디자인명세서

LLM based Excel Creating Platform

by 방기호, 정관용, 박동민, 이상훈, 윤규성

TEAM 2

Instructor: 이은석

Teaching Assistant: 김영경, 김진영, 최동욱, 허진석

Document Date: 01 June, 2025

Faculty: SungKyunKwan University

Contents

List of Figures

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 2)이다. 시스템 개발자는 크게 Front-end와 Back-end 개발자로 나뉜다. 그리고 소프트웨어 공학 개론 수업의 교수, 조교, 참여 학생이다.

1.2. Scope

LLM에 대한 이해. Front-end와 Back-end 구조에 관한 이해.

1.3. Objective

이 소프트웨어 설계 문서의 주요 목적은 LLM based Excel Creating Platform 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 요구사항을 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 Diagram을 이용하여 서술한다.
4) System Architecture - Frontend: Frontend 시스템의 구조를 Class Diagram, Sequence Diagram 을 이용하여 서술한다.
5) System Architecture - Backend: Backend시스템의 구조를 Sequence Diagram, Class Diagram을 이용하여 서술한다.
6) Protocol Design: 클라이언트와 서버의 커뮤니케이션을 프로토콜 디자인을 서술한다.
7) Database Design: 시스템의 Database Requirements를 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, 툴에 대해 설명하고 개발 범위에 대해 설명한다.

2.2. Applied Diagrams

2.2.1. Used Tools

Draw.io / Mermaid live editor: 다이어그램을 그리기 위한 웹 사이트이다.

dbDiagram.io: 데이터베이스의 ERD를 그리기 위 웹 사이트이다.

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. User는 채팅을 통해 원하는 엑셀 파일을 만들도록 LLM을 통해 명령할 수 있다.

2.2.3. Sequence Diagram

Sequence Diagram은 시간순서로 행동별로 어떤 객체와 어떻게 상호작용을 하는지 표현하는 Diagram이다. Event Diagram이라고도부르는데, 시나리오와 관련된 객체와 시나리오의 기능 수행에 필요한 객체 간에 교환되는 메시지를 순서대로 표현한다. 시나리오와 관련된 객체는 사용자, Front-end, Back-end, GPT가 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. 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.

2.2.6. Project Scope

본 문서에서는 엑셀을 공부하지 않은 비전문가들이 LLM을 통해 간편하게 엑셀 파일을 생성할 수 있도록 보조하기 위해 만들어졌다.

2.2.7. References

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

3

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. 프론트엔드로부터 사용자가 자연어로 명령을 내리면 이에 따른 요청이 서버로 전해진다. 자연어 명령은 LLM을 통해 다뤄지고 동시에 MySQL 데이터베이스에 채팅 히스토리로 저장된다. LLM을 통한 명령은 Excel Management System을 통해 엑셀 파일을 생성하며 데이터베이스로 전송되고 사용자는 이 엑셀파일을 받아볼 수 있게 된다.

