Software Requirements Specification

for

LLM 활용
청년정책 추천 플랫폼

SWE3002_41: Prof. Eun Seok Lee

Team 3: 강민규, 기민성, 김정원, 김희수, 박시온, 조우열

SungKyunKwan University

2025.11.09.

Table of Contents

Table of Figures

Table of Tables

1. Introduction

1.1 Purpose

이 문서는 LLM 을 활용하여 사용자의 조건에 맞는 청년정책을 추천하는 시스템 개발에 필요한 요구사항을 명확하게 정의하는 것을 목적으로 한다. 본 플랫폼은 기존의 파편화된 정책 정보 제공 방식과 단순 키워드 검색의 한계를 극복하기 위해, RAG 기술과 LLM 을 활용한다. 사용자의 개인화된 프로필 및 자연어 질문(prompt)을 기반으로, 외부 정책 데이터베이스(DB)의 최신 정보를 참조하여 가장 적합한 청년 정책을 추천하고, 그 추천 사유까지 함께 제공하는 지능형 시스템 구현을 목표로 한다.

이에 본 문서는 해당 플랫폼이 제공해야 할 전반적인 기능적 요구사항(RAG 기반 추천, 정책 상세 검색, 관심 정책 관리 등)과 비기능적 요구사항(성능, 보안, 신뢰성 등), 시스템의 동작 범위, 인터페이스 및 제약 사항을 상세히 기술한다.

본 명세서의 대상 독자는 프로젝트 개발팀(Team 3), UI/UX 디자이너, 플랫폼 평가자 및 기타 시스템 이해관계자들이며, 이들 모두가 프로젝트의 최종 목표와 세부 사항에 대해 공통된 기준을 가지고 성공적으로 프로젝트를 기획, 개발, 테스트 및 평가할 수 있도록 돕는 공식 가이드라인으로 사용된다.

1.2 Project Scope

1.2.1. Product Identification

본 문서에서 정의하는 소프트웨어 시스템은 'LLM 을 활용한 청년정책 추천 플랫폼' 웹사이트이다.

1.2.2. Product Scope (In/Out of Scope)

본 플랫폼은 대한민국의 청년이 자신의 개인적 조건(프로필)에 따라 참여 가능한 청년 정책을 쉽고 빠르게 탐색하고 활용할 수 있도록 돕는 지능형 웹 기반 검색 시스템이다.

주요 수행 작업 (In Scope):

- **RAG 기반 맞춤형 추천:** 사용자의 초기 설문(Survey) 정보와 자연어 프롬프트를 입력 받아, 기존 LLM 의 Knowledge Cut-off 및 Hallucination 문제를 해결한 RAG 기술로 최적화된 정책과 추천 사유를 제공한다.
- **상세 조건 검색:** 키워드, 지역, 취업 상태, 학력 등 다양한 필터를 사용한 상세 검색 기능을 제공한다.
- **정책 정보 제공:** 최신/인기 정책 목록, 정책 상세 정보(설명, 지원 내용, 신청 방법, 신청 링크)를 제공한다.

수행하지 않을 작업 (Out of Scope):

- 본 플랫폼은 정책 정보의 검색과 추천, 가이드에 중점을 둔다. 정책의 실제 신청/접수 기능이나 정부 24 등 외부 신청 사이트와의 직접적인 인증(SSO) 및 연동은 포함하지 않는다.
- 플랫폼 관리자를 위한 정책 데이터의 직접 생성, 수정, 삭제(CMS) 기능은 본 범위에 포함되지 않는다. (정책 데이터는 별도의 js 스크립트를 통해 외부 API 로부터 일괄 수집하는 것을 전제로 한다.)

1.2.3. Objectives and Goals

본 소프트웨어의 궁극적인 목표는 청년 1 인당 정책 수혜율을 높이는 것이다. 이를 달성하기 위한 주요 목적은 다음과 같다:

- **정보 파편화 해결**: 지자체와 기관마다 흩어져 있는 정책 정보를 단일 플랫폼에서 통합 제공한다.
- **접근성 향상**: 복잡한 전문 용어와 절차를 RAG 기반의 쉬운 자연어 검색 및 상세 설명으로 해소한다.
- **맞춤형 안내 제공**: 개인 프로필과 관심사를 기반으로 사용자에게 실질적으로 필요한 정책을 선별하여 추천한다.

이를 통해 사용자는 정보 탐색에 드는 시간을 획기적으로 줄이고, 자신도 몰랐던 혜택을 빠짐없이 누릴 수 있게 되는 이점을 얻을 수 있다.

1.3 Definitions, Acronyms, and Abbreviations

Table 1.1: Table of Definitions, Acronyms, and Abbreviations

약어	설명
LLM	Large Language Model (대규모 언어 모델)
RAG	Retrieval-Augmented Generation (검색-증강 생성)
API	Application Programming Interface
JWT	JSON Web Token (JWT 인증 미들웨어)
CMS	Content Management System (콘텐츠 관리 시스템)
B2G	Business-to-Government (B2G 활용성)

용어	정의
LANGCHAIN	LLM 과 외부 데이터 소스(API, DB 등)를 연결해주는 프레임워크
KNOWLEDGE CUT-OFF	LLM 이 특정 시점까지의 데이터로만 학습되어 최신 정보를 모르는 한계
HALLUCINATION	환각. LLM 이 사실에 근거하지 않은 그럴듯한 거짓 정보를 생성하는 현상
SURVEY	사용자 초기정보(나이, 거주지, 소득, 관심 분야 등)를 수집하는 입력 폼
PROMPT	LLM 에게 작업을 지시하기 위해 사용자가 입력하는 자연어 질문 또는 명령
NODE.JS	백엔드 서버 구축에 사용되는 JavaScript 런타임 환경
REACT	프론트엔드(웹사이트) UI 구축에 사용되는 JavaScript 라이브러리
MYSQL	본 프로젝트에서 사용하는 데이터베이스 관리 시스템(DBMS)
BCRYPT	auth.js 에서 사용자 비밀번호 암호화(해싱)에 사용되는 라이브러리

1.4 References

1. IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications, In IEEEXplore Digital Library
   https://people.eecs.ku.edu/~saiedian/812/Project/Wiegers-Resources/Chapter%2010/Software%20Requirements%20Specification%20Template.docx
2. 2025 Fall 41 class Team 3 깃허브 주소
   https://github.com/skkuse/2025fall_41class_team3
3. 백엔드 소스 코드 (Node.js) (auth.js, mypage.js, policies.js 등)
4. RAG/Search 스크립트 (Python) (recommend.py 등)
5. 정책 데이터 수집 스크립트 (api_save.js)

1.5 Overview

본 문서는 총 3 장으로 구성된다. 1 장(Introduction)에서는 본 문서의 목적, 시스템의 범위, 용어 정의, 참조 문서 등 프로젝트 개요를 기술한다. 2 장(Overall Description)에서는 제품의 전반적인 관점, 주요 기능, 사용자 특성, 제약 사항 및 가정, 의존성 등을 설명한다. 3 장(Specific Requirements)에서는 외부 인터페이스, Use Case 기반의 기능 요구사항, 비기능적 요구사항 및 시스템 아키텍처 등을 상세히 기술한다.

본 명세서에 기술된 모든 요구사항은 Team 3 팀원 전원의 협의를 통해 작성되었으며, 프로젝트 설계 및 구현의 기준으로 사용된다.

2. Overall Description

2.1 Product Perspective

본 제품은 청년 사용자와 청년 정책 DB 를 연결하는 웹 서비스 플랫폼이다. 정부와 지자체에서 제공하는 수많은 청년 정책 정보가 파편화되어 있고 용어가 복잡하여, 정작 정책 수혜 대상인 청년들의 정책 체감도와 수혜율이 낮은 문제를 해결하는 것을 개발 동기로 한다.

기존의 단순 키워드 검색이나 일반 LLM 챗봇이 가진 최신 정보 반영 문제(Knowledge Cut-off) 및 정보 왜곡 문제(Hallucination)를 RAG 기술로 극복하는 것을 핵심 목적으로 한다.

이를 통해, 청년 사용자 개개인의 상황에 최적화된 정책을 AI 가 추천하고(AI 기반 맞춤 정책 추천), 복잡한 신청 절차를 단계별로 안내하여(정책 활용 Step-by-step 가이드 제공), 청년 1 인당 정책 수혜율을 실질적으로 높이는 효과를 기대한다.

2.1.1. System Interfaces

본 플랫폼은 독립적으로 동작하지 않으며, 다음과 같은 외부 시스템 및 Actor 와 상호작용한다.

