 한다.

### **2.2.8 References**

IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements •  
Specifications, In IEEE Xplore Digital Library

# 3

## System Architecture

### - Overall

#### 3.1 Objectives

이 챕터에서는 front end 설계에서 back end 설계에 이르는 프로젝트 어플리케이션의 시스템 구성에 대해 설명한다.

#### 3.2 System Organization

이 서비스는 client-server 모델을 적용하여 설계되었으며, front end 에서 user 와 service 간의 상호작용을 담당한다. client 는 user 와의 interaction 을 담당하고 있는데 login 과정과 커리큘럼 선택, 그리고 코딩 과정이 그것이다. client 는 user 가 입력한 ID 나 password, 그리고 직접 문제 풀이로 작성한 코드를 server 에 보내게 되는 데 이때 HTTP 통신을 이용한다. server 는 client 로부터 데이터를 받아서 database 에 있는 data 를 이용해 user 에 대한 data 를 읽게 된다. Database 와의 communication 은 MySQL 을 이용한다. user 가 보낸 코드는 AI model 로 전달되어서 해당 코드에 대한 분석과 정답 유무, 필요한 서비스 등이 산출되고 이것은 다시 user 의 front end 로 전달된다. 또한 user 가 작성한 코드와 그에 대한 AI model 의 산출물 등은 database 로 전해져서 user data 로 관리된다. 이 user data 는 user 가 다음 교육을 진행할 때와 포트폴리오를 생성하는 데 이용된다. 또한 user 가 선택한 커리큘럼 또한 user data 로 database 에 전해져서 관리되어 지게 된다. 만약 신규 이용자가 계정을 생성을 하려고 할 때에는 ID 가 기존에 database 에 있던 ID 와 겹치지 않게 확인하는 작업 또한 진행된다.

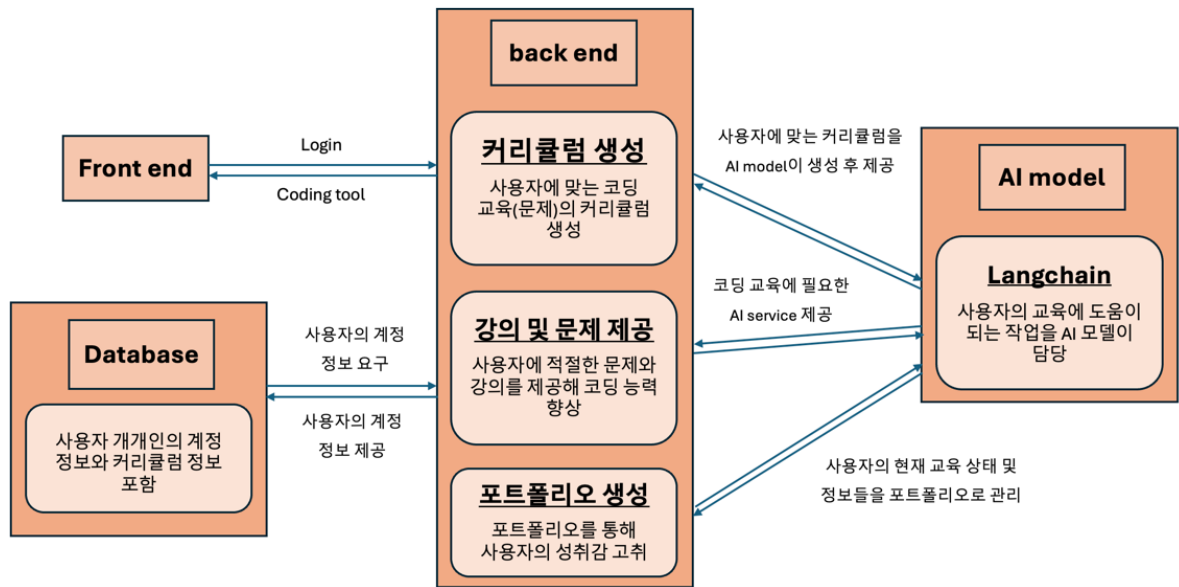


Figure 3.1 System Design

### 3.2.1 System Diagram

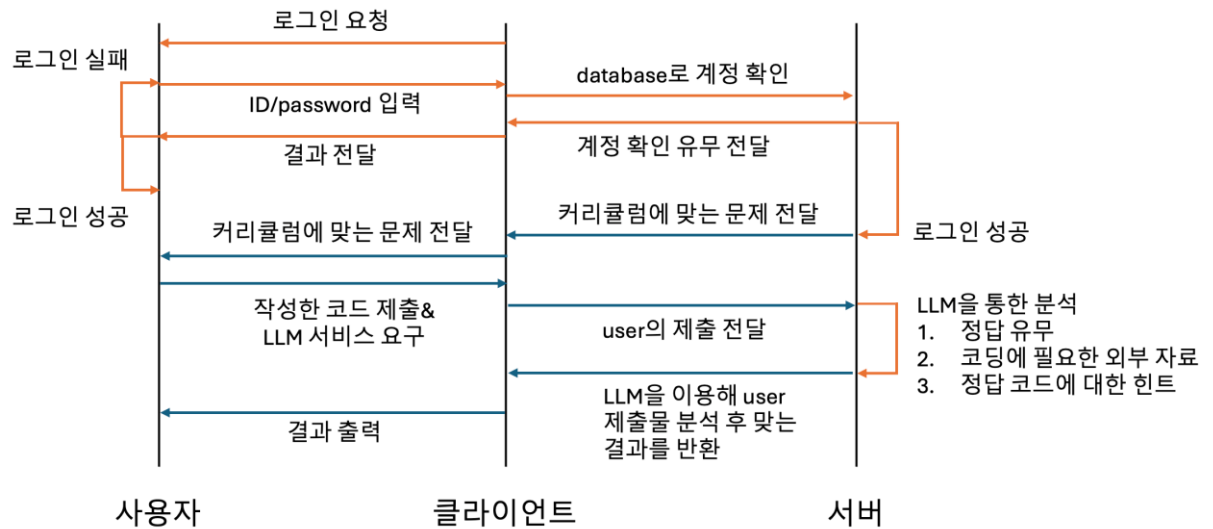


Figure 3.2 Overall System Diagram

### 3.3 Use Case Diagram

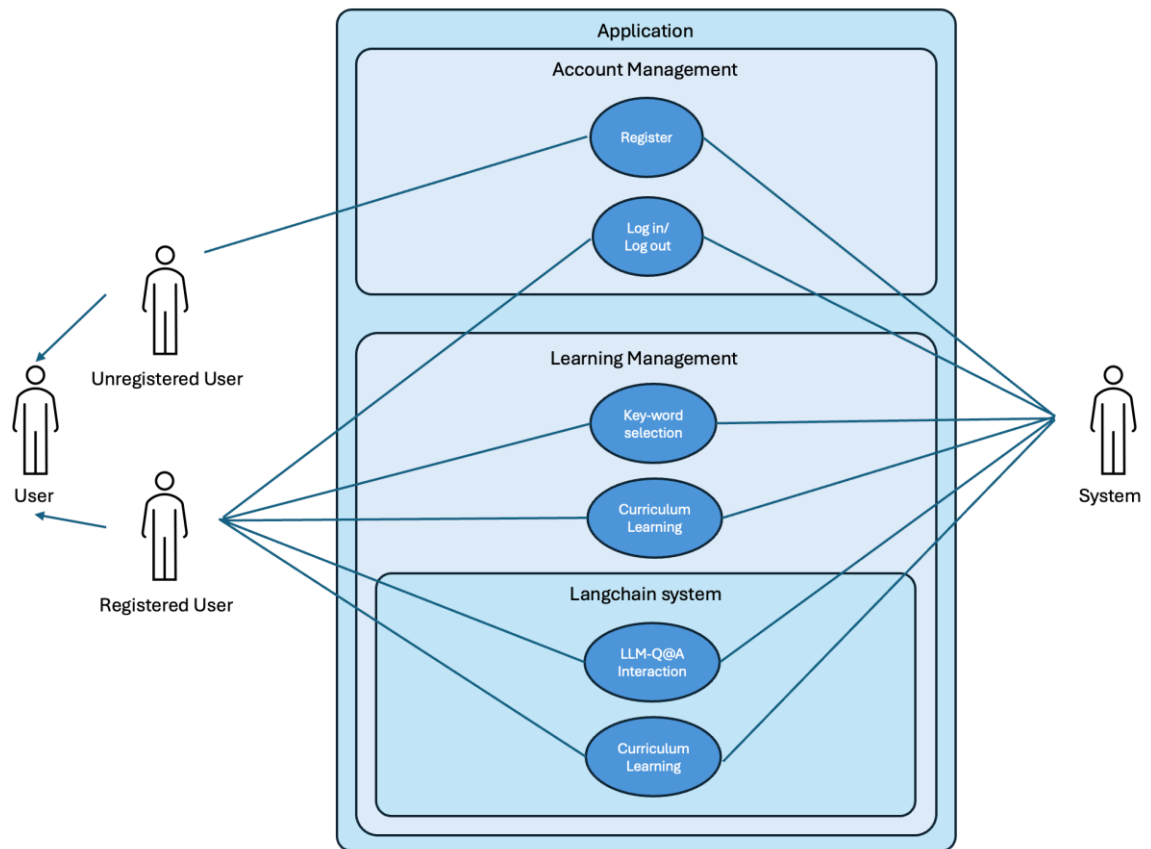


Figure 3.3 System Use Case Diagram

# 4

## System architecture

### -Fronted

#### 4.1 Objectives

이 장에서는 프론트엔드 시스템의 구조 및 속성, 기능에 대해 설명하고 과정에 대한 간단한 도식으로 표현한다.

##### 4.1.1 profile

사용자 계정으로 나타내지는 프로필 클래스 세부 정보이다. 사용자는 시스템을 이용하기 위해 아래 클래스에 맞게 기본 정보를 입력해야 한다. 사용자는 등록 후 자신의 정보를 수정할 수 있다. ID 와 Password 이외의 정보는 선택사항으로 두어 개인정보의 과도한 이용 및 유출 위험을 막는다. 또한 첫 로그인이라고 판단될 경우 keyword 선택을 통한 커리큘럼을 생성한다.

#### Objectives

- 필수 사항
  - 1) ID(email address) : 사용자의 이메일. 이를 아이디로 사용
  - 2) Password : 사용자가 지정한 비밀번호
  - 3) Keyword : 사용자가 회원 가입 시 선택하는 키워드
- 선택 사항
  - 1) Name : 사용자의 이름

- 2) Age : 사용자의 나이
- 3) Gender : 사용자의 성별
- 4) Phone number : 사용자의 연락처

## Methods

- 1) Create\_ID() : ID 를 포함한 계정 생성
- 2) Get\_ID() : 생성된 계정 받기
- 3) Get\_keyword() : 키워드 받기
- 4) Recommend\_Cr() : 키워드 바탕으로 커리큘럼 추천