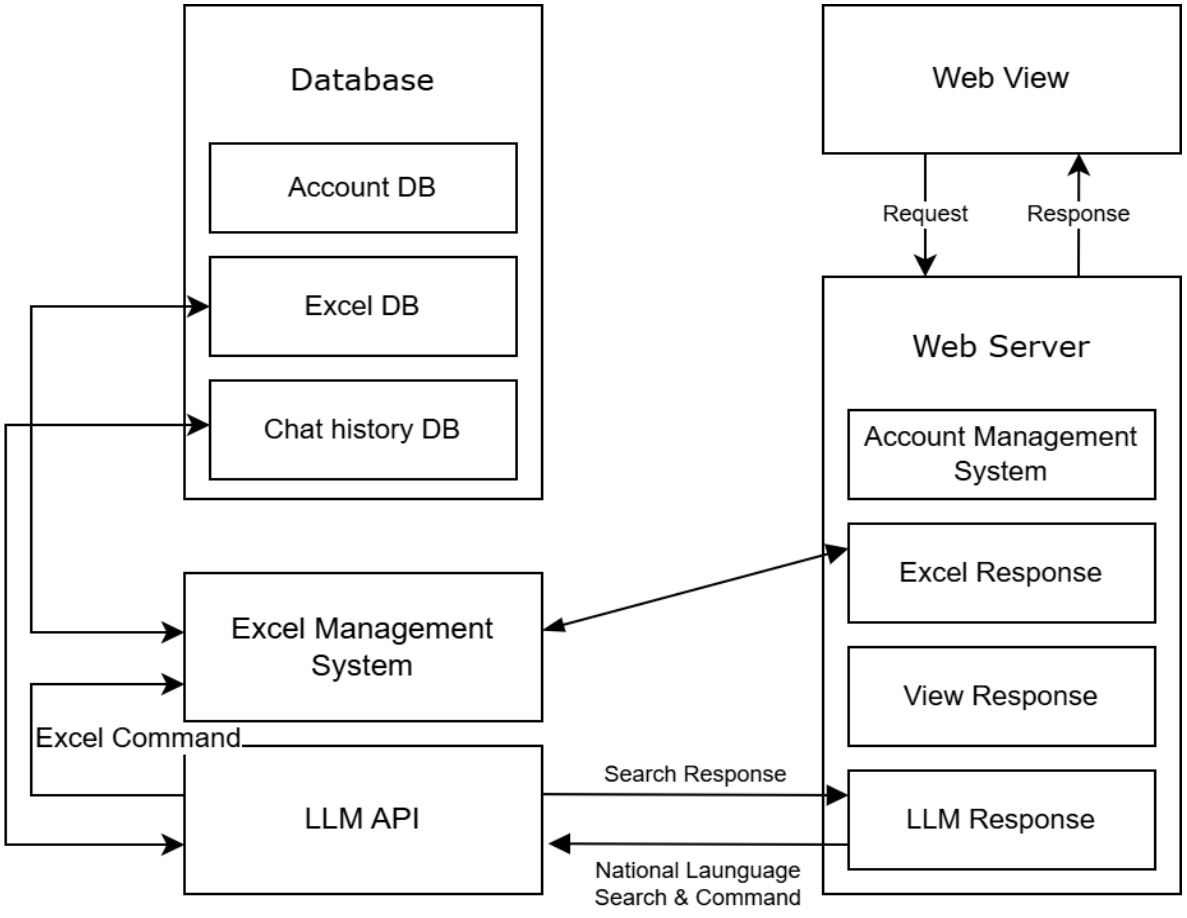

Figure 3.1: Overall System Architecture

3.2.1. System Diagram

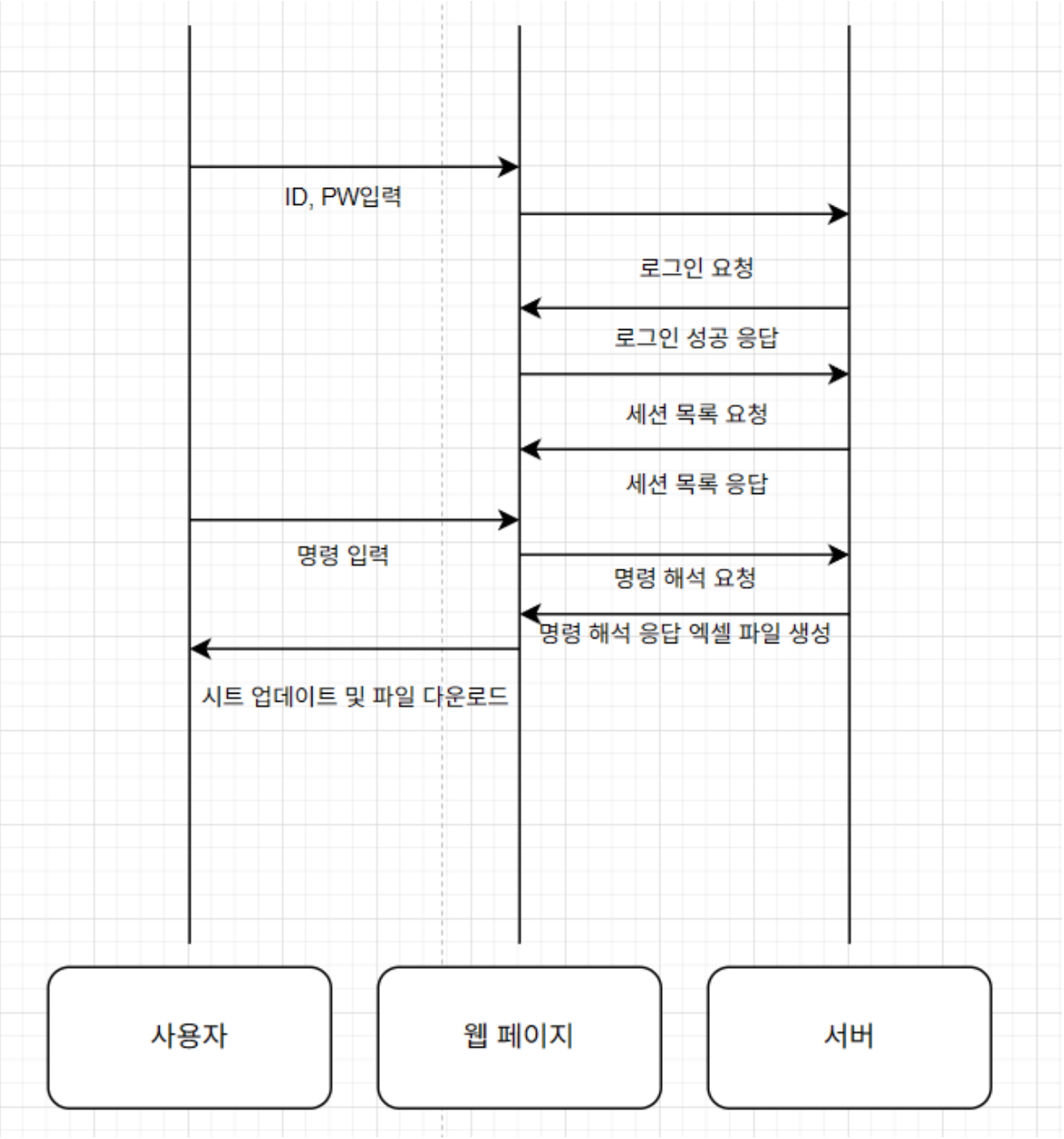

Figure 3.2: System Diagram-Overall

3.3. Use Case Diagram

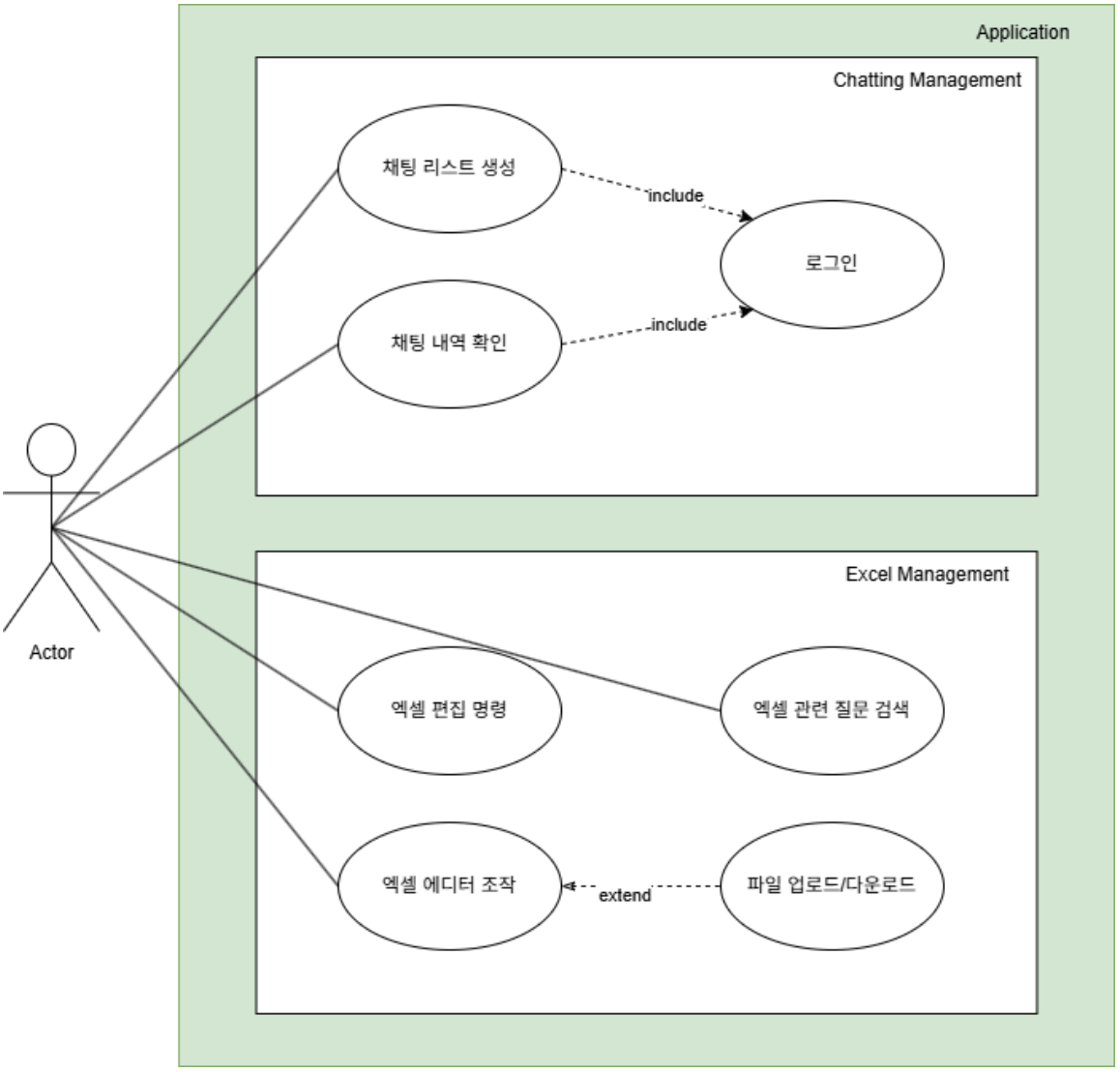

Figure 3.3: Use Case Diagram

4

System Architecture - Frontend

4.1. Objectives

본 장에서는 프론트엔드 시스템의 구조와 핵심 구성 요소, 그리고 사용자 인터페이스의 동작 흐름을 설명한다. UI는 로그인 화면과 메인 화면으로 구성되며, 이들과 서버 간의 상호작용을 클래스 다이어그램(정적 구조)과 시퀀스 다이어그램(시간 순 흐름)을 통해 분석한다. 본 다이어그램은 로그인부터 세션 불러오기, 명령 처리, 시트 반영 및 저장까지의 전체 과정을 시각적으로 표현한다.

4.2 Class Diagram – 자연어 기반 엑셀 편집 시스템 구조

본 클래스 다이어그램은 자연어 기반 엑셀 수식 추천 시스템의 핵심 구성 요소 간 관계를 시각적으로 표현한 것이다. 사용자는 UI를 통해 로그인 및 명령어를 입력하며, 입력된 명령은 서버(Server)에서 처리된다. 서버는 채팅 기록(ChatHistory)과 엑셀 시트(ExcelSheet)를 포함한 세션(Session)을 관리하며, 명령 해석 및 수식 적용 또한 서버 내에서 수행된다. 모든 데이터 흐름은 UI와 Server 간의 상호작용을 통해 이루어진다.

