성균관대학교
SUNG KYUN KWAN UNIVERSITY

AI-Tutoring CTF platform

Software Design Specification

소프트웨어공학개론

41분반 Team 2

팀장 홍형근

팀원 김태영

김혜인

송재현

신영섭

임석현

목차

List of Tables

List of Figures

1. Introduction

1.1. Objective

본 문서는 2023년 1학기 성균관대학교 소프트웨어공학개론 41분반 2팀에 의해 개발되는 “AI-Tutoring CTF Platform” 서비스의 소프트웨어 디자인 명세서이다.

본 문서는 개발팀을 비롯한 성균관대학교 소프트웨어공학개론 수강생, 조교, 교수 모두가 열람할 수 있다. 누구나 본 문서를 재배포하거나 수정할 수 있지만, 상업적 용도로 활용 시 개발팀의 허가가 필요하다.

본 문서는 웹 어플리케이션 기반의 “AI-Tutoring CTF Platform” 서비스에 대한 디자인 명세를 제시하려는 목적으로 작성되었다. 디자인은 이전에 작성된 요구사항 명세서를 기반으로 작성되었다. 본 서비스는 사이버 보안을 공부하려는 사용자에게 도움이 되는 문제와 문제 해결에 도움을 줄 수 있는 보조적인 기능을 제공한다. 이를 위해 웹 서버, 웹 인터페이스, 사용자 관리 시스템 등의 시스템이 개발되며, 디자인에 관련한 세부사항은 이하 문서에 명시한다.

1.2. Applied Diagrams

이 장에서는 본 문서에서 디자인을 명세화 하기 위해 사용한 다이어그램(Diagram)에 대한 설명을 담고 있다.

1.2.1. UML

Unified Modeling Language (UML)은 소프트웨어 디자인 설계를 시각화하기 위한 표준 모델링 언어이다. UML은 시스템의 시각화를 위해 여러 종류의 다이어그램을 사용하며, 본 문서에 사용된 개별 다이어그램에 대한 설명은 후술한다.

1.2.2. Class Diagram

Class diagram은 시스템의 클래스, 속성, 메소드 및 개체 간 관계를 통해 시스템의 구조를 설명하는 정적 구조 다이어그램이다.

1.2.3. Sequence Diagram

Sequence diagram은 개체가 서로 작동하는 방식 및 순서를 보여주는 상호 작용 다이어그램이다. 시스템의 작동 과정을 시간 순서대로 나타낼 수 있다.

1.2.4. Context Diagram

Context diagram은 시스템의 운영적 측면을 표현할 수 있다. 시스템의 경계, 외부 개체, 인터페이스를 나타냄으로써 시스템을 식별할 수 있다.

1.2.5. Deployment Diagram

Deployment diagram은 시스템의 물리적 구성 요소 간 위상적 관계를 나타낸다. 본 문서에서는 각 소프트웨어 컴포넌트가 어느 하드웨어 리소스에 배치되는지를 표현하였다.

1.2.6. State Diagram

State diagram은 시스템이나 프로세스의 상태와 그 상태들 사이의 전이를 나타낸다. 이를 통해 시스템의 동작 흐름과 상태 변화를 시각적으로 파악할 수 있다.

1.3 Applied Tools

이 장에서는 본 문서에서 디자인을 명세화 하기 위해 사용한 tool에 대한 설명을 담고 있다.

1.3.1. diagrams.net

Figure 1. Logo of diagrams.net

diagrams.net은 오픈소스 기반의 클라우드 기반 다이어그램 편집기로서, 다양한 종류의 다이어그램을 만들고 편집할 수 있도록 도와준다. 이를 통해 사용자는 간단한 흐름도부터 복잡한 프로세스 흐름, 기술적인 아키텍처 다이어그램, 조직도, UML, ER다이어그램 등 다양한 유형의 다이어그램을 제작할 수 있다. 사용자가 제작 할 다이어그램 유형을 선택하면 diagrams.net에서는 해당 유형을 위한 다양한 도구와 요소를 제공한다. diagrams.net은 다이어그램을 다양한 파일 형식(PNG, JPG, PDF 등)으로 내보낼 수 있도록 하여 사용자에게 편의를 제공한다. 또한, 오픈소스 기반이므로, 개인적 용도 혹은 기업에서 사용하더라도 무료로 사용이 가능하다.

1.3.2. Figma

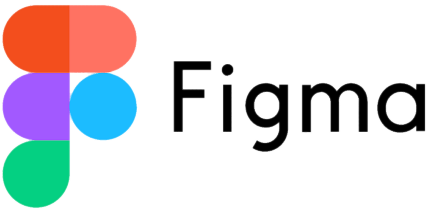

Figure 2. Logo of Figma

Figma는 협업을 위한 클라우드 기반 디자인 툴이다. 클라우드 기반으로 작동하므로, 사용자는 어디서든 인터넷 연결만 있으면 접속하여 작업이 가능하다. 또한, 작업물을 공유하여 여러 사용자가 동시에

작업 할 수 있으며, 프로젝트의 진행상황을 실시간으로 공유하고 피드백이 가능하다.

Figma는 그래픽 디자인, UI/UX디자인, 웹 디자인 등 다양한 분야에 활용할 수 있다. 사용자는 디자인 요소를 만들고 그림판에서 그림을 그리는 것과 같이 작업 할 수 있으며, 벡터 그래픽 도구, 디자인 시스템, 라이브 미리보기, 스타일 가이드, 팀 라이브러리, 플러그인 등 다양한 기능을 활용하여 효율적인 작업이 가능하다.

1.4. References

- Git Community, “Git”. https://git-scm.com/
- Facebook, “React”. https://reactjs.org/
- OpenJS Foundation, “Node.js”. https://nodejs.org/
- Nest, “Nest.js” https://nestjs.com
- Docker Incorporation, “Docker”. https://www.docker.com/
- Oracle, “MySQL”. https://www.mysql.com/

2. Overall System Architecture

2.1 Purpose

본 장의 목적은 전체 시스템 구조를 효과적으로 이해하기 위함이다. 시스템 구조를 여러 종류의 다이어그램을 통해 표현한다.

2.2. System Organization

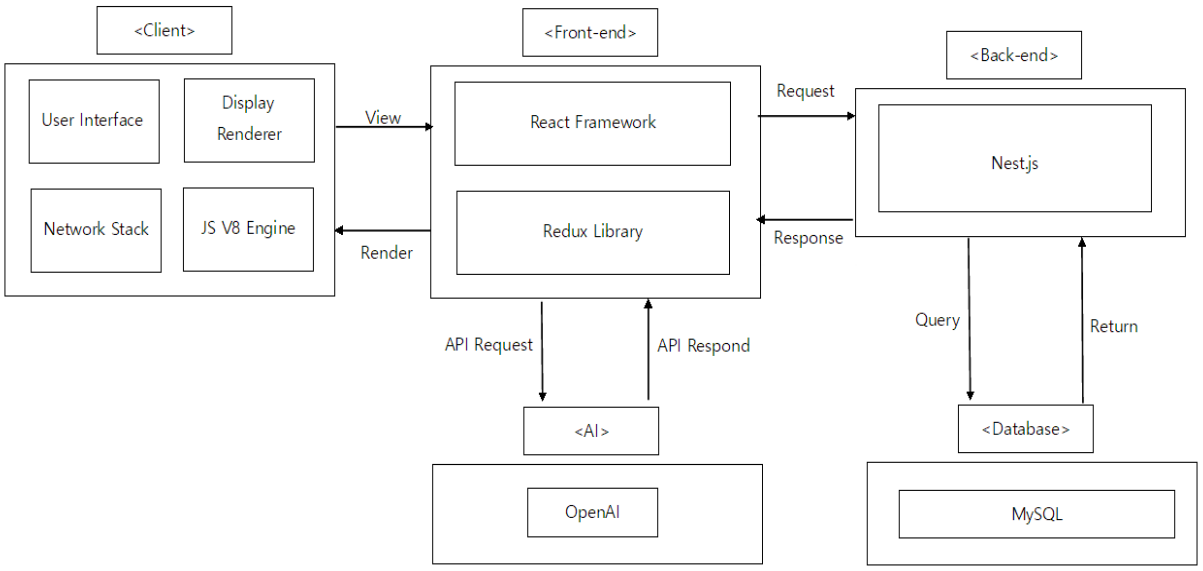

Figure 3. Overall System Organization

본 시스템은 크게 사용자, 프론트엔드, 백엔드로 구성된다. 사용자는 웹 브라우저에서 제공되는 렌더링과 JS 엔진을 통해 시스템의 UI에 접근할 수 있다. 본 시스템은 사용자가 최신 버전의 Chrome 웹 브라우저를 이용한다는 것을 상정한다.

프론트엔드는 사용자와의 상호작용을 처리하는 역할을 담당한다. 프론트엔드 서버는 React 등의 오픈 소스 프레임워크 및 라이브러리를 통해 구현된다. 프론트엔드 애플리케이션은 빌드된 정적 파일로 구성되어 웹 서버를 통해 클라이언트에게 제공된다. 프론트엔드 애플리케이션은 사용자 측에서 실행되며, 사용자의 요청을 처리하고 필요하다면 백엔드 서버와 통신하여 데이터에 접근한다.

백엔드 서버는 데이터에 대한 열람, 갱신, 삭제 등을 처리하는 역할을 담당한다. 프론트엔드 서버로부터 요청을 받을 시 사전에 정의된 비즈니스 로직을 수행 후 응답을 반환한다. 이를 위해 데이터베이스와 상호 작용을 하여 필요한 데이터를 처리한다. 백엔드 서버는 Nest와 같은 오픈 소스 프레임워크나 MySQL과 같은 데이터베이스를 사용하여 구현된다.