- **사용자 (청년):** React 로 구현된 애플리케이션을 통해 시스템과 상호작용하는 핵심 Actor
- **관리자 (지자체 담당자):** 제안서에서 정의된 잠재적 사용자. (B2G 확장 시) 별도의 관리자 페이지를 통해 데이터 리포트에 접근 가능
- **국무조정실 청년 정책 API:** 플랫폼의 정책 DB(MySQL)에 최신 정책 정보를 제공하는 핵심 외부 데이터 소스
- **OpenAI API:** RAG 모델에서 검색된 정책 정보를 바탕으로 사용자 맞춤형 추천 사유를 생성하는 외부 LLM 시스템

2.1.2. User Interfaces

인터페이스는 React 로 구현된 반응형 웹사이트를 통해 제공한다. 모든 인터페이스는 데스크톱, 노트북, 스마트폰 등 다양한 디바이스에서 일관된 사용자 경험을 제공해야 한다.

주요 인터페이스 화면은 UI 시안을 기반으로 하며, 다음을 포함한다:

- **홈 화면**: '최신 정책' 및 '인기 정책' 목록, '나를 위한 맞춤 정책' 추천 기능 접근 버튼 제공
- **로그인 / 회원가입 화면**: auth.js 와 연동되어 사용자 인증 및 초기 프로필(Survey) 입력을 처리
- **맞춤 정책 추천 화면**: 사용자가 자연어 프롬프트를 입력, RAG 추천 결과를 배지(badge)와 함께 리스트로 확인하는 화면
- **정책 상세 화면**: 정책의 상세 설명, 지원 내용, 신청 방법, 신청 링크 제공
- **마이페이지**: 사용자 정보 수정, 관심 정책(스크랩) 목록 조회 제공

2.1.3. Hardware Interfaces

본 시스템은 웹 서비스이므로, 표준적인 인터넷 연결이 가능한 데스크톱, 노트북, 스마트폰 등의 하드웨어를 통해 접근 가능하다. 별도의 특수 하드웨어는 요구되지 않는다.

2.1.4. Software Interfaces

본 시스템은 다음과 같은 소프트웨어 기술 스택을 기반으로 한다.

- Frontend (Client): React (v18 이상) 기반으로 개발되며, Chrome, Safari, Edge 등 최신 웹 브라우저에서의 실행 보장
- Backend (Server): Node.js (v18 이상) 환경에서 실행
- Database: MySQL (v8.0 이상) 사용

- **AI Framework:** LangChain (Python)을 기반으로 RAG 파이프라인 구축
- **External APIs:** OpenAI API, 국무조정실 청년정책 API

2.1.5. Communication Interfaces

- 사용자(Client, React)와 백엔드 서버(Node.js) 간의 통신은 HTTPS 프로토콜 기반의 RESTful API (JSON 형식)를 통해 이루어진다.
- 백엔드 서버는 db.js 를 통해 MySQL DB 와 통신한다.
- 백엔드 서버는 api_save.js 및 recommend.py 스크립트를 통해 외부 API(국무조정실, OpenAI)와 HTTPS 로 통신한다.

2.1.6. Memory Constraints

- 정책 DB(policies 테이블)에 저장될 정책의 총량은 api_save.js 스크립트를 통해 수집 가능한 모든 '승인' 상태의 유효한 정책으로, 약 수천 개(1,000 ~ 5,000 개) 수준으로 예상된다.
- 사용자 정보(users 테이블)는 서비스 사용자 수에 비례하여 증가한다.

2.1.7. Operations

- **사용자:** 시스템의 주요 사용자는 '청년 사용자'이다. (2.3 챕터에서 상세 기술)
- **운영 (배포):** 시스템은 24/7 가용성을 목표로 클라우드 환경(AWS)에서 운영되어야 한다.
- **운영 (데이터 동기화):** 정책 DB 최신성 유지를 위해, api_save.js 스크립트는 스케줄러를 통해 주기적으로 실행되어 국무조정실 API 로부터 데이터를 동기화하고 만료된 정책을 정리해야 한다.

2.2 Product Functions

2.2.1. 회원가입 및 프로파일링 (Survey)

사용자는 플랫폼의 개인화 기능(맞춤 추천, 정책 스크랩 등)을 이용하기 위해 회원가입을 할 수 있다. 이 과정에서 이메일, 암호화된 비밀번호뿐만 아니라, RAG 추천의 기반이 되는 초기 설문(Survey) 데이터(거주지, 소득, 관심 분야 등)를 입력한다. 이 정보는 MySQL users 테이블에 안전하게 저장된다.

2.2.2. 로그인 및 마이페이지 (계정 관리)

기가입자는 auth.js 의 /login API 를 통해 로그인하여 서비스를 이용한다. 로그인한 사용자는 '마이페이지' 기능에 접근할 수 있다. 이 페이지에서 회원가입 시 입력했던 본인의 프로필 정보를 조회하고, 거주지나 소득, 관심 분야 등 변경된 정보를 직접 수정할 수 있다. 수정된 정보는 AI 추천의 정확도에 즉시 반영된다.

2.2.3. RAG 기반 맞춤형 정책 추천

UI 에서 사용자의 자연어 질문(prompt)을 입력 받으면, 백엔드는 policies.js 의 /recommend API 를 호출한다. 이 API 는 recommend.py 스크립트를 실행, 저장된 사용자 프로필(users DB)과 프롬프트를 복합적으로 분석한다. RAG 모델은 policies DB 의 최신 정보만을 실시간으로 참조하여, Hallucination 없이 사용자에게 최적화된 정책 목록과 추천 사유를 생성하여 반환한다.

2.2.4. 정책 탐색 및 관리 (검색, 스크랩, 리뷰)

RAG 추천 외에도 사용자는 다양한 방법으로 정책을 탐색할 수 있다.

- **상세 검색**: policies.js 의 /search API 를 통해 키워드, 지역, 학력 등 상세 필터로 정책을 검색
- **목록 조회**: 홈 화면에서 '최신 정책'및 '인기 정책' 조회
- **상세 조회**: 정책 클릭 시 상세 정보 확인

2.2.5. 정책 활용 가이드 (Step-by-step)

policies DB 에서 제공되는 신청 방법, 제출 서류 등의 복잡한 텍스트 정보를, 정책 상세 페이지에서 시각화 된 단계별 가이드(예시: 1. 서류 준비 -> 2. 링크 접수 -> 3. 심사)로 재구성하여 제공한다. 이는 정보 비대칭을 해소하고 신청 과정에서의 사용자 이탈을 방지하는 것을 목표로 한다.

2.2.6. 정책 데이터 분석 (B2G 기능)

향후 B2G 확장 시, users 테이블의 프로필 및 policies/recommend API 호출 로그(사용자 수요 데이터)를 분석하여, 지자체/정부에 정책 사각지대 및 인기 정책 리포트를 제공하는 관리자 기능을 포함한다.

2.3 User Classes and Characteristics

플랫폼의 주요 사용자 유형과 일반적인 특성은 다음과 같다.

2.3.1. 청년 사용자 (핵심 사용자)

정의: 정책 수혜 대상이 되는 최종 사용자(End-User)이다.

특성:

- ✓ 본인의 조건(나이, 소득, 지역, 관심사)에 맞는 정책을 찾고자 하는 강한 동기를 가지고 있다.

- ✓ 복잡한 행정 용어에 익숙하지 않으며, 여러 정부 사이트에 흩어진 정보를 탐색하는 데 많은 시간을 들이기 어렵다.
- ✓ React 기반의 현대적인 웹 인터페이스 및 RAG 챗봇과 상호작용할 수 있는 기본적인 기술 소양을 갖추고 있다.
- ✓ 회원가입 및 프로필 정보를 입력하고, policies.js 및 mypage.js 의 개인화된 기능을 적극적으로 활용한다.

2.3.2. 지자체/정부 담당자 (잠재적 사용자, B2G)

정의: B2G 활용성에 기반한 2 차 사용자(관리자/분석가)이다.

특성:

- ✓ 플랫폼에 축적된 사용자 수요 데이터(users 프로필 통계, /recommend API 의 프롬프트 키워드, /popular 정책 조회수 등)를 분석하여 정책의 사각지대를 파악하고 효율적인 예산 배분을 원하는 요구가 있다.
- ✓ (향후 확장 시) 2.2.6 기능에 정의된 별도의 관리자 페이지 및 데이터 리포트에 접근할 수 있는 권한을 가진다.
- ✓ 이 사용자는 본 개발팀(Team 3)과는 다른 별개의 System Administrator 이다.

2.4 Design and Implementation Constraints

본 '청년정책 추천 플랫폼'은 다음 목록에 기술된 설계 및 개발 제약 조건을 준수해야 한다.

- **아키텍처**: 시스템은 반드시 RAG 아키텍처를 기반으로 해야 한다.
- **Frontend**: 프론트엔드 웹사이트는 React 프레임워크를 사용하여 개발해야 한다.