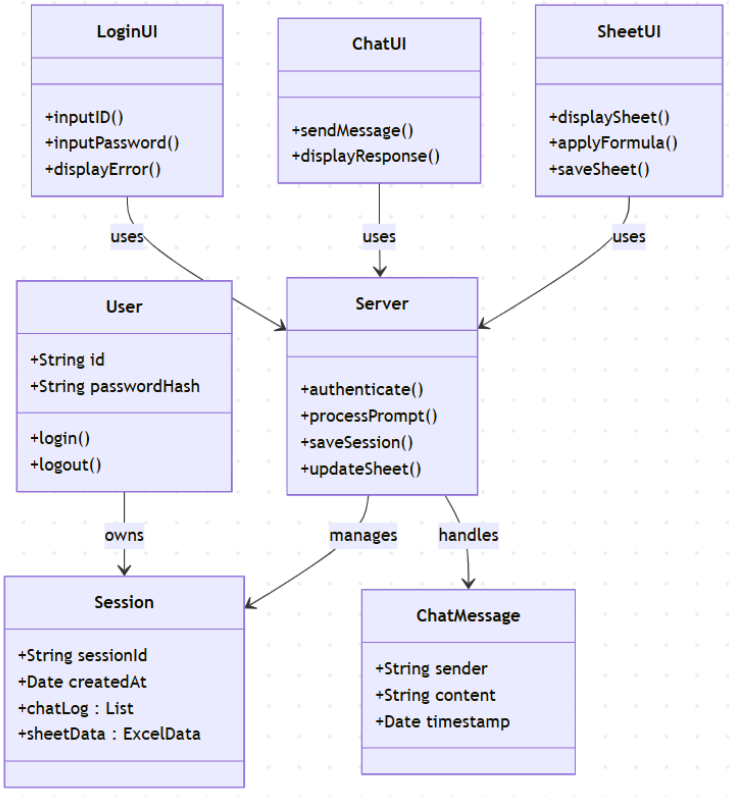

Figure 4.1: Class Diagram – 자연어 기반 엑셀 편집 시스템 구조

4.3 Sequence Diagram – 전체 사용자 흐름

본 시퀀스 다이어그램은 사용자가 로그인한 후 자연어 명령을 통해 엑셀 시트를 편집하고 저장하기까지의 과정을 시간 순서대로 나타낸 것이다. 로그인 성공 시 초기 세션 정보와 시트 데이터가 로딩되며, 이후 사용자가 입력한 명령은 서버(Server)에서 처리되어 시트(ExcelCanvas)에 반영된다. 사용자는 이전 채팅 기록(ChatHistory)을 불러올 수 있고, 최종적으로 수정된 시트를 저장하면 서버에 해당 내용이 반영된다. 전체 과정은 UI와 Server 간의 메시지 흐름을 중심으로 구성된다.

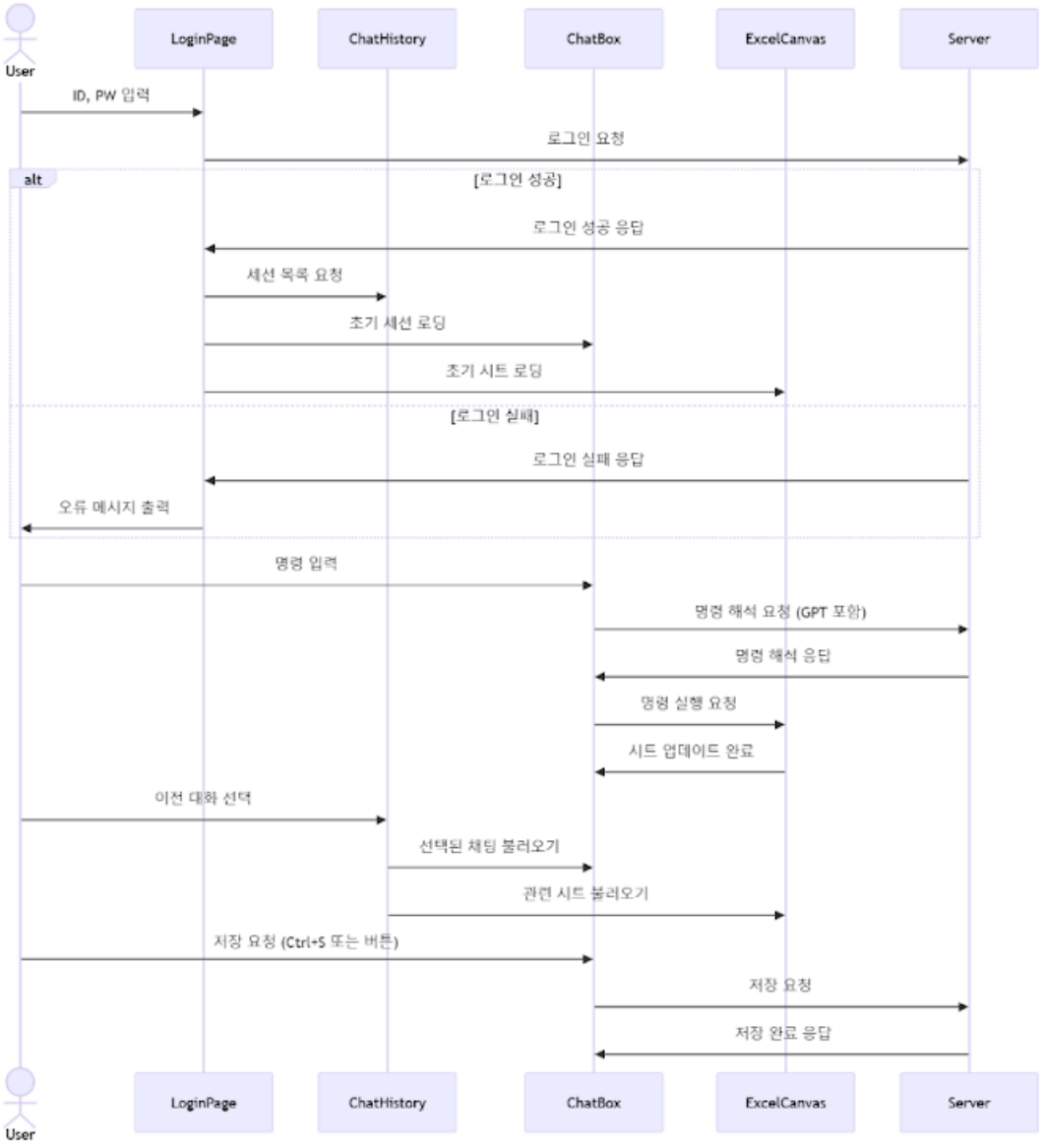

Figure 4.2: Sequence Diagram – 사용자 전체 흐름

5

System Architecture - Backend

5.1. Objectives

이 챕터는 back-end 시스템의 구조에 대해서 기술한다.