2.3. System Context

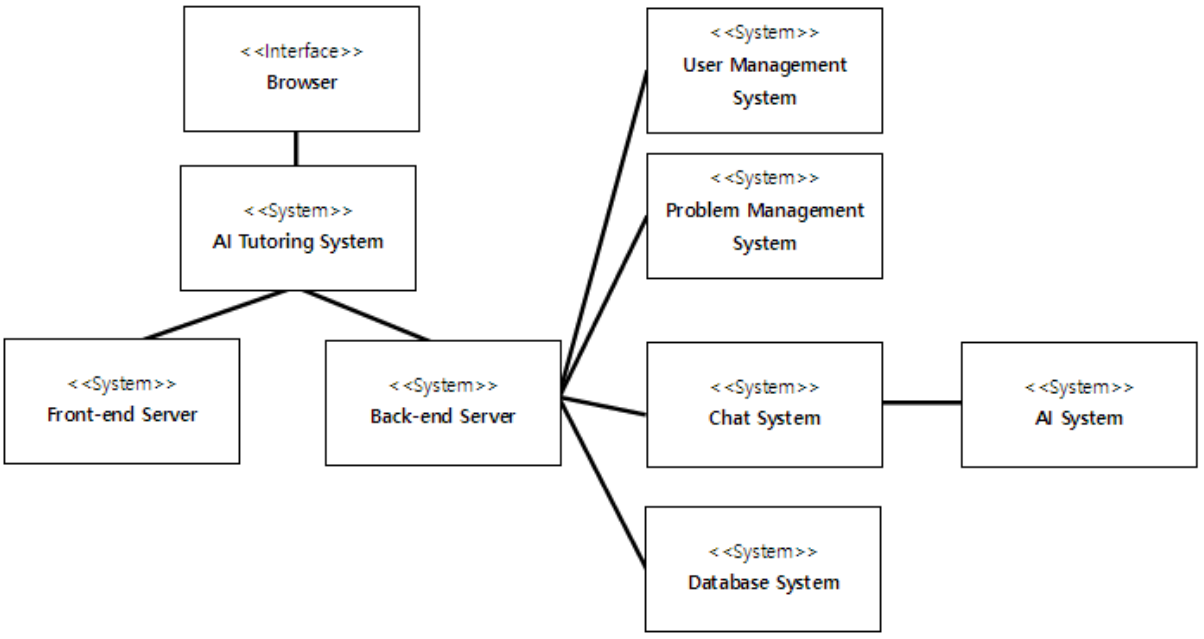

Figure 4. Context Model

2.4. Sequence Diagram

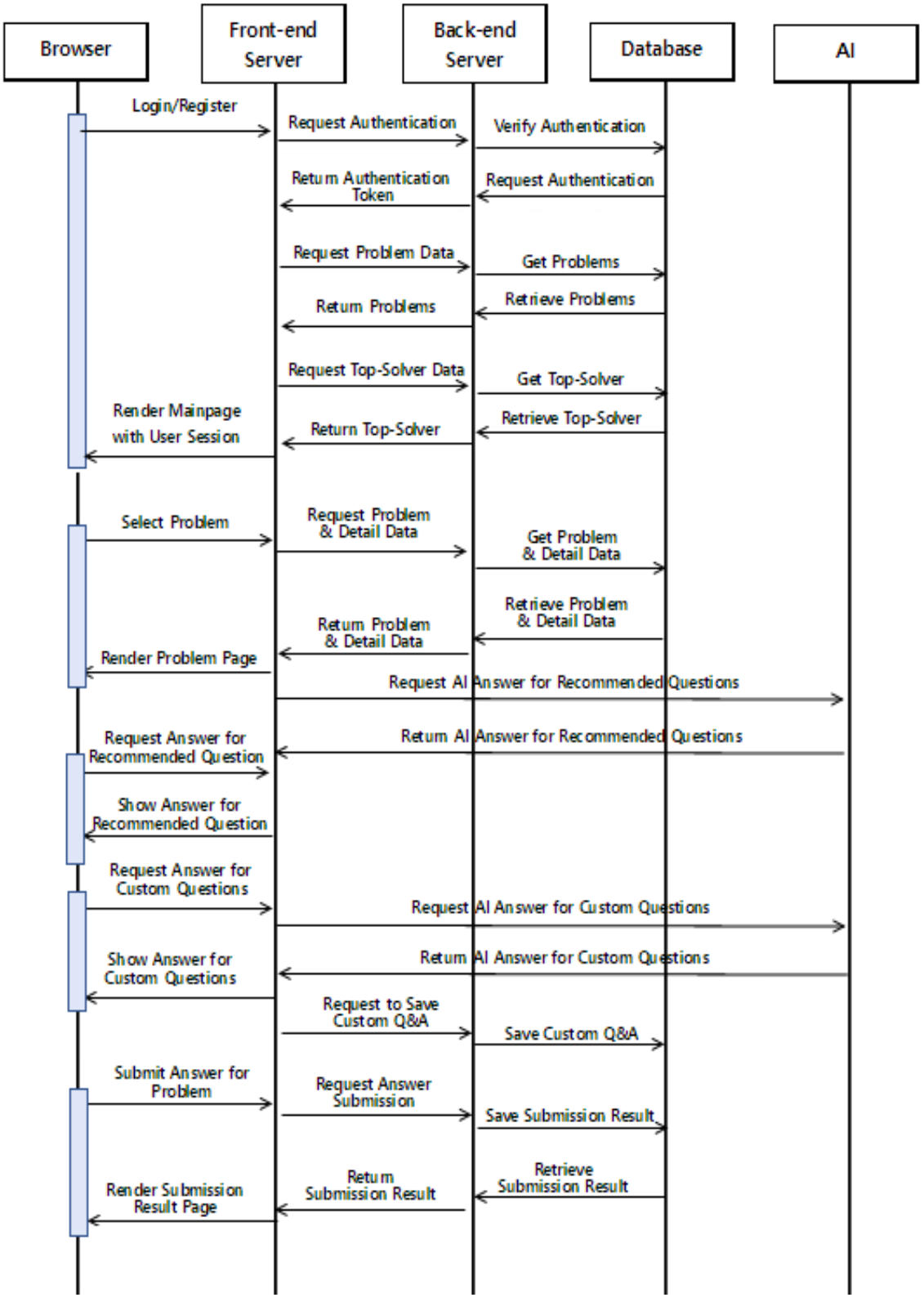

Figure 5. Sequence Diagram

2.5. Deployment Diagram

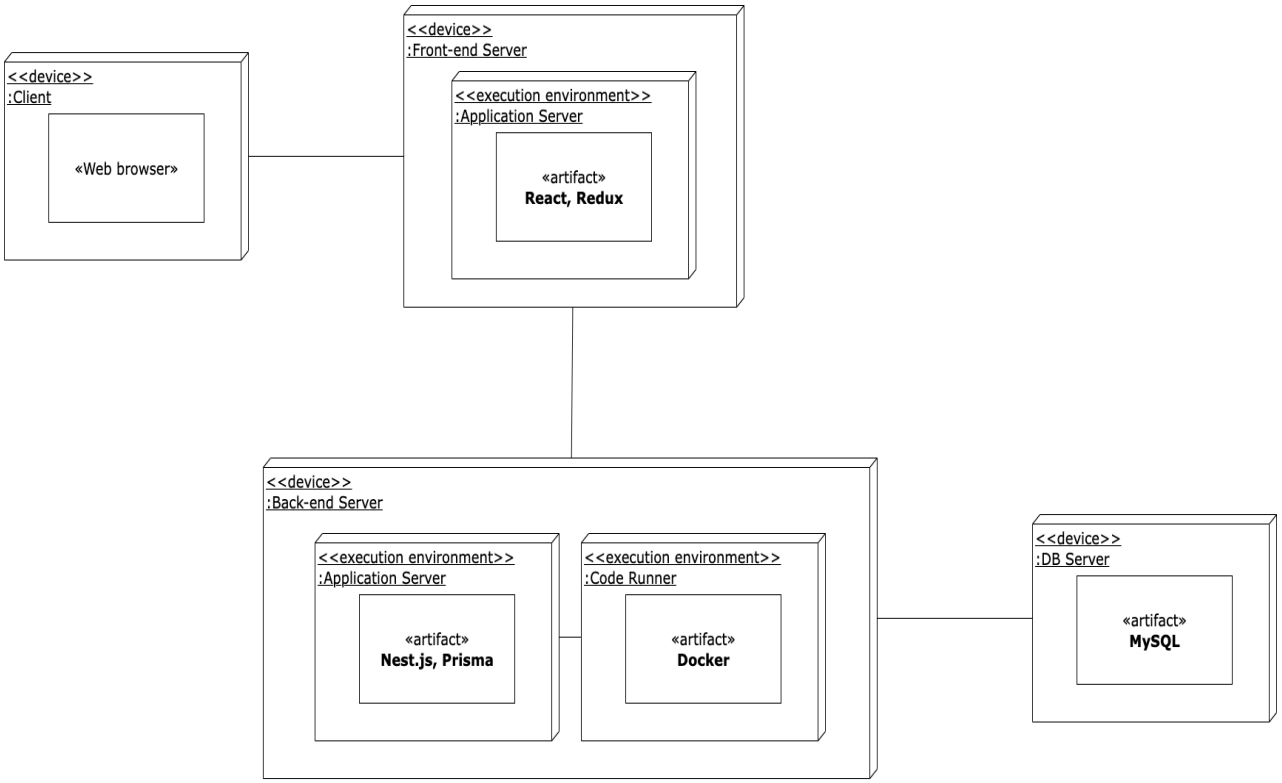

Figure 6. Deployment Diagram

3. Frontend System Architecture

3.1. Purpose

본 장에서는 유저와의 상호작용을 담당하는 프론트 엔드 시스템의 구조와 컴포넌트, 서브 컴포넌트, 메소드 등을 기술한다. 시스템을 구성하는 각 페이지는 컴포넌트에 해당되며, 각 페이지에는 사용자와의 상호작용을 위한 서브컴포넌트가 존재한다. 각 컴포넌트에 상응하는 클래스 다이어그램을 통해, 컴포넌트의 구성 및 동작 방식을 설명한다.

3.2. Overall Architecture

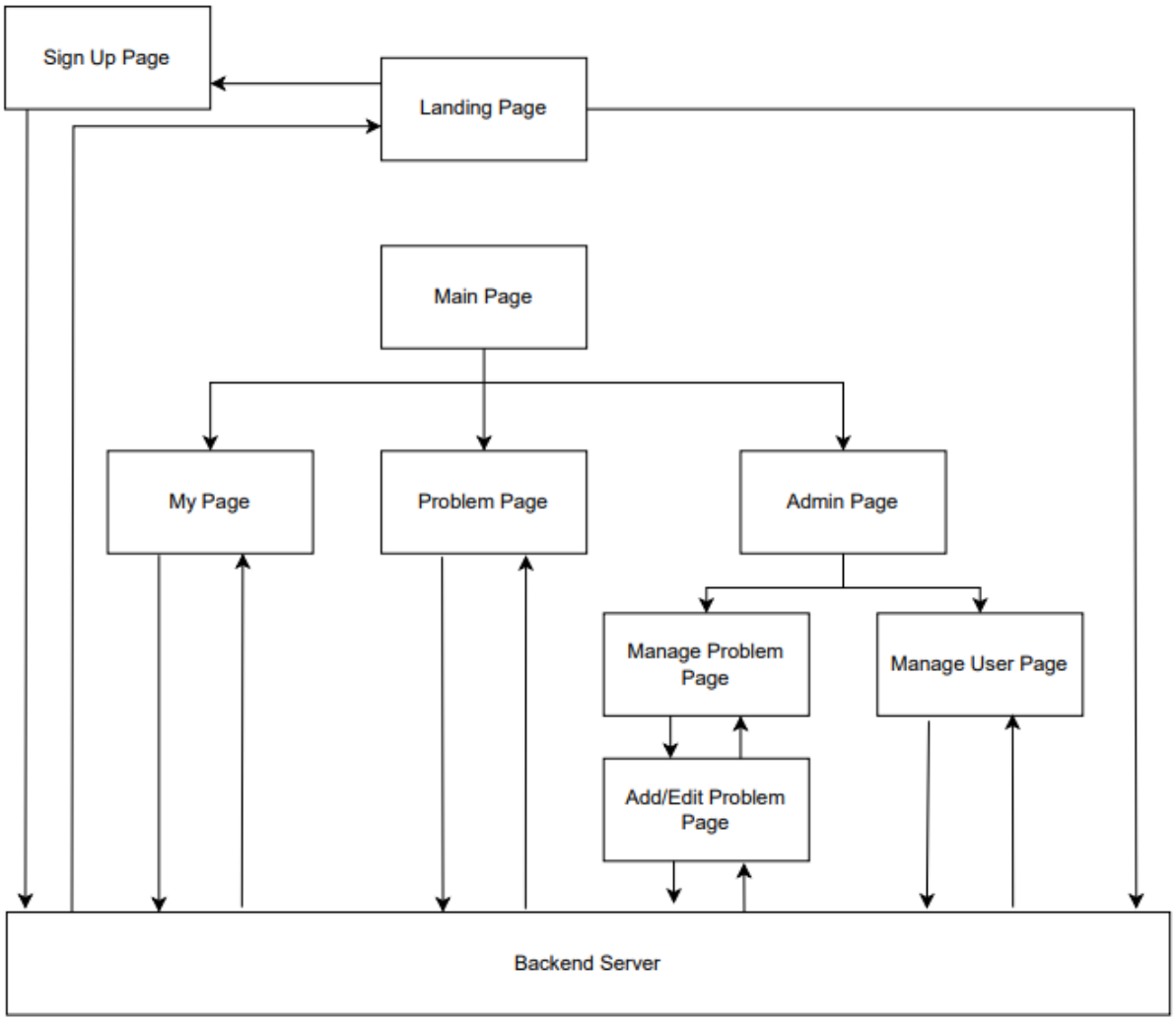

Figure 7. Overall Architecture of Frontend System

본 서비스의 시스템은 크게 Frontend와 backend, DB로 구성되어 있다. Frontend는 서버에 정보를 입출력하는 역할을 수행한다. 또한, Frontend 단에서 private route를 구현하여 로그인이 이루어지지 않은 경우, 본 시스템을 사용할 수 없고 Landing page로 리다이렉션된다. 로그인이 이루어진 후, Main page로 리다이렉션된 후, Main page 내에서 문제 목록 조회 및 리더보드 확인 등의 본 시스템의 기능을 이용할 수 있고, My page, Admin page, Problem page로 접근하여 본 시스템의 다른 기능들을 이용할 수 있다. Backend 및 Database는 유저의 로그인, 문제 정보, AI-tutor chat 등 시스템의 주요 기능 수행 시 Frontend와 상호작용할 수 있다.

3.3. Landing Page

본 시스템인 AI-Tutoring CTF Platform에 처음으로 접속하면 보여지는 페이지이다. 이 페이지에서는 사용자가 본인의 아이디와 비밀번호를 입력하여 로그인을 하거나, 회원가입을 할 수 있다.

3.3.1. Subcomponents

- Logo_Button: 본 시스템의 로고를 나타내는 Security Edu이다. 사용자가 미인증 상태이므로, landing page로 이동하는 버튼 기능을 수행한다.
- ID_Input: 사용자가 아이디를 입력하는 창이다.
- Password_Input: 사용자가 비밀번호를 입력하는 창이다.
- Login_Button: 사용자가 아이디와 패스워드를 입력한 후 로그인을 하기 위한 버튼이다.
- Sign_Up_Button: 사용자가 회원가입을 하기 위한 버튼이다.

3.3.2. Methods

- Show_Landing_Page(): 본 페이지인 Landing Page를 페이지를 띄운다.
- Login(): 사용자가 LogIn 버튼을 눌렀을 때 사용자가 입력한 아이디와 비밀번호를 입력받아 로그인을 실행한다. 아이디와 비밀번호가 적절하면 Show_Main_Page()를, 그렇지 않으면 Show_Error_Message()를 호출한다.
- Sign_Up(): 사용자가 Sign_Up 버튼을 눌렀을 때 Sign Up 페이지로 리다이렉션 한다.
- Show_Error_Message(): 아이디나 비밀번호 오류등으로 인해 로그인에 실패한 경우 에러 메시지를 출력한다. 메시지 확인 버튼을 누르면 Show_Landing_Page()가 호출된다.
- Show_Sign_Up_Page(): Sign Up Page로 리다이렉션한다.
- Show_Main_Page(): Main Page로 리다이렉션한다.

3.3.3. State Diagram

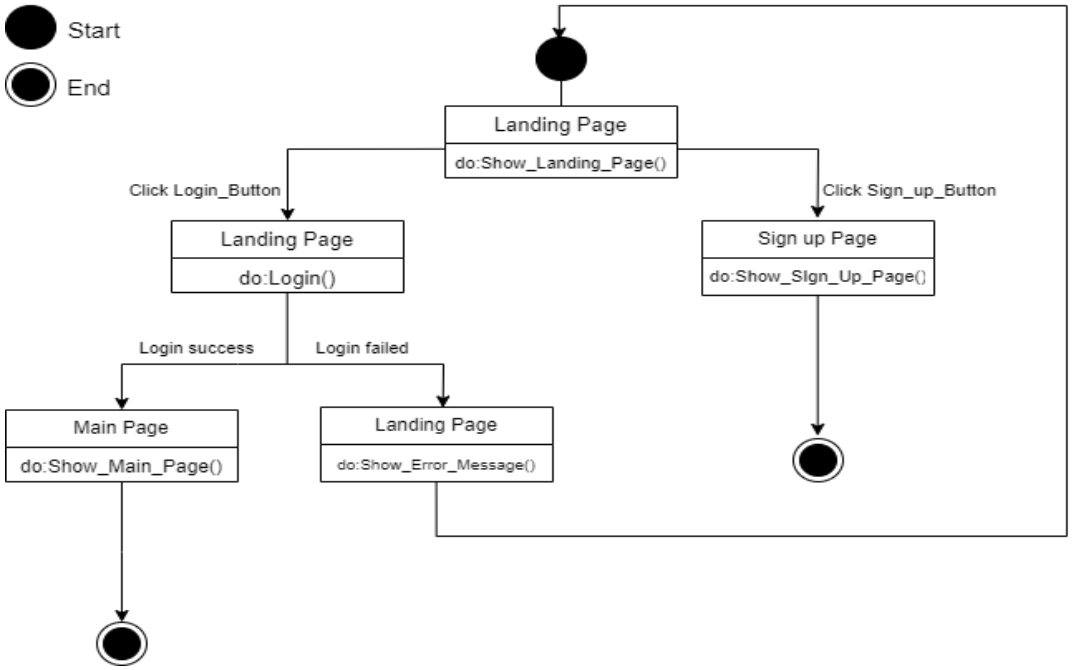

Figure 8. State Diagram of Landing Page

3.4. Sign Up Page

본 시스템에 가입을 하기 위한 페이지이다.