- **Backend:** 백엔드 API 서버는 Node.js 환경에서 개발해야 한다.
- **Database:** 사용자 정보, 정책 데이터, 리뷰 등을 저장하는 데이터베이스는 MySQL 을 사용해야 한다.
- **AI 연동:**
    - AI 추천 모델은 LangChain 프레임워크(Python)를 기반으로 구축해야 한다.
    - LLM 은 OpenAI API (GPT 계열)를 활용해야 한다.
- **데이터 출처:** 정책 데이터의 주 출처는 국무조정실 청년 정책 API 여야 하며, api_save.js 스크립트를 통해 주기적으로 동기화되어야 한다.
- **보안:** 사용자 비밀번호는 반드시 bcrypt 등의 검증된 알고리즘을 사용하여 단방향 암호화(해시) 후 저장해야 한다.
- **인증:** API 접근 제어는 JWT 기반의 인증 미들웨어를 사용해야 한다.

2.5 Assumptions and Dependencies

본 명세서에 기술된 요구 사항들은 다음의 가정과 의존성 요소를 기반으로 한다.

- Assumptions:
    - **사용자 정보 제공의 정확성:** 사용자는 RAG 기반 맞춤형 추천의 정확도를 높이기 위해, 회원가입 및 마이페이지 수정 시 자신의 거주지, 나이, 소득 수준, 관심 분야 등의 개인 프로필 정보를 기꺼이, 그리고 정확하게 제공할 것이라고 가정한다. 부정확한 정보는 추천 품질 저하로 이어질 수 있다.
- Dependencies:
    - **국무조정실 청년 정책 API:** 본 시스템은 정책 데이터의 최신성과 정확성을 국무조정실 청년 정책 API 에 전적으로 의존한다. 이 API 가 안정적으로 24 시간 운영되지 않거나, 데이터 제공에 오류/지연이 발생할 경우, api_save.js 를 통한 DB 동기화 실패 및

search_v2.py 의 검색 결과 누락 등 시스템의 핵심 기능이 원활하게 작동하지 않을 수 있다.

- **OpenAI API:** RAG 기반 맞춤형 추천 및 추천 사유 생성 기능의 품질과 속도는 OpenAI API 의 성능과 안정성에 직접적으로 의존한다. API 장애 또는 응답 지연 시, 맞춤형 추천 기능이 실패하거나 지연될 수 있다.

3. External Interface Requirements

3.1 External Interfaces

3.1.1 User Interface (UI)

사용자가 직접 상호작용하는 웹사이트의 주요 화면 인터페이스를 정의한다.

이름	마우스 및 키보드를 통한 웹 입력 처리
목적/내용	시스템 사용자가 키보드(텍스트 입력)와 마우스(클릭, 스크롤)의 입력을 통해 웹 애플리케이션 시스템에 명령 전달
입력 주체/ 출력 목적지	사용자 / 웹 브라우저 (클라이언트)
범위/정확도/ 허용 오차	• 범위: 화면에 렌더링된 버튼, 입력 필드, 링크, 탭 등 상호작용 가능한 모든 HTML 요소. • 정확도: 유저의 마우스 및 키보드 입력에 따른 정확도 • 허용 오차: 해당 없음
단위	버튼 클릭, 링크 클릭, 텍스트 입력
시간/속도	비정기적인 사용자의 입력 / 즉각적인 사용자 명령 수행 (UI 반응)
타 입출력과의 관계	입력 내용에 따라 클라이언트(React)에서 처리되거나, Client-Server API 를 통해 서버(Node.js)로 명령을 요청
데이터 형식 및 구성	Text (회원가입 정보, 로그인, 정책 검색어, 추천 프롬프트), Mouse Click (이벤트)
명령 형식	각 버튼 및 링크 클릭에 따른 API 호출 또는 페이지 라우팅(Routing) 명령 매핑
종료 메시지	해당 없음

Table 3.1: Processing Web Inputs via Mouse and Keyboard

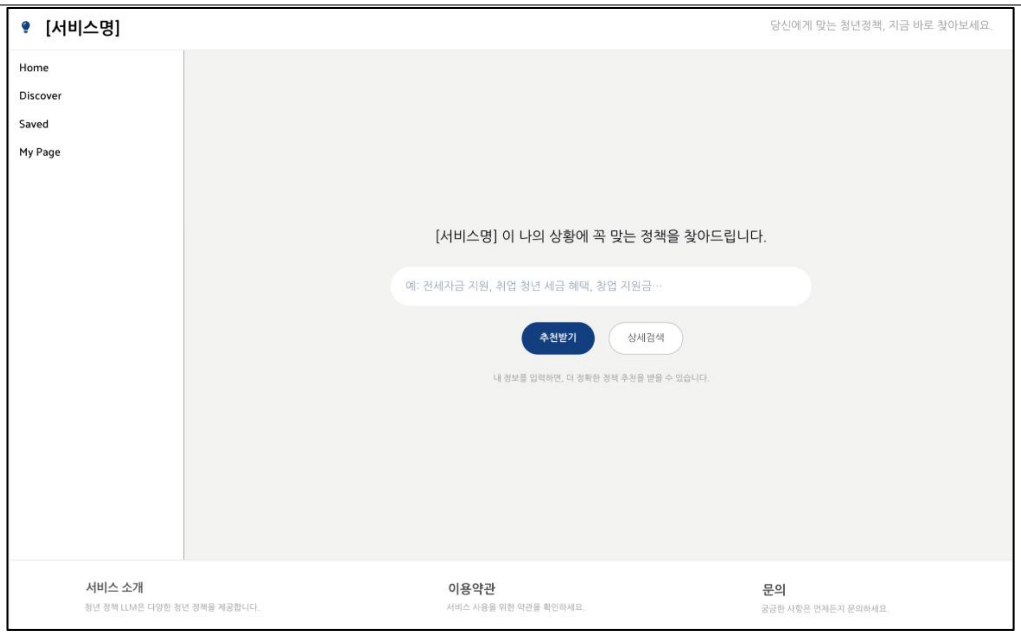
이름	웹 브라우저를 통한 메인 화면 출력
목적/내용	시스템 사용자가 Home 에 접속했을 때 제공되는 기본 인터페이스 정책 탐색(최신/인기) 및 맞춤 추천 기능으로의 진입점 제공
입력 주체/ 출력 목적지	클라이언트 / 사용자
범위/정확도/ 허용 오차	해당 없음
단위	화면 (웹 페이지)
시간/속도	사용자의 'Home' 메뉴 클릭 또는 메인 URL 접속에 따른 즉각적인 화면 전환
타 입출력과의 관계	페이지 로드 시, API(<code>GET /api/policies/recent</code>, <code>GET /api/policies/popular</code>)를 호출하여 받은 JSON 데이터를 화면에 렌더링
화면 형식 및 구성	 Figure 3.1: Main User Interface • 좌측 (내비게이션 바): Home, Discover, Saved, My Page 메뉴 • 중앙 (컨텐츠 영역): 서비스명, 자연어 검색창, 추천받기 버튼, 상세검색 버튼 • 중앙 (데이터 영역): '최신 정책' 목록과 '인기 정책' 목록이 표시됨
윈도우 형식 및 구성	• React 컴포넌트 기반의 SPA(Single Page Application) 레이아웃. • 내비게이션 바, 컨텐츠 영역, 푸터 영역으로 구성
데이터 형식 및 구성	HTML, CSS, JavaScript, Text (정책명)
명령 형식	해당 없음
종료 메시지	해당 없음

Table 3.2: Main Page Web Interface Output

이름	사용자 등록 인터페이스
목적/내용	시스템에 신규 사용자의 프로필 정보(개인정보, 관심사 등) 등록
입력 주체/ 출력 목적지	사용자 / 서버
범위/정확도/ 허용 오차	API 명세서에 정의된 각 필드의 유효성 규칙을 따름
단위	화면 (웹 Form)
시간/속도	'가입하기' 버튼 클릭 시 서버에 비동기 요청 전송, 수 초 내 응답
타 입출력과의 관계	<code>GET /api/auth/check-email</code>, <code>GET /api/auth/check-nickname</code>을 통해 입력값의 유효성/중복을 실시간으로 확인
화면 형식 및 구성	• 필요한 정보를 입력받기 위한 텍스트 필드, 드롭다운, 체크박스 슬롯 • 계정 정보 등록을 위한 '가입하기' 버튼
윈도우 형식 및 구성	회원가입 Form 페이지
데이터 형식 및 구성	JSON (서버로 전송되는 Request Body)
명령 형식	'가입하기' 버튼 클릭에 따른 API 호출 명령 매핑
종료 메시지	• 성공(201): "회원가입 성공" 또는 자동 로그인 처리 • 실패(400): "필수 값 누락" • 실패(409): "이메일 중복"

Table 3.3: User Registration Interface

이름	맞춤 정책 추천 인터페이스
목적/내용	사용자의 자연어 프롬프트와 기존 저장된 프로필 정보를 기반으로 RAG 맞춤형 정책을 추천받는 인터페이스
입력 주체/ 출력 목적지	사용자 / 서버
범위/정확도/ 허용 오차	사용자의 남은 추천 횟수가 0 보다 커야 함
단위	화면 (웹 Form)
시간/속도	'추천 받기' 버튼 클릭 시 RAG 파이프라인(Python 스크립트 호출 포함)이 실행되며, 수 초~수십 초 이내 응답
타 입출력과의 관계	인증된 사용자의 email 과 prompt 텍스트가 recommend.py 스크립트의 입력으로 전달
화면 형식 및 구성	• 자연어 prompt 를 입력하기 위한 텍스트 입력창 • 추천 실행을 위한 추천 받기 버튼 • 추천 결과가 카드 목록 형태로 출력되는 영역
윈도우 형식 및 구성	메인 페이지의 중앙 컨텐츠 영역 또는 별도의 추천 페이지
데이터 형식 및 구성	입력: Text 출력: JSON (서버로부터 받은 recommendations 배열)
명령 형식	'추천 받기' 버튼 클릭에 따른 API 호출
종료 메시지	• 실패(400): "오늘 추천 횟수가 모두 소진되었습니다" • 실패(401): "인증 실패: 토큰 없음"

Table 3.4: Custom Policy Recommendation Interface

3.1.2 Software Interface (Client-Server API)

웹 클라이언트(React)와 백엔드 서버(Node.js) 간의 RESTful API 인터페이스를 정의한다.