## Class Diagram

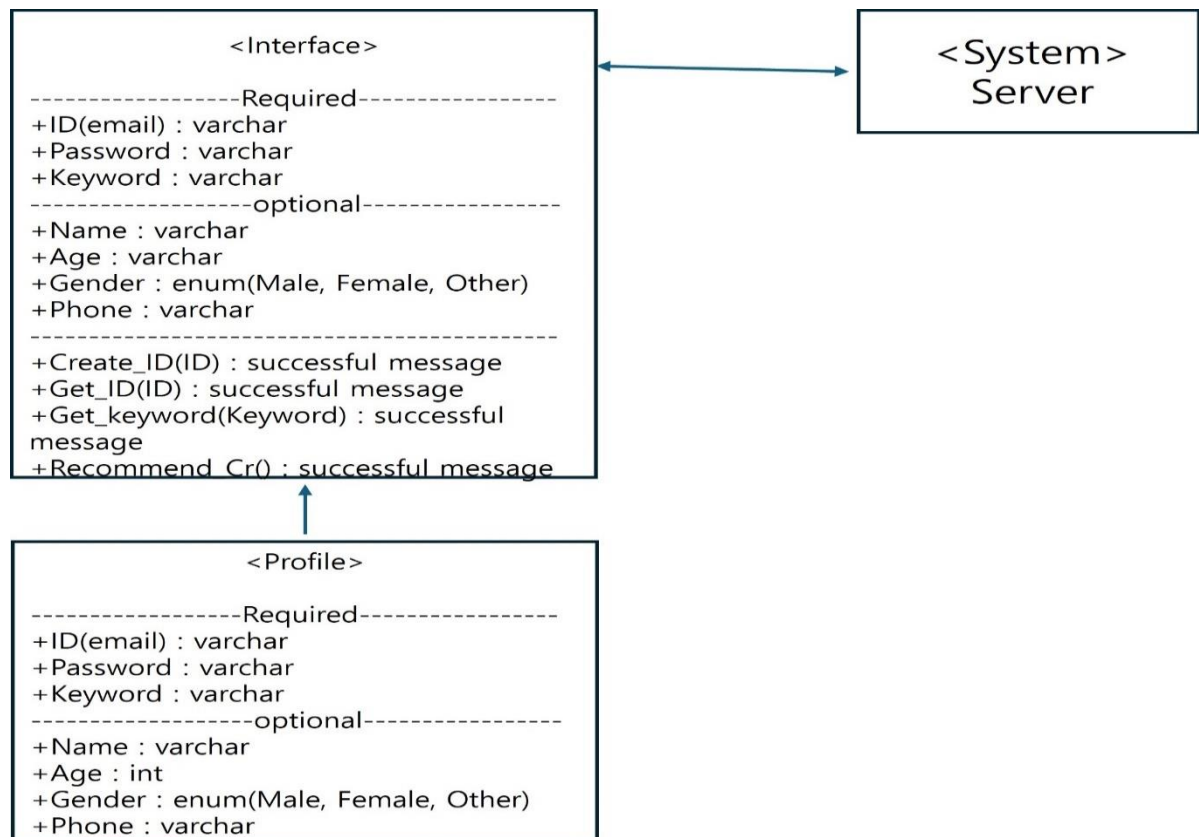


Figure 4.1 Profile Class Diagram

## Sequence Diagram

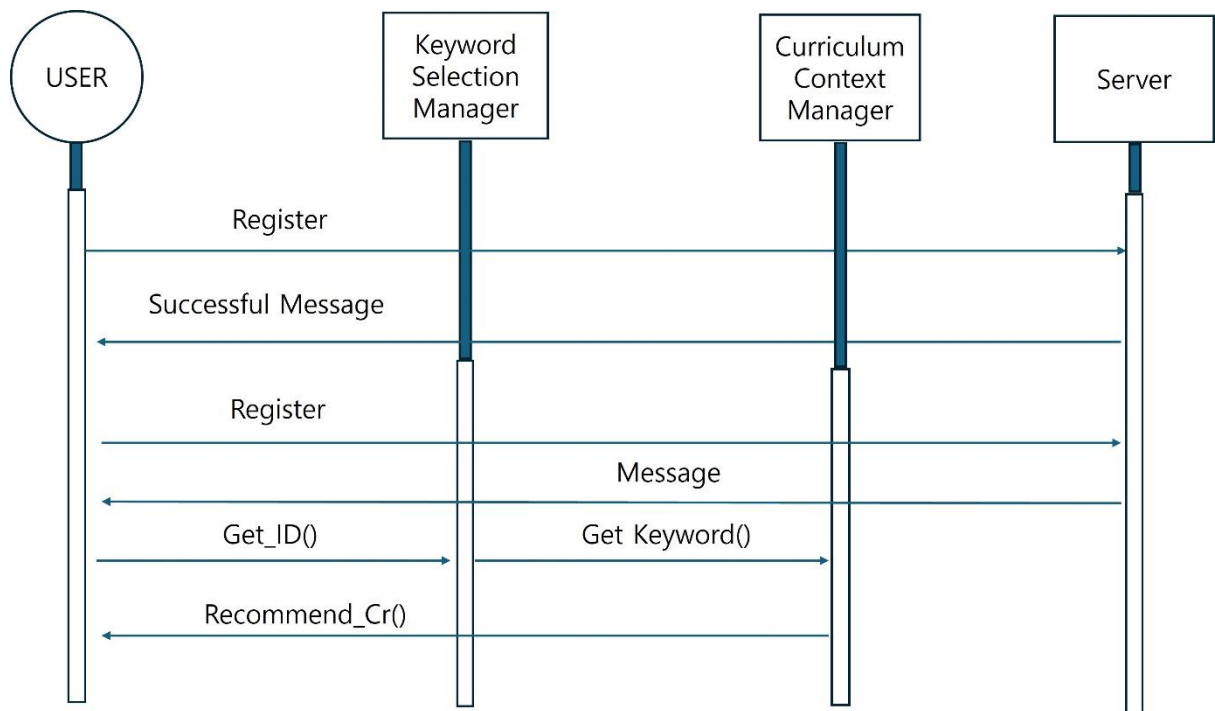


Figure 4.2 Profile Sequence Diagram

### 4.1.2 LLM Q&A

사용자가 커리큘럼에 해당하는 과정을 수강하고 있을 때, 필요시 LLM 을 이용하여 코드에 관해 질의 응답을 할 수 있습니다. 이때, 주어진 문제에 대한 해결책 혹은 도움이 되는 강의 추천 등을 받을 수 있으며, 이 과정에서 경우에 따라 LLM Q&A Manager 혹은 RAG Manager 를 통하게 됩니다. RAG manager 의 경우 외부 소스를 참조하여 강의 추천 등을 하게 됩니다.

## Objectives

- 1) ID : 사용자의 ID
- 2) Code : 사용자의 코드
- 3) External Source : LLM 모델이 참고하는 외부 데이터 set
- 4) Code : 사용자가 문제 풀이를 위해 작성한 Code
- 5) MSG : 사용자의 코드 및 자연어가 저장된 형태



## Methods

- 1) Get\_msg() : 사용자로부터 자연어 입력
- 2) Send\_msg() : 받은 자연어를 서버로 전달
- 3) Get\_ans() : 외부 소스에서 참고하여 생성한 대답을 서버로부터 받기
- 4) Send\_ans() : 서버로부터 받은 답변을 사용자에게 전달
- 5) Put\_Msg, Receive\_Msg : 받은 Msg 를 알맞은 형태로 변형하여 외부 소스에 전달, 참조

## Class Diagram

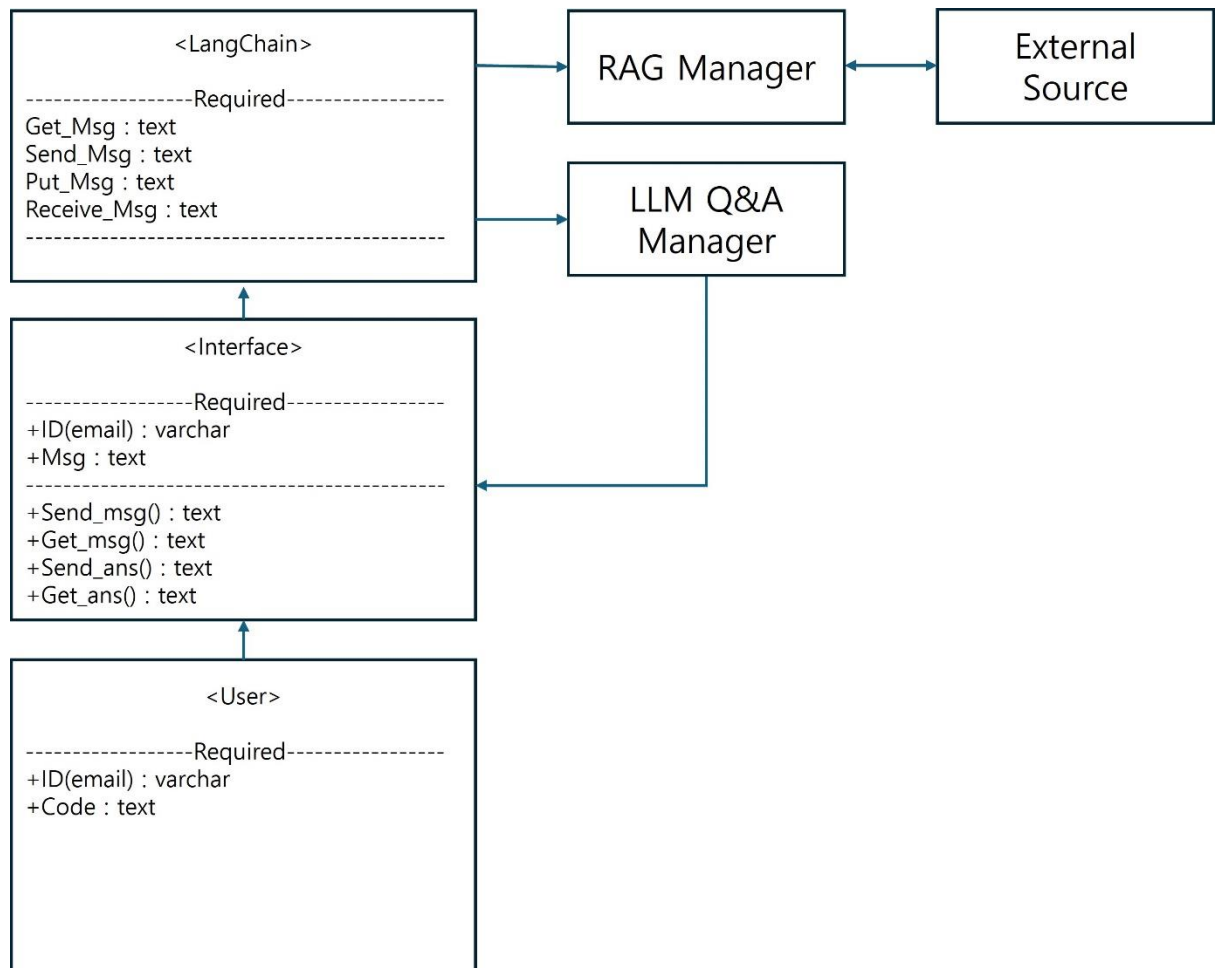


Figure 4.3 LLM Q&A Class Diagram

## Sequence Diagram

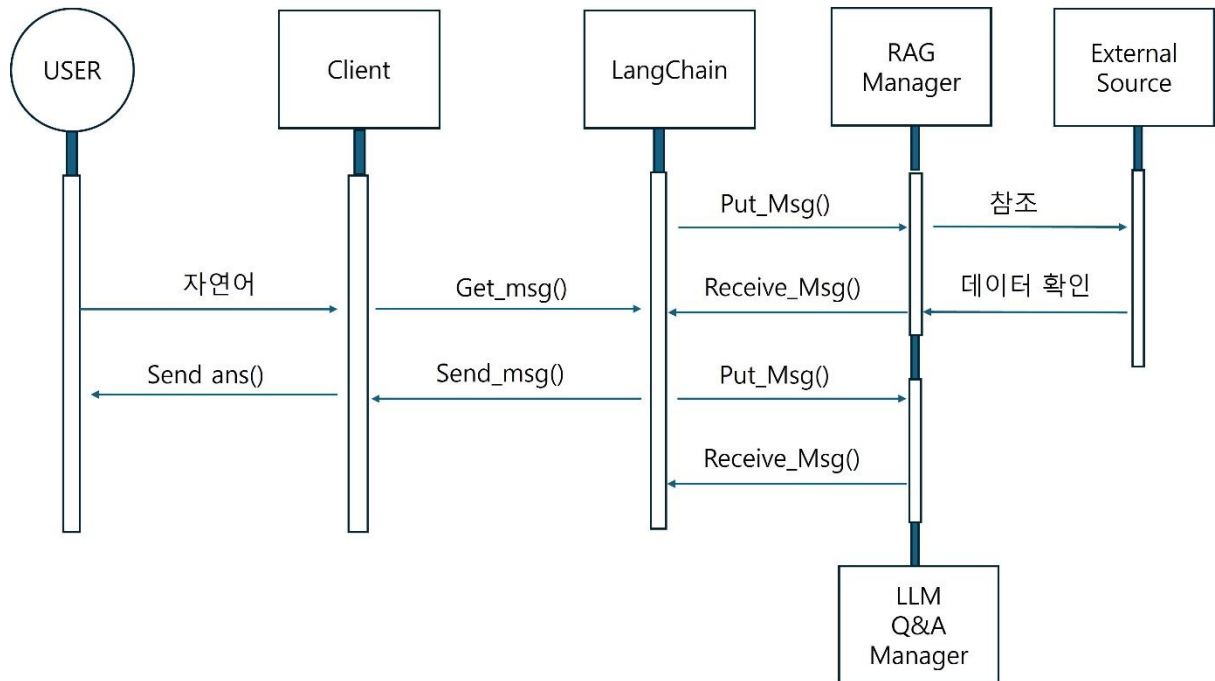


Figure 4.4 LLM Q&A Sequence Diagram

### 4.1.3 Dialogue save

사용자는 LLM 과의 나눈 대화를 각 커리큘럼에 맞게 database 에 저장한다. 이를 통해 이후 LLM 의 능력을 향상시키고 맥락에 더 알맞게 대답할 수 있도록 update 할 수 있으며, 학습을 재개할 때도 저장된 대화를 바탕으로 좀 더 사용자가 원하는 대답을 말하게 할 수 있다.

## Objectives

- 1) ID : 사용자의 ID
- 2) External Source : LLM 모델이 참고하는 외부 데이터 set
- 3) Dialogue : 사용자와 LLM 간의 대화 내용

## Methods

- 1) Get\_DL() , Send\_DL() : Dialogue 를 받기, 전달하기

- 2) Update() : 참조한 정보 및 데이터베이스를 바탕으로 맥락에 알맞게 조정
- 3) Query, Return Status : Dialogue 와 관련된 정보를 외부 소스에서 참조.