3.4.1. Subcomponents

- Logo_Button: 본 시스템의 로고를 나타내는 Security Edu이다. 사용자가 미인증 상태이므로, landing page로 이동하는 버튼 기능을 수행한다.
- ID_Input: 사용자가 가입할 아이디를 입력하는 창이다.
- Password_Input: 사용자가 비밀번호를 입력하는 창이다.
- Password_Confirm: 사용자가 가입할 비밀번호를 재입력하는 창이다.
- User_Name_Input: 사용자가 가입후 사용할 사용자 이름을 입력하는 창이다.
- Email_Input: 사용자의 이메일을 입력하는 창이다.
- Sign_Up_Button: 사용자가 가입 정보를 모두 입력하고나서 해당정보로 가입을 원할때 클릭하는 버튼이다.

3.4.2. Methods

- Show_Sign_Up_Page(): 본 페이지인 Sign_Up_Page를 띄운다.
- Click_Logo(): 사용자가 로고를 눌렀을 때 본 시스템의 초기 페이지로 돌아간다. 로그인이 되지 않은 상황에서는 Landing Page, 로그인이 된 상황에서는 Main Page로 리다이렉션 된다.
- Sign_Up(): 사용자가 Sign_Up_Button을 눌렀을 때 사용자가 입력한 아이디, 비밀번호, 사용자

이름을 입력받아 회원가입을 실행한다. 아이디와 사용자 이름에 대해 중복검사를 실시하여 중복될시, 사용자로부터 재입력을 요청한다.

- Show_Error_Message(): 아이디 중복이나 비밀번호 오류등으로 인해 로그인에 실패한 경우 에러 메시지를 출력한다. 메시지 확인 버튼을 누르면 Show_Landing_Page()가 호출된다.

3.4.3. State Diagram

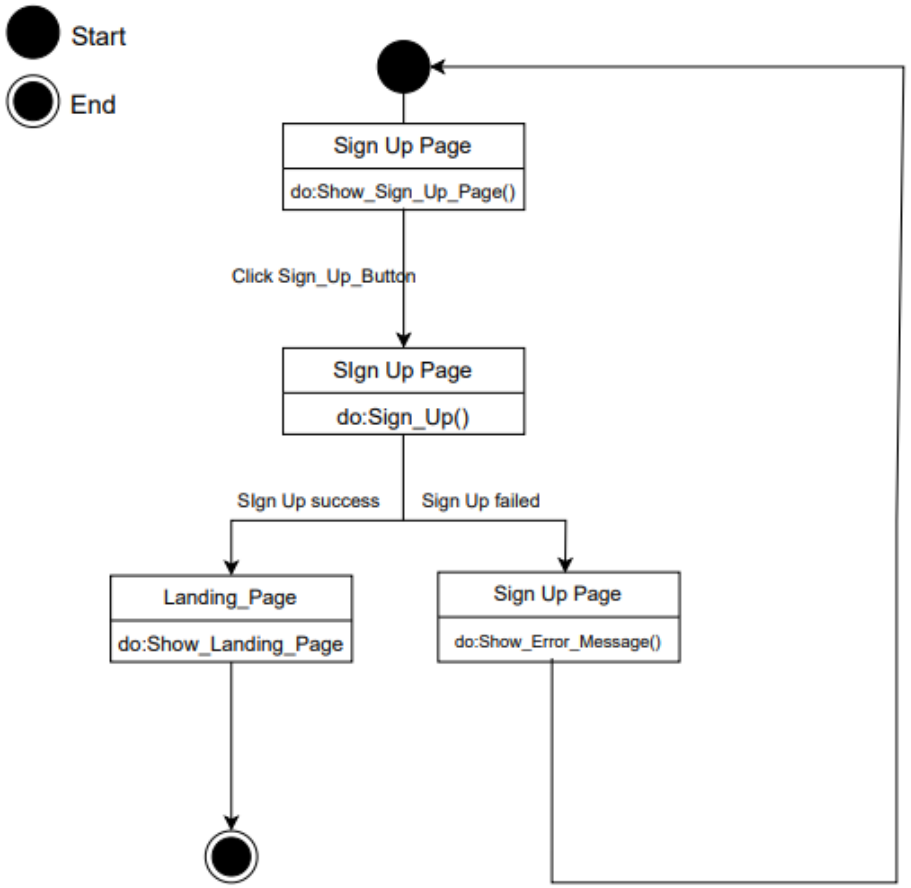

Figure 9. State Diagram of Sign Up Page

3.5. Main Page

사용자가 로그인에 성공하고 나서 보여지는 페이지이다. 이 페이지에는 문제리스트와 랭킹이 나타난다. 문제리스트 중 사용자가 문제를 선택하면 해당 문제에 대한 Problem Page로 리다이렉션 된다. 문제리스트 중 사용자가 풀이를 시도한 이력이 있는 문제들에 대해서는 해당 문제 선택란 옆에 사용자의 풀이에 대한 정답여부가 표시된다. 랭킹에는 본 시스템내에서 누적 맞은 문제 수가 가장 많은 User들의 User Name이 표시된다.

3.5.1. Subcomponents

Main Page는 크게 세가지 부분으로 구성된다.

- Head: 메인페이지의 상단에 위치한 부분이다.
- ProblemLists: 문제의 목록이 위치한 부분이다.
- Ranks: User들의 랭킹이 표시되는 부분이다.

(1) Head는 아래의 속성을 가진다.

- Logo_Button: 본 시스템의 로고를 나타내는 Security Edu이다. 사용자가 인증 상태이므로, main page로 이동하는 버튼 기능을 수행한다.
- My_Page_Button: 사용자가 개인정보를 열람하기 위해 My Page로 이동하기 위한 버튼이다.
- Admin_Page_Button: 관리자가 사용자나 문제를 관리하기 위한 페이지인 Admin Page로 이동하기 위한 버튼이다. 로그인 한 계정이 관리자 계정인 경우에 나타난다.
- Logout_Button: 로그인을 한 사용자가 로그아웃을 하기 위한 버튼이다.

(2) ProblemLists는 Problem들을 table형식으로 표현한 것이며, 각 Problem은 아래의 속성을 가진다.

- Problem_Number: 문제의 고유번호이다.
- Problem_Title: 문제의 제목이다.
- Problem_Description: 문제에 대한 간략한 설명이다.
- Correctness: 사용자가 해당 문제를 맞혔는지, 틀렸는지, 시도하지 않았는지에 대한 정보이다.

(3) Ranks는 본 시스템에서 순위가 높은 Ranker들을 table형식으로 표현한 것이며, Ranker는 아래의 속성을 가진다.

- Rank: 해당 랭커의 순위를 나타낸다.
- User_Name: 해당 랭커의 User name이다.
- Solved_Problems: 해당 랭커가 맞힌 문제의 개수이다.

3.5.2. Methods

(1) Head에서의 Method들은 아래와 같다.

- Show_Main_Page(): 로고를 클릭했을 때 동작한다. 사용자가 인증 상태이므로, 본 페이지인 Main Page를 띄운다.
- Show_My_Page(): 사용자가 My_Page_Button를 눌렀을 때 My Page으로 리다이렉션 한다.
- Show_Admin_Page(): 관리자가 Admin_Page_Button를 눌렀을 때 Admin Page으로 리다이렉션 한다.
- Logout(): 사용자가 Logout_Button을 눌렀을 때 로그아웃을 실행한다.

- Show_Landing_Page(): Landing Page로 리다이렉션 한다.

(2) Problem에서의 Method는 아래와 같다.

- Show_Problem_Page(): Problem을 클릭했을 때 해당 Problem Page로 리다이렉션 한다.

(3) Ranker에서만 동작하는 Method는 현재 버전을 개발하는 단계에서는 구현하지 않는다. 이후 버전에서 필요시 추가하도록 한다.

3.5.3. State Diagram

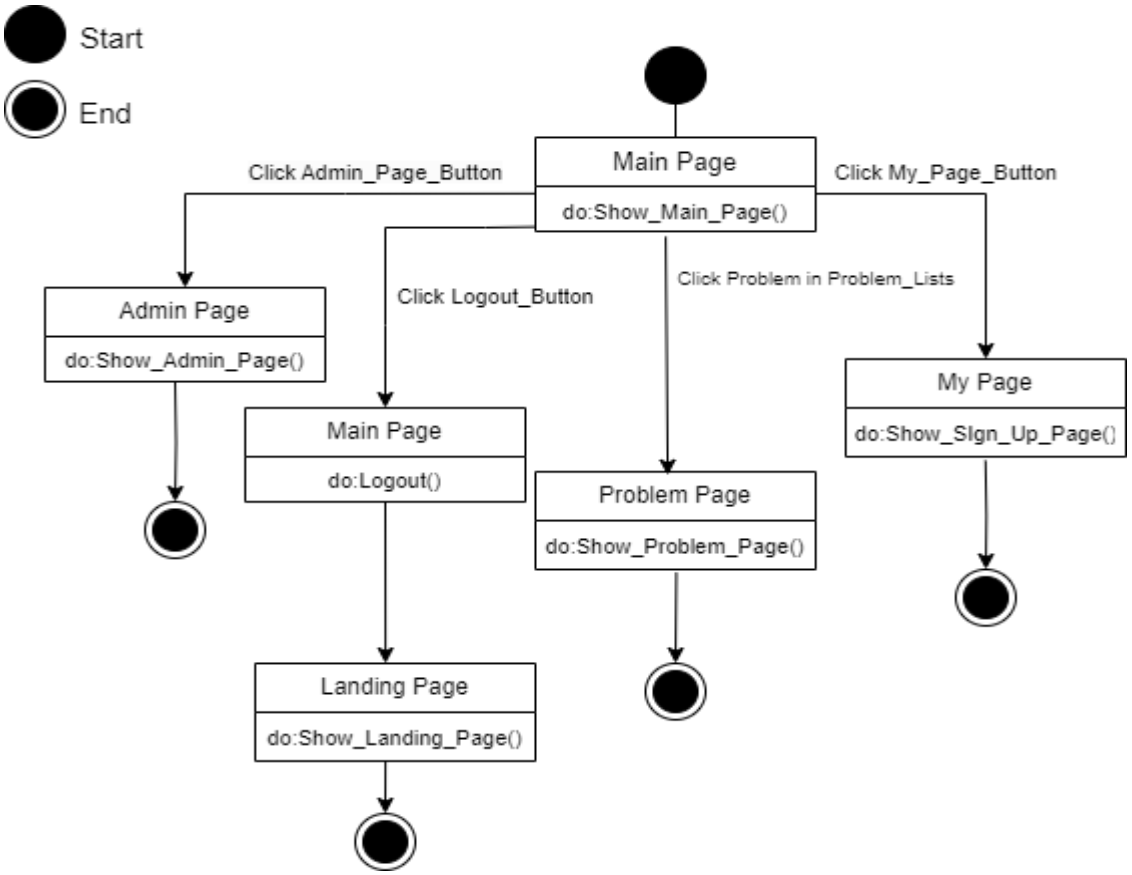

Figure 10. State Diagram of Main Page

3.6. My Page

사용자가 로그인을 한 이후에 화면 우측 상단의 My_Page_Button을 눌렀을 때 표시되는 페이지로, 사용자의 계정 정보조회를 하거나, 사용자 정보를 수정할 수 있는 페이지이다. 사용자는 Username 변경, password 변경, Email 변경, Ranking 조회, 계정삭제 행위를 할 수 있다.

3.6.1. Subcomponents

- Logo_Button: 본 시스템의 로고를 나타내는 Security Edu이다. 사용자가 미인증 상태이므로, landing page로 이동하는 버튼 기능을 수행한다.
- Change_Username_Button: 사용자가 자신의 Username 변경을 하기 위한 버튼이다.
- Change_Password_Button: 사용자가 자신의 비밀번호 변경을 하기 위한 버튼이다.

- Change_Email_Button: 사용자가 자신의 이메일 변경을 하기 위한 버튼이다.
- My_Ranking_Button: 사용자가 자신의 랭킹을 조회하기 위한 버튼이다.
- Delete_Button: 사용자가 자신의 계정을 삭제하기 위한 버튼이다.
- Logout_Button: 로그인을 한 사용자가 로그아웃을 하기 위한 버튼이다.
- Screen: 사용자가 Change_Username_Button, Change_Password_Button, My_Ranking_Button, Delete_Button을 선택했을 때 각 선택사항별 동작을 하기 위한 컴포넌트이다.
- Change_Username_Screen: 사용자 이름을 수정하기 위한 스크린이다.
- Change_Password_Screen: 사용자의 비밀번호 수정을 하기 위한 스크린이다.
- Change_Email_Screen: 사용자의 비밀번호 수정을 하기 위한 스크린이다.
- Ranking_Screen: 사용자의 랭킹을 조회하기 위한 스크린이다.
- Delete_Account_Screen: 사용자의 계정 삭제를 위한 스크린이다.

3.6.2. Methods

- Show_Main_Page(): 메인 페이지로 리다이렉션한다.
- Show_Change_Username_Screen(): Screen을 통해 사용자에게 새 Username, 비밀번호를 요청한다.
- Change_Username(): Show_Change_Username_Screen이 호출 된 후 사용자가 입력한 새 Username이 다른 사용자의 Username과 중복되지 않고, 비밀번호가 옳을 시 Username을 변경한다.
- Show_Change_Password_Screen(): Screen을 통해 사용자에게 현재비밀번호, 새 비밀번호, 새 비밀번호 확인 항목을 요청한다.
- Change_Password(): Show_Change_Password_Screen()을 통해 요청한 입력이 적절할 시 비밀번호를 변경한다.
- Show_Change_Email_Screen(): Screen을 통해 사용자에게 새 Email, 비밀번호를 요청한다.
- Change_Email(): Show_Change_Email_Screen()이 호출된 후 사용자가 입력한 새 Email이 다른 사용자의 Email과 중복되지 않고, 비밀번호가 옳을 시 이메일을 변경한다.
- Show_Ranking_Screen(): Screen에 사용자의 랭킹정보, 맞은 문제 정보를 띄운다.
- Delete_Account(): Screen에서 비밀번호를 요청한 뒤, 다시 한번 계정삭제 의사를 확인한 뒤, 계정삭제 의사가 확인 되면 계정을 삭제한다.
- Show_Landing_Page(): Landing Page로 리다이렉션한다.
- Show_My_Page(): 본 페이지를 띄운다.