5.2. Overall Architecture

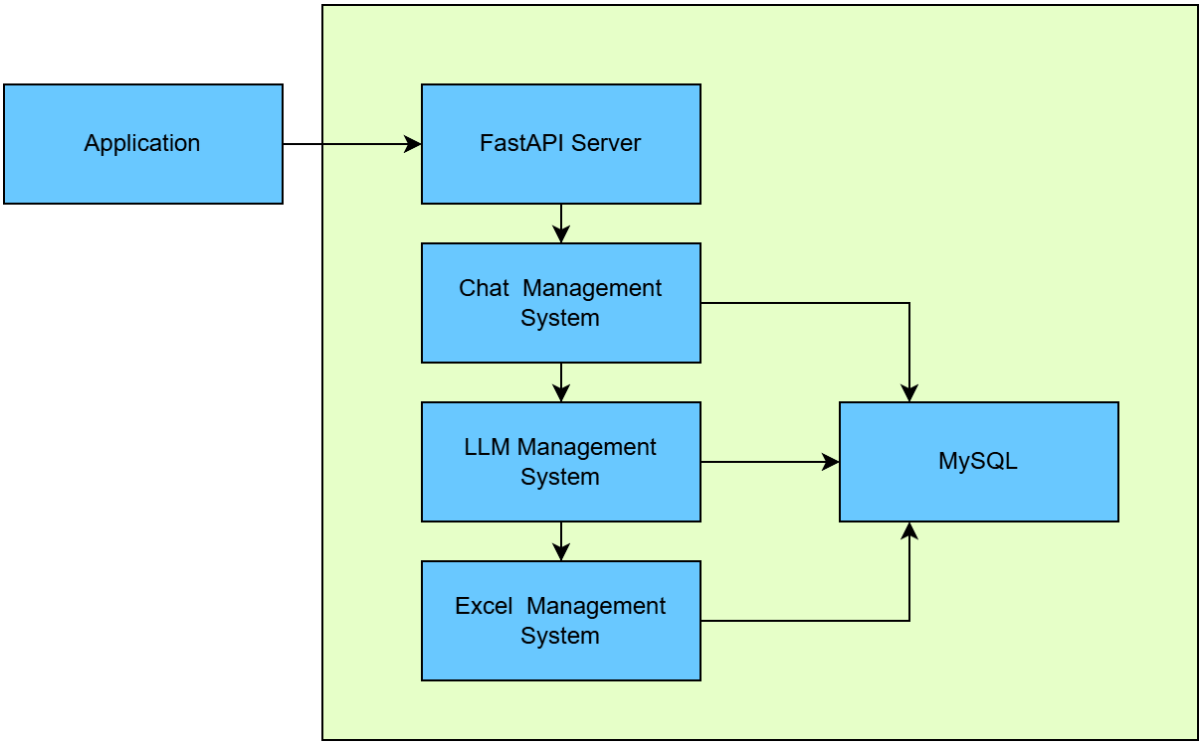

Figure 5.1: Overall Architecture

사용자로 하여금 엑셀을 학습할 필요 없이 자연어를 통해 원하는 엑셀 파일을 편집 할 수 있도록 지원하는 것이 시스템의 핵심 목표이다. front-end는 부터 요청을 받아서 FastAPI server에서 해당 요청을 처리한다. back-end에서 처리하는 요청으로는 사용자 인증, 채팅 세션 관리, 자연어 명령 처리, 엑셀 파일 조작 등이 있으며, OpenAI GPT API를 통해 자연어를 엑셀 명령어로 변환하여 사용자의 요구사항을 충족한다. 또한, 데이터베이스로는 MySQL을 사용한다.

5.3. Subcomponents

5.3.1. User Management System

Class Diagram

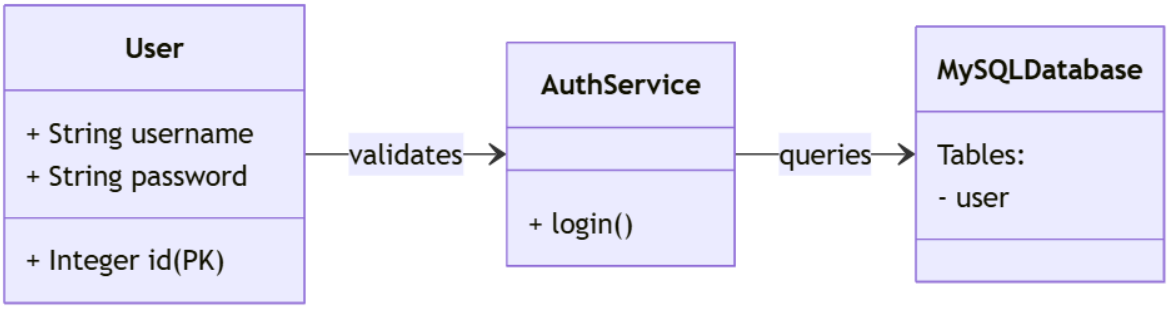

Figure 5.2: User Management System Class Diagram

AuthService는 사용자 인증과 로그인 처리를 담당하며, 이메일과 비밀번호 검증을 위해 데이터베이스와 상호작용한다.

Sequence Diagram

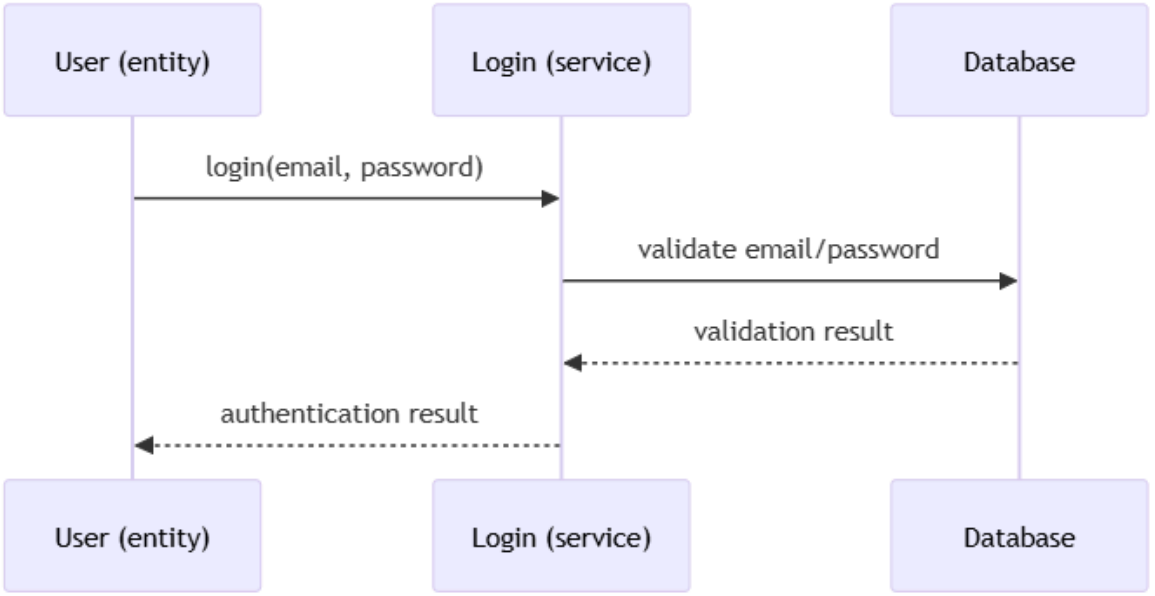

Figure 5.3: Sequence Diagram - User Management System

5.3.2. Chat Session System

Class Diagram

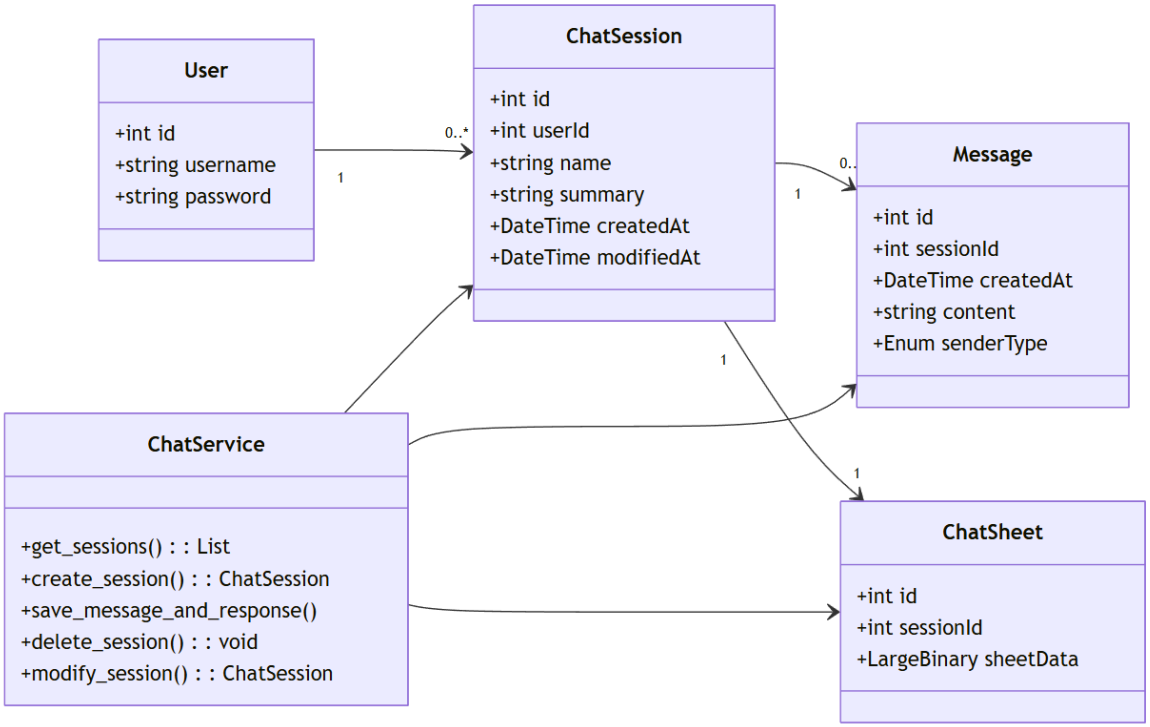

Figure 5.4: Class Diagram - Chat Session System

ChatService는 채팅 세션 생성, 메시지 저장 및 조회, 시트 데이터를 관리하는 역할을 수행하며, 사용자의 대화와 엑셀 연동 기능을 지원한다.