### Class Diagram

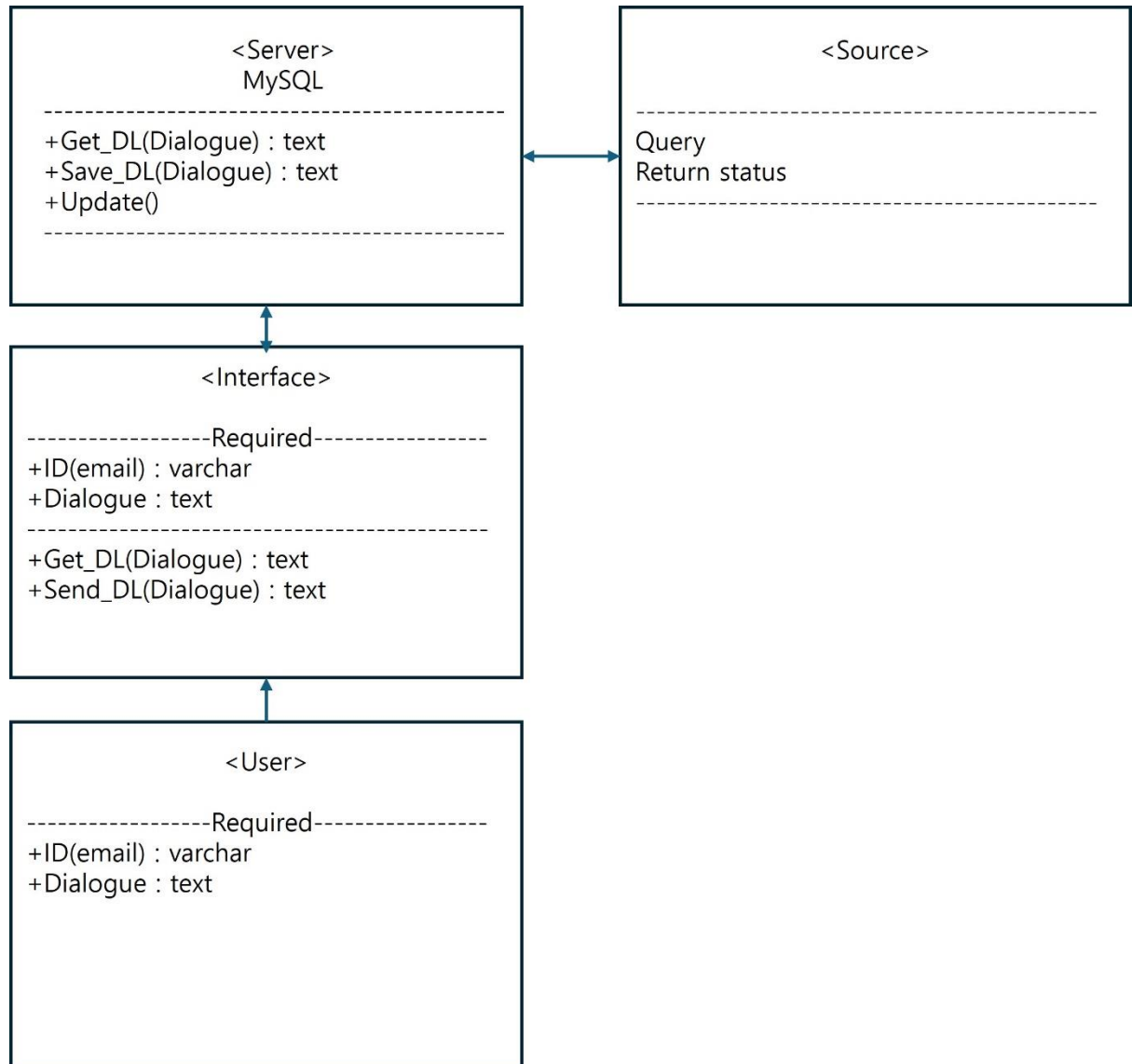


Figure 4.5 Dialogue Save Class Diagram

### Sequence diagram

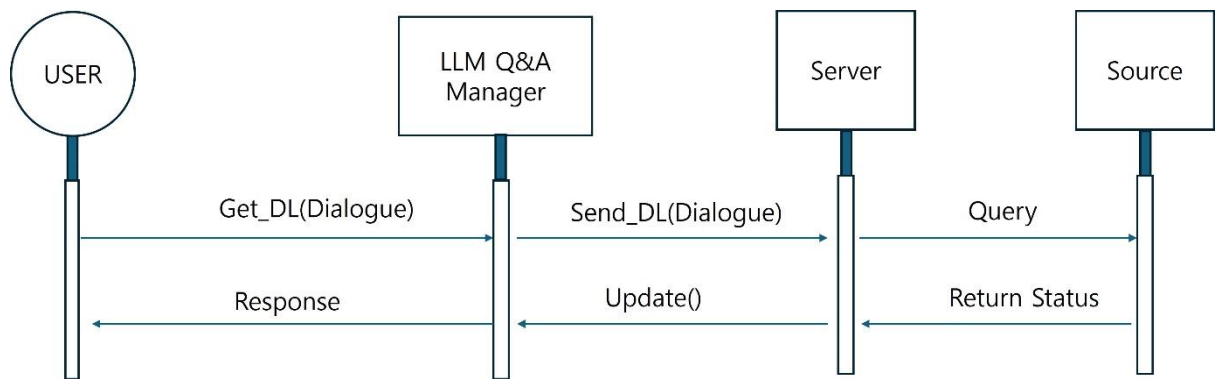


Figure 4.6 Dialogue Save Sequence Diagram

# 5

## System Architecture

### – Backend

#### 5.1 Objectives

해당 챕터는 back-end 시스템의 구조에 대해서 설명한다.

#### 5.2 Objectives

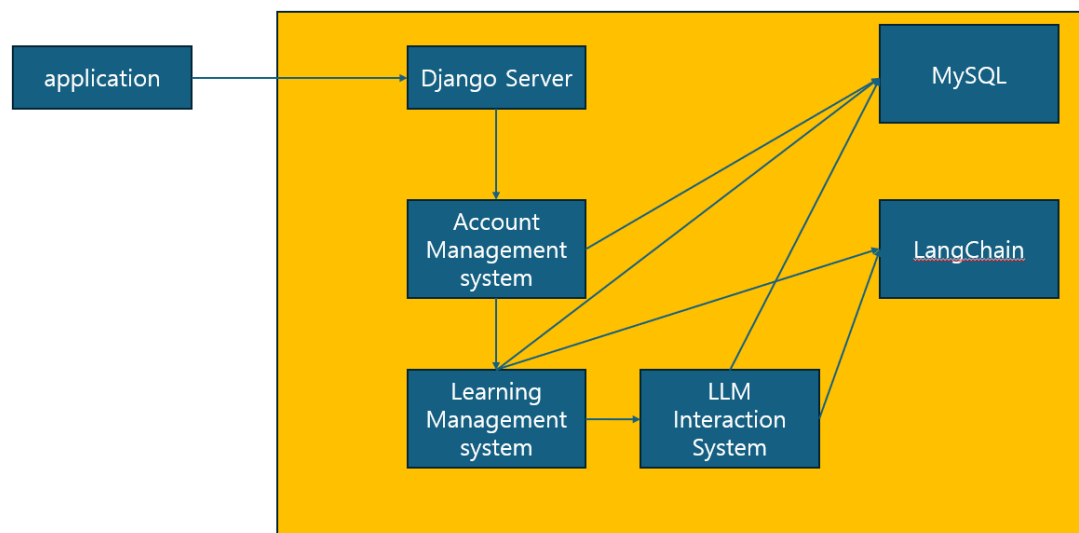


Figure 5.1 Overall Architecture

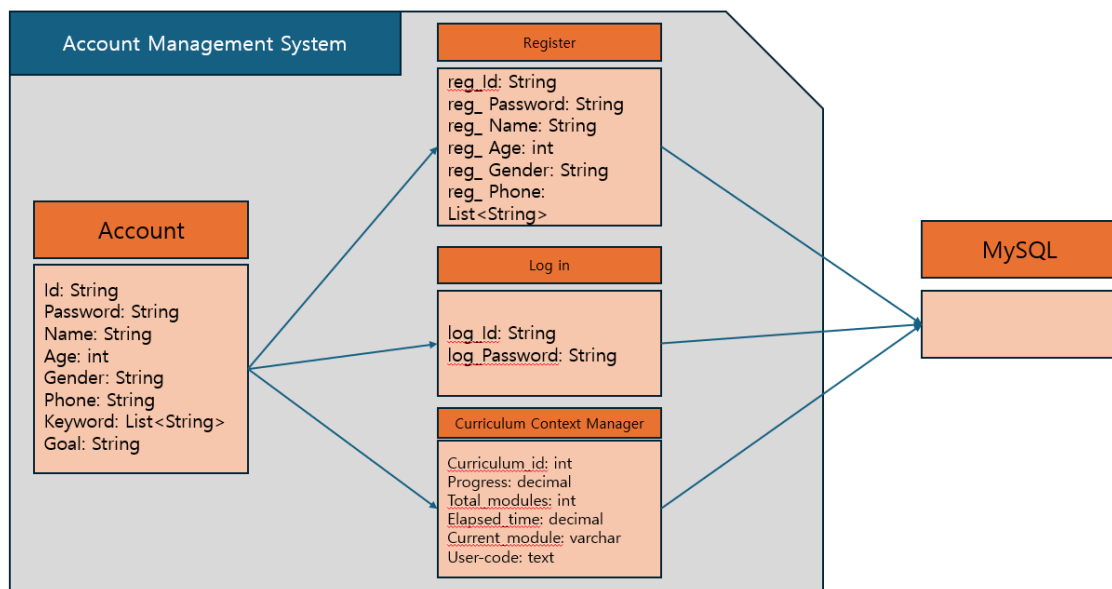
사용자로 하여금 관심있는 분야를 선택하고 해당분야에 대해서 학습을 도와주는 본 시스템의 구조는 위와 같다. Front-end 로부터 요청이 들어오면 Django server 에서

해당 요청을 처리하는 function 을 실행한다. Back-end 에서 처리하는 요청으로는 회원가입 처리, 로그인 처리, 커리큘럼 생성 처리, 강의 및 문제 제공, LLM 을 이용한 질의, 포트폴리오 생성 등이 있으며, Mysql 데이터베이스를 이용해서 처리한다.

## 5.3 Subcomponents

### 5.3.1 Account management System

#### Class Diagram



**Figure 5.2 Overall Architecture - Account Management System**

Register class: 입력 받은 Email 주소가 중복되어 있는지 확인한다. 비밀번호가 올바른 양식으로 입력 받았는지 확인한다. 형식이 올바르면 입력 받은 정보를 Database 에 새롭게 추가한다.

Login class: 입력 받은 데이터가 기존 Database 에 존재하는지 확인하고, 정보가 일치한다면 해당 class 의 정보를 불러온다.

Curriculum Context Manger class: 로그아웃 시에 progress 의 정도를 갱신하고 해당 account 의 상태를 로그아웃 상태로 바꾼다.