3.6.3. State Diagram

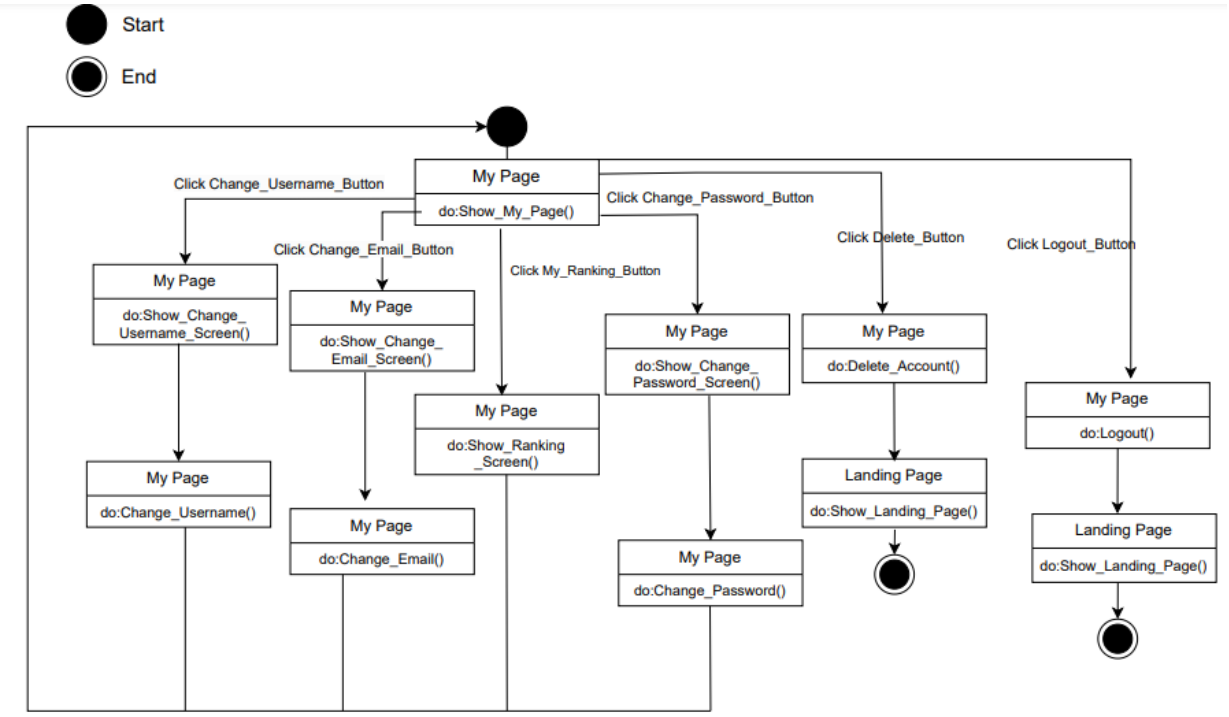

Figure 11. State Diagram of My Page

3.7. Problem Page

사용자가 Main Page의 문제리스트 중에서 풀이하고 싶은 문제를 클릭하였을 때 해당하는 문제를 풀이 할 수 있는 환경을 제공하는 페이지이다. 화면에는 문제의 내용과 정답 제출란, 그리고 문제 풀이가 어려울 때 AI-Tutor에게 질문을 할 수 있는 채팅창이 제공된다. 해당 문제에서 AI-Tutor에게 질문하기 좋은 질문거리들을 추천질문으로 제공하여, 사용자는 AI-Tutor에게 보다 편리하게 질문을 할 수 있도록 한다.

3.7.1. Subcomponents

- Logo_Button: 본 시스템의 로고를 나타내는 Security Edu이다. 사용자가 인증 상태이므로, Main Page로 이동하는 버튼 기능을 수행한다.
- Chat_Input: AI-Tutor으로 전송할 질문을 작성하는 form이다.
- Chat_Button: AI-Tutor으로 질문을 전송하는 버튼이다.
- Logout_Button: 로그인을 한 사용자가 로그아웃을 하기 위한 버튼이다.
- Submit_Button: 사용자가 자신이 선택한 정답을 제출하기 위한 버튼이다.
- Recommend_Question_Button: 사용자가 추천 질문에 대한 답변을 확인하기 위한 버튼이다. 이 버튼의 실행은 Chat_Input에 Recommend_Question_Button에 존재하는 값을 입력한 뒤 Chat_Button을 클릭하는 것과 기능상 동일하다.

3.7.2. Methods

- Show_Problem_Page(): 본 페이지를 띄운다.
- Submit(): 사용자가 작성한 답안을 DB에 저장된 답과 비교하기 위해 백엔드 서버에 요청을 보낸다.
- Show_Main_Page(): Main Page로 리다이렉션한다.
- Logout(): 사용자가 Logout_Button을 눌렀을 때 로그아웃을 실행한다.
- Show_Landing_Page(): Landing Page로 리다이렉션한다.
- Chat(): AI-Tutor으로 질문을 포함한 요청을 보낸다.
- Chat_Recommend(): Recommend_Question_Button을 눌렀을 시, 해당 추천 질문을 포함한 요청을 보낸다.

3.7.3. State Diagram

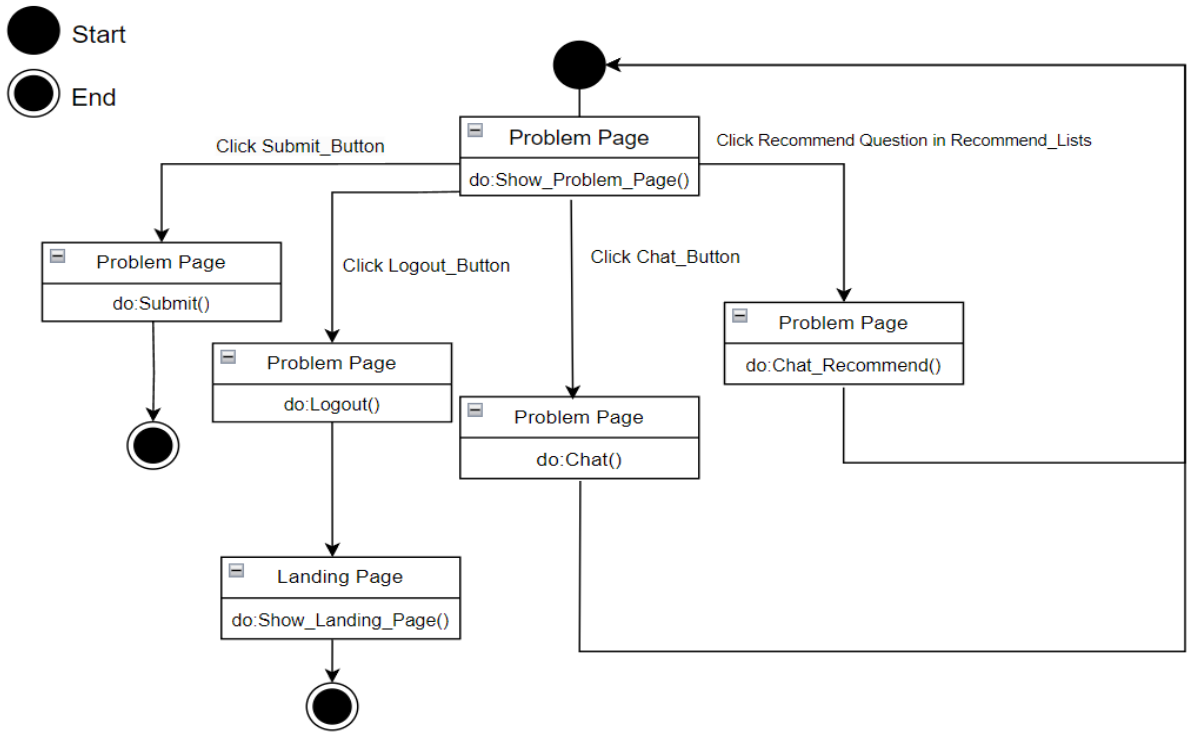

Figure 12. State Diagram of Problem Page

3.8. Admin Page

본 시스템의 관리자가 접근하는 페이지이다. 관리자가 본 페이지에서 Problem_Manage_Button을 클릭하면 본 시스템의 Problem 현황을 관리할 수 있는 Manage Problem Page로 리다이렉션 된다. 관리자가 본 페이지에서 User_Manage_Button을 클릭하면 본 시스템에 가입된 사용자들을 관리 할 수 있는 User Manage Page로 리다이렉션 된다.

3.8.1. Subcomponents

- Logo_Button: 본 시스템의 로고를 나타내는 Security Edu이다. 사용자가 인증된 상태이므로, main

page로 이동하는 버튼 기능을 수행한다.

- User_Manage_Button: user manage page로 이동하는 버튼이다.
- Problem_Manage_Button: problem manage page로 이동하는 버튼이다.
- Logout_Button: 로그인을 한 사용자가 로그아웃을 하기 위한 버튼이다.

3.8.2. Methods

- Show_Admin_Page(): 본 페이지인 Admin Page를 띄운다.
- Show_Main_Page(): Main Page로 리다이렉션한다.
- Show_Manage_Problem_Page(): Manage Problem Page로 리다이렉션한다.
- Show_Landing_Page(): Landing Page로 리다이렉션한다.
- Show_Manage_User_Page(): Manage User Page로 리다이렉션한다.
- Logout(): 사용자가 Logout_Button을 눌렀을 때 로그아웃을 실행한다.

3.8.3. State Diagram

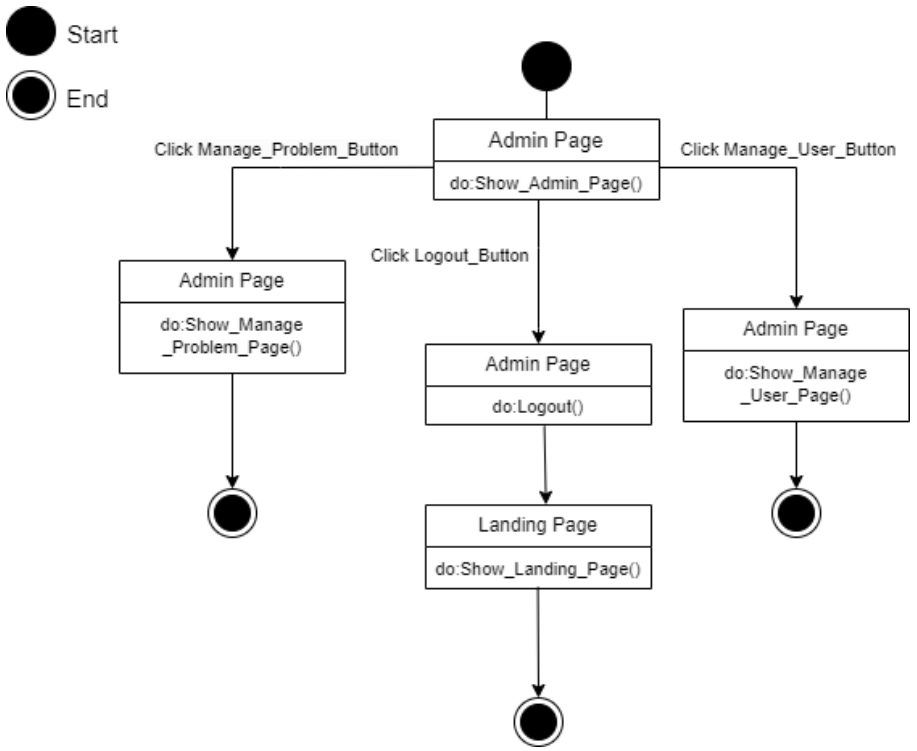

Figure 13. State Diagram of Admin Page

3.9. Problem Manage Page

본 시스템의 관리자가 본 시스템의 Problem 을 관리할 수 있는 페이지이다. 본 페이지에서는 업로드 된 모든 리스트가 나타나며, 각 문제 옆에 해당 문제를 수정, 삭제, 활성화 여부 설정을 할 수 있도록 하는 버튼이 나타난다. 본 페이지의 하단에는 새로운 문제를 추가를 위한 Add New Problem 버튼이 있다.

3.9.1. Subcomponents

- Logo_Button: 본 시스템의 로고를 나타내는 Security Edu이다. 사용자가 인증된 상태이므로, main page로 이동하는 버튼 기능을 수행한다.
- Add_Button: 새로운 문제를 생성하기 위한 버튼이다.
- Edit_Button: 기존 문제의 속성을 수정하기 위한 버튼이다.
- Hide_Button: 기존 문제를 숨기기 위한 버튼이다.
- Delete_Button: 기존 문제를 삭제하기 위한 버튼이다.
- Logout_Button: 로그인을 한 사용자가 로그아웃을 하기 위한 버튼이다.

3.9.2. Methods

- Show_Main_Page(): Main Page로 리다이렉션한다.
- Show_Add_Edit_Page(): Add/Edit Problem Page로 리다이렉션한다.
- Show_Landing_Page(): Landing Page로 리다이렉션한다.
- Logout(): 사용자가 Logout_Button을 눌렀을 때 로그아웃을 실행한다.
- Hide(): 해당 문제를 숨김 처리한다.
- Delete(): 해당 문제를 삭제한다.
- Logout(): 사용자가 Logout_Button을 눌렀을 때 로그아웃을 실행한다.