항목	설명
이름	'청년정책 추천 플랫폼' API (Client ↔ Server)
목적/내용	웹 클라이언트와 Node.js 백엔드 서버 간의 통신. 사용자 인증, 마이페이지 관리, 정책 조회 및 RAG 추천 기능 제공
데이터 형식	모든 요청(Request) 및 응답(Response)은 json 형식을 기본으로 사용
인증 방식	bearerAuth (JWT). /api/auth/login 또는 /api/auth/signup 으로 발급받은 Access Token 을 HTTP Header 의 Authorization 필드에 Bearer <token> 형태로 전송
주요 엔드포인트 명세	1. Auth API (/api/auth) • signup: 사용자 프로필 정보(email, nickname, password, birthDate, location, income, education 등)를 받아 계정 생성 • login: email, password 를 받아 인증을 수행하고 token (Access Token) 및 refreshToken 반환 • refresh: refreshToken 을 받아 유효성 검증 및 새로운 token (Access Token) 및 refreshToken 발급 • check-email: email 쿼리 파라미터로 DB 중복 여부를 {"data": {"exists": ...}} 형태로 반환 • check-nickname: nickname 쿼리 파라미터로 DB 중복 및 비속어 포함 여부를 {"data": {"exists": ...}} 형태로 반환 2. MyPage API (/api/mypage) • basic: 인증된 사용자의 기본 정보(email, nickname, birthDate, location 등) 조회 • detail: 인증된 사용자의 모든 상세 프로필 정보(소득, 학력 등) 조회 • edit: 인증된 사용자의 프로필 정보(nickname, location, interests 등) 수정 • recommend: 사용자의 남은 RAG 추천 횟수 조회 3. Policies API (/api/policies) • recent: DB 에서 최신 정책 3 개 조회 • popular: DB 에서 조회수 기준 인기 정책 3 개 조회 • id: policyId 를 받아 해당 정책의 상세 정보(설명, 지원 내용, 신청 방법 등)를 DB 에서 조회

	• search: 쿼리 파라미터를 활용하여 필터링된 정책 목록 반환 • recommend: prompt 쿼리 파라미터와 인증된 사용자 email 을 recommend.py 로 전달하여 RAG 기반 추천 결과 반환 (추천 횟수 1 차감)

Table 3.5: Software Interface - 'Youth Policy Recommendation Platform' API

3.1.3 Software Interface (Server-External API)

백엔드 서버가 RAG 및 데이터 수집을 위해 의존하는 외부 서비스 인터페이스를 정의한다.

항목	설명
이름	외부 청년 정책 정보 API
목적/내용	원천 정책 데이터를 수집하여 MySQL DB 에 동기화하기 위함. js 스크립트가 주기적으로 이 API 호출
인터페이스 방식	HTTP GET 요청
데이터 형식	JSON 형식으로 데이터 수신
주요 로직	• API 를 페이지별로 호출하여 승인 상태, 신청 기간 등을 기준으로 유효한 정책 필터링 • 지역 코드, 학력, 직업 코드 등을 참조하여 원본 코드를 사람이 읽을 수 있는 텍스트로 변환 • 변환된 데이터를 테이블에 INSERT … ON DUPLICATE KEY UPDATE 로 저장 • 마감이거나 신청 기간이 지난 정책을 DB 에서 삭제

Table 3.6: Software Interface - External Policy Information API (Data Information)

항목	설명
이름	OpenAI API
목적/내용	RAG 파이프라인의 Generation 단계 수행. 정책 필터링, 우선순위 결정, 추천 사유 생성을 위해 LLM 호출
인터페이스 방식	OpenAI Python Client (openai.OpenAI(api_key=...))를 통한 HTTPS API 호출
데이터 형식	OpenAI Python Client 가 관리하는 JSON 객체
주요 로직	• 필터링된 정책 목록과 사용자 프로필, 프롬프트를 LLM 에 전달하여 가장 적합한 정책명 5 개를 텍스트로 추출 • 선정된 정책 목록과 상세 매칭 정보를 LLM 에 전달하여, 각 정책별 맞춤형 추천 사유를 JSON 형식으로 생성

Table 3.7: Software Interface - External LLM API

3.1.4 Software Interface (Internal Sub-system)

항목	설명
이름	내부 RAG/Search 서브시스템 인터페이스
목적/내용	메인 서버가 검색 및 RAG 추천 로직을 Python 스크립트에 위임하기 위한 동기식 인터페이스
인터페이스 방식	Node.js 를 통한 프로세스 실행
데이터 형식	입력: JSON 문자열이 Command-line Argument 로 전달 출력: Python 스크립트의 출력을 JSON 문자열로 간주하여 수신 및 파싱
주요 인터페이스	1. 정책 검색 • 입력: { "keyword": "...", "sido": "...", ... } 형태의 검색 필터 JSON 문자열 • 출력: [{ "id": 1, "plcyNm": "..." }, ...] 형태의 필터링된 정책 목록 JSON 문자열 2. 정책 추천 • 입력: email (사용자 ID) 문자열, prompt (사용자 입력 프롬프트) 문자열 • 출력: [{ "id": 123, "plcyNm": "...", "reason": "...", "badges": [...] }, ...] 형태의 RAG 추천 결과 JSON 문자열

Table 3.8: Software Interface - Internal Subsystem (Node.js → Python)

3.1.5 Hardware Interface

이름	시스템에서 사용 가능한 디바이스
목적/내용	클라이언트: 사용자가 키보드, 마우스를 사용해 웹사이트에 접근하고 정책을 추천받음 서버: 백엔드 로직(Node.js, Python)을 실행하고 외부 요청 처리 데이터베이스: mysql 를 통해 연결된 policies, users 등의 데이터를 저장/관리
입력 주체/ 출력 목적지	사용자/클라이언트 디바이스 (웹 브라우저), 클라이언트/서버 (Node.js), 서버/데이터베이스 (MySQL)
범위/정확도/ 허용 오차	해당 없음
단위	해당 없음
시간/속도	사용자의 입력/즉각적인 UI 처리 및 비동기 API 요청
타 입출력과의 관계	해당 없음
데이터 형식 및 구성	해당 없음
명령 형식	키보드 입력, 마우스 클릭
종료 메시지	해당 없음

Table 3.9: Hardware Interface

3.1.6 Communication Interface

이름	호스트 서버 – 클라이언트
목적/내용	각 클라이언트(웹 브라우저)에서 호스트 서버에 접속을 요청, 사용자가 입력한 JSON 데이터를 서버에서 전달받아 이에 해당하는 JSON 응답을 제공
입력 주체/ 출력 목적지	클라이언트(웹 브라우저)와 호스트 서버(Node.js)
범위/정확도/ 허용 오차	해당 없음
단위	패킷 (HTTP)
시간/속도	최소 10Mbps 이상 (권장)
타 입출력과의 관계	사용자 입력(UI)이 API(JSON)로 변환되어 본 통신을 통해 서버로 전송
데이터 형식	json 을 이용한 요청/응답 데이터
명령 형식	HTTPS/HTTP 프로토콜 (GET, POST, PUT, DELETE 메서드)
종료 메시지	HTTP 응답 코드 (예: 200, 201, 400, 401, 404, 500)

Table 3.10: Communication Interface - Host Server-Client (Client↔ Server)

이름	호스트 서버 – 외부 API (정책/LLM)
목적/내용	1. 외부 정책 API 에서 원본 정책 데이터를 수집 2. OpenAI 클라이언트를 통해 LLM(GPT) API 를 호출하여 RAG 추천 사유를 생성
입력 주체/ 출력 목적지	호스트 서버(Node.js/Python)와 외부 서비스(정책 API, OpenAI API)
범위/정확도/ 허용 오차	해당 없음
단위	패킷 (HTTPS)
시간/속도	외부 API 의 응답 속도에 의존적
타 입출력과의 관계	소프트웨어 인터페이스가 본 통신을 통해 실행
데이터 형식	JSON 요청 및 응답
명령 형식	HTTPS 프로토콜 (GET, POST 메서드)
종료 메시지	HTTP 응답 코드