## Sequence Diagram

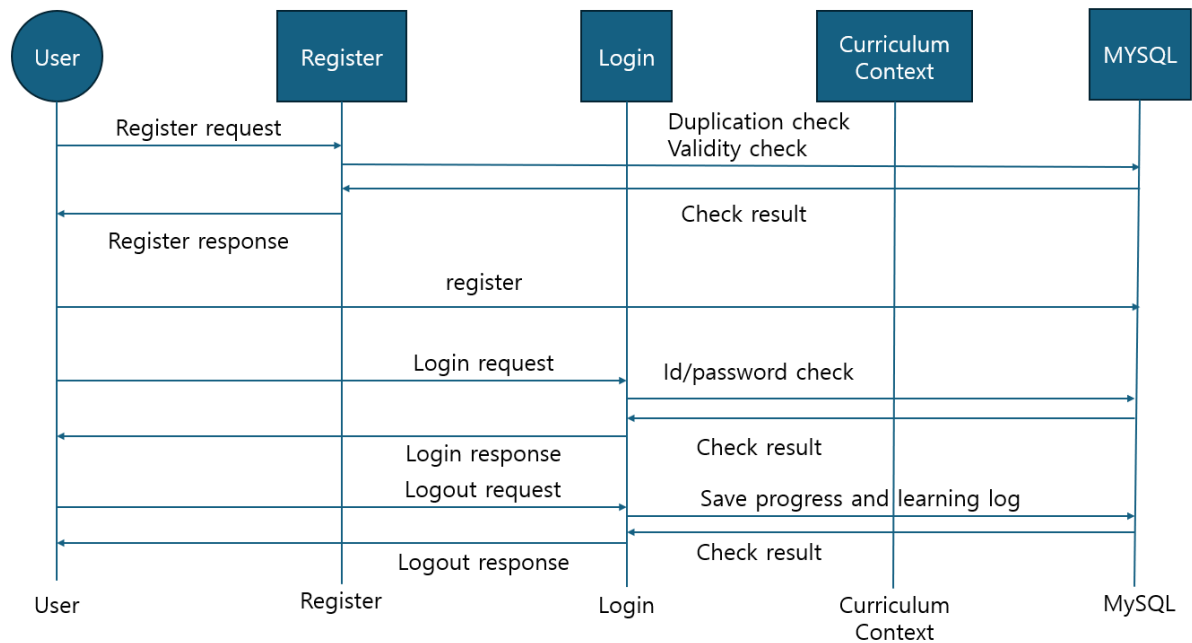


Figure 5.3 Sequence Diagram - Account Management System

## 5.3.2. Learning Management System

### Class Diagram

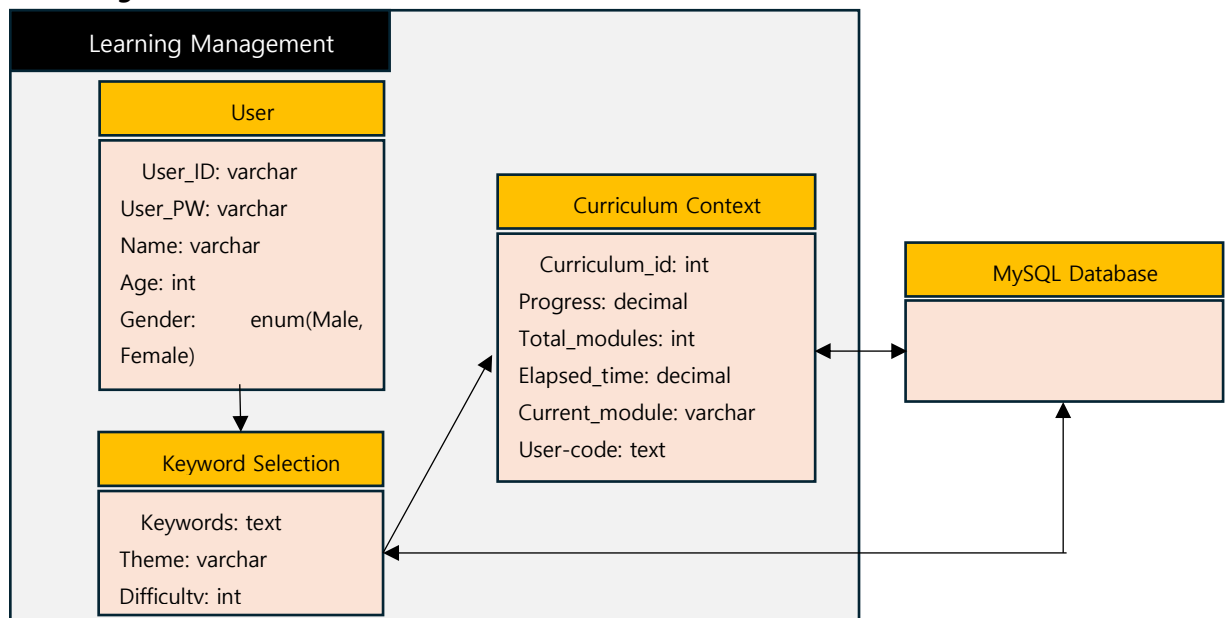
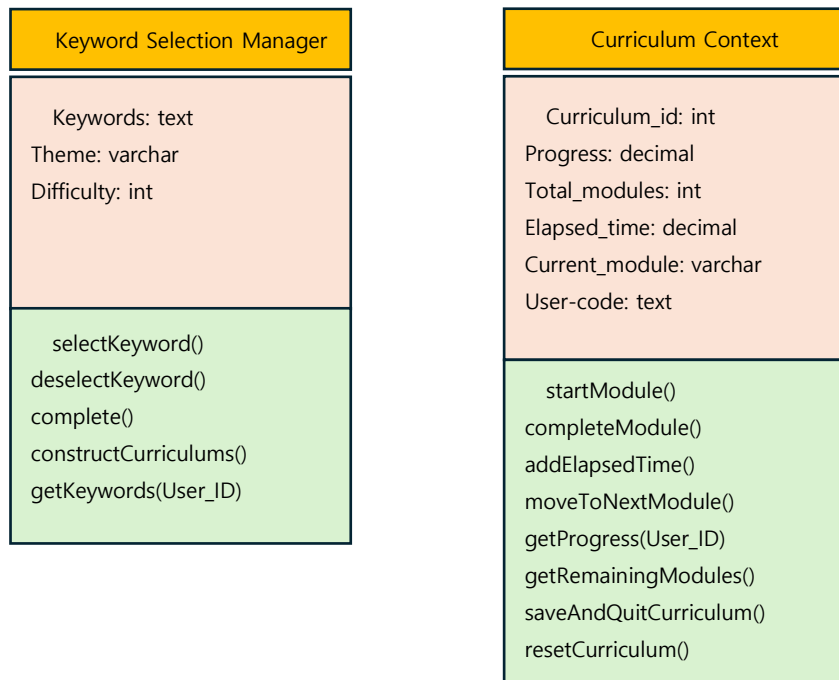