3.9.3. State Diagram

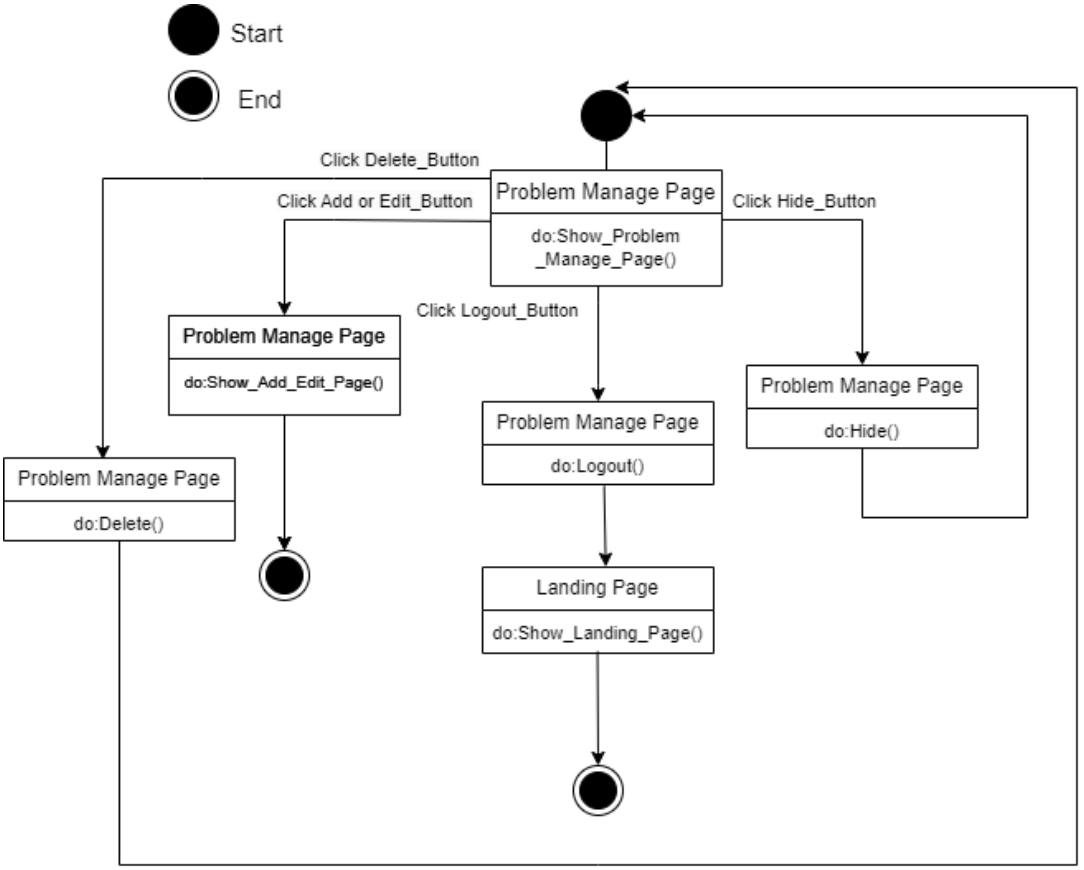

Figure 14. State Diagram of Problem Manage Page

3.10. User Manage Page

본 시스템의 관리자가 본 시스템의 User를 관리할 수 있는 페이지이다. 본 페이지에서는 본 시스템에 가입된 User들의 목록이 나타난다. 각 User별 User Name이 나타나며, 그 옆에는 Info 버튼, Ban 버튼이 존재한다. Info버튼은 해당 User의 정보를 나타내는 팝업을 띄운다. 해당 팝업창에서 별도의 버튼을 통해 User를 관리자로 승격시킬 수 있다. Ban 버튼을 클릭하면 팝업창에서 기간을 선택하여 해당 사용자의 서비스 이용을 그동안 중지시킬 수 있다.

3.10.1. Subcomponents

- Logo_Button: 본 시스템의 로고를 나타내는 Security Edu이다. 사용자가 인증된 상태이므로, main page로 이동하는 버튼 기능을 수행한다.
- Info_Button: 선택한 사용자의 정보를 확인하기 위한 버튼이다.
- Ban_Button: 선택한 사용자의 이용을 정지시키기 위한 버튼이다.
- Logout_Button: 로그인을 한 사용자가 로그아웃을 하기 위한 버튼이다.

3.10.2. Methods

- Show_User_manage_Page(): User Manage Page로 리다이렉션한다.
- Logout(): 사용자가 Logout_Button을 눌렀을 때 로그아웃을 실행한다.
- Ban(): Ban_Button을 눌렀을 때 관리자로부터 기간을 입력 받는다. 선택한 사용자의 이용을 관리자가 입력한 기간동안 정지시키는 요청을 전송한다.
- Info(): 선택한 사용자의 정보를 열람하기 위해 백엔드 서버로 요청을 전송한다.
- Show_Landing_Page(): Landing Page로 리다이렉션한다.

3.10.3. State Diagram

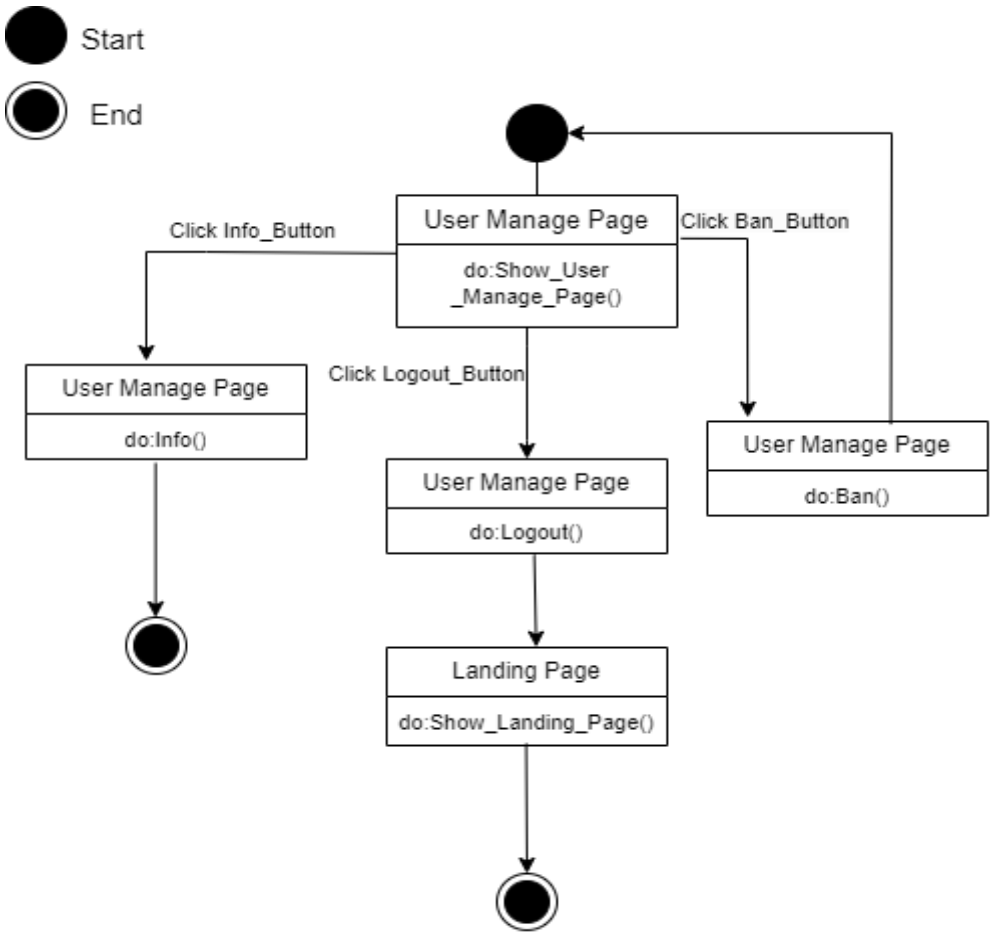

Figure 15. State Diagram of User Manage Page

3.11. Add/Edit Problem Page

관리자가 Problem을 추가하거나 수정할 때 접근하는 페이지이다. 문제를 추가할 때는 문제 내용 입력창, 선택지 입력창, 문제 정답 입력창이 제공된다. 문제를 수정할 때는 문제를 추가할 때와 동일한 형식의 창이 기존 내용으로 채워진 채로 제공되며, 관리자는 기존 내용을 수정하여 새로운 내용을 저장 할 수 있다.

3.11.1. Subcomponents

- Logo_Button: 본 시스템의 로고를 나타내는 Security Edu이다. 사용자가 인증된 상태이므로, main page로 이동하는 버튼 기능을 수행한다.
- Problem_Input: 제목, 내용, 추천질문 목록, 정답 등 문제를 구성하는 속성들에 대한 form이다.
- Add/Edit_Button: 작성된 form을 기반으로 기존 문제를 수정하거나, 새로운 문제를 추가하는 작업을 실행하는 버튼이다.
- Logout_Button: 로그인을 한 사용자가 로그아웃을 하기 위한 버튼이다.

3.11.2. Methods

- Show_Problem_Page(): 본 페이지를 띄운다.
- Submit(): 사용자가 작성한 form을 DB에 반영하기 위해 백엔드 서버에 요청을 보낸다.
- Show_Main_Page(): Main Page로 리다이렉션한다.
- Logout(): 사용자가 Logout_Button을 눌렀을 때 로그아웃을 실행한다.
- Show_Landing_Page(): Landing Page로 리다이렉션한다.

3.11.3. State Diagram

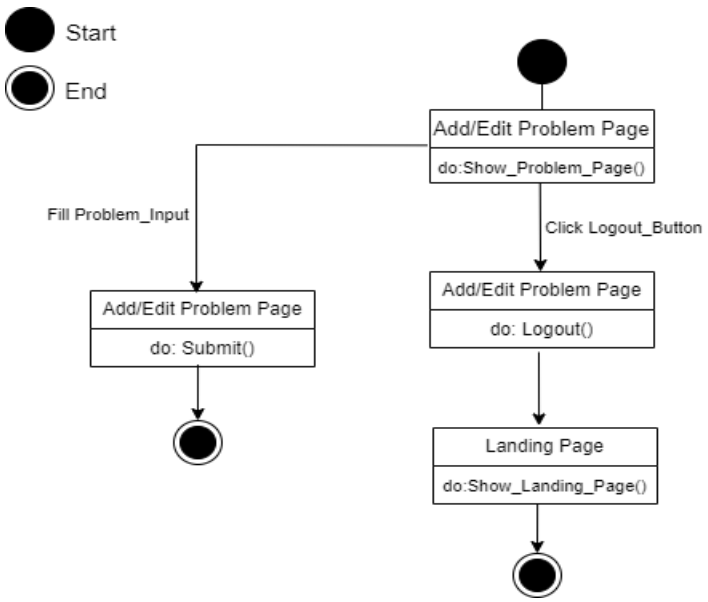

Figure 16. State Diagram of Add/Edit Problem Page

4. Backend System Architecture

4.1. Purpose

본 장의 목표는 백엔드 서버의 세부 구조를 명확히 하기 위함이다. 백엔드 서버의 각 컴포넌트와 데이터베이스 간의 관계를 시각화하여 백엔드 서버의 전반적인 구조를 파악할 수 있다. 각 컴포넌트의 세부 사항을 파악하기 위해 각 컴포넌트를 구성하는 클래스와 동작 흐름을 Class Diagram과 Sequence Diagram을 통해 나타낸다.

4.2. Overall Architecture

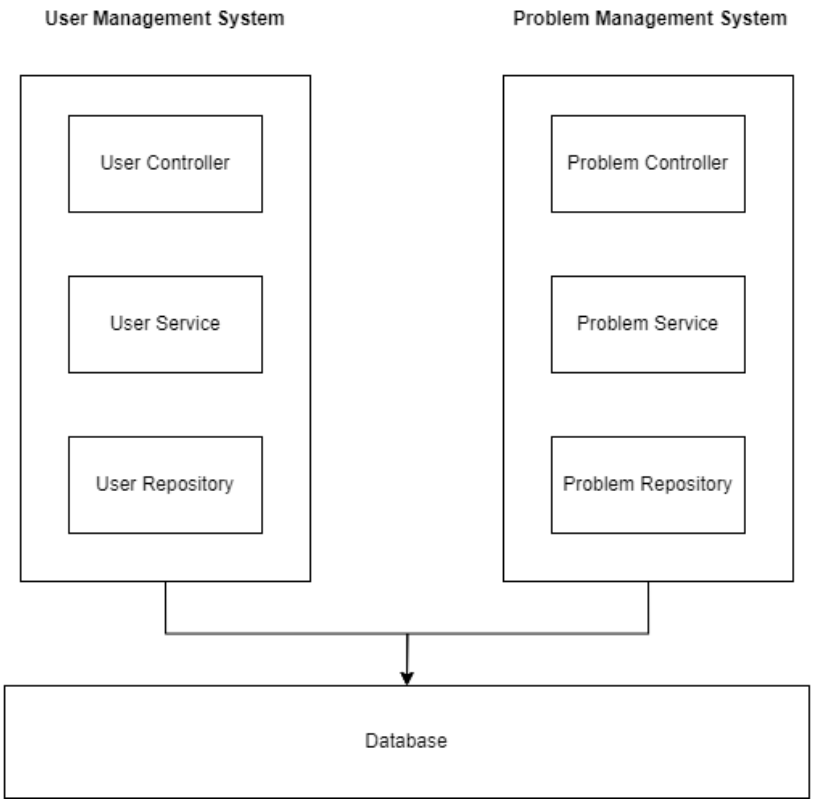

Figure 17. Backend Architecture

4.3. Subsystems

4.3.1. User Management System

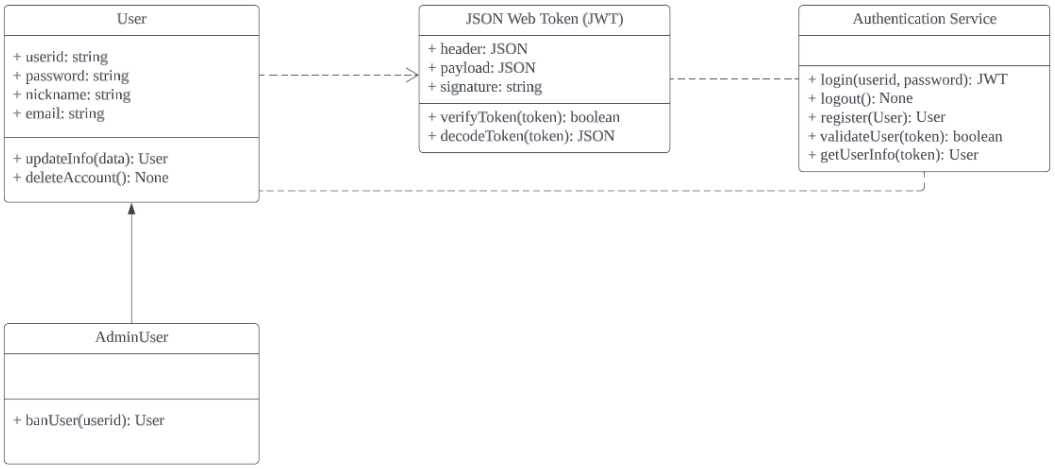

Figure 18. Class Diagram of User Management System

본 시스템에 로그인하는 객체는 User로 관리되며, 회원가입, 로그인, 로그아웃 등의 기능을 수행할 수 있다. 로그인 시, 사용자가 userid와 password를 서버에 보내면, authentication service는 사용자 인증을 처리한다. Authentication service는 제공된 정보를 사용해 사용자를 확인 후, 유효한 사용자인 경우에 JWT token을 헤더에 설정하고 cookie를 생성하여 반환하게 되며, 이후 클라이언트가 서버에 요청을 보낼 때마다 해당 JWT token을 헤더에 포함하여 전송한다. 로그아웃 시에는 cookie를 삭제하여 클라이언트에서의 token을 만료시킨다.