Sequence Diagram

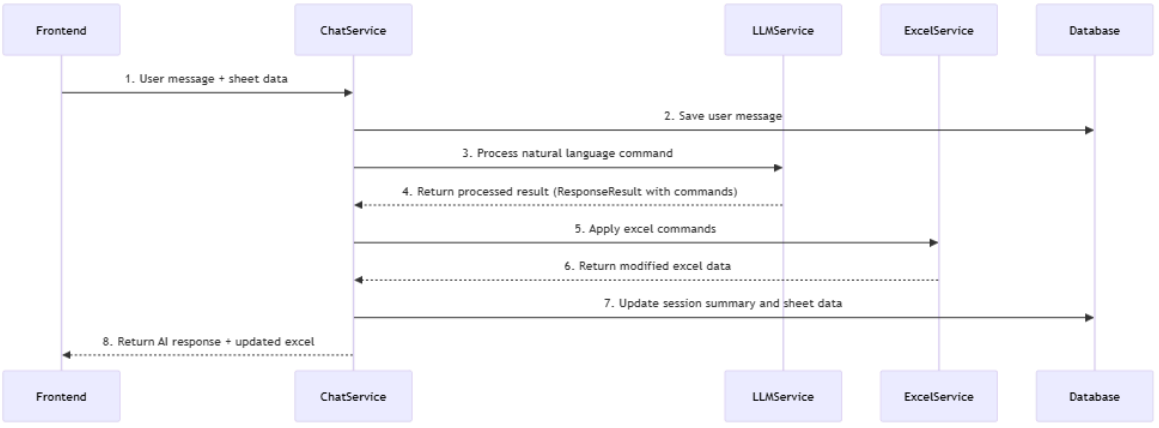

Figure 5.5: Sequence Diagram - Chat Session System

5.3.3. Excel LLM System

Class Diagram

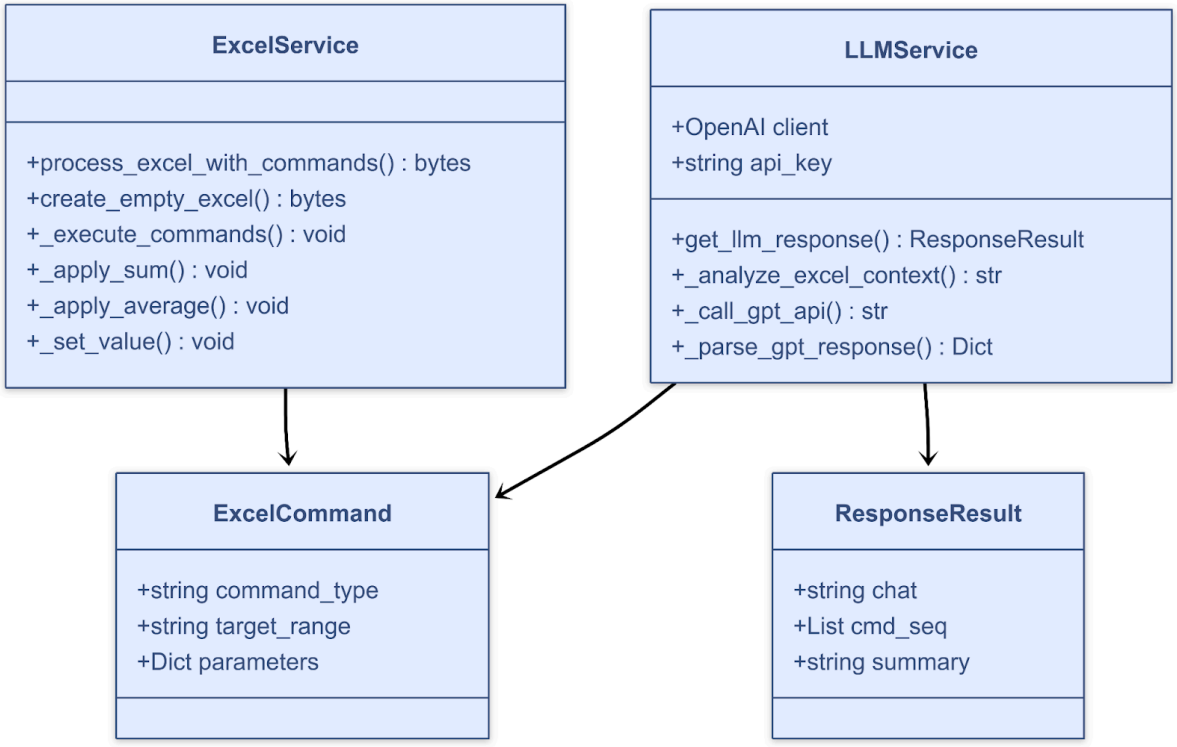

Figure 5.6: Class Diagram - Excel LLM System

엑셀 LLM 서비스는 각 채팅 메세지에 대한 답변을 하는 역할을 수행합니다. 일차적으로 LLM으로 자연어 명령을 분석하여 JSON 형식으로 반환합니다. 이후 엑셀 파일 수정 사항이 있다면 ExcelCommand를 ExcelService에 전달하여 엑셀 파일을 수정합니다.

Sequence Diagram

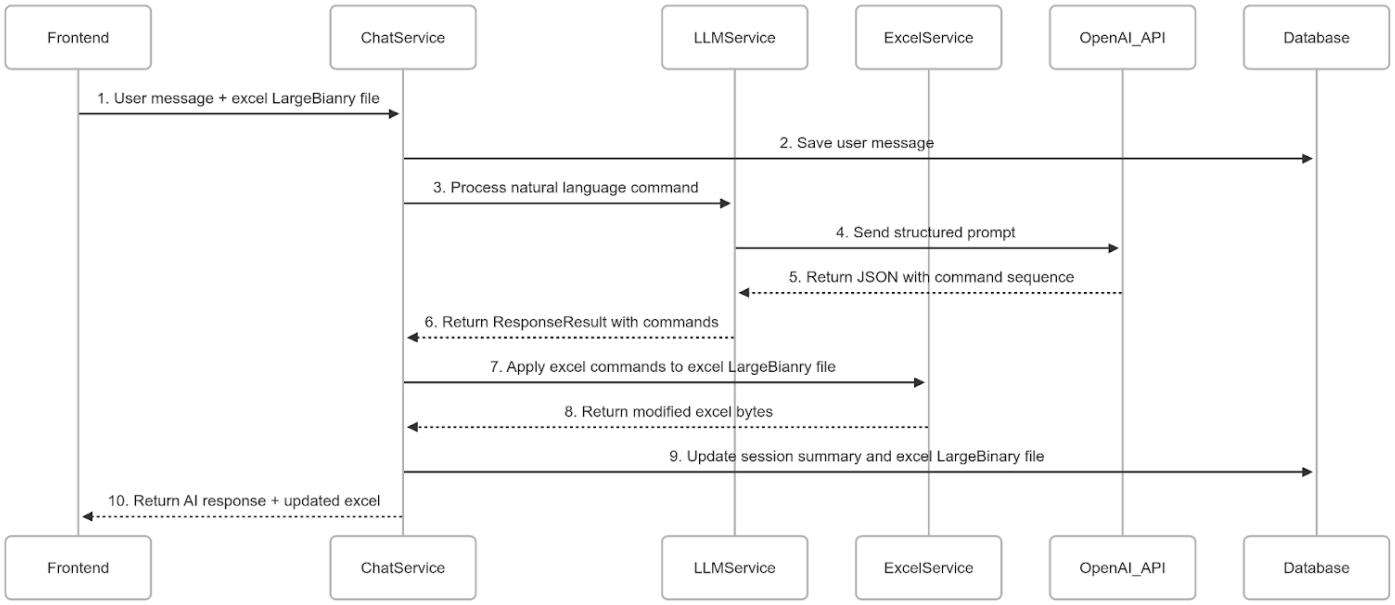

Figure 5.7: Sequence Diagram - Excel LLM System

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. HTTP

HTTP(HyperText Transfer Protocol)는 웹에서 데이터를 교환하는 데 사용되는 표준 프로토콜이다. 클라이언트와 서버 간의 요청(Request)과 응답(Response) 구조로 동작하며, 일반적으로 웹 페이지를 불러오거나 API를 호출할 때 사용된다. HTTP는 상태를 유지하지 않는(stateless) 프로토콜로, 각 요청은 독립적으로 처리된다.