Figure 5.4 Overall Architecture



**Figure 5.5 Detailed Class Diagrams**



## Sequence Diagram

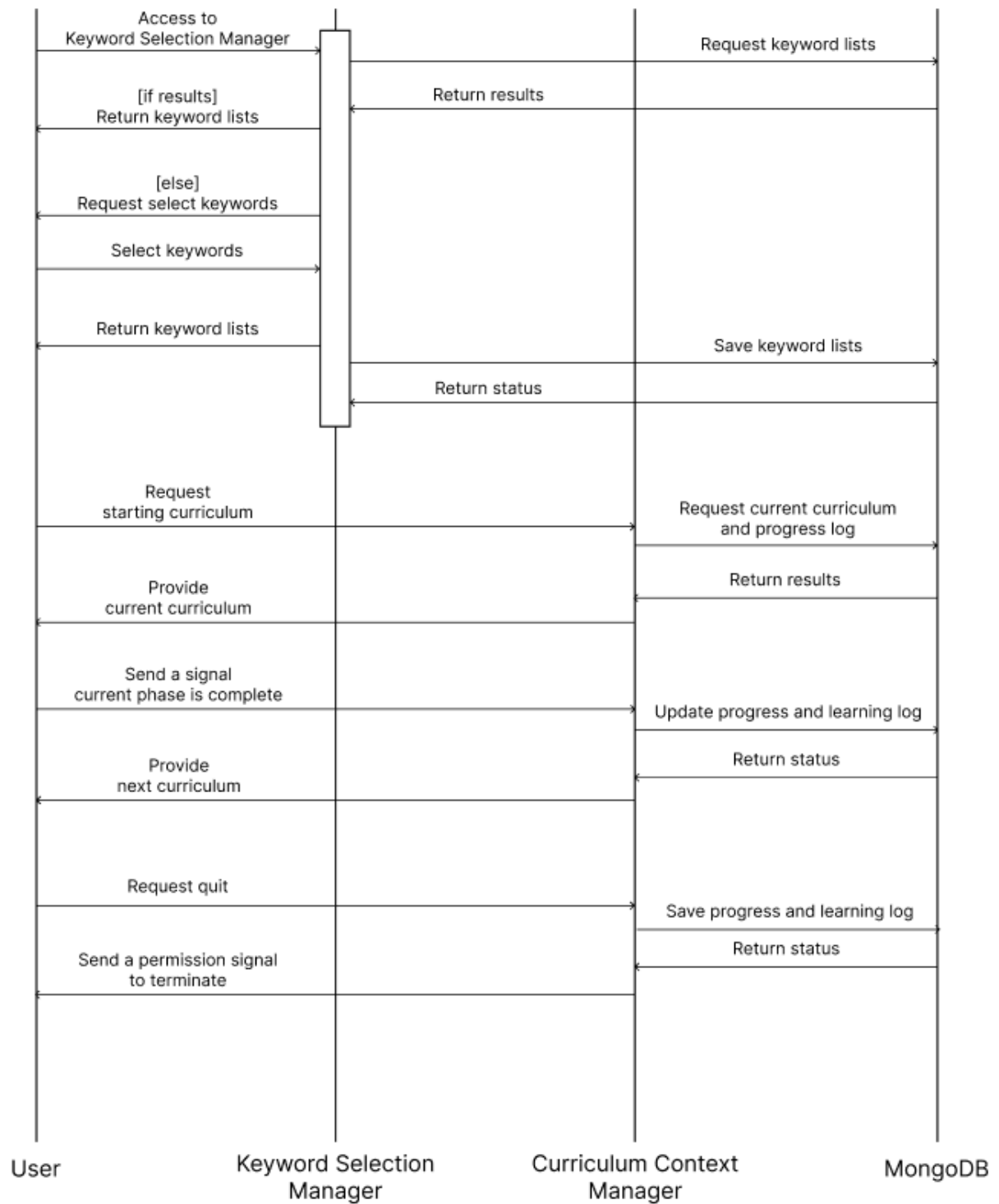


Figure 5.6 Detailed Sequence Diagram

### 5.3.3. LLM Interaction System

#### Class Diagram

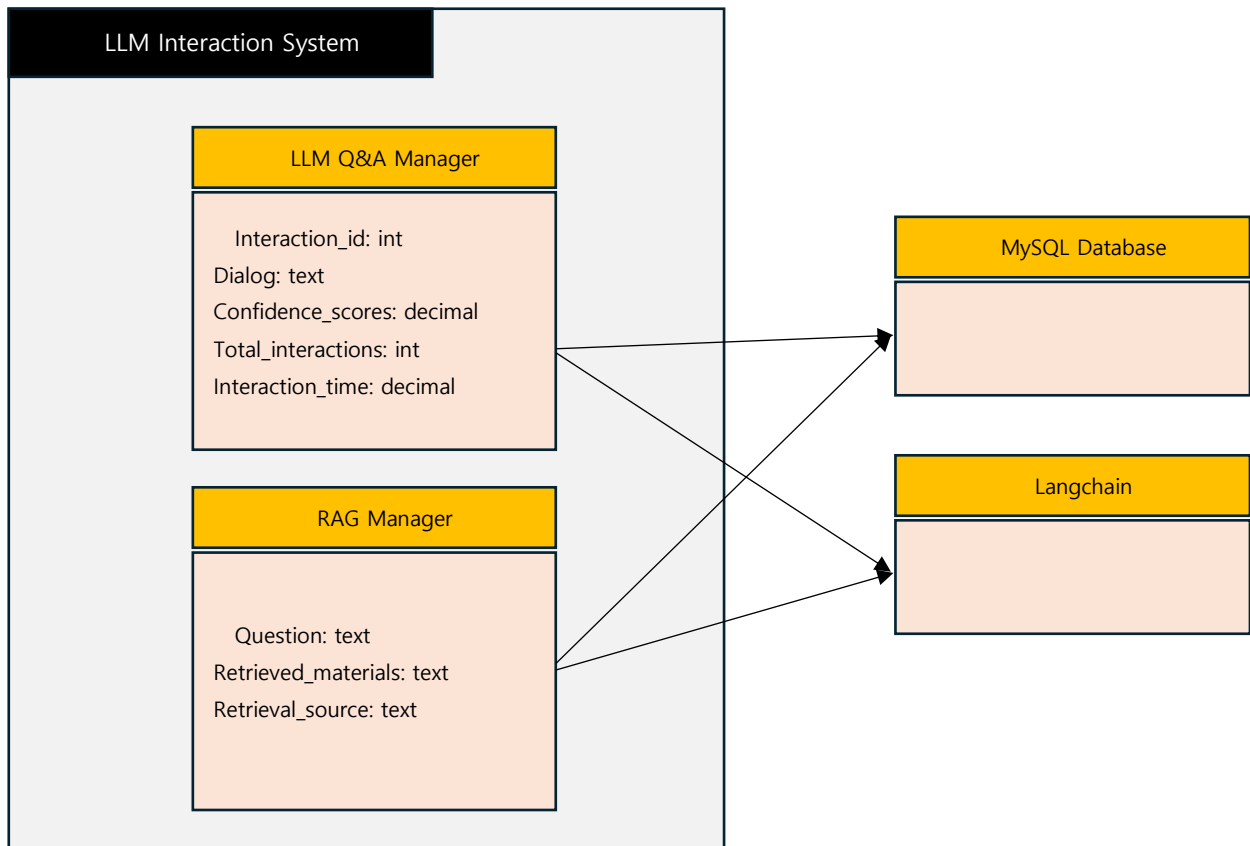


Figure 5.7 Overall Architecture

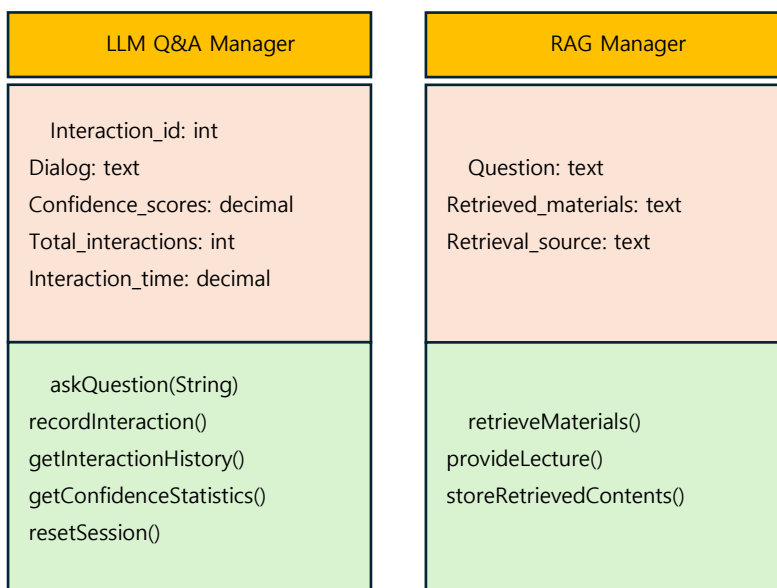


Figure 5.8 Detailed Class Diagram

## Sequence Diagram

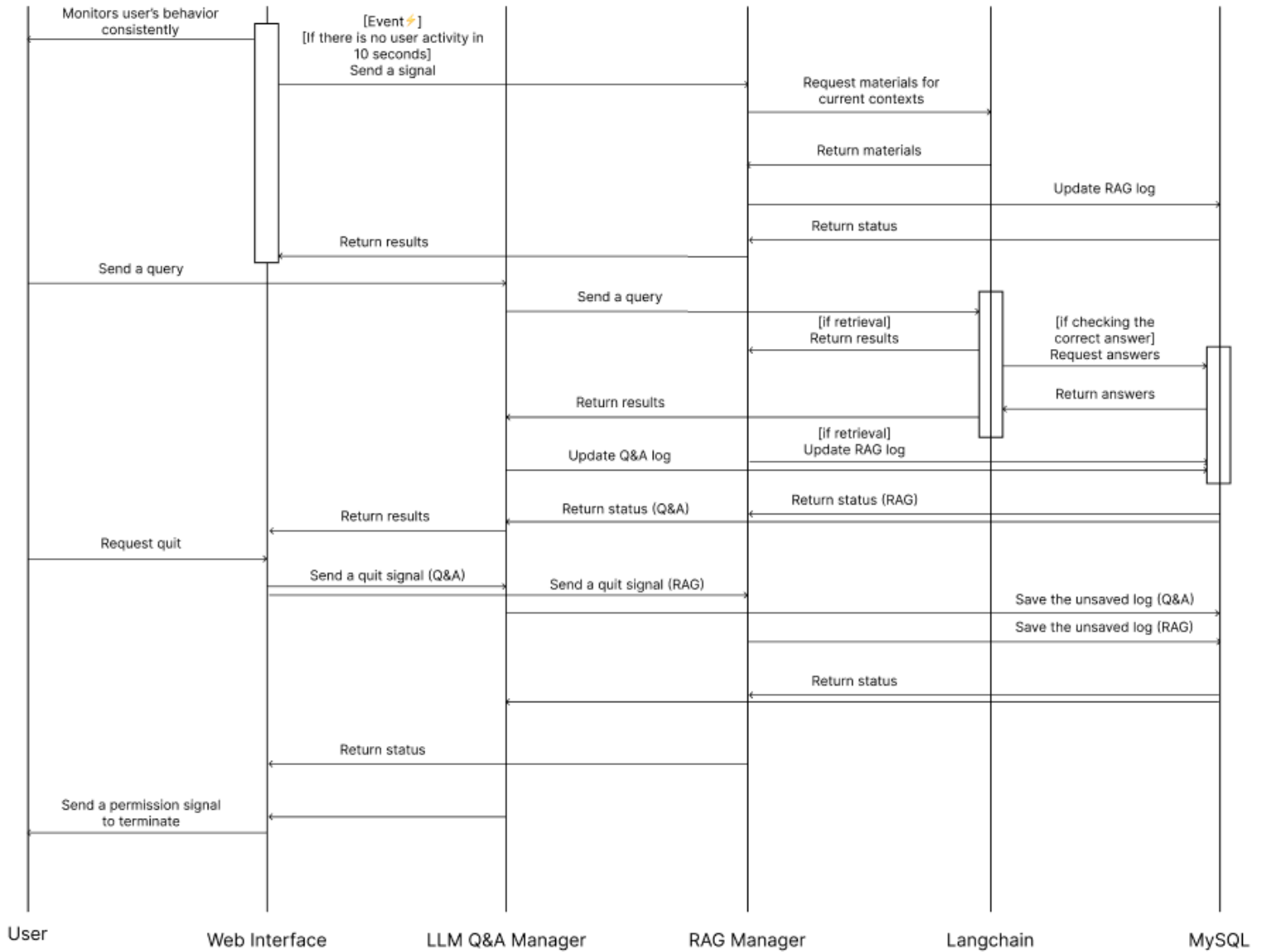


Figure 5.9 Detailed Sequence Diagram

LLM interaction system 은 웹 인터페이스 상태를 추적하여 10 초 이상 사용자가 어떠한 행동도 하지 않는다면 어려움을 겪는다고 판단하여 추가 학습을 위한 정보를 제공해준다. Q&A session 에서는 아래의 3 가지 일이 가능하다.

1. 일반적인 Q&A 는 단순히 사용자와 LLM 의 interaction 으로 달성한다.
2. 만약 자료 제공을 원하는 Q&A 라면 LLM 은 외부 웹 환경에서 소스를 검색하고, LLM 은 이를 가공하여 사용자에게 제공한다.
3. 코드 작성의 정답 여부를 물어보는 Q&A 라면 LLM 은 MySQL 데이터베이스에서 관련된 정답들을 가져온 후, 이들과의 유사성을 바탕으로 결과를 제공한다.

# 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

Ajax 는 Asynchronous JavaScript and XML 의 약자로 JavaScript 와 XML 형식을 이용한 비동기적 정보 교환 기법이다. Ajax 는 JSON, XML, HTML 그리고 일반 텍스트 형식을 포함한 다양한 포맷을 주고받을 수 있다. Ajax 의 장점은 사용자의 이벤트가 있을 때 전체 페이지를 리프레쉬 하지 않고서도 비동기적으로 페이지를 업데이트 시켜 줄 수 있다.

### 6.3 TLS and HTTPS

TLS(Transport Layer Security)는 국제 인터넷 표준화 기구에서 표준으로 인정한 정보의 무결성을 보장하기 위한 암호화 프로토콜이다. HTTPS(HyperText Transfer Protocol Secure)는 HTTP 를 TLS 위에서 패킹해서 보안성을 강화한 것이다.

### 6.4 CSRF and CSRF Token

CSRF 는 사용자의 의도와 상관없이 악의적인 웹사이트가 사용자를 대신하여 원치 않는 요청을 서버에 보낼 수 있도록 하는 공격 기법이다. CSRF Token 은 CSRF 공격을

방지하기 위한 보안 기법이다. 서버가 각 사용자 요청마다 고유한 토큰을 생성하고 이를 클라이언트와 공유하여 요청의 유효성을 확인하는 기법이다.

## 6.5 Authentication

### 6.5.1 Register

#### Request

Attribute	Detail	
Protocol	HTTPS	
Method	POST	
Success Response Body	ID(Email)	사용자 이메일
	Password	사용자 비밀번호
	Name	사용자 이름
	age	사용자 나이
	gender	사용자 성별
	Phone number	사용자 휴대전화 번호
	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	Message	가입완료 메시지
	Session ID	접근하기 위한 session ID
Failure Response Body	Message	Failure message

Figure 6.2 Table of Register Response

## 6.5.2 Login

### Request

Attribute	Detail	
Protocol	HTTPS	
Method	POST	
Success Response Body	ID(Email)	사용자 이메일
	Password	사용자 비밀번호
	CSRF Token	CSRF Token issued by server

Figure 6.3 Table of Login 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
	Session Token	Token for ID session
Failure Response Body	Message	Failure message

**Figure 6.4 Table of Login Response**

## 6.5.3 Logout

### Request

Attribute	Detail	
Protocol	HTTPS	
Method	POST	
Success Response Body	Session ID	Session ID to access

	진행률	업데이트하기 위한 진행률
	학습 로그	사용자의 학습 로그
	CSRF Token	CSRF Token issued by server

**Figure 6.5 Table of Logout 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 6.6 Table of Logout Response**



## 6.5.4 Email Duplication Check

### Request

Attribute	Detail	
Protocol	HTTPS	
Method	Ajax/ POST	
Success Response Body	ID(Email)	사용자 이메일

Figure 6.7 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	이메일의 중복 여부 확인
Failure Response Body	Message	Failure message

Figure 6.8 Table of Email Duplication Check Response

### 6.5.5 Request keyword lists

#### Request

Attribute	Detail	
Protocol	HTTPS	
Method	POST	
Success Response Body	Session ID	Session ID to access
	CSRF Token	CSRF Token issued by server

Figure 6.9 Table of Request Keyword Lists 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
	Keyword list	저장된 키워드들의 리스트
Failure Response Body	Message	Failure message

Figure 6.10 Table of Request Keyword Lists Response

## 6.5.6 Save keyword lists

### Request

Attribute	Detail	
Protocol	HTTPS	
Method	POST	
Success Response Body	Session ID	Session ID to access
	Keyword list	사용자가 관심 있는 키워드들의 리스트
	CSRF Token	CSRF Token issued by server

Figure 6.11 Table of Save Keyword Lists 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 6.12 Table of Save Keyword Lists Response

## 6.5.7 Progress Update

### Request

Attribute	Detail	
Protocol	HTTPS	
Method	POST	
Success Response Body	Session ID	Session ID to access
	CSRF Token	CSRF Token
	커리큘럼 ID	커리큘럼 ID
	진행률	진행률

Figure 6.13 Table of Process Update Request

### Response

Attribute	Detail	
Protocol	HTTPS	
Success Code	200 (OK)	
Failure Code	HTTP error code = 400(Bad Request, overlap)	
Success Response Body	Message	진행률 저장 완료 메세지
Failure Response Body	Message	실패 메세지

Figure 6.14 Table of Process Update Response

### 6.5.8 LLM Interaction

#### Request

Attribute	Detail	
Protocol	HTTPS	
Method	POST	
Success Response Body	Session ID	Session ID to access
	CSRF Token	CSRF Token
	Curriculum_Id	커리큘럼 ID
	User Question	사용자가 LLM 에 요청하는 질문

Figure 6.15 Table of LLM Interaction Request

#### Response

Attribute	Detail	
Protocol	HTTPS	
Success Code	200 (OK)	
Failure Code	HTTP error code = 400(Bad Request, overlap)	
Success Response Body	LLM Answer	사용자의 질문에 대한 LLM 의 답변 (자료, 정답 유무, 질문에 대한 답변 등등이 가능함)
Failure Response Body	Message	실패 메세지

Figure 6.16 Table of LLM Interaction Response

### 6.5.9 Study Code Update

#### Request

Attribute	Detail	
Protocol	HTTPS	
Method	POST	
Success Response Body	Session ID	Session ID to access
	CSRF Token	CSRF Token
	모듈 ID	모듈 ID
	Code	사용자가 학습한 코드

Figure 6.17 Table of Study Code Update Request

#### Response

Attribute	Detail	
Protocol	HTTPS	
Success Code	200 (OK)	
Failure Code	HTTP error code = 400(Bad Request, overlap)	
Success Response Body	Message	코드 저장 완료 메세지
Failure Response Body	Message	실패 메세지

Figure 6.18 Table of Study Code Update Response

### 6.5.10 Study Note Update

#### Request

Attribute	Detail	
Protocol	HTTPS	
Method	POST	
Success Response Body	Session ID	Session ID to access
	CSRF Token	CSRF Token
	모듈 ID	모듈 ID
	Note	사용자가 학습 과정에 작성한 메모들

Figure 6.19 Table of Study Note Update Request

#### Response

Attribute	Detail	
Protocol	HTTPS	
Success Code	200 (OK)	
Failure Code	HTTP error code = 400(Bad Request, overlap)	
Success Response Body	Message	노트 저장 완료 메세지
Failure Response Body	Message	실패 메세지

Figure 6.20 Table of Study Note Update Response

### 6.5.11 Study Note Update (TrainingLog)

#### Request

Attribute	Detail	
Protocol	HTTPS	
Method	POST	
Success Response Body	Session ID	Session ID to access
	CSRF Token	CSRF Token
	모듈 ID	모듈 ID
	Note	사용자가 학습 과정에 작성한 메모들
	duration_hours	사용자가 해당 모듈을 학습한 시간

Figure 6.21 Table of Training Request

#### Response

Attribute	Detail	
Protocol	HTTPS	
Success Code	200 (OK)	
Failure Code	HTTP error code = 400(Bad Request, overlap)	
Success Response Body	Message	노트 저장 완료 메세지
Failure Response Body	Message	실패 메시지

Figure 6.22 Table of Training Log Response



# Database Design

## 7.1 Objectives

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

## 7.2 ER Diagram

본 애플리케이션 시스템은 총 6가지 entity로 이루어져 있다; Account, Curriculum, Module, CurriculumProgress, TrainingLog, UserCode. ER-Diagram은 entity간의 관계, 그리고 entity와 attribute의 관계를 다이어그램으로 설명한다. 각 entity의 primary key는 해당 entity 내부의 attribute 이름 옆에 열쇠 기호로 표시되어 있다. 각 entity마다 대응되는 개수는 entity를 연결하는 선 주변에 표기되어 있어 확인 가능하다.

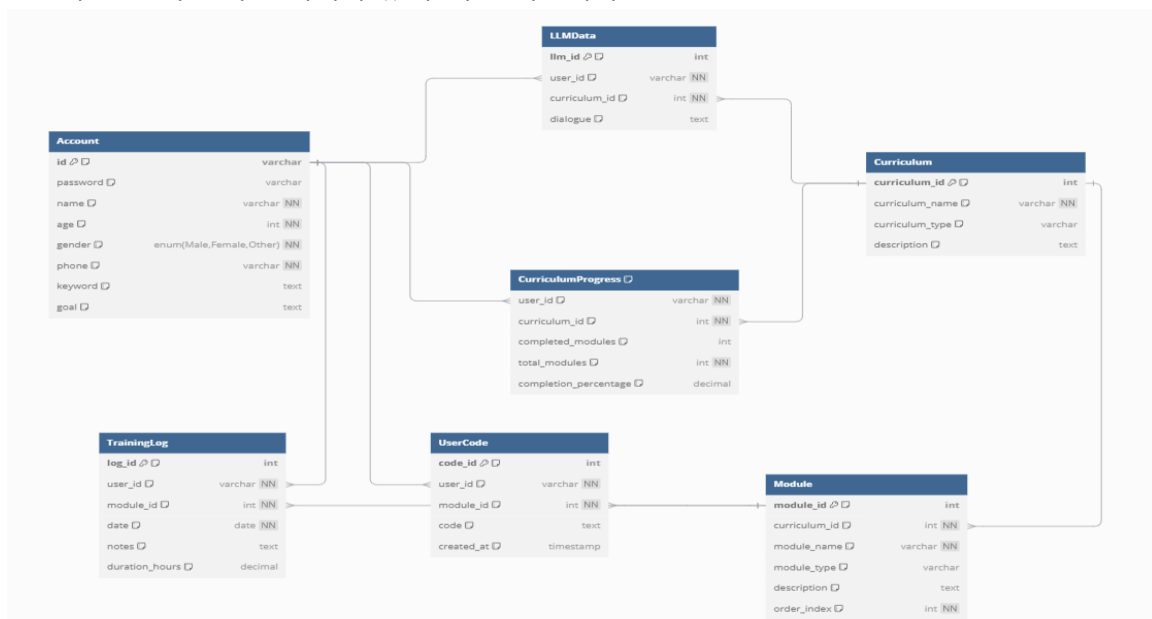


Figure 7.1 ER Diagram

### 7.2.1 Account Table

Account	
id	varchar
password	varchar
name	varchar NN
age	int NN
gender	enum(Male, Female, Other) NN
phone	varchar NN
keyword	text
goal	text

Figure 7.2 Account Table

Account Entity 는 애플리케이션 사용자에게 해당한다. Account Entity 는 id, pwd, name, age, gender, phone, keyword, goal 을 attribute 로 담고 있다. 이 중 id 가 앞에서 말했던 것과 같이 열쇠 기호를 포함하고 있기에 primary key 로 존재한다. 이 attribute 들은 사용자가 서비스에 가입할 때 가입하는 정보이다.