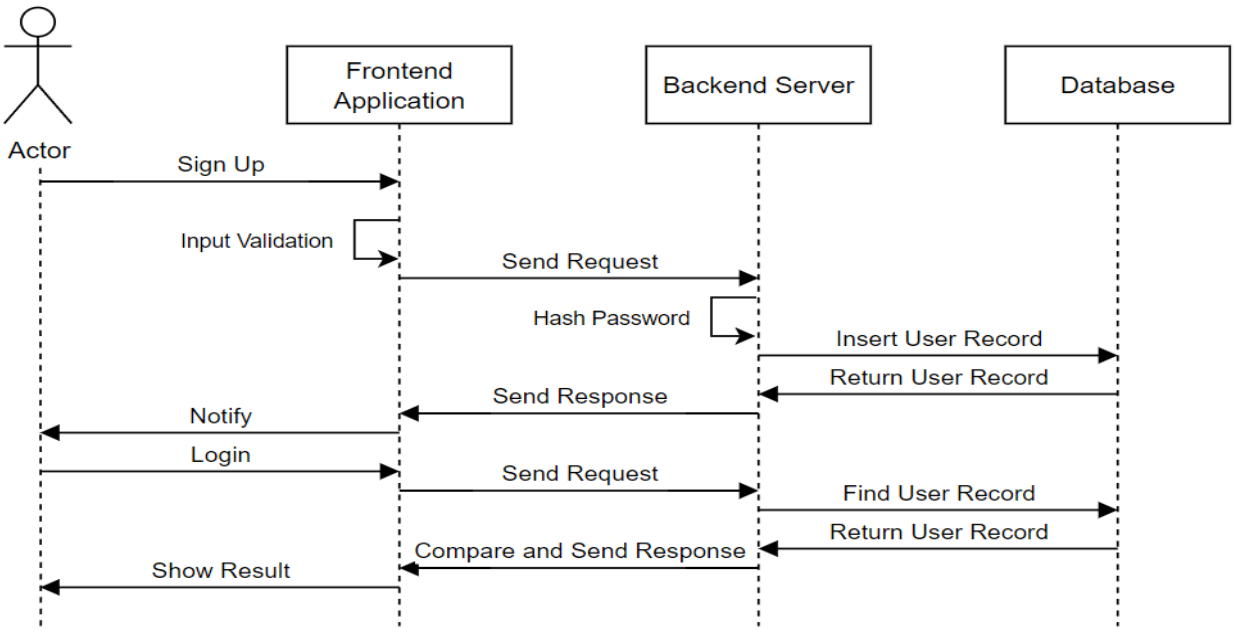

Figure 19. Sequence Diagram of User Management System

4.3.2. Problem Management System

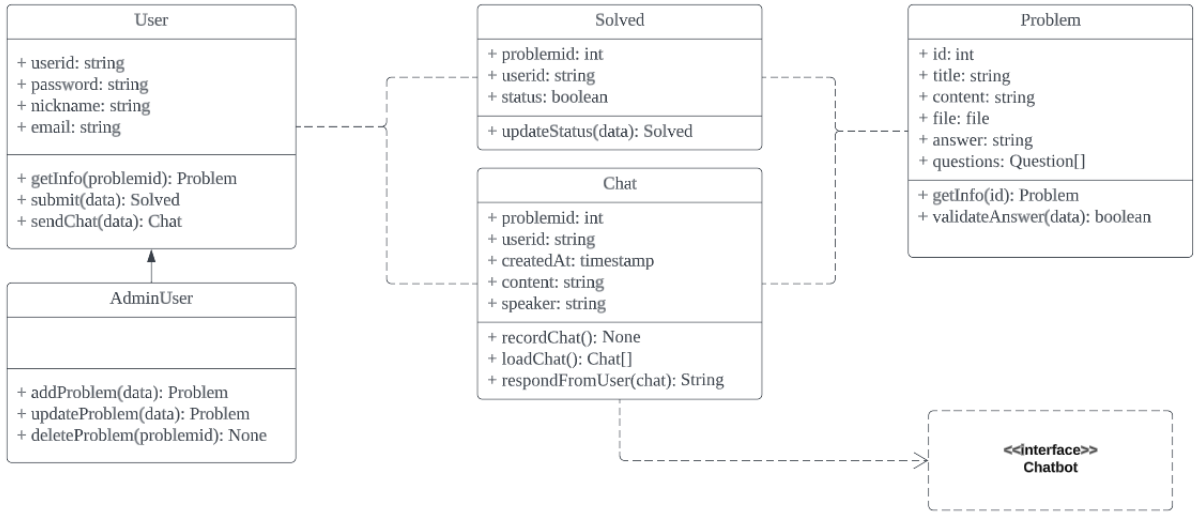

Figure 20. Class Diagram of Problem Management System

문제 관리 시스템에서는 사용자의 입장에서 문제를 어떻게 관리하고 상호작용 하는 지를 다루게 된다. 일반 사용자는 문제를 해결하기 위해 answer 속성을 제외한 문제 정보에 접근할 수 있다. 또한, 사용자가 도출한 문제의 답을 제출할 때, 실제 문제의 정답과 비교하고 그 결과가 Solved record로 기록된다. 사용자는 본 서비스에서 제공하는 AI chatbot과 질의응답을 할 수 있으며, 이에 따른 기록은 문제별로 저장되어 풀이 도중 세션이 종료되어도 추후 이어서 문제를 해결할 수 있다. 개별 문제의 생성, 갱신, 삭제 작업은 오직 관리자만이 수행할 수 있다.

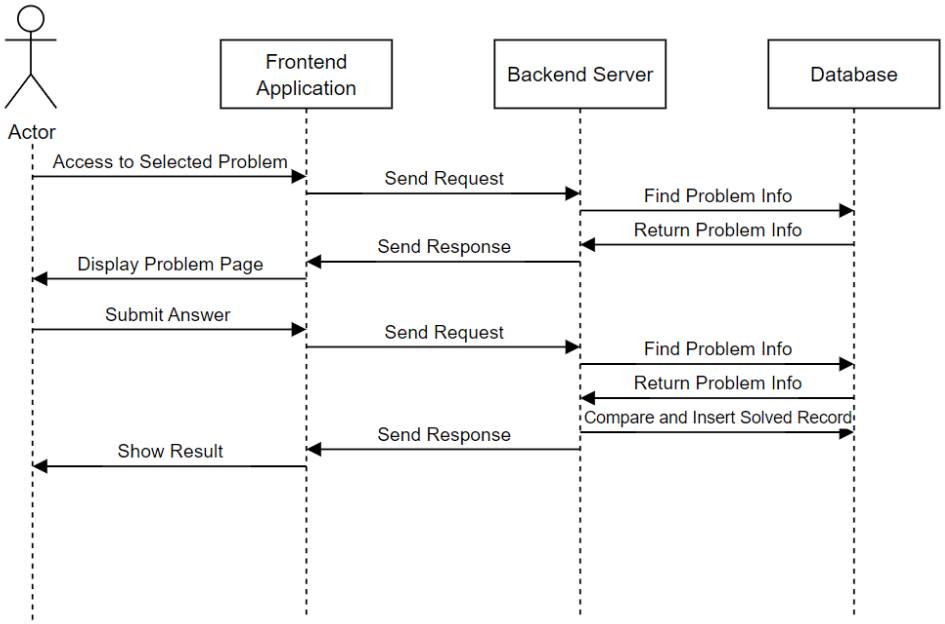

Figure 21. Problem Management System Sequence Diagram

5. Protocol Design

본 장에서는 사용자와의 상호작용 및 인터페이스 제공을 담당하는 프론트엔드 시스템과 전체 서비스의 원활한 작동을 위해 데이터를 관리하는 백엔드 시스템이 어떠한 규약에 의거하여 설계되었는지에 대한 내용을 기술한다. 또한, 두 시스템의 상호작용을 위한 API를 기술한다.

5.1. HTTP

Hypertext Transfer Protocol (HTTP)는 클라이언트와 서버 간 하이퍼텍스트 문서를 전달하는 데 사용되는 프로토콜이다. HTTP는 클라이언트가 요청을 보내면, 서버는 요청된 리소스나 오류가 포함된 응답을 보내는 요청-응답 모델을 따른다. 본 서비스의 백엔드 시스템은 포트 3000 또는 임의의 지정된 포트에서 프론트엔드 시스템과의 TCP/IP 연결을 상정하고 있으며, 프론트엔드 시스템은 포트 3001 또는 임의의 지정된 포트에서 백엔드 시스템과 연결을 설정할 수 있다. 사용자는 웹 브라우저를 통해 HTTP의 기본 포트 번호 80이나 HTTPS의 기본 포트 번호 443에서 백엔드 시스템과의 통신을 수행함을 상정한다. HTTP는 stateless protocol이기 때문에, 각 요청-응답은 독립적으로 수행되며 서버는 클라이언트의 상태를 유지하지 않는다.

5.2. REST API

Representational State Transfer (REST)는 웹 애플리케이션을 만드는 데 특화된 클라이언트-서버 간 효율적인 데이터 통신을 위한 API 설계 아키텍처이다. Application Programming Interface (API)는 두 소프트웨어 간의 상호 작용을 위한 인터페이스를 의미하며, 본 문서에서의 API는 클라이언트와 서버 간 통신을 위해 정의된 일련의 규칙을 의미한다.

본 시스템의 API는 REST의 설계 규칙을 준수하여 작성할 것을 상정한다. 단, 시스템 설계 시 고려되어야 하는 사항의 우선순위에 따라 규칙을 임의로 변경하여 적용할 수 있다. 본 시스템의 API에 대한 설계는 5.5.에 명시되어 있다.

5.3. JSON

Javascript Object Notation (JSON)은 구조화된 데이터를 표현하기 위한 텍스트 기반의 표준 형식이다. JSON은 key-value 쌍으로 구성되어 있으며, 간결하고 쉬운 문법을 갖고 있기 때문에 웹 애플리케이션에서 API 개발에 주로 사용된다. 본 시스템의 API 또한 JSON 형식의 데이터를 사용한다고 상정한다.

5.4. JWT

Json Web Token (JWT)는 당사자 간에 정보를 안전하게 전송하기 위한 인터넷 표준이다. 검증 및 인증을 위한 모든 정보가 토큰에 포함되어 있기 때문에, 서버는 상태를 저장하고 관리하기 위한 별도의 리소스를 할당할 필요 없이 애플리케이션 및 API의 보안성을 높일 수 있다.

5.5. JSX

Javascript Syntax Extension (JSX)은 React에서 컴포넌트 기반의 UI를 구축하기 위한 확장 문법이다. JSX은 HTML 마크업과 유사한 문법이지만 실제로는 Javascript의 동작을 전제하므로, 유저 인터페이스 설계 시 재사용성을 높일 수 있다. 본 시스템은 JSX와 Typescript가 결합된 TSX 기반의 유저 인터페이스 구축을 상정하여, single page application 기반의 UI를 구축하는 것을 목표로 한다.

5.6. API documentation

Table 1. User Login API

이름	사용자 로그인		
Component	Authentication	Content Type	application/json
Method	POST	URL	auth/login
Request	User ID, Password		
Response	Token		

Table 2. User Logout API

이름	사용자 로그아웃		
Component	Authentication	Content Type	application/json
Method	POST	URL	auth/logout
Request	Token		
Response	Message		

Table 3. User Signup API

이름	사용자 회원가입		
Component	Authentication	Content Type	application/json
Method	POST	URL	auth/signup
Request	User Id, User password, Nickname, Email		
Response	Message		

Table 4. Reissue Access Token API

이름	**Access Token** 재발급		
Component	Authentication	Content Type	application/json
Method	POST	URL	auth/refresh
Request	-		
Response	Access Token		

Table 5. List Problems API

이름	모든 문제 목록 조회		
Component	Problem	Content Type	application/json
Method	GET	URL	problems/
Request	-		
Response	[(Problem ID, Problem Title), …]		

Table 6. Create Problem API

이름	신규 문제 생성		
Component	Problem	Content Type	application/json
Method	POST	URL	problems/
Request	Admin Token, Problem Title, Problem Content, Problem Answer		
Response	Created Problem Data		

Table 7. Retrieve Problem Detail API

이름	문제 세부 정보 조회		
Component	Problem	Content Type	application/json
Method	GET	URL	problems/{problem_id}
Request	-		
Response	Problem ID, Problem Title, Problem Content, Problem File		

Table 8. Update Problem Detail API

이름	문제 세부 정보 갱신		
Component	Problem	Content Type	application/json
Method	PATCH	URL	problems/{problem_id}
Request	Admin Token, (Problem Title, Problem Content, Problem Answer, Problem isHidden)		
Response	Problem ID, Problem Title, Problem Content, Problem File		

Table 9. Delete Problem API

이름	문제 삭제		
Component	Problem	Content Type	application/json
Method	DELETE	URL	problems/{problem_id}
Request	Admin Token		
Response	Access Token		

Table 10. List Questions of Problem API

이름	문제의 질문 목록 조회		
Component	Problem	Content Type	application/json
Method	GET	URL	problems/{problem_id}/questions
Request	-		
Response	Access Token		

Table 11. Update Question of Problem API

이름	문제의 질문 갱신		
Component	Problem	Content Type	application/json
Method	PATCH	URL	problems/{problem_id}/questions/{question_id}
Request	Admin Token, (Question Content, Question Answer)		
Response	Updated Question		

Table 12. Delete Question of Problem API

이름	문제의 질문 삭제		
Component	Problem	Content Type	application/json
Method	DELETE	URL	problems/{problem_id}/questions/{question_id}
Request	Admin Token		
Response	Deleted Question		

Table 13. Upload File of Problem API

이름	문제의 파일 추가		
Component	Problem	Content Type	application/json
Method	POST	URL	problems/{problem_id}/file
Request	Admin Token, File Data		
Response	Updated Problem		

Table 14. Delete File of Problem API

이름	문제의 파일 삭제		
Component	Problem	Content Type	application/json
Method	DELETE	URL	problems/{problem_id}/file
Request	Admin Token		
Response	Deleted Question		

Table 15. Retrieve User Detail API

이름	사용자 세부 정보 조회		
Component	User	Content Type	application/json
Method	GET	URL	users/{user_id}
Request	-		
Response	User Data except password		

Table 16. List User Leaderboard API

이름	사용자 랭킹 조회		
Component	User	Content Type	application/json
Method	GET	URL	users/leaderboard
Request	-		
Response	[{User ID, Number of Solved Problems}...]		