6.3. Axios

Axios는 JavaScript에서 널리 사용되는 Promise 기반 HTTP 클라이언트 라이브러리이다. 브라우저와 Node.js 환경에서 모두 사용할 수 있으며, JSON 데이터 처리를 손쉽게 해주고 요청 및 응답의 인터셉터(interceptor), 에러 처리, 타임아웃 설정 등 다양한 편의 기능을 제공한다. 특히, XMLHttpRequest나 fetch API보다 사용성이 뛰어나고 코드가 간결하여 REST API와의 통신 시 널리 활용된다.

6.4. Authentication

6.4.1. Log In

Request

Attribute	Detail	
Protocol	HTTP	
Method	POST	
Request Body	username	Username(for identification)
	password	User password

Figure 6.1: Table of Log in Request

Response

Attribute	Detail	
Protocol	HTTP	
Success Code	200 OK	
Failure Code	HTTP error code - 400(Bad Request) - 401(Unauthorized) - 422(Validation Error)	
Success Response Body	username	Username
	userId	User ID from database(for identification)
Failure Response Body	Message	Failure message

Figure 6.2: Table of Log in Response

6.5. ChatSession

6.5.1 Get Sessions

Request

Attribute	Detail	
Protocol	HTTP	
Method	GET	
Query Parmeter	UserId	Unique identifier for the user

Figure 6.3: Table of Get Sessions Request

Response

Attribute	Detail		
Protocol	HTTP		
Success Code	200 OK		
Failure Code	HTTP error code - 404(Not Found) - 422(Validation Error)		
Success Response Body	List	id	Unique identifier for the session
		userId	Unique identifier for the user
		name	Name of session
		modifiedAt	Timestamp of the last message in the chat.
Failure Response Body	Message		Failure message

Figure 6.4: Table of Get Sessions Response

6.5.2. Create Session

Request

Attribute	Detail		
Protocol	HTTP		
Success Code	200 OK		
Failure Code	HTTP error code - 404(Not Found) - 422(Validation Error)		
Success Response Body	sessionId		Unique identifier for the session
	sessionName		Name of session
	sheetData		Binary data of an Excel (.xlsx) file
	Message	id	Unique identifier for the message
		createdAt	Timestamp of when the chat was created
		content	Content of the message
		senderType	Sender of the message ('USER' or 'AI')
Failure Response Body	Message		Failure message

Figure 6.5: Table of Create Session Request

Response

Attribute	Detail	
Protocol	HTTP	
Success Code	200 OK	
Failure Code	HTTP error code - 201(Created) - 422(Validation Error)	
Success Response Body	id	Unique identifier for the session
	message	User input in natural language
	sheetData	Binary data of an Excel (.xlsx) file
Failure Response Body	Message	Failure message

Figure 6.6: Table of Create Session Response

6.5.3. Modify Session

Request

Attribute	Detail	
Protocol	HTTP	
Method	PUT	
Path Variable	sessionId	Unique identifier for the session
Request Body	name	Name of session

Figure 6.7: Table of Modify Session Request

Response

Attribute	Detail	
Protocol	HTTP	
Success Code	200 OK	
Failure Code	HTTP error code - 404(Not Found) - 422(Validation Error)	
Success Response Body	id	Unique identifier for the session
	userId	Unique identifier for the user
	name	Name of session
	modifiedAt	Timestamp of the last message in the chat.
Failure Response Body	Message	Failure message

Figure 6.8: Table of Modify Session Response

6.5.4. Delete Session

Request

Attribute	Detail	
Protocol	HTTP	
Method	Delete	
Path Variable	sessionId	Unique identifier for the session

Figure 6.9: Table of Delete Session Request

Response

Attribute	Detail	
Protocol	HTTP	
Success Code	204 No Content	
Failure Code	HTTP error code - 404(Not Found) - 422(Validation Error)	
Failure Response Body	Message	Failure message

Figure 6.10: Table of Delete Session Response

6.6. Message

6.6.1 Get Messages

Request

Attribute	Detail	
Protocol	HTTP	
Method	GET	
Path Variable	sessionId	Unique identifier for the session

Figure 6.11: Table of Get Messages Request

Response

Attribute			Detail
Protocol	HTTP		
Success Code	200 OK		
Failure Code	HTTP error code - 404(Not Found) - 422(Validation Error)		
Success Response Body	sessionId		Unique identifier for the session
	userId		Unique identifier for the user
	name		Name of session
	modifiedAt		Timestamp of the last message in the chat.
	sheetData		Binary data of an Excel (.xlsx) file
	Message (list)	id	Unique identifier for the message
		createdAt	Timestamp of when the chat was created
		content	Content of the message
		senderType	Sender of the message ('USER' or 'AI')
Failure Response Body	Message		Failure message

Figure 6.12: Table of Get Messages Response

6.6.2 Send Message

Request

Attribute	Detail	
Protocol	HTTP	
Method	POST	
Path Variable	sessionId	Unique identifier for the session
Request Body	message	User input in natural language
	sheetData	Binary data of an Excel (.xlsx) file

Figure 6.13: Table of Send Message Request

Response

Attribute	Detail		
Protocol	HTTP		
Success Code	200 OK		
Failure Code	HTTP error code - 404(Not Found) - 422(Validation Error)		
Success Response Body	sheetData		Binary data of an Excel (.xlsx) file
	Message	id	Unique identifier for the message
		createdAt	Timestamp of when the chat was created
		content	Content of the message
		senderType	Sender of the message ('USER' or 'AI')
Failure Response Body	Message		Failure message

Figure 6.14: Table of Send Message Response

7

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 4가지 entity로 이루어져 있다; User, Chat Session, Message, Chat Sheet. 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.

7.2.1. user

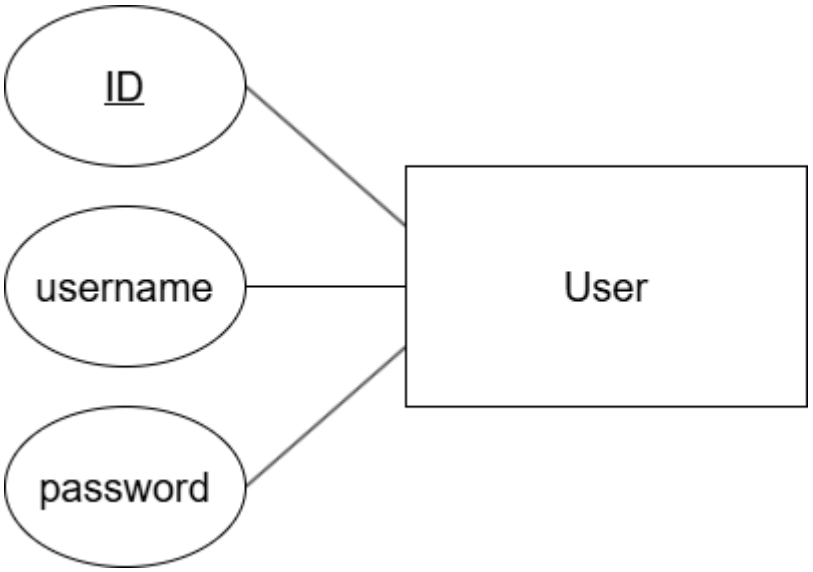

Figure 7.1: ER Diagram, Entity, User

User Entity는 애플리케이션 사용자에 해당한다. User Entity는 ID, username, password 를 attribute로 가지며, ID가 primary key이다. 이 attribute들은 애플리케이션 회원 가입 때 사용자가 기입한 정보이다.