Table 3.11: Communication Interface - Server-External API (Server ↔ External)

이름	호스트 서버 – 데이터베이스
목적/내용	백엔드 서버(Node.js, Python 스크립트)가 mysql 또는 pymysql 라이브러리를 사용해 데이터베이스에 접속하고, SQL 쿼리를 실행하여 데이터 CRUD(Create, Read, Update, Delete)
입력 주체/ 출력 목적지	호스트 서버(Node.js/Python)와 데이터베이스 서버(MySQL)
범위/정확도/ 허용 오차	connectionLimit: 10
단위	패킷 (MySQL Protocol)
시간/속도	DB 쿼리 실행 속도
타 입출력과의 관계	auth.js, mypage.js, policies.js 등 대부분의 API 로직이 DB 쿼리 포함
데이터 형식	• SQL 쿼리 (SELECT, INSERT, UPDATE, DELETE 등) • DB 응답 (Rows, AffectedRows 등)
명령 형식	TCP/IP 를 통한 MySQL 프로토콜 통신
종료 메시지	쿼리 성공 또는 에러 (SQL Error)

Table 3.12: Communication Interface - Server-Database (Server ↔ DB)

이름	호스트 서버 – 내부 서브시스템
목적/내용	메인 서버가 검색 및 RAG 추천 기능을 내부 Python 스크립트에 위임을 위한 통신
입력 주체/ 출력 목적지	Node.js 프로세스와 Python3 자식 프로세스
범위/정확도/ 허용 오차	해당 없음
단위	Stream
시간/속도	Python 스크립트 실행 속도
타 입출력과의 관계	search, recommend API 가 인터페이스 호출
데이터 형식	• 입력: Command-line Arguments • 출력: Standard Output
명령 형식	프로세스 생성 및 파이프를 통한 데이터 수신
종료 메시지	Python 프로세스 종료 코드

Table 3.13: Communication Interface - Server-Internal Subsystem (Node.js ↔ Python)

3.2 Function Requirement

3.2.1. Use Case

Use case name	Register (회원가입 및 프로파일링)
Actor	미등록 사용자 (청년)
Description	미등록 사용자가 시스템의 모든 개인화 기능을 이용하기 위해 계정 및 초기 프로필 정보를 등록하는 과정
Normal Course	1. 모든 사용자는 웹사이트 접속 후 화면 중앙에 로그인 페이지가 나타난다. 2. 등록 후 사용자에게만 서비스가 제공되는 것을 확인한 후, 미등록 사용자는 로그인 페이지에서 등록 버튼을 클릭한다. 3. 사용자가 페이지를 등록하도록 등록 페이지로 이동한다. 4. 사용자는 등록 양식에 따라 추가 정보를 입력해야 한다. 필요한 정보는 다음을 포함한다. (a) ID (이메일 주소) (b) Password (c) 생년월일 (d) 연소득을 비롯한 개인정보 5. 시스템은 이메일 주소가 올바른지 확인하고 암호를 찾는 상황을 준비하기 위해 지정된 전자 메일 주소로 확인 코드를 보낸다. 6. 양식을 작성한 후 사용자가 등록되고 양식 끝에 있는 등록 버튼을 클릭하면 로그인 페이지로 돌아간다.
Precondition	• 사용자가 아직 시스템에 등록되지 않아야 한다. • 사용자가 올바른 정보를 입력해야 한다. • 동일한 이메일 주소를 다른 사용자의 이메일 주소와 중복해서는 안 된다. • 잘못된 입력이 있는 경우, 시스템은 이메일 주소 및 비밀번호의 형식을 확인해야

	한다.
Post Condition	• users 테이블에 암호화된 password 및 모든 프로필 정보가 포함된 새 레코드가 생성된다. • 사용자는 '등록된 사용자' 상태가 된다.
Assumptions	해당 없음

Table 3.14: Use Case of Register

Use case name	Log-in / Log-out (로그인/로그아웃)
Actor	등록된 사용자
Description	• **로그인**: 시스템에 등록된 사용자가 서비스 사용 인증을 받는 프로세스 • **로그아웃**: 로그인한 사용자가 시스템에서 나가는 프로세스
Normal Course	**로그인**: 1. 시스템에 이미 회원으로 등록한 사용자가 시스템에서 서비스를 사용하려고 한다. 2. 등록을 위해 설정한 전자 메일 주소와 암호를 입력한다. 3. 정보가 올바르면, 시스템은 사용자가 시스템에 접속할 수 있도록 하며, 이제 사용자는 시스템에서 제공하는 모든 서비스를 이용할 수 있다. **로그아웃**: 1. 시스템을 종료하려면 'logout' 버튼을 클릭한다. 사용자가 로그아웃 하지 않고 응용 프로그램을 닫은 경우 시스템이 해당 사용자에 대한 세션을 임의로 닫는다.
Precondition	• **로그인**: 사용자가 시스템에 이미 등록되어 있어야 한다. • **로그아웃**: 사용자가 로그인 상태여야 한다.
Post Condition	사용자가 온라인 상태여야 한다.
Assumptions	해당 없음

Table 3.15: Use Case of Log-in/out

Use case name	Recommendation (RAG 맞춤 정책 추천)
Actor	등록된 사용자
Description	추천 과정은 회원으로 등록된 사용자가 시스템에서 제공하는 청년정책 추천 서비스를 통해 자신의 맞춤 정책을 통해 확인하는 프로세스
Normal Course	1. 모든 사용자는 로그인 후 채팅창, 최신 정책과 조회수 기반으로 인기 있는 정책이 나타난다. 2. 채팅창에 질의를 한다. 3. 질의를 하면 질의에 맞는 사용자 맞춤 정책들이 뜬다. 4. 추천 정책들을 누르고 상세정보를 확인한다.
Precondition	사용자가 로그인 상태여야 한다.
Post Condition	인기있는 정책 저장을 위해 시스템에 해당 사용자의 키워드가 저장된다.
Assumptions	해당 없음

Table 3.16: Use Case of Recommendation

Use case name	Floating Policy
Actor	등록된 사용자
Description	플로팅 과정은 회원으로 등록된 사용자가 최신 정책과 조회수 기반의 인기정책을 확인하는 프로세스
Normal Course	• 모든 사용자는 로그인 후 채팅창, 최신 정책과 조회수 기반으로 인기 있는 정책이 나타난다. • 정책을 누르고 상세정보를 확인한다.
Precondition	사용자가 로그인 상태여야 한다.
Post Condition	해당 없음
Assumptions	해당 없음

Table 3.17: Use Case of Floating Policy

Use case name	Detailed Policy
Actor	등록된 사용자
Description	추천 과정은 회원으로 등록된 사용자가 시스템에서 제공하는 청년정책 추천 서비스를 통해 자신의 맞춤 정책을 통해 확인하는 프로세스
Normal Course	• 모든 사용자는 로그인 후 채팅창, 최신 정책과 조회수 기반으로 인기 있는 정책이 나타난다. • 정책을 누르고 상세정보를 확인한다.
Precondition	사용자가 로그인 상태여야 한다.
Post Condition	해당 없음
Assumptions	해당 없음