Table 17. Update User Detail API

이름	사용자 세부 정보 갱신		
Component	User	Content Type	application/json
Method	PATCH	URL	users/{user_id}
Request	Authentication Token, User Nickname, User Email, User Password		
Response	User Data except Password		

Table 18. Ban User API

이름	사용자 정지		
Component	User	Content Type	application/json
Method	PATCH	URL	users/{user_id}/ban
Request	Admin Token, Ban/Unban, Unbanned Date		
Response	User Data except Password		

Table 19. Promote/Demote User API

이름	관리자 권한 부여 및 회수		
Component	User	Content Type	application/json
Method	PATCH	URL	users/{user_id}/admin
Request	Admin Token, isAdmin (true/false)		
Response	User Data except Password		

Table 20. List Solved Problems of User API

이름	사용자의 해결 문제 목록 조회		
Component	User	Content Type	application/json
Method	GET	URL	users/{user_id}/solved
Request	-		
Response	[{User ID, Number of Solved Problems}...]		

Table 21. Submit Problem API

이름	사용자의 문제 제출		
Component	User	Content Type	application/json
Method	POST	URL	users/{user_id}/solved/{problem_id}
Request	Submitted Value		
Response	Solved Status		

Table 22. List Chatlog of Problem API

이름	사용자의 해당 문제에 대한 채팅 목록 조회		
Component	User	Content Type	application/json
Method	GET	URL	users/{user_id}/solved/{problem_id}/chat
Request	Authorization Token		
Response	[{Speaker, Created Time, Content}...]		

Table 23. Add Chatlog of Problem API

이름	사용자의 해당 문제에 대한 채팅 추가		
Component	User	Content Type	application/json
Method	POST	URL	users/{user_id}/solved/{problem_id}/chat
Request	Speaker, Content		
Response	[{User ID, Number of Solved Problems}...]		

6. Database Design

6.1. Entity-Relationship Diagram

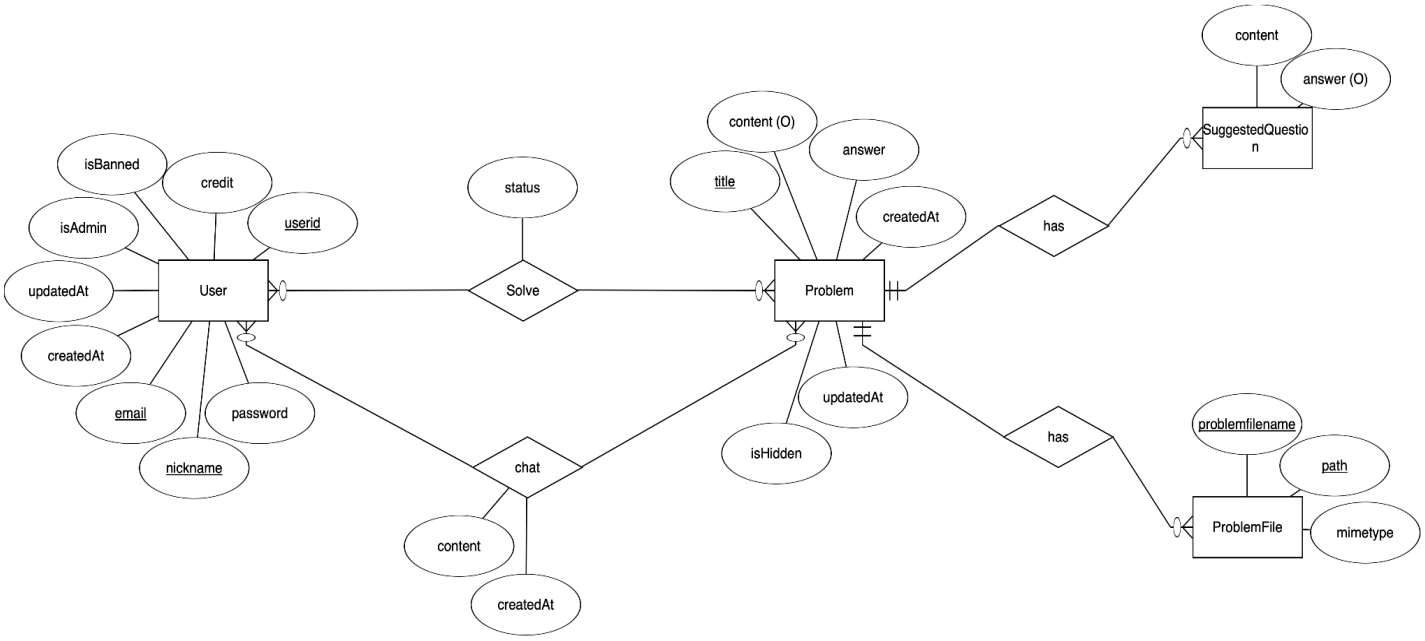

Figure 22. Entity-Relation Diagram

위의 'Entity-Relationship Diagram'(이하 'ERD')는 본 프로젝트에서 활용하는 데이터와 그 관계에 대해 표현한 것이다. 관련 데이터 속성이 많은 엔티티는 User와 Problem이다. User는 여러 개의 Problem과 Solved 관계를 맺을 수 있으며, AI와 Chat을 수행할 수 있고, Problem은 다양한 User가 풀어볼 수 있어 각각 m:n의 관계에 있다. Problem은 SuggestedQuestion과 ProblemFile을 가지고 있는데, 하나의 문제에는 3개의 추천질문이 있고, 하나의 문제에는 다양한 이미지 파일이 필요할 수 있다. 그 관계는 1:m으로 표현되어 있다. 추천질문에 대한 답은 엔티티의 속성으로 활용되지만, api를 통해 데이터가 요청되었을 때 바로 얻을 수도 있기에 optional한 속성이다.

6.2. Relational Schema

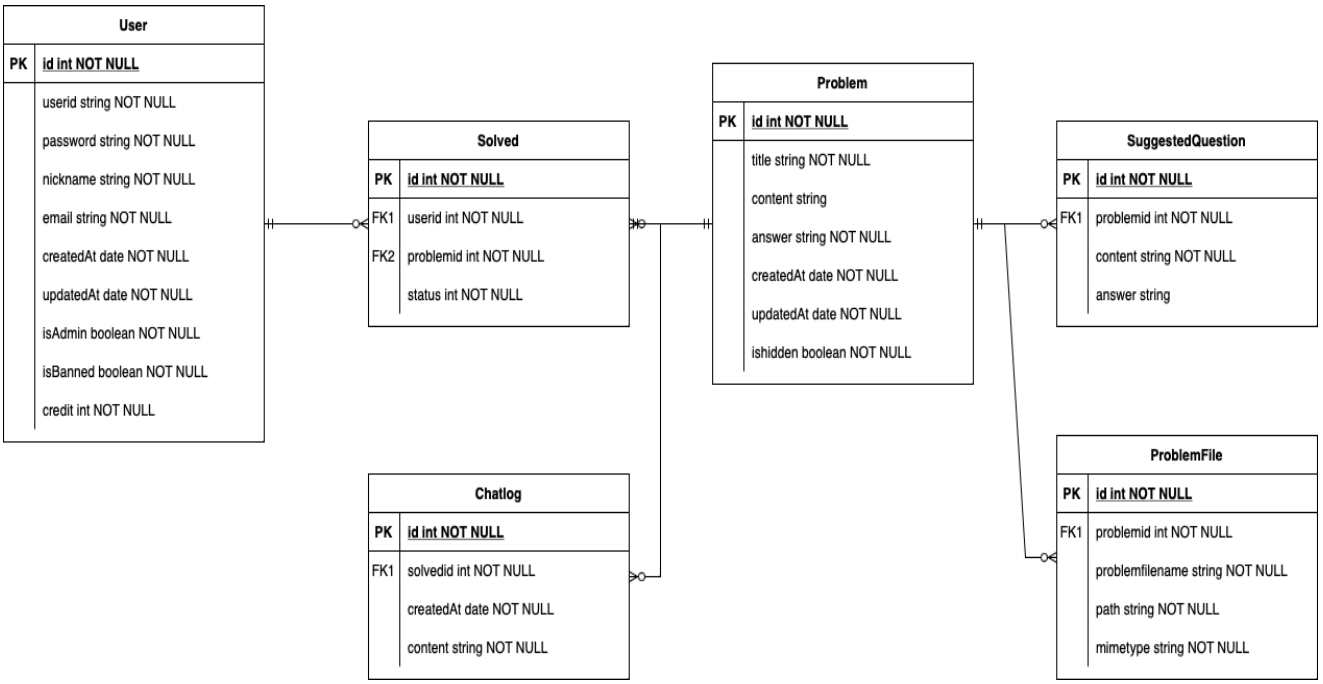

Figure 23. Relational Schema

위 Relational Schema를 통해 6.1.에서 보인 ERD의 엔티티와 관계를 표로 구성하였을 때의 그 관계에 대해 표현하였다. 특히 m:n 관계의 solve와 chat을 구현하기 위해 solved라는 표를 하나 더 생성해 각 user와 problem에 1:m 관계를 만들고, solved에 1:m으로 chatlog을 관계지은 것이 큰 특징이다. 생략되었던 각 표의 id 역시 PK로서 표현되어 있다.

7. Testing Plan

7.1. Objective

본 장에서는 요구사항 명세서에서 기술한 시나리오를 바탕으로 통합 시스템이 동작하는 지 확인한다. 유닛 테스트, 통합 테스트 과정을 거쳐 시스템 출시 전에 발생할 수 있는 잠재적인 오류 혹은 결함을 찾아내고자 한다. 이를 위해 세가지 관점으로 나누어 테스트 과정을 설계하며, 세부 테스트 케이스를 설명한다.

7.2. Testing Policy

7.2.1. Development Testing

본 시스템 개발 과정 중에 발생할 수 있는 리스크와 비용을 줄이는 것을 목표로 하며, 먼저 시스템을 구성하는 서브 시스템 혹은 컴포넌트 단위의 독립적인 유닛 테스트를 진행한다. 유닛 테스트는

서비스하고자 하는 기능이 예상대로 동작하는 지를 중점적으로 확인한다. 더불어 구현 컴포넌트의 유지보수성을 높이고 잠재적인 결함을 발견하기 위해 상호 간의 코드 리뷰를 진행한다. 독립 테스트 진행 후, 각 컴포넌트들을 순차적으로 통합하여 통합 테스트를 진행한다. 통합 테스트는 기능적인 항목 뿐만 아니라 비기능적인 성능, 보안, 안정성 등을 검증해야 한다. 이를 위해 정적 코드 분석, 데이터 흐름 분석 등을 수행할 수 있다.

7.2.2. Release Testing

본 시스템의 새로운 버전 출시에 따른 배포 과정이 정상적으로 수행될 수 있는 지를 검증한다. 시스템이 최신 버전의 온전한 상태로 배포되어 작동에 어떠한 하자가 없어야 한다. 일반적으로 목표로 하는 기능을 포함한 알파 테스트를 거쳐, 그 과정에서 발견한 피드백 및 결함을 반영하여 베타 테스트를 진행한다. 알파 테스트의 경우 개발 진행 조직 내부에서 진행하며, 베타 테스트의 경우 제한된 사용자에게 공개하여 진행한다.

7.2.3. User Testing

본 시스템의 예상 사용자가 사용하는 과정을 시험함으로써, 본 시스템이 모든 요구사항 명세를 충족하는 지 검증한다. 이를 위해 시스템 사용 시나리오를 작성하고, 이 시나리오에서 AI-Tutoring CTF 시스템이 모든 Use Case를 충족할 수 있는지를 검증한다. 검증 결과를 바탕으로 AI-Tutoring CTF 시스템의 베타 버전의 유용성을 평가하고, 최종 결과물로서 실제 환경에 Deployment 될 것인지를 결정한다.

7.3. Test Case

7.3.1. Sign Up

Table 24. User Signup Test

Element	Description
Summary	본 시스템에 접근하기 위한 계정을 생성한다.
Pre-Condition	● 사용자는 로그인을 하지 않은 anonymous 상태이다. ● 사용자의 인터넷 환경이 원활하다. ● 가입한 적이 없는 사용자이다.
Test Steps	1. 랜딩 페이지에서 ‘Sign Up’ 버튼을 클릭해 나타나는 회원가입 페이지로 이동 2. 요구하는 정보를 모두 기입 3. ‘Create Account’ 버튼을 클릭하면, 기입한 정보로 회원정보가 생성.
Expected Results	생성된 계정으로 로그인 되면서 토큰 정보를 저장하고, 메인 페이지로 이동.

7.3.2. Login

Table 25. User Login Test

Element	Description
Summary	본 시스템에 접근하기 위해서는 로그인을 수행해야 한다.
Pre-Condition	● 사용자는 로그인을 하지 않은 anonymous 상태이다. ● 사용자의 인터넷 환경이 원활하다. ● 사용자는 본 서비스의 회원이다.
Test Steps	1. 랜딩 페이지에서 아이디와 비밀번호를 기입 2. '로그인' 버튼을 클릭해 동일 회원 정보 검토 3. 서버에서 토큰 생성 후 토큰 정보 응답
Expected Results	응답받은 토큰 정보를 저장하고, 메인 페이지로 이동.

7.3.3. Block Unauthenticated Access

Table 26. Block Unauthenticated Access Test

Element	Description
Summary	본 시스템에 접근할 때 권한이 없는 사용자나 차단된 사용자는 접속을 차단한다.
Pre-Condition	● 사용자의 인터넷 환경이 원활하다.
Test Steps	1. 메인 페이지 요청 시 헤더에 토큰정보 유무 확인 2. 만약 토큰정보가 없다면, 메인 페이지 접속 차단. 3. 토큰이 나타내는 유저가 차단된 유저라면, 접속 차단.
Expected Results	경고 메시지와 함께 랜딩 페이지로 이동.