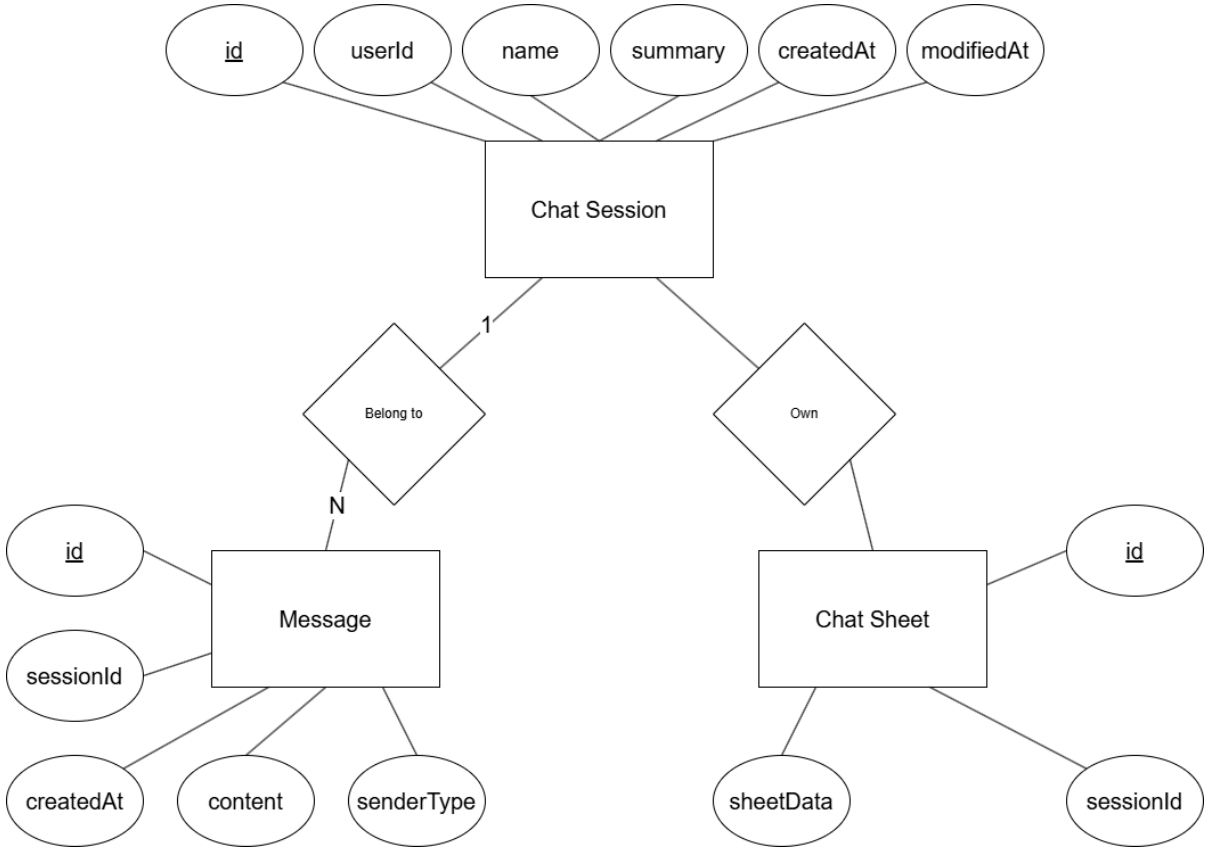

Figure 7.2: ER Diagram, Entity, Chat Session & Message & Chat Sheet

7.2.2. Chat Session

Chat Session Entity는 사용자별 채팅 세션을 나타낸다. primary key인 id, User의 id를 참조한 userid, 세션의 이름을 나타내는 name, 세션 요약 정보인 summary, 세션 생성 일시와 수정 일시를 나타내는 createdAt, modifiedAt attribute를 가지고 있다.

7.2.3. Message

Message Entity는 각 채팅 세션에서 주고 받은 메시지를 나타낸다. primary key인 id, ,Chat Session의 id에서 참조한 sessionId, 메시지 생성 일시를 나타낸 createdAt, 메시지 내용을 저장하는 content, 메시지 발신자 구분을 위한 senderType attribute를 가지고 있다.

7.2.4. Chat Sheet

Chat Sheet Entity는 채팅 세션과 연결된 엑셀 시트 데이터를 나타낸다. primary key인 id,Chat Session의 id에서 참조한 sessionId, 엑셀 시트 데이터를 Large Binary 형식으로 저장한 sheetData attribute를 가지고 있다.

7.3. Relational Schema

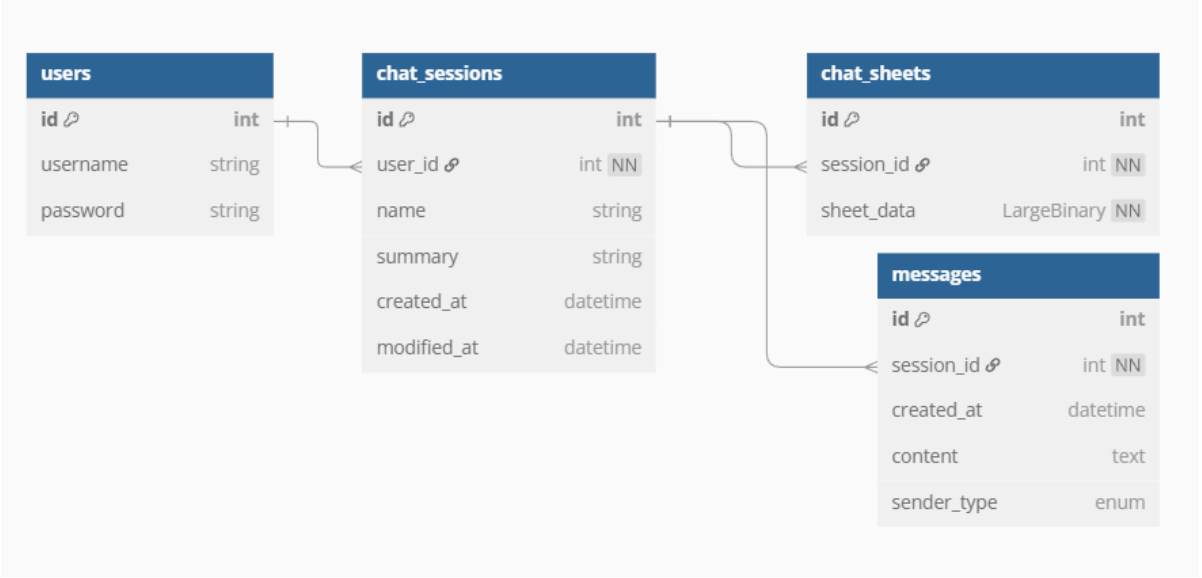

Figure 7.3: Relational Schema

7.4. SQL DDL

7.4.1. User

```sql
CREATE TABLE User (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(100) NOT NULL UNIQUE,
    password VARCHAR(100) NOT NULL
);
```

Figure 7.4: User sql schema

7.4.2. Chat Session

```sql
CREATE TABLE ChatSession (
    id INT AUTO_INCREMENT PRIMARY KEY,
    userId INT NOT NULL,
    name VARCHAR(255),
    summary VARCHAR(500),
    createdAt DATETIME DEFAULT CURRENT_TIMESTAMP,
    modifiedAt DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    FOREIGN KEY (userId) REFERENCES User(id)
        ON DELETE CASCADE
);
```

Figure 7.5: Chat Session sql schema

7.4.3. Message

```sql
CREATE TABLE Message (
    id INT AUTO_INCREMENT PRIMARY KEY,
    sessionId INT NOT NULL,
    createdAt DATETIME DEFAULT CURRENT_TIMESTAMP,
    content TEXT NOT NULL,
    senderType ENUM('USER', 'AI') NOT NULL,
    FOREIGN KEY (sessionId) REFERENCES ChatSession(id)
        ON DELETE CASCADE
);
```

Figure 7.6: Message sql schema

7.4.4. Chat Sheet

```sql
CREATE TABLE ChatSheet (
    id INT AUTO_INCREMENT PRIMARY KEY,
    sessionId INT NOT NULL,
    sheetData LargeBinary NOT NULL,
    FOREIGN KEY (sessionId) REFERENCES ChatSession(id)
        ON DELETE CASCADE
);
```

Figure 7.7: Chat Sheet sql schema

8

Testing Plan

8.1. Objectives

본 장에서는 개발 테스트(Development Testing)와 사용자 테스트(User Testing)의 두 가지 주요 하위 그룹으로 테스트 계획을 설명한다. 이러한 테스트의 목적은 애플리케이션의 잠재적인 오류와 결함을 발견하고 수정하여 최종적으로 안정적인 서비스를 사용자에게 제공하기 위함이다.