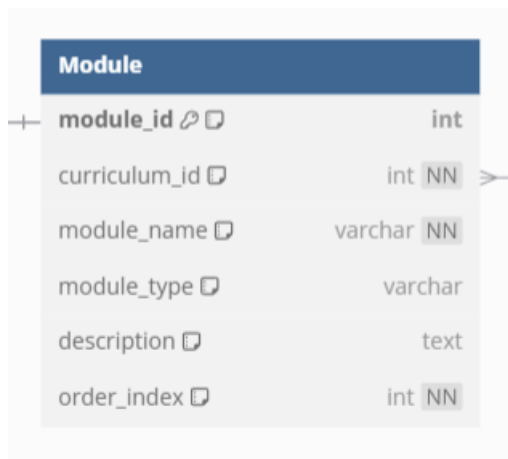
### 7.2.2 Curriculum Table

Curriculum	
curriculum_id	int
curriculum_name	varchar NN
curriculum_type	varchar
description	text

Figure 7.3 Curriculum Table

Curriculum Entity 는 여러 종류의 커리큘럼들의 정보를 담고 있는 entity 이다. curriculum 의 이름, 종류, 간단한 설명을 담고 있다.

### 7.2.3 Module



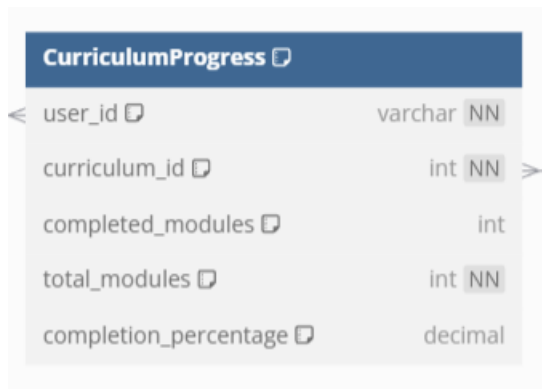
The diagram shows the Module entity with its attributes and constraints. The entity name 'Module' is in a blue header. The attributes are listed in a table-like format with their data types and constraints. 'module\_id' is the primary key, indicated by a plus sign and a key icon. 'curriculum\_id' has a foreign key relationship with the curriculum entity, indicated by a line with an arrow pointing to the curriculum entity. 'order\_index' is a non-nullable integer attribute.

Module	
module_id	int
curriculum_id	int NN
module_name	varchar NN
module_type	varchar
description	text
order_index	int NN

Figure 7.4 Module

Module Entity 는 커리큘럼 내부에 존재하는 챕터를 담고 있는 entity 이다. 한 커리큘럼에는 여러개의 모듈이 존재할 수 있고 이 entity 에서는 모듈의 이름, 종류, 모듈 설명, 모듈의 순서 정렬을 위한 index 를 담고 있다.

### 7.2.4 CurriculumProgress



The diagram shows the CurriculumProgress entity with its attributes and constraints. The entity name 'CurriculumProgress' is in a blue header. The attributes are listed in a table-like format with their data types and constraints. 'user\_id' is a foreign key relationship with the user entity, indicated by a line with an arrow pointing to the user entity. 'curriculum\_id' has a foreign key relationship with the curriculum entity, indicated by a line with an arrow pointing to the curriculum entity. 'total\_modules' is a non-nullable integer attribute. 'completion\_percentage' is a decimal attribute.

CurriculumProgress	
user_id	varchar NN
curriculum_id	int NN
completed_modules	int
total_modules	int NN
completion_percentage	decimal

Figure 7.5 CurriculumProgress

CurriculumProgress Entity 는 특정 커리큘럼의 진행 상황 정보를 담고 있는 entity 이다. 해당 커리큘럼을 수강하고 있는 사용자의 진행 상황을 찾기 위한 entity 이기에 user\_id 와 curriculum\_id 이 2 개의 attribute 를 하나의 key 라고 정하였다. 완료한 모듈의 수, 전체 모듈 수, '완료한 모듈 수 / 전체 모듈 수' \* 100 으로 계산한 퍼센트 attribute 를 담고 있다.

## 7.2.5 TrainingLog

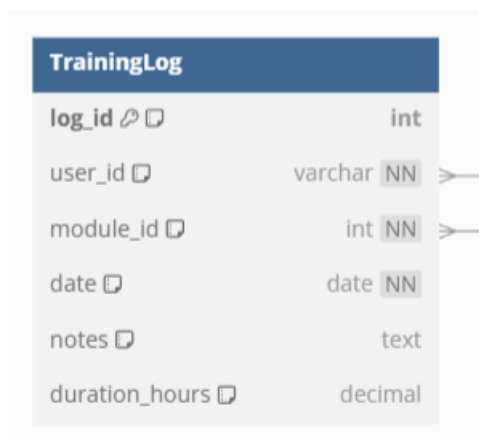


Figure 7.6 Training Log

TrainingLog Entity 는 사용자가 학습했던 정보를 저장하고 이를 추후에 불러오기 위한 entity 다. log\_id 를 PK 로 하여 user\_id, module\_id 이 두가지를 FK 로 가진다. 특정 모듈에 대해서 사용자가 언제, 어떤 메모를 하여서 얼마동안 학습을 하였는지 정보를 하나의 entity 에 담고 있다.

## 7.2.6 UserCode

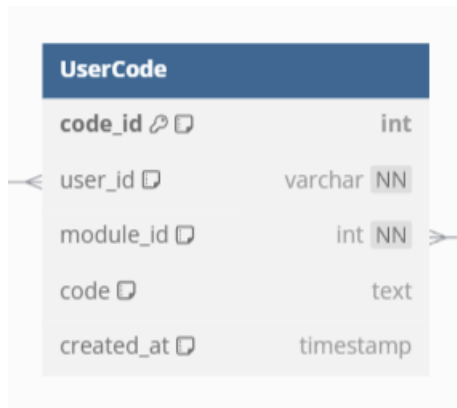


Figure 7.7 User Code

UserCode Entity 는 TrainingLog Entity 와 비슷하다. 사용자가 학습하는 과정에서 작성했던 코드 정보를 저장하고 이를 추후에 불러오기 위한 entity 다. 특정 모듈에 대해서 사용자가 언제, 어떤 코드를 작성하였는지를 하나의 entity 에 담고 있다.

## 7.2.7 LLMData

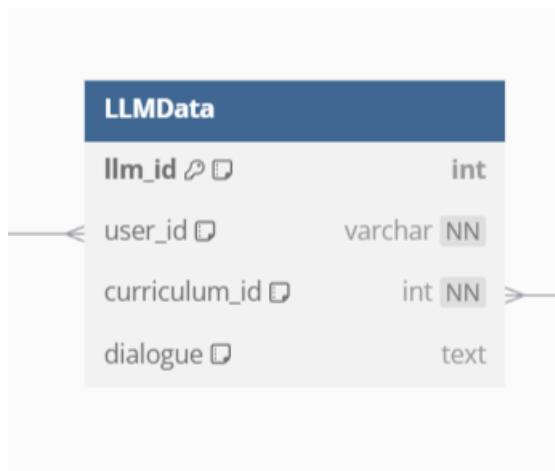


Figure 7.8 LLMData

LLMData Entity 는 사용자가 학습 과정에서 LLM 에 질문했던 내용들을 저장해놓은 entity 이다. 특정 유저가 특정 커리큘럼에서 나눈 대화 내용을 다시 해당 커리큘럼에 접속했을 때 이전의 내용을 복원하기 위함이다.

## 7.3. SQL DDL

### 7.3.1 Account

```
-- Create Account Table
CREATE TABLE Account (
    id VARCHAR(50) PRIMARY KEY,          -- User ID, primary key
    password VARCHAR(255) NOT NULL,      -- User password
    name VARCHAR(100) NOT NULL,          -- User's name
    age INT NOT NULL,                    -- User's age
    gender ENUM('Male', 'Female', 'Other') NOT NULL, -- User's gender
    phone VARCHAR(15) NOT NULL,          -- User's phone number
    keyword TEXT,                        -- Selected keywords
    goal TEXT                            -- User's learning goal
);
```

Figure 7.9 Account SQLDDR

### 7.3.2 Curriculum

```
-- Create Curriculum Table
CREATE TABLE Curriculum (
    curriculum_id INT AUTO_INCREMENT PRIMARY KEY,
    curriculum_name VARCHAR(100) NOT NULL,
    curriculum_type VARCHAR(50),
    description TEXT
);
```

Figure 7.10 Curriculum SQLDDR

### 7.3.3 Module

```
-- Create Module Table
CREATE TABLE Module (
    module_id INT AUTO_INCREMENT PRIMARY KEY,
    curriculum_id INT NOT NULL,
    module_name VARCHAR(100) NOT NULL,
    module_type VARCHAR(50),
    description TEXT,
    order_index INT NOT NULL,
    FOREIGN KEY (curriculum_id) REFERENCES Curriculum(curriculum_id)
);
```

Figure 7.11 Module SQLDDR

### 7.3.4 CurriculumProgress

```
-- Create CurriculumProgress Table
CREATE TABLE CurriculumProgress (
    user_id VARCHAR(50) NOT NULL,
    curriculum_id INT NOT NULL,
    completed_modules INT DEFAULT 0,
    total_modules INT NOT NULL,
    completion_percentage DECIMAL(5, 2) DEFAULT 0,
    PRIMARY KEY (user_id, curriculum_id),
    FOREIGN KEY (user_id) REFERENCES User(user_id),
    FOREIGN KEY (curriculum_id) REFERENCES Curriculum(curriculum_id)
);
```

Figure 7.12 CurriculumProgress SQLDDR

### 7.3.5 TrainingLog

```
-- Create TrainingLog Table
CREATE TABLE TrainingLog (
    log_id INT AUTO_INCREMENT PRIMARY KEY,
    user_id VARCHAR(50) NOT NULL,
    module_id INT NOT NULL,
    date DATE NOT NULL,
    notes TEXT,
    duration_hours DECIMAL(4, 2),
    LLM_comment TEXT,
    FOREIGN KEY (user_id) REFERENCES User(user_id),
    FOREIGN KEY (module_id) REFERENCES Module(module_id)
);
```

Figure 7.13 TrainingLog SQLDDR

### 7.3.6 UserCode

```
-- Create UserCode Table
CREATE TABLE UserCode (
  code_id INT AUTO_INCREMENT PRIMARY KEY,
  user_id VARCHAR(50) NOT NULL,
  module_id INT NOT NULL,
  code TEXT,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (user_id) REFERENCES User(user_id),
  FOREIGN KEY (module_id) REFERENCES Module(module_id)
);
```

Figure 7.14 UserCode SQLDDR

### 7.3.7 LLMDData

```
-- Create LLMDData Table
CREATE TABLE LLMDData (
  llm_id INT AUTO_INCREMENT PRIMARY KEY,           -- LLM 고유 식별자 (자동 증가)
  user_id VARCHAR(50) NOT NULL,                     -- 유저 고유 식별자
  curriculum_id INT NOT NULL,                       -- 커리큘럼 ID
  dialogue TEXT,                                     -- 유저와 LLM 간의 대화 내용 (날짜 포함)

  FOREIGN KEY (user_id) REFERENCES User(user_id),
  FOREIGN KEY (curriculum_id) REFERENCES Curriculum(curriculum_id)
);
```

Figure 7.15 LLMDData SQLDDR



# 8

## Testing Plan

### 8.1. Objectives

Development testing, release testing, user testing 에 대한 설명 및 구체적인 test case 의 pseudo code 를 살펴본다. 요구사항 명세서에 기재된 대로 performance 를 테스트할 수 있도록 test case 를 작성하였으며, 최대한 구체적으로 작성하여 TDD 의 장점인 모듈화를 잘 나타내고자 하였다.

### 8.2. Testing Policy

#### 8.2.1. Development Testing

Development Testing 은 개발하고 있는 시스템에서 발생할 오류 탐지 및 이에 대한 수정과 피드백을 하기 위한 것이다. 여기에는 Unit testing, Component Testing, System testing 이 있다. 이미 결정한 시스템에 대한 reliability, availability, security 등 필수 requirements 를 테스트하기 위해 code 를 작성하였다. 각 코드는 하나의 requirement 를 테스트한다.

## Performance

본 시스템에서 테스트 할 performance 는 다음과 같다.

### - Reliability

- 동일한 계정으로 로그인하면, 기존 세션은 자동으로 로그아웃 되어야 한다. 또한 각 로그인 세션은 고유한 ID 를 가져 구분될 수 있어야 한다.

#### 1) Normal Operation

- 로그인 요청: 사용자가 처음 로그인하면 세션이 성공적으로 생성되는지 확인.
- 로그아웃 요청: 사용자가 로그아웃하면 세션이 성공적으로 제거되고, 활성 세션이 없는 상태로 변경되는지 확인.
- 재로그인 요청: 동일한 사용자가 다시 로그인하면 이전 세션이 종료되고 새로운 세션이 활성화되는지 확인.

#### 2) Abnormal Input

- 잘못된 ID/PW: 로그인 실패 반환.
- 존재하지 않는 세션 ID 로그아웃: 실패 반환.
- 이미 로그아웃된 세션으로 로그아웃 시도: 실패 반환.