Table 3.18: Use Case of Detailed Policy

3.2.2. Use Case Diagram

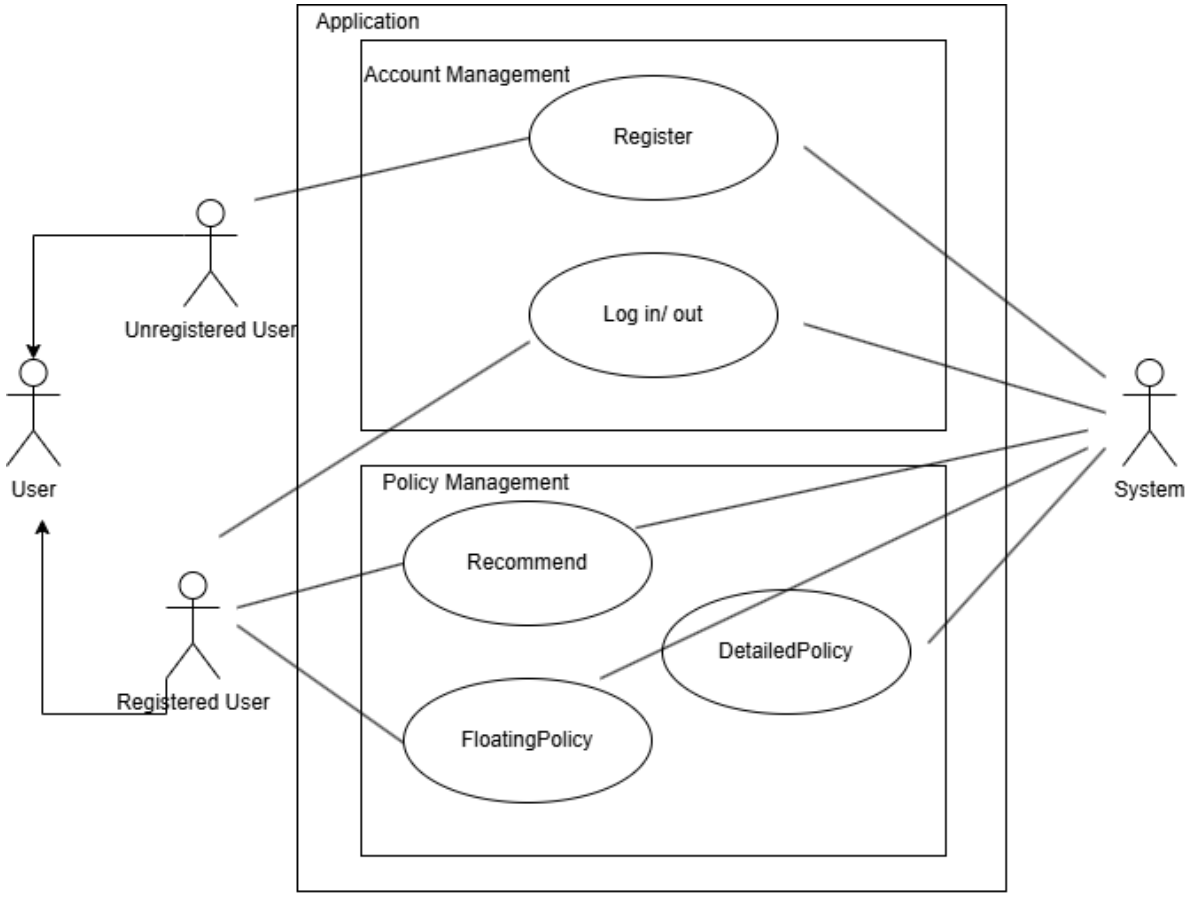


Figure 3.2: Use Case Diagram

3.2.3. Data Dictionary

Table 3.19: Table of Data Dictionary

Field	Key	Constraint	Description
email	PK	Not Null, VARCHAR(100)	이메일(로그인 ID)
nickname		Not Null, VARCHAR(50)	닉네임
password		Not Null, VARCHAR(255)	비밀번호 해시
birthDate		Not Null, DATE	생년월일
location		Not Null, VARCHAR(255)	거주 지역
maritalStatus		Not Null, VARCHAR(100)	혼인 상태
income		Not Null, INT	소득
education		Not Null, VARCHAR(100)	학력
major		Not Null, VARCHAR(100)	전공
employmentstatus		Not Null, VARCHAR(100)	취업 상태
interests		Not Null, TEXT	관심 분야
created_at		DEFAULT CURRENT_TIMESTAMP, DATETIME	생성 시각
refreshToken		NULL 허용, TEXT	리프레시 토큰

Field	Key	Constraint	Description
id	PK	Not Null, INT, AUTO_INCREMENT	정책 ID
plcyNm	UK	VARCHAR(255), UNIQUE, NULL 허용	정책명
lclsfNm		VARCHAR(100), NULL 허용	대분류명
plcyKywdNm		VARCHAR(255), NULL 허용	정책 키워드
plcyExplnCn		TEXT, NULL 허용	정책 설명
plcySprtCn		TEXT, NULL 허용	지원 내용
sprtSclLmtYn		VARCHAR(10), NULL 허용	소득 한도 여부
srngMthdCn		TEXT, NULL 허용	선정 방법
sprtSclCnt		VARCHAR(50), NULL 허용	지원 인원 수
ptcpPrpTrgtCn		TEXT, NULL 허용	참여/대상 설명
inqCnt		INT, NULL 허용	조회수
plcyAprvSttsCd		VARCHAR(50), NULL 허용	승인 상태 코드
aplyYmd		VARCHAR(50), NULL 허용	신청일(문자)
bizPrdSeCd		VARCHAR(50), NULL 허용	사업 기간 유형
bizPrdBgngYmd		VARCHAR(50), NULL 허용	사업 시작일
bizPrdEndYmd		VARCHAR(50), NULL 허용	사업 종료일
zipCd		TEXT, NULL 허용	지역/우편번호 정보
sprtTrgtMinAge		VARCHAR(10), NULL 허용	최소 연령
sprtTrgtMaxAge		VARCHAR(10), NULL 허용	최대 연령
sprtTrgtAgeLmtYn		VARCHAR(10), NULL 허용	연령 제한 여부
mrgSttsCd		VARCHAR(20), NULL 허용	혼인 상태 코드
earnCndSeCd		VARCHAR(20), NULL 허용	소득 조건 구분 코드
earnMinAmt		VARCHAR(50), NULL 허용	최소 소득
earnMaxAmt		VARCHAR(50), NULL 허용	최대 소득
schoolCd		VARCHAR(50), NULL 허용	학력 코드
jobCd		VARCHAR(50), NULL 허용	직업 코드
plcyMajorCd		VARCHAR(50), NULL 허용	전공(정책) 코드
sbizCd		VARCHAR(50), NULL 허용	세부 사업 코드
plcyAplyMthdCn		TEXT, NULL 허용	신청 방법

3.2.4. Data Flow Diagram

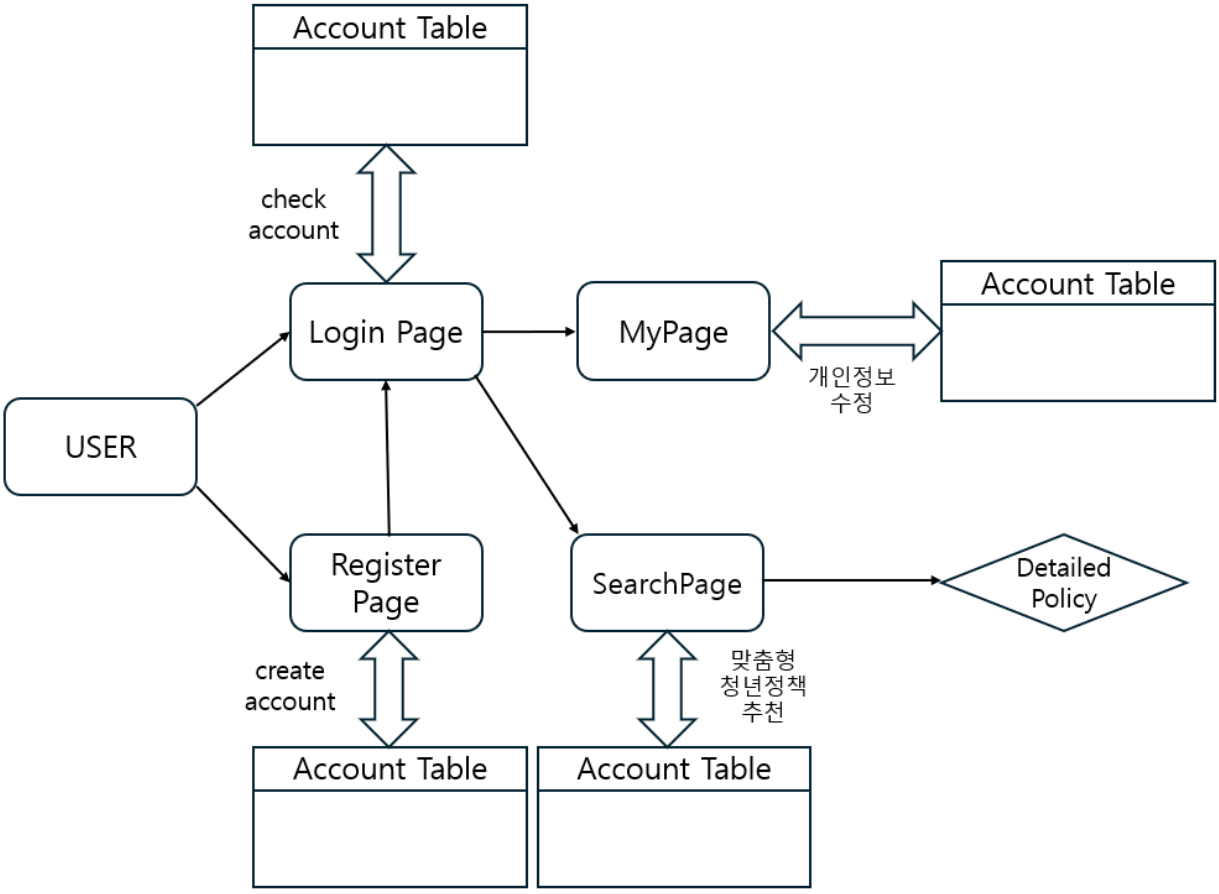


Figure 3.3: Data Flow Diagram

3.3 Performance Requirements

본 시스템의 성능 요구사항은 정적 요구사항과 동적요구사항으로 나누어 다음과 같이 정의한다. 예측에 기반한 내용이며 실제 구현시 달라질 수 있다.