8.2. Testing Policy

8.2.1. Development Testing

개발 단계에서 발생할 수 있는 오류를 조기에 발견하고 수정하기 위한 테스트를 수행한다. 프로젝트 특성상 로그인 기능은 단순히 DB에서 사용자 ID/PW 조회로만 처리하며, 보안 기능(암호화, 세션 관리 등)은 구현하지 않는다. 프론트엔드와 백엔드 간의 데이터 흐름 검증 및 팀원 간의 코드 리뷰를 통해 주요 기능의 완성도를 높인다. 주요 테스트 항목으로는 프론트엔드와 백엔드 간의 API 연동 테스트, 입력된 명령어에 대한 결과 데이터 처리 및 시각화 검증, 그리고 엑셀 시트 생성과 데이터 조회 등 각 기능의 기본 동작 확인을 포함한다.

Performance
사용자가 요청한 엑셀 생성/조회 작업에 대해 간단한 경우는 10초 이내, 복잡한 경우(시트에 많은 작업이 포함되는 경우)는 1분 이내에 완료되는 것을 목표로 테스트한다. 또한 데이터베이스에 결과가 저장되고 프론트엔드로 응답이 전달되기까지 5초 이내를 목표로 한다. 최대 1000명의 사용자 계정 데이터를 관리할 수 있도록 테스트하며 대규모 부하는 요구사항에서 제외한다.

Reliability
시스템이 정상적으로 동작할 수 있도록 각 하위 모듈(프론트엔드, 백엔드, DB)에 대해 단위 테스트를 실시하여 오류를 반복적으로 확인하고 수정하며, 통합 단계에서도 전체 시스템의 동작을 검증하여 안정성을 확보한다.

Security
본 프로젝트에서는 로그인 페이지가 형식적으로만 존재하며, 실제로는 보안 기능(암호화, 세션 관리 등)을 구현하지 않는다. 따라서 보안 관련 테스트는 최소화하며, DB 접근 테스트 시 단순 사용자 인증(DB 조회)만을 검증한다.

8.2.2. User Testing

실제 사용자(개발팀원 및 수강생)를 대상으로 로컬 환경에서 소프트웨어를 시연하며 테스트를 진행한다. 사용자는 지정된 시나리오(정상 사용, 잘못된 입력, 예상치 못한 상황 등)를 수행하며 오류를 발견하고, 피드백을 수집하여 기능 개선에 반영한다. 사용자의 시나리오 진행 상황을 기록하여 테스트 완료 후 분석 자료로 활용하며, 사용자 인터페이스(UI)와 사용자 경험(UX)의 사용성도 함께 검토한다.

8.2.3. Testing Case

성능(Performance)과 신뢰성(Reliability)에 대해 각각 최소 5개 이상의 테스트 케이스와 시나리오를 설계하여 소프트웨어를 테스트한다. 각 테스트 케이스에 대해 결과를 기록하고 평가 보고서를 작성하여 품질을 확인한다.

8.2.4. Test Data Plan

테스트 수행을 위한 데이터셋을 별도로 구성한다. 예를 들어, 엑셀 파일의 기본 시트 데이터와 사용자의 계정 정보를 사전에 생성하여 각 테스트 단계에서 동일한 조건으로 테스트가 이루어지도록 준비한다. 이를 통해 테스트 결과의 일관성과 신뢰성을 높인다.

8.2.5. Test Result Reporting

테스트를 완료한 후 각 테스트 케이스별로 결과를 기록하고, 발견된 오류와 개선 사항을 정리하여 팀 내부 공유 문서로 관리한다. 최종적으로 테스트 평가 보고서를 작성하여 프로젝트 산출물로 제출한다.

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 9.1: HTML Logo

HTML은 웹 페이지를 위한 마크업 언어이며 W3C가 HTML과 CSS 표준의 공동 책임자이다. HTML을 통 해 제목, 단락, 목록 등 본문을 위한 구조적 의미를 나타내는 것뿐만 아니라 링크, 인용과 그 밖의 항목으로 구조적 문서를 만들 수 있는 방법을 제공받을 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 웹 사이트의 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. Docker

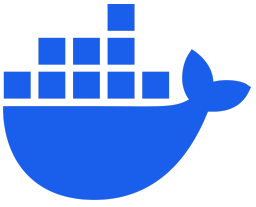

Figure 9.5: Docker Logo

Docker는 컨테이너 기반의 가상화 플랫폼으로, 데이터베이스(MySQL)를 손쉽게 배포하고 관리할 수 있도록 도와준다. 이를 통해 개발과 운영 환경에서 동일한 데이터베이스 환경을 유지할 수 있으며, 로컬 개발 환경에서도 손쉽게 MySQL 서버를 구동할 수 있다. 따라서 우리는 본 소프트웨어의 데이터베이스(MySQL) 운영을 위한 환경을 표준화하고 관리하기 위해 Docker를 사용하였다.

9.3.3. MySQL

Figure 9.6: MySQL Logo

MySQL은 전 세계적으로 널리 사용되는 관계형 데이터베이스 관리 시스템으로, 안정성과 성능이 검증된 데이터베이스 솔루션이다. 데이터 무결성 보장, 다양한 쿼리 처리, 트랜잭션 관리 등이 가능하여, 사용자 정보, 메시지, 엑셀 데이터 등 프로젝트의 핵심 데이터를 안전하게 저장하고 관리할 수 있다. 따라서 우리는 본 소프트웨어의 데이터 관리 및 안정성을 위해 MySQL을 사용하였다.

9.3.4. FastAPI

Figure 9.7: FastAPI Logo

FastAPI는 Python 기반의 현대적인 웹 프레임워크로, 빠른 개발과 높은 성능을 제공한다. 타입 힌트를 기반으로 한 자동 문서화 기능을 지원하여 Swagger UI와의 통합이 용이하며, RESTful API 설계와 비동기 처리를 통해 높은 확장성을 확보할 수 있다. 따라서 우리는 본 소프트웨어의 API 서버를 구축하고 Swagger UI를 통해 개발자와 사용자가 API를 쉽게 테스트하고 이해할 수 있도록 FastAPI를 사용하였다.

9.3.5. GPT 4.1

Figure 9.8: GPT4.1 Logo

GPT-4.1은 대규모 언어 모델로서 이전 버전보다 향상된 이해력과 정확성을 제공한다. 그리고 코드 작성과 디버깅, 창의적 글쓰기, 언어 번역 등에서 향상된 성능을 발휘한다. 또한 GPT-4.1은 더 많은 학습 데이터와 최적화된 알고리즘 덕분에 응답의 신뢰성과 일관성이 높다. 그리고 GPT-4.1은 Excel 데이터와 사용자의 요청을 추상화하여 반환하지 않으면서 일정 이상의 속도를 보장하기 때문에 제작하려는 프로그램에 가장 적합한 LLM이다.

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 license를 요구하는 기술 및 software의 사용을 지양한다.
- 사용되는 AI는 OpenAI의 GPT 모델을 기반으로 하며, 명령당 사용되는 토큰 수는 최소화하도록 개발한다.
- 외부 API의 사용량 제한을 고려하여, 요청 큐잉, 결과 캐싱, 실패 시 대체 안내 등의 예외 처리 방안을 포함하여 안정적으로 운영되도록 한다.
- 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
2025/06/01	Introduction and System Architecture	1.0	방기호
2025/06/01	System Architecture - Frontend and LATEX	1.0	이상훈
2025/06/01	System Architecture – Backend and Protocol Design	1.0	박동민
2025/06/01	System Architecture – Backend and Database Design	1.0	정관용
2025/06/01	Overall architecture, Testing and Development Plan	1.0	윤규성