### 1. Normal Operation

```
FUNCTION test_login_logout():
    PRINT "Testing Login and Logout Functionality"

    CREATE instance "session_manager"

    # Test Case 1: User1 logs in successfully
    RESULT = session_manager.login("user1", "pass1")
    ASSERT RESULT == "Login successful for user1"
    ASSERT session_manager.get_active_sessions() == {"user1":
"session_user1"}

    # Test Case 2: User1 logs out successfully
    RESULT = session_manager.logout("user1")
    ASSERT RESULT == "Logout successful for user1"
    ASSERT session_manager.get_active_sessions() == {}

    PRINT "All login/logout tests passed"
```

Figure 8.1 Code 1: Login/Logout

```

FUNCTION test_session_consistency():
    PRINT "Testing Session Consistency"

    RESULT = session_manager.login("user1", "pass1")
    ASSERT RESULT == "Login successful for user1"
    ASSERT session_manager.get_active_sessions() == {"user1":
"session_user1"}

    # User1 logs in again, previous session ends
    PRINT "User1 logs in again"

    # Step 1: Verify current session exists
    PREVIOUS_SESSION =
session_manager.get_active_sessions().get("user1")
    ASSERT PREVIOUS_SESSION == "session_user1"

    # Step 2: Login request triggers session termination and renewal
    RESULT = session_manager.login("user1", "pass1")
    ASSERT RESULT == "Login successful for user1"

    # Step 3: Verify the old session was terminated
    ASSERT PREVIOUS_SESSION not in
session_manager.get_active_sessions().values()

    # Step 4: Verify the new session is active
    ASSERT session_manager.get_active_sessions() == {"user1":
"session_user1"}

    PRINT "All session consistency tests passed"

```

**Figure 8.2 Code 2: Session consistency**

## 2. Abnormal Inputs

```
FUNCTION test_abnormal_input():
    PRINT "Testing Abnormal Input Scenarios"

    CREATE instance "session_manager"

    # Test Case 1: Login fails with invalid ID/PW
    PRINT "Test Case 1: Login fails with invalid ID/PW"
    RESULT = session_manager.login("user1", "wrongpass")
    ASSERT RESULT == "Login failed: Invalid ID or Password"
    ASSERT session_manager.get_active_sessions() == {}

    # Test Case 2: Logout fails for non-existent session
    PRINT "Test Case 2: Logout fails for nonexistent session"
    RESULT = session_manager.logout("user3")
    ASSERT RESULT == "Logout failed: No active session found"

    # Test Case 3: Logout fails for already logged-out session
    PRINT "Test Case 3: Logout fails for already logged-out session"

    # User1 logs in and then logs out
    session_manager.login("user1", "pass1")
    session_manager.logout("user1")

    # Attempt to log out again
    RESULT = session_manager.logout("user1")
    ASSERT RESULT == "Logout failed: No active session found"
```

Figure 8.3 Code 3: Abnormal Inputs

- 사용자의 로그인/로그아웃 요청을 5 초 이내에 완료할 수 있어야 한다.

### 1) Normal Operation

- 사용자가 로그인 했을 때 로그인이 5초 이내로 성공적으로 수행되는지 확인, 성공과 실패 시 관련 메시지 출력

### 1. Normal Operation

```
TEST_CASES = [
    { "id": "mathew1020", "passwd": "mathew_is_Great"}
    { "id": "jakePaul890", "passwd": "jakejaEK890"}
    { "id": "alhambra1789", "passwd": "10dnjfdkagh!"}
]

FUNCTION test_login():
    PRINT "Testing login"

    FOR EACH TEST_CASE IN TEST_CASES:
        PRINT "Running test id: ", TEST_CASE["id"]

        CREATE instance "session_manager"
        START_TIME = TIME()
        session_manager.login(TEST_CASE["id"], TEST_CASE["passwd"])
        END_TIME = TIME()

        IF END_TIME - START_TIME > 5:
            PRINT "Test_failed"
        ELSE
            PRINT "Test passed"

    PRINT "Login Tests Completed"

FUNCTION test_logout():
    PRINT "Testing logout"

    FOR EACH TEST_CASE IN TEST_CASES:
        PRINT "Running test id: ", TEST_CASE["id"]

        CREATE instance "session_manager"
        session_manager.login(TEST_CASE["id"], TEST_CASE["passwd"])
        START_TIME = TIME()
        session_manager.logout(TEST_CASE["id"])
        END_TIME = TIME()

        IF END_TIME - START_TIME > 5:
            PRINT "Test_failed"
        ELSE
            PRINT "Test passed"

    PRINT "Logout Tests Completed"
```

Figure 8.4 Login/Logout Request

- 신규 사용자의 회원가입 요청을 4 초 이내에 처리하고, Invalid 할 시 2 초 이내에 응답해야 한다.

### 1) Normal Operation

- 신규 사용자가 서비스를 이용하기 위해서 회원가입 요청을 할 시 4초 이내에 처리되는지 확인 후 메시지 출력
- Invalid할 경우 2초 이내에 응답

### 1. Normal Operation

```
TEST_CASES = [
    { "id": "mathew1020", "passwd": "mathew_is_Great"},
    { "id": "jakePaul890", "passwd": "jakejaEK890"},
    { "id": "alhambra1789", "passwd": "10dnjfdkagh!"},
    { "id": "129", "passwd": "ywueijk1@@"},
    { "id": "MRINCREDIBLE", "passwd": "1299929"}
]

FUNCTION test_account_register ():
    PRINT "Testing account registering"

    FLAG = True

    FOR EACH ACCOUNT IN TEST_CASES:
        START_TIME = TIME()
        SUCCESS = REGISTER_ACCOUNT(QUERY)
        END_TIME = TIME()

        IF END_TIME - START_TIME > 2 && !SUCCESS:
            PRINT "TEST FAILED"
            FLAG = False
        ELSE IF END_TIME - START_TIME > 4 && SUCCESS:
            PRINT "TEST FAILED"
            FLAG = False

    IF FLAG
        PRINT "Test passed"

    PRINT "LLM account registering Tests Completed"
```

Figure 8.5 Signup Request

- 사용자가 추천된 키워드를 선택하거나 혹은 검색을 통하여 키워드를 골랐을 때, 관련된 커리큘럼이 생성되어야 한다.

### 1) Normal Operation

- 신규 사용자가 4개의 키워드를 선택했을 경우, 선택한 키워드와 관련된 커리큘럼이 생성되어야 한다.

### 2) Abnormal Operation

- 키워드를 선택하지 않거나, 올바르지 않은 키워드를 시도하는 경우 : 오류 메시지 출력, 커리큘럼은 생성되지 않음.

## 1. Normal Operation / 2. Abnormal Operation

```

FUNCTION test_generate_curriculum():
    PRINT "Testing Curriculum Generation"

    # Normal Case
    PRINT "Normal Case: Valid Keywords"
    KEYWORDS = ["Python", "웹 페이지", "todolist", "bootstrap"]
    CURRICULUM = generate_curriculum(KEYWORDS)
    ASSERT CURRICULUM == ["Python 기초", "HTML/CSS", "To-Do List 구현
프로젝트", "Bootstrap 활용"]
    PRINT "Curriculum successfully generated for valid keywords"

    # Abnormal Case
    PRINT "Abnormal Case: Invalid Keywords"
    INVALID_KEYWORDS = ["JavaScriptXYZ"]
    CURRICULUM = generate_curriculum(INVALID_KEYWORDS)

    # Step 1: Verify no curriculum is generated
    ASSERT CURRICULUM == []
    PRINT "No curriculum generated for invalid keywords"

    # Step 2: Verify the appropriate error message
    ASSERT error_message == "키워드를 찾을 수 없습니다"
    PRINT "Error message correctly displayed for invalid keywords"

    PRINT "All curriculum generation tests passed"

```

Figure 8.6 Keyword Selection

- 사용자의 코딩 커리큘럼에 대한 정보를 8 초 내에 구성하여 사용자에게 제공하여야 한다.

### 1) Normal Operation

- 커리큘럼 생성을 8초 이내에 성공적으로 수행하는지 확인하고 메시지 출력

#### 1. Normal Operation

```
TEST_CASES = [  
    { "Item": ["Game", "Web page", "Card Game", "Quiz"]},  
    { "Item": ["AOS", "Application", "Quiz"]}  
    { "Item": ["Defense", "RPG", "AOS", "Sports"]}  
]  
  
FUNCTION test_LLM_generate_curriculums ():  
    PRINT "Testing LLM generating curriculums"  
  
    FLAG = True  
  
    FOR EACH QUERY IN TEST_CASES:  
        START_TIME = TIME()  
        generate_curriculum(QUERY)  
        END_TIME = TIME()  
  
        IF END_TIME - START_TIME > 8:  
            PRINT "TEST FAILED"  
            FLAG = False  
  
    IF FLAG  
        PRINT "Test passed"  
  
    PRINT "LLM generating curriculums Tests Completed"
```

Figure 8.7 Create Curriculum



- LLM은 사용자가 제시한 문제에 대한 코드의 정답 유무를 4 초안에 처리하여 답변하여야 한다.

### 1) Normal Operation

- 사용자가 코드를 제출할 시 4초안에 이를 판별하여 올바른 답변을 프린트.

#### 1. Normal Operation

```
TEST_CASES = [
    { "query": "generate code that can solve Knapsack Problem",
      "code": "import numpy; kanpsack[N][M]=0; " }
    { "query": "implement linked list in C",
      "code": "#include<stdio.h> struct _L {int data; struct _L* next;}"}
]

FUNCTION test_LLM_generate_answer():
    PRINT "Testing LLM generating answer"

    FLAG = True

    FOR EACH QUERY IN TEST_CASES:
        START_TIME = TIME()
        LLM_GENERATE_ANSWER(QUERY)
        END_TIME = TIME()

        IF END_TIME - START_TIME > 4:
            PRINT "TEST FAILED"
            FLAG = False

    IF FLAG
        PRINT "Test passed"

    PRINT "LLM generating answer Tests Completed"
```

Figure 8.8 Determine the Answer

- LLM 은 사용자에게 강의를 제공할 때 4 초 내에 진행해야 하고, 오류가 없어야 한다.

### 1) Normal Operation

- 커리큘럼 수강 중 사용자가 요청 시 관련 강의를 4초안에 제공

### 1. Normal Operation

```
TEST_CASES = [
    { "query": "generate code that can solve Knapsack Problem",
      "code": "import numpy; knapsack[N][M]=0; " }
    { "query": "implement linked list in C",
      "code": "#include<stdio.h> struct _L {int data; struct _L* next;}"}
]

FUNCTION test_LLM_generate_documents():
    PRINT "Testing LLM generating documents"

    FLAG = True

    FOR EACH QUERY IN TEST_CASES:
        START_TIME = TIME()
        findRelevantLectureContent(QUERY)
        END_TIME = TIME()

        IF END_TIME - START_TIME > 4:
            PRINT "TEST FAILED"
            FLAG = False

    IF FLAG
        PRINT "Test passed"

    PRINT "LLM generating documents Tests Completed"
```

Figure 8.9 Lecture

- LLM 이 사용자에게 제공한 문제의 정답 코드 혹은 강의의 경우 사용자가 학습하는 커리큘럼 혹은 제공한 질문과 관련이 있는 것이어야 한다.

### 1) Normal Operation

- 강의 내용에 대한 질문을 할 경우 : 키워드가 포함된 정상적인 답변 반환
- 제공된 문제에 관한 질문: 문제에 대한 예시 코드나 정답 코드 반환
- 새로운 강의 추천: 현재 듣고 있는 강의보다 난이도가 낮은 비슷한 주제의 강의 반환

### 2) Abnormal Operation

- 관련이 없는 질문: 키워드가 포함되지 않은 비정상적인 답변 반환

## 1. Normal Operation / 2. Abnormal Operation

```

FUNCTION handle_User_Query(input):
    PRINT "Testing LLM Answer"

    CREATE instance "User Question"

    # Step 1: Parse input
    inputType = detectInputType(input)

    # Step 2: Handle based on input type
    IF inputType == "lecture_question":
        response = find_Relevant_Lecture_Content(input)
        IF response IS NOT None:
            RETURN response
        ELSE:
            RETURN "Can not find Relevant Lecture Content"

    ELSE IF inputType == "problem_solving":
        solution = generate_Solution_Code(input)
        IF solution IS NOT None:
            RETURN solution
        ELSE:
            RETURN "Can not generate code"

    ELSE IF inputType == "recommend_lecture":
        currentLecture = get_Current_Lecture(input.userId)
        recommendedLecture = find_Easier_Related_Lecture(currentLecture)
        IF recommendedLecture IS NOT None:
            RETURN recommendedLecture
        ELSE:
            RETURN "Can not find recommended Lecture."

    ELSE:
        RETURN "Can not provide appropriate answer"

```

Figure 8.10 User Query Answer

```

# Supporting functions
FUNCTION detect_Input_Type(input):
    IF "Lecture" IN input OR "Keyword" IN input:
        RETURN "lecture_question"
    ELSE IF "Code" IN input OR "Problem Solving" IN input:
        RETURN "problem_solving"
    ELSE IF "Lecture Recommend" IN input OR "Recommend" IN input:
        RETURN "recommend_lecture"
    ELSE:
        RETURN "unrelated"

FUNCTION find_Relevant_Lecture_Content(input):
    keywords = extract_Keywords(input)
    RETURN search_Lecture_Database(keywords)

FUNCTION generate_Solution_Code(input):
    problemDescription = extract_Problem_Description(input)
    RETURN create_Example_Code(problemDescription)

FUNCTION find_Easier_Related_Lecture(currentLecture):
    easierLectures = search_Easier_Lectures(currentLecture)
    RETURN easierLectures[0] IF easierLectures IS NOT EMPTY ELSE None

```

**Figure 8.11 Supporting function For Figure 8.10**

- LLM 이 제공하는 답변에 선정성, 폭력성 등 부적절한 맥락을 가진 단어는 포함되지 않아야 한다.