3.3.1. Static Numerical Requirement

- **지원 단말기**: 시스템은 React 기반 웹사이트를 지원하는 모든 최신 웹 브라우저(Chrome, Safari, Edge 등)가 설치된 데스크톱, 노트북, 스마트폰 등에서 정상적으로 동작해야 한다.
- **동시 사용자 수**: 시스템은 여러 명의 청년 사용자가 동시에 접속하여 핵심 기능을 사용할 수 있도록 지원해야 한다.

- **데이터 처리 용량:**
    - policies 테이블은 api_save.js 를 통해 수집되는 약 1,000 ~ 5,000 개의 정책 데이터를 저장 및 관리할 수 있어야 한다.
    - users 테이블은 최소한 100 명 이상의 사용자 계정 및 프로필 정보를 관리할 수 있어야 한다.

3.3.2. Dynamic Numerical Requirement

- **RAG 맞춤형 추천:** 사용자가 프롬프트 입력 후 '추천 받기' 버튼을 클릭한 시점부터 추천 결과가 화면에 표시될 때까지 걸리는 시간은, 95%의 경우 대략 15 초 이내에 완료되어야 한다. (단, 이는 특히 OpenAI API 의 응답 속도에 직접적인 영향을 받는다.)
- **상세 검색:** 사용자가 상세 검색을 요청한 시점부터 결과 목록이 표시될 때까지 걸리는 시간은 약 5 초 이내에 완료되어야 한다. (DB 에서 필터링하는 시간을 포함한다.)
- **일반 API 요청:**
    - 회원가입 및 로그인/로그아웃 요청은 약 3 초 이내에 완료되어야 한다.
    - 홈 화면의 정책 목록 조회, 정책 상세 조회, 마이페이지 프로필/관심 정책 조회 등 DB 조회 기반 API 는 약 3 초 이내에 완료되어야 한다.
    - 프로필 수정, 관심 정책 추가/삭제, 리뷰 작성/수정/삭제 등 DB 쓰기 기반 API 는 약 3 초 이내에 완료되어야 한다.

3.4 Logical Database Requirements

본 시스템은 MySQL 데이터베이스 서비스를 이용하여 데이터를 관리한다.

정보 유형: 데이터베이스는 users, policies 의 2 개 핵심 테이블을 포함한다.

- **users 테이블**: 사용자의 계정 정보(email, 암호화된)뿐만 아니라, RAG 맞춤 추천의 핵심 기반이 되는 개인 프로필 정보(birthDate, income, location, interests 등)를 저장한다. 또한 개인화 기능을 위한 recommendCount 정보, 인증을 위한 refreshToken 을 저장한다.
- **policies 테이블**: api_save.js 를 통해 외부 API(국무조정실)로부터 수집된 모든 청년 정책의 원본 및 가공된 데이터를 저장한다.

접근 권한:

- **일반 사용자**: 사용자는 mypage.js API 를 통해 본인의 users 레코드에 대해서만 조회 및 수정 권한을 갖는다.
- **시스템 (Backend)**: 백엔드 서버는 모든 테이블에 대해 CRUD 권한을 갖는다.
- **관리자 (B2G)**: (향후 확장 시) 관리자는 users 및 policies 테이블의 데이터를 조회/분석할 수 있는 읽기 권한을 갖는다.

무결성 제약: users 테이블의 email 필드는 Primary Key(PK)로, 고유해야 하며 중복될 수 없다. 이는 auth.js 의 GET /api/auth/check-email 및 POST /api/auth/signup 로직을 통해 보장된다.

3.5 Design Constraints

3.5.1. Standards Compliance

- **웹 표준**: 프론트엔드(React) 코드는 W3C 에서 권고하는 HTML5, CSS3, ECMAScript 6 (ES6+)의 최신 웹 표준을 준수해야 한다.
- **백엔드 표준**: 백엔드(Node.js) 코드는 일관된 JavaScript 코딩 스타일 가이드(예: Prettier, ESLint)를 따라야 한다.

- **Python 표준**: RAG 스크립트(recommend.py)는 PEP 8 Python 스타일 가이드를 준수해야 한다.
- **API 표준**: 모든 클라이언트-서버 통신은 RESTful API 원칙을 따르며, API 명세서에 정의된 규격을 준수해야 한다.

3.5.2. Software and Hardware Constraints

- **Frontend Framework**: 프론트엔드 웹사이트는 반드시 React 프레임워크를 사용하여 개발해야 한다. (2.4 절 제약조건)
- **Backend Runtime**: 백엔드 API 서버는 반드시 Node.js 및 Express 환경에서 개발해야 한다. (2.4 절 제약조건)
- **Database System**: 데이터베이스는 MySQL 을 사용해야 한다. (2.4 절 제약조건)
- **AI Stack**: AI 모델은 LangChain 프레임워크(Python) 기반으로 OpenAI API 를 활용해야 한다. (2.4 절 제약조건)
- **데이터 소스**: 정책 데이터의 주 출처는 국무조정실 청년 정책 API 여야 한다. (2.4 절 제약조건)
- **브라우저 호환성**: 시스템은 Chrome, Safari, Edge 의 최신 버전에서 모든 기능이 정상 동작해야 한다.
- **모바일 호환성**: 반응형 UI 는 iOS 16 이상, Android 11 이상의 모바일 환경에서 주요 기능(추천, 검색, 조회)이 동일하게 작동해야 한다.
- **하드웨어 최소 사양**: 사용자는 2GB RAM 이상의 메모리를 탑재한 기기(PC, 모바일)에서 서비스를 원활히 이용할 수 있어야 한다.

3.5.3. UI/UX Constraints

- **디자인 시스템**: 모든 UI 컴포넌트(버튼, 폰트, 색상 팔레트)는 사전에 정의된 디자인 시스템을 준수하여 일관성을 유지해야 한다.
- **성능 연동**: 백엔드 API 응답 시간(3.3.2 동적 요구사항)에 맞춰, 1 초 이상 소요되는 작업(특히 RAG 추천)은 사용자에게 로딩 스피너 또는 스켈레톤 UI 와 같은 시각적 피드백을 제공해야 한다.
- **이미지 로딩**: 정책 상세 페이지 등에 포함될 수 있는 이미지 리소스는 최적화되어, 3 초 이내에 로드되어야 한다.

3.5.4. Security Constraints

- **인증**: API 접근 제어는 반드시 authenticate 미들웨어와 같은 JWT 기반 인증을 사용해야 한다.
- **비밀번호**: 사용자 비밀번호는 bcrypt 를 사용하여 복호화가 불가능한 해시 값으로 변환하여 users 테이블에 저장해야 한다.
- **데이터 전송**: 모든 클라이언트-서버 통신은 HTTPS 프로토콜을 통해 암호화되어야 한다.
- **외부 라이브러리**: package.json 에 포함되는 모든 외부 라이브러리는 보안 취약점이 검증된 오픈소스만 사용해야 한다.

3.6 Software System Attributes

'청년정책 추천 플랫폼'이 제공해야 하는 기능적 요구사항 외의 비기능적 요구사항, 즉 시스템의 품질 속성을 정의한다. 이 속성들은 시스템이 얼마나 안정적이고, 안전하며, 효율적으로 운영될 수 있는지를 결정한다. 본 명세서는 Reliability(신뢰성), Availability(가용성), Security(보안성), Maintainability(유지보수성), Portability(휴대성)의 5 가지 핵심 속성을 기술한다.

3.6.1 Reliability

Reliability(신뢰성)은 시스템 배포 시 사용자가 신뢰할 수 있는 정확한 정보를 일관되게 제공하고, 데이터 무결성을 보장하는 데 필요한 요소를 의미한다.

- **정보 신뢰성 (RAG 기반 Hallucination 방지)**
    - 본 플랫폼은 LLM 의 가장 큰 신뢰성 문제인 'Hallucination(환각)' 문제를 방지하기 위해 RAG 아키텍처를 채택한다.
    - recommend.py 의 함수는 LLM 이 임의의 정보를 생성하는 것이 아니라, policies DB 에서 사전에 필터링 된 정책 문서를 기반으로 추천 사유를 생성하도록 강제한다.
    - 사용되는 시스템 프롬프트("역할: 한국 청년정책 추천 에디터. 사실만 사용해...")는 LLM 이 사실에 기반한 답변만 생성하도록 하여 정보의 신뢰성을 확립한다.
- **데이터 신뢰성 (유효 데이터 선별)**
    - '정책 추천' use case 실행 시, 사용자의 프로필(users 테이블)과 정책 조건(policies 테이블)을 정확하게 비교하여 자격이 없는 정책이 추천되는 오류를 최소화해야 한다.
    - 시스템 신뢰성의 근간이 되는 원천 데이터는 스크립트를 통해 신뢰할 수 있는 정책만 선별하여 구축한다.
    - 해당 스크립트는 외부 API 호출 시 승인된 정책만을 필터링하고, 신청이 마감된 정책을 제외하여 DB 에 저장함으로써 데이터셋 자체의 신뢰성을 보장한다.