7.3.4. Main Page

Table 27. Main Page Test

Element	Description
Summary	메인 페이지에 나타나는 문제목록과 리더보드가 정상적으로 표시된다.
Pre-Condition	● 사용자는 로그인 상태이다. ● 사용자의 인터넷 환경이 원활하다.

Test Steps	1. 랜딩 페이지에서 로그인 작업 수행 및 메인페이지 이동 2. 메인페이지의 문제목록 및 리더보드를 확인.
Expected Results	정상적인 문제목록 및 리더보드 확인

7.3.5. Access to Problem

Table 28. Access to Problem Test

Element	Description
Summary	메인 페이지의 문제 목록에서 원하는 문제를 선택한다.
Pre-Condition	● 사용자는 로그인상태이다. ● 사용자의 인터넷 환경이 원활하다.
Test Steps	1. 랜딩페이지에서 로그인 작업 수행 2. 메인페이지의 문제목록에서 문제 선택
Expected Results	선택한 문제풀이 페이지로 이동한다.

7.3.6. Problem Page

Table 29. Problem Page Test

Element	Description
Summary	문제 풀이 화면에서 문제에 대한 설명과 챗봇과 했던 이전 대화를 볼 수 있다.
Pre-Condition	● 사용자는 로그인 상태이다. ● 사용자의 인터넷 환경이 원활하다.
Test Steps	1. 메인 페이지에서 문제를 선택하여 문제 페이지로 이동 2. 문제 페이지에서 문제 풀이 화면의 구성 요소가 정상적으로 배치되었는지 확인
Expected Results	문제 풀이 화면에서 정상적으로 문제에 대한 설명과 사용자의 챗봇의 대화 내역을 확인할 수 있다.

7.3.7. AI-Tutor (Chatbot)

Table 30. AI-Tutor (Chatbot) Test

Element	Description
Summary	인터페이스에서 챗봇에 보낼 질문을 작성하고 챗봇으로부터 얻은 대답을 확인할 수 있다.
Pre-Condition	● 사용자는 로그인 상태이다. ● 사용자의 인터넷 환경이 원활하다.
Test Steps	1. 챗봇에게 할 질문 작성 2. 작성된 질문을 제출 3. 반환된 대답을 대화창에 표시
Expected Results	대화창에 챗봇의 대답이 표시된다.

7.3.8. Suggested Question

Table 31. Suggested Question Test

Element	Description
Summary	사용자가 추천질문을 선택하면 그 대답을 확인할 수 있다.
Pre-Condition	● 사용자는 로그인상태이다. ● 사용자의 인터넷 환경이 원활하다.
Test Steps	1. 사용자가 추천질문 중 하나를 선택 2. 추천질문에 대응되는 대답을 DB에서 탐색 3. 유효한 대답이 DB에 있을 경우, 대화창에 표시 4. 대답이 null 값일 경우, openai API로 요청 전송 5. 대답을 대화창에 표시
Expected Results	대화창에 추천질문에 대한 대답이 표시된다.

7.3.9. Clear Chatlog

Table 32. Clear Chatlog Test

Element	Description
Summary	챗봇과 했던 기존 대화를 대화창에서 삭제할 수 있다.
Pre-Condition	● 사용자는 로그인상태이다. ● 사용자의 인터넷 환경이 원활하다.

Test Steps	1. 사용자가 대화 초기화 버튼을 누름. 2. DB에서 현재 문제에 대한 사용자의 질문 내용을 삭제 3. 페이지를 재시작
Expected Results	대화창에서 기존의 대화 내용이 삭제된다.

7.3.10. Problem Submission

Table 33. Problem Submission Test

Element	Description
Summary	정답을 제출한 후 그 결과를 확인할 수 있다.
Pre-Condition	● 사용자는 로그인 상태이다. ● 사용자의 인터넷 환경이 원활하다.
Test Steps	1. 사용자 정답 제출 버튼을 누름. 2. 제출된 정답과 DB에 저장된 정답이 동일한지 비교 3. 결과를 화면에 표시
Expected Results	정답인지 아닌지 결과가 화면에 표시된다.

7.3.11. List of Solved Problems

Table 34. List of Solved Problems Test

Element	Description
Summary	사용자가 풀어본 문제의 목록을 확인할 수 있다.
Pre-Condition	● 사용자는 로그인 상태이다. ● 사용자의 인터넷 환경이 원활하다.
Test Steps	1. 사용자가 MyPage의 풀어본 문제 탭으로 이동 2. 풀어본 문제 목록이 화면에 표시
Expected Results	사용자가 풀어본 문제 목록이 화면에 표시된다.

7.3.12. Update User Information

Table 35. Update User Information Test

Element	Description
Summary	사용자가 자신의 정보를 수정할 수 있다.
Pre-Condition	● 사용자는 로그인 상태이다. ● 사용자의 인터넷 환경이 원활하다.
Test Steps	1. MyPage에서 사용자가 정보 수정란에서 새로운 정보를 입력 2. 입력된 정보를 바탕으로 DB를 갱신
Expected Results	사용자의 정보가 수정된다.

7.3.13. Block Admin Page Access

Table 36. Block Admin Page Access Test

Element	Description
Summary	일반 사용자는 관리자 페이지에 접근할 수 없다.
Pre-Condition	● 사용자의 인터넷 환경이 원활하다. ● 사용자는 본 서비스의 회원이다. ● 사용자는 관리자 권한이 없다.
Test Steps	1. 일반 사용자가 관리자 페이지에 접근 시도 2. 접근 거부 페이지로 리다이렉트
Expected Results	사용자가 접근 거부 페이지로 이동한다.

7.3.14. Admin Page Access

Table 37. Admin Page Access Test

Element	Description
Summary	관리자 권한을 소유한 사용자가 관리자 페이지를 접근할 수 있다.
Pre-Condition	● 사용자는 로그인 상태이다. ● 사용자의 인터넷 환경이 원활하다. ● 사용자는 관리자 권한을 갖고 있다.

Test Steps	1. 관리자 권한을 소유한 유저가 메인페이지로 접근 2. 메인페이지에서 'Admin Page' 버튼을 눌러 관리자 페이지로 이동
Expected Results	사용자가 정상적으로 관리자 페이지로 이동된다.

7.3.15. Create Problem

Table 38. Create Problem Test

Element	Description
Summary	관리자 페이지의 문제 관리 페이지에서 문제를 추가할 수 있다.
Pre-Condition	● 사용자는 로그인 상태이다. ● 사용자의 인터넷 환경이 원활하다. ● 사용자는 관리자 권한을 갖고 있다..
Test Steps	1. 관리자 권한을 소유한 사용자가 관리자 페이지로 이동 2. 사용자가 관리자 페이지에서 'Manage Problem' 버튼을 클릭하여 문제 관리 페이지로 이동 3. 문제 관리 페이지에서 'Add new problem' 버튼을 클릭하여 문제 추가 page로 이동. 4. 문제 추가 page에서 추가할 문제에 대한 정보를 입력 후 문제 추가
Expected Results	추가된 문제가 정상적으로 메인 페이지의 문제 선택 화면에 추가되고 문제에 접근할 수 있다.

7.3.16. Update Problem

Table 39. Update Problem Test

Element	Description
Summary	관리자 페이지의 문제 관리 페이지에서 문제를 수정할 수 있다.
Pre-Condition	● 사용자는 로그인 상태이다. ● 사용자의 인터넷 환경이 원활하다. ● 사용자는 관리자 권한을 갖고 있다..
Test Steps	1. 관리자 권한을 소유한 사용자가 관리자 페이지로 이동 2. 사용자가 관리자 페이지에서 'Manage Problem' 버튼을 클릭하여 문제 관리 페이지로 이동 3. 문제 관리 페이지에서 이미 존재하는 문제를 클릭하여 문제 수정 page로 이동. 4. 문제 수정 page에서 수정할 문제에 대한 정보를 입력 후 문제 갱신
Expected Results	갱신된 문제가 정상적으로 반영된다.

7.3.17. Manage User

Table 40. Manage User Test

Element	Description
Summary	관리자가 관리자 페이지의 사용자 관리 페이지에서 사용자 정보를 수정할 수 있다.
Pre-Condition	• 사용자는 로그인 상태이다. • 사용자의 인터넷 환경이 원활하다. • 사용자는 관리자 권한을 갖고 있다..
Test Steps	1. 관리자 권한을 소유한 사용자가 관리자 페이지로 이동 2. 사용자가 관리자 페이지에서 'Manage User' 버튼을 클릭하여 사용자 관리 페이지로 이동 3. 사용자에게 관리자 권한을 주거나 해당 기간동안 접근을 차단하는 등 사용자 정보를 수정
Expected Results	갱신된 정보가 정상적으로 반영된다.

8. Development Design

8.1. Objective

본 장에서는 개발을 함에 있어서 사용할 개발 환경 및 개발하는 과정에 있어서 지켜야할 제약 사항들에 대해 서술한다.

8.2. Development Environment

8.2.1. Git

Figure 24. Git Logo

Git은 분산 버전 관리 시스템(Distributed Version Control System)으로, 프로그래밍 코드의 변경 이력을 관리하는 도구이다. 이를 통해 여러 사람이 동시에 작업하고, 코드 변경 사항을 추적하고, 변경

내용을 병합할 수 있다. Git은 변경 이력을 스냅샷으로 관리하여 파일의 상태를 저장하고, 필요한 경우 이전 상태로 돌아갈 수 있게 해준다. 또한, 다양한 원격 저장소를 통해 코드를 공유하고 협업할 수 있다. 본 시스템을 개발함에 있어서도 각 컴포넌트 개발을 브랜치로 나눠서 진행 할 수 있고, 코드 리뷰 및 통합에 용이하므로, Git을 활용한다.

8.2.2. React

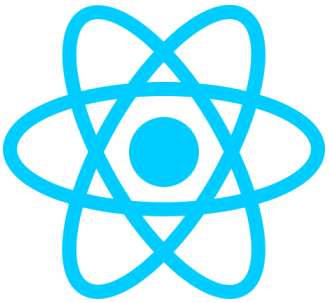

Figure 25. React Logo

React는 웹 애플리케이션 개발에서 사용되는 JavaScript 라이브러리로, 컴포넌트 기반 접근 방식과 가상 DOM을 활용하여 개발자들에게 다양한 이점을 제공한다. 이를 통해 개발자들은 재사용 가능한 UI 요소를 구축하고, 성능을 최적화하며, 데이터 흐름을 예측 가능하게 유지할 수 있다. 또한, React는 다양한 커뮤니티와 생태계를 가지고 있어 개발자들은 지식과 리소스를 공유하고 협업을 강화할 수 있다. React를 사용함으로써 개발자들은 유연하고 효율적인 웹 애플리케이션을 개발할 수 있다. 따라서 본 시스템을 개발하는 과정에 있어서 React를 사용하여 유지 보수성을 확보하도록 한다.

8.2.3. Redux

Figure 26. Redux Logo

Redux는 JavaScript 애플리케이션의 상태 관리를 위한 예측 가능한 상태 컨테이너로, 중앙 집중화된 상태 관리, 예측 가능한 상태 변화, 시간 여행 디버깅, 확장성과 재사용성, 그리고 생태계와 커뮤니티의 이점을 제공한다. Redux를 활용함으로써 개발자들은 복잡한 애플리케이션에서 상태를 효율적으로 관리하고, 컴포넌트 간의 상태 공유와 통신을 용이하게 할 수 있다. 또한, 불변 객체와 리듀서를 통해 상태 변화를 예측 가능하게 유지하며, 시간 여행 디버깅을 통해 버그를 추적하고 수정할 수 있다. Redux는 확장성과 재사용성을 갖추고 있으며, 다양한 생태계와 커뮤니티를 통해 개발자들은 리덕스 기능을 보완하고 개발 프로세스를 향상시킬 수 있다. 따라서 본 시스템은 Redux를 시스템의 상태 관리 라이브러리로서 활용한다.

8.2.4. Nest.js

Figure 27. Nest.js Logo

Nest.js는 Node.js를 기반으로 한 프레임워크로, 서버 측 애플리케이션의 개발을 위해 사용된다. Nest.js는 Angular의 구조와 문법을 일부 차용하여 구성되어 있어 Angular 개발자들에게 익숙한 환경을 제공한다. 이 프레임워크는 모듈, 미들웨어, 컨트롤러, 서비스 등의 개념을 사용하여 확장 가능하고 모듈화된 애플리케이션을 구축할 수 있도록 도와준다. Nest.js는 TypeScript를 기본 언어로 지원하며, 강력한 의존성 주입(Dependency Injection) 기능과 함께 테스트 가능하고 유지 보수가 용이한 애플리케이션을 개발할 수 있도록 지원한다. 또한, 다양한 데이터베이스, 인증 및 인가 시스템, 웹 소켓과 같은 다양한 기능과 라이브러리를 통합하여 개발자들에게 편리한 환경을 제공한다. 따라서 본 시스템은 백엔드 개발에 Nest.js를 활용하도록 한다.

8.3. Development Constraints

본 시스템은 소프트웨어 요구사항 명세서와 소프트웨어 디자인 명세서에 기술된 내용을 기반으로 설계 및 구현된다. 이외의 세부 사항은 개발진의 기술 능력을 반영하여, 다음 사항을 준수해야 한다.

- 범용적으로 사용되며 검증된 기술 및 서비스를 사용한다.
- 상업용 라이선스가 필요하거나 추가 비용을 지불 해야하는 서비스는 사용을 지양한다.

- 오픈소스 소프트웨어 사용을 지향한다.
- 시스템의 가용성과 확장성을 고려하여 설계 및 구현한다
- 시스템의 개발 및 유지보수 비용을 고려하여 설계 및 구현한다.

9. Supporting Information

9.1. Software Design Specification

소프트웨어 디자인 명세서는 IEEE 권장 사항에 맞추어 작성되었다 (IEEE Standard for Information Technology Systems Design Software Design Descriptions, IEEE-Std-1016). 다만 본 시스템의 설계 사항을 용이하게 파악할 수 있도록 본래의 양식에서 일부 추가되거나 제외된 항목이 있다.

9.2. Document History

Table 41. Document History

Date	Version	Description	Writer
2023.05.14	0.0.1	초본 작성	홍형근 외 5인
2023.05.21	1.0.0	최종본 작성	홍형근 외 5인