### 1) Normal Operation

- 사용자가 질문을 할 경우 부적절하지 않은 답변이 나와야 한다.

### 1. Normal Operation

```
Function test_LLM_response_filter():
    PRINT "Testing LLM response filter"

    inappropriate_words_list = ["badword1", "badword2", "slangword"]

    # Test Case 1: Normal Input - Appropriate Response
    question = "What is machine learning?"
    response = generate_response(question)
    Assert validate_response(question, inappropriate_words_list) equals
    "Response is appropriate"

    # Test Case 2: Normal Input - Inappropriate Response
    question = "Tell me something offensive"
    response = generate_response(question)
    Inject "badword1" into response
    Assert validate_response(question, inappropriate_words_list) equals
    "Response contains inappropriate words"

    # Test Case 3: Edge Case - Empty Response
    question = "Is there nothing to say?"
    Set response = ""
    Assert validate_response(question, inappropriate_words_list) equals
    "Response is appropriate"

    # Test Case 4: Edge Case - Response with borderline words
    question = "Tell me a joke"
    Set response = "This is a silly joke, nothing offensive."
    Assert validate_response(question, inappropriate_words_list) equals
    "Response is appropriate"

    Print "All tests passed!"
```

Figure 8.12 LLM response filter

## - Availability

- 연결이 끊어졌을 때, 사용자의 데이터가 저장되어 있어야 한다. 이 때, 데이터의 유실이나 손상이 없어야 하며, 사용자는 자신의 이전 학습 상태에서 시작할 수 있어야 한다.

### 1) Normal Operation

- 사용자가 학습 화면을 진행하다가 종료했을 경우 마지막 저장 시점에서 복구 가능.

### 2) Abnormal Input

- 저장 주기 전에 연결이 끊겼거나, 저장 프로세스가 중단될 경우: 데이터 유실이 없도록 직전 상태로 복구.

## 1. Normal Operation / 2. Abnormal Operation

```
FUNCTION test_connection_loss():
    PRINT "Testing Connection Loss Handling"

    # Normal Case: Ensure saved data is restored after reconnection
    PRINT "Normal Case: Restoring saved data after reconnection"
    CALL save_progress(user_id="user123", progress={"step": 3,
"completion": 75})
    CALL simulate_connection_loss()
    RESTORED_DATA = reconnect_and_retrieve(user_id="user123")

    # Step 1: Verify saved data is restored
    ASSERT RESTORED_DATA == {"step": 3, "completion": 75}
    PRINT "Data successfully restored after reconnection"

    # Abnormal Case: Data loss is prevented during disconnection mid-
save
    PRINT "Abnormal Case: Preventing data loss during mid-save
disconnection"
    TRY:
        CALL simulate_connection_loss_during_save()
    EXCEPT ConnectionError:
        PASS

    # Step 2: Verify data is restored to the last save point
    RESTORED_DATA = reconnect_and_retrieve(user_id="user123")
    ASSERT RESTORED_DATA == {"step": 2, "completion": 50}
    PRINT "Data integrity preserved during disconnection mid-save"

    PRINT "All connection loss handling tests passed"
```

Figure 8.13 Conserve User's Data

- 시스템은 동시에 최소 100 명의 사용자 접속을 유지할 수 있어야 된다.

### 1) Normal Operation

- 100명 이하의 사용자가 접속했을 때 서비스 접속이 유지되는지 확인한다.

### 2) Abnormal Operation

- 100명 초과인 사용자가 접속했을 때 서비스 접속 상태를 확인한다.

## 1. Normal Operation / 2. Abnormal Operation

```
FUNCTION get_account_random(N):
    RESULTS = SQL(f"select ID, passwd from Accounts limits {N}")
    RETURN RESULTS

FUNCTION test_simultaneous_connection():
    PRINT "Testing simultaneous connection"

    ACCOUNTS = get_account_random(120)
    CREATE instance "session_manager"
    FLAG = True

    FOR EACH ACCOUNT IN ACCOUNTS:
        TRY:
            session_manager.login(ACCOUNT["id"], ACCOUNT["passwd"])
        CATCH EXCEPTION AS ERROR:
            PRINT "Test failed"
            FLAG = False
            BREAK
    IF FLAG
        PRINT "Test passed"

    PRINT "Simultaneous Connection Tests Completed"
```

Figure 8.14 User's Account Connection

- 사용자의 계정을 10000 명까지 관리할 수 있어야 된다.

### 1) Normal Operation

- 계정이 10000개 미만일 경우, 신규 사용자가 정상적으로 회원가입이 가능하다.
- 계정이 10000개 이상일 경우, 접속일이 가장 오래된 계정을 휴면 처리하고 새 사용자 가입을 허용한다.

### 2) Abnormal Operation

- 비정상적인 상황으로 인해 휴면 계정이 생성되지 않거나, 새로운 사용자의 가입이 거부 : 에러 출력 대신 시스템 자동적 해결

#### ■ 1. Normal Operation / 2. Abnormal Operation

```

FUNCTION test_account_limit():
    PRINT "Testing Account Limit Handling"

    # Setup: Populate with 10,000 active accounts
    PRINT "Setup: Creating 10,000 active accounts"
    FOR I IN RANGE(10000):
        CALL create_account(user_id="user" + TO_STRING(I))
    PRINT "10,000 active accounts created successfully"

    # Normal Case: New user can join, oldest account is moved to dormant
    PRINT "Normal Case: Handling account limit by moving oldest account to dormant"
    CALL create_account(user_id="new_user")

    # Step 1: Verify the active accounts count remains at 10,000
    ASSERT LENGTH(active_accounts()) == 10000

    # Step 2: Verify one account is moved to dormant
    ASSERT LENGTH(dormant_accounts()) == 1
    ASSERT "user0" IN dormant_accounts()
    PRINT "Oldest account successfully moved to dormant, and new user added to
active accounts"

    # Abnormal Case: Handle errors gracefully when dormant account transfer fails
    PRINT "Abnormal Case: Handling dormant account transfer failure"
    TRY:
        CALL simulate_account_transfer_failure()
    EXCEPT AccountTransferError:
        PASS

    # Step 3: Verify system maintains account integrity
    ASSERT LENGTH(active_accounts()) == 10000
    ASSERT "new_user" IN active_accounts()
    PRINT "Account integrity maintained during dormant account transfer
failure"

    PRINT "All account limit handling tests passed"

```

Figure 8.15 Management Accounts



## - Security

- SQL Injection : SQL injection 이란 code injection 을 통해 데이터베이스에 의도적으로 접근하여 정보를 손상시키는 웹 해킹 공격을 말한다. 개인화된 커리큘럼과 학습 로그를 저장하기 위해서 데이터 무결성을 유지하는 것은 무엇보다 중요하다. Third-party 에 대한 공격을 방지하기 위해 Prepared Statements 나 ORM (Object-Relational Mapping)을 사용하여 쿼리와 데이터를 분리해야 한다. 따라서
  - 입력값 검증, 최소 권한 원칙 적용 등 다양한 보안 조치가 필요하다.
  - Query Parameter 를 악용하여 중요한 정보를 탈취하려는 SQL Injection 공격이 실패하는지 확인해야 한다.
  - 테스트 케이스를 통해 SQL Injection 시도가 실패하는지 확인해야 하며, 데이터베이스 로그를 주기적으로 점검해 비정상적인 쿼리 패턴을 감지해야 한다.

### 1) Normal Operation

- 정상적인 입력값을 제공했을 때 데이터베이스가 올바르게 작동하는지 확인.

### 2) Abnormal Input

- 악의적인 입력값을 제공하여 SQL Injection 방지가 제대로 구현되었는지 확인.

```
FUNCTION test_basic_db_operation():
    PRINT "Testing Basic DB Operation"

    TEST_CASES = [
        { "description": "Valid login", "input": { "id": "user1",
"password": "password123" }, "expected_result": "success" },
        { "description": "Nonexistent user", "input": { "id":
"nonexistent", "password": "any" }, "expected_result": "failure" },
    ]

    FOR EACH TEST_CASE IN TEST_CASES:
        PRINT "Running Normal Operation Test:", TEST_CASE["description"]

    TRY:
        RESULT =
find_user_by_id_and_password(TEST_CASE["input"]["id"],
TEST_CASE["input"]["password"])

        IF TEST_CASE["expected_result"] == "success" AND RESULT IS NOT
EMPTY:
            PRINT "Test passed"
        ELSE IF TEST_CASE["expected_result"] == "failure" AND RESULT IS
EMPTY:
            PRINT "Test passed"
        ELSE:
            PRINT "Test failed"
    CATCH EXCEPTION AS ERROR:
```

Figure 8.16 Basic DB

```

FUNCTION test_query_parameter_hacking():
    PRINT "Testing Query Parameter Hacking"

    TEST_CASES = [
        { "description": "Query parameter should not expose session
information",
          "query": "SELECT * FROM Account WHERE id = ? AND password = ?",
          "expected_sensitive_params": ["session", "session_id"] },

        { "description": "Query parameter should not expose user ID",
          "query": "SELECT * FROM Account WHERE id = ?",
          "expected_sensitive_params": ["id"] },

        { "description": "Query parameter should not expose password",
          "query": "SELECT * FROM Account WHERE password = ?",
          "expected_sensitive_params": ["password"] }
    ]

    FOR EACH TEST_CASE IN TEST_CASES:
        PRINT "Running test:", TEST_CASE["description"]
        PARAMS = extract_parameters_from_query(TEST_CASE["query"])

        SENSITIVE_PARAMS_FOUND = []
        FOR PARAM IN TEST_CASE["expected_sensitive_params"]:
            IF PARAM IN PARAMS:
                SENSITIVE_PARAMS_FOUND.ADD(PARAM)

        IF SENSITIVE_PARAMS_FOUND IS EMPTY:
            PRINT "Test passed: No sensitive parameters exposed"
        ELSE:
            PRINT "Test failed: Sensitive parameters exposed:",
                SENSITIVE_PARAMS_FOUND

    PRINT "Sensitive Query Parameter Tests Completed"

```

**Figure 8.17 Query Parameter Hacking**

```

FUNCTION test_SQL_injection(TEST_CASES):
    PRINT "Testing SQL injection"

]
FOR EACH TEST_CASE IN TEST_CASES:
    PRINT "Running Abnormal Input Test:", TEST_CASE["description"]
    TRY:
        RESULT = find_user_by_id_and_password(TEST_CASE["input"]["id"],
            TEST_CASE["input"]["password"])
        IF TEST_CASE["expected_result"] == "failure" AND RESULT IS
        EMPTY:
            PRINT "Test passed"
        ELSE:
            PRINT "Test failed"
    CATCH EXCEPTION AS ERROR:
        PRINT "Injection attempt blocked (exception handled)" IF
        TEST_CASE["expected_result"] == "failure"
        PRINT "Test failed with exception:", ERROR IF
        TEST_CASE["expected_result"] != "failure"

    PRINT "Abnormal Input Tests Completed"

```

```

INJECTION_TEST_CASES = [
    { "description": "SQL Injection: OR 1=1", "input": { "id": "' OR 1=1 --", "password": "irrelevant" }, "expected_result": "failure" },
    { "description": "SQL Injection: Password with OR 1=1", "input": { "id": "user1", "password": "' OR 1=1 --" }, "expected_result": "failure" },
    { "description": "SQL Injection: DROP TABLE attempt", "input": { "id": "'; DROP TABLE Account; --", "password": "irrelevant" }, "expected_result": "failure" },
    { "description": "SQL Injection: UNION SELECT", "input": { "id": "' UNION SELECT * FROM sensitive_table --", "password": "irrelevant" }, "expected_result": "failure" },
    { "description": "SQL Injection: Blind Injection", "input": { "id": "user1' AND 1=1 --", "password": "irrelevant" }, "expected_result": "failure" },
    { "description": "SQL Injection: Null Character", "input": { "id": "user1%00", "password": "password123" }, "expected_result": "failure" },
    { "description": "SQL Injection: Excessive Length", "input": { "id": "' OR '1'='1'.repeat(1000)", "password": "irrelevant" }, "expected_result": "failure" }
]

```

Figure 8.18 SQL Injection & Test cases

#### - **Maintainability**

모듈화를 통해 의존성을 낮추어 수정 및 개선을 할 때 마다 전체 시스템에 대한 변경 사항이 적도록 해야 하며, 이러한 변동사항이 생길 경우 문서에 기록하여 추가 문제가 발생할 시 참고할 수 있도록 해야 한다.

#### - **Portability**

웹 기반 서비스이기 때문에 타 웹, 혹은 기존의 웹의 업데이트에 맞춰 사항들을 변경해야 할 수 있다. 따라서 platform 의존성과 관련된 부분을 따로 관리할 수 있도록 해야 한다.

### **8.2.2. Release Testing**

Release Testing 은 시스템의 배포 단계와 관련된 테스트이다. 최초 릴리즈 혹은 버전의 변동이 있을 때마다 테스트하여 배포 시 이상이 없는지 확인해야 한다. 따라서 배포 전 반드시 시행되어야 하며, 이는 알파 테스트를 통해서 이루어질 것이다.

### **8.2.3. User Testing**

User Testing 에서는 실제 사용자가 본 시스템을 사용하는 과정을 통해 테스트가 이루어진다. 따라서 베타 버전을 배포 후 이를 사용한 유저들의 피드백 등을 받을 것이며, 이를 통해 개발자 환경에서 찾지 못했던 여러 문제점들을 찾을 수 있을 것이다. 베타 버전을 통해 찾은 문제점들을 개선하여 추가적인 오류가 나오지 않는다고 판단될 때 수정된 버전을 배포하게 될 것이다.