- **데이터 무결성 (DB 트랜잭션)**
    - 데이터베이스 트랜잭션을 적용하여, 두 작업 중 하나만 성공해 발생하는 데이터 불일치 상태를 방지하고 데이터 무결성을 보장한다.
- **오류 처리**
    - 정책 데이터 원본(공공 데이터 API 등)에 연결할 수 없거나 응답이 지연될 경우, 시스템이 중단되지 않고 사용자에게 "데이터를 불러오는 데 실패했습니다"와 같은 명확한 오류 메시지를 반환한다.

3.6.2 Availability

Availability(가용성)은 사용자가 필요로 할 때 시스템이 중단 없이 작동하고, 항상 최신의 유효한 정보에 접근할 수 있도록 보장하는 데 필요한 요소를 의미한다.

- **정보 가용성 (최신 데이터 동기화)**
    - 시스템은 신규/업데이트 정책 저장 및 만료 정책 삭제 기능을 통해 주기적인 체크포인트를 수행한다.
    - 외부 API 의 최신 정보를 DB 에 동기화하여, LLM 의 'Knowledge Cut-off' 한계를 극복하고 사용자에게 항상 신청 가능한 최신 정책 정보의 가용성을 보장한다.
- **서비스 가용성 (세션 복구)**
    - 사용자의 액세스 토큰이 1 시간 후 만료되더라도, refresh 엔드포인트를 통해 refreshToken 으로 새로운 액세스 토큰을 '재시작' 할 수 있다.
    - 이러한 복구 메커니즘은 사용자가 재로그인해야 하는 불편함 없이 서비스를 지속적으로 이용할 수 있도록 보장한다.

- **리소스 가용성**
    - 백엔드 서버는 데이터베이스 커넥션 풀을 관리한다. 이는 다수의 동시 사용자 접속 시 DB 커넥션 고갈로 인한 서비스 장애를 방지하고 안정적인 시스템 가용성을 보장한다.
    - **목표 가용성**: 시스템은 99.9%의 가용성(Uptime)을 목표로 한다. (월간 약 43 분의 장애 허용)
    - **유지보수**: 데이터베이스 스키마 변경, 서버 업데이트 등의 정기 점검은 사용자가 가장 적은 시간대에 수행되어야 한다.
    - **장애 복구**: 심각한 시스템 장애 발생 시, 5 분 이내에 서버 자동 재시작 등의 초기 복구 절차가 완료되어야 한다.

3.6.3 Security

Security(보안성)은 악의적인 액세스, 사용, 데이터 파괴로부터 시스템과 사용자의 민감한 개인정보를 보호하는 요소를 의미한다.

- **사용자 인증**
    - authenticate 미들웨어를 통해 검증되고 유효한 토큰을 소유한 사용자만이 마이페이지, 정책 추천 등 핵심 API 에 접근할 수 있다.
- **비밀번호 암호화 (Encryption)**
    - '특정 암호화 기술' 로 bcrypt 를 사용한다.
    - auth.js 의 signup API 는 사용자 비밀번호를 단방향 암호화하여 DB 에 저장하며, login 시에는 bcrypt.compare 를 통해 원본 비밀번호 노출 없이 안전하게 인증을 수행한다.

- **사용자 인가 (Authorization)**
    - mypage.js 의 모든 API 는 토큰에서 추출된 사용자 이메일을 SQL 쿼리의 <WHERE email = ?> 조건으로 사용한다. 이는 인증된 사용자가 오직 본인의 정보만 조회하고 수정할 수 있도록 보장하여 악의적인 액세스 를 차단한다.
- **입력값 검증**
    - '데이터 무결성 확인'의 일환으로 auth/check-nickname API 는 DB 중복 검사 외에도, 비속어 집합을 포함한 닉네임 생성을 400 Bad Request 로 차단하여 부적절한 데이터 입력을 방지한다.

3.6.4 Maintainability

Maintainability(유지보수성)은 소프트웨어 자체의 수정 용이성, 기능 개선, 확장성과 관련된 속성을 의미하며, 이는 모듈성, 인터페이스, 복잡성 관리와 관련된다.

- **모듈성 (API 라우팅)**
    - 시스템은 모듈성을 확보하기 위해 API 로직을 authRouter (/api/auth), policyRouter (/api/policies), mypageRouter (/api/mypage)로 명확하게 분리한다. 이를 통해 각 도메인의 수정이 다른 도메인에 미치는 영향을 최소화할 수 있다.
- **모듈성 (RAG 로직 분리)**
    - 복잡성이 높은 RAG 추천 및 검색 로직은 policies.js 가 직접 수행하지 않고, 명확한 '인터페이스' 를 통해 Python 스크립트에 책임을 위임한다.
    - 이를 통해 AI/Python 로직과 웹 서버 로직을 분리하여, 각 부분을 독립적으로 수정할 수 있다.

- 로깅(Logging)
    - 주요 시스템 이벤트(회원가입, 로그인 실패, 정책 추천 요청, DB 오류 등)는 로그 파일로 기록되어야 하며, 이는 오류 추적 및 디버깅에 활용된다.
- **정책 관리**
    - policies 테이블에 새로운 정책을 추가, 수정, 삭제하는 작업(예: 관리자 페이지 또는 배치 스크립트)은 시스템 전체를 재시작하지 않고도 적용 가능해야 한다.
- **설정 관리**
    - 여러 파일이 DB 접속 정보, API 키, JWT 비밀키 등을 .env 파일에서 관리한다. 이는 코드를 수정하지 않고도 개발, 운영 환경 설정을 쉽게 변경할 수 있게 하여 유지보수성을 높일 수 있다.
- **코드 표준**
    - 모든 백엔드/프론트엔드 코드는 사전에 정의된 코딩 컨벤션(스타일 가이드)을 준수해야 한다.

3.6.5 Portability

Portability(휴대성)은 소프트웨어를 하나의 호스트 시스템에서 다른 호스트 시스템/운영 체제로 쉽게 이식하는 것과 관련된 요소들을 설명한다.

- **환경 종속성 분리**
    - 시스템은 데이터베이스 주소(DB_HOST), API 키(API_KEY) 등 모든 환경 종속적인 설정을 .env 파일로 분리한다.
    - 이는 호스트 의존적인 코드의 비율을 최소화하여, 코드 변경 없이 다른 호스트 시스템(로컬, 테스트 서버 등)으로 쉽게 이식할 수 있게 한다.

- **표준 기술 스택 및 브라우저 호환성**
    - 본 플랫폼은 Node.js, React, Python3, MySQL 등 특정 운영 체제의 사용에 구애받지 않는 표준적이고 이식성이 높은 기술 스택으로 구성된다. 이 기술들은 Windows, macOS, Linux 등 다양한 운영 체제에서 동일하게 작동한다.
- **의존성 관리**
    - package.json 파일은 Node.js 의 모든 의존성을 명시한다. npm install 명령어를 통해 어떠한 호스트 환경에서도 동일한 개발 환경을 신속하게 구축할 수 있다.
- **배포 환경**
    - 백엔드 애플리케이션은 Docker 컨테이너화를 통해 AWS, kt cloud 등 다양한 클라우드 환경이나 온프레미스 서버에 쉽게 배포될 수 있어야 한다.

3.7 Organizing the Specific Requirements

본 플랫폼은 사용자의 상세 프로필과 자연어 입력을 기반으로 AI(RAG) 모델을 연동하여 맞춤형 정책을 추천하는 복합 시스템이다. 따라서 세부 요구사항이 광범위하며, 특정 상황과 기능에 따라 요구사항이 달라진다. 본 섹션에서는 세부 요구사항들을 다양한 기준(사용자 클래스, 기능, 객체, 상호작용 등)으로 체계화하여 구성한다.

3.7.1 System Mode

본 시스템은 사용자의 인증 상태에 따라 두 가지 주요 모드로 구분된다.

- 1. **로그인 모드 (Authenticated User Mode):** JWT 토큰 인증(authenticate 미들웨어)을 통과한 '청년' 사용자가 접근하는 모드
    - **기능:** AI 정책 추천, 정책 검색, 상세 조회, 관심 정책 관리, 리뷰 작성, 마이페이지 등 모든 핵심 기능 사용 가능

- 2. **시스템 관리 모드 (Administrator Mode):** 관리자가 시스템 데이터베이스를 관리하는 모드
    - **기능:** 외부 '청년정책 API'로부터 최신 데이터를 동기화하고, 만료된 정책을 일괄 삭제하여 데이터의 무결성과 최신성을 유지

3.7.2 User Class

본 시스템의 요구사항은 Class Diagram 에 정의된 주요 Actor 를 기준으로 분류할 수 있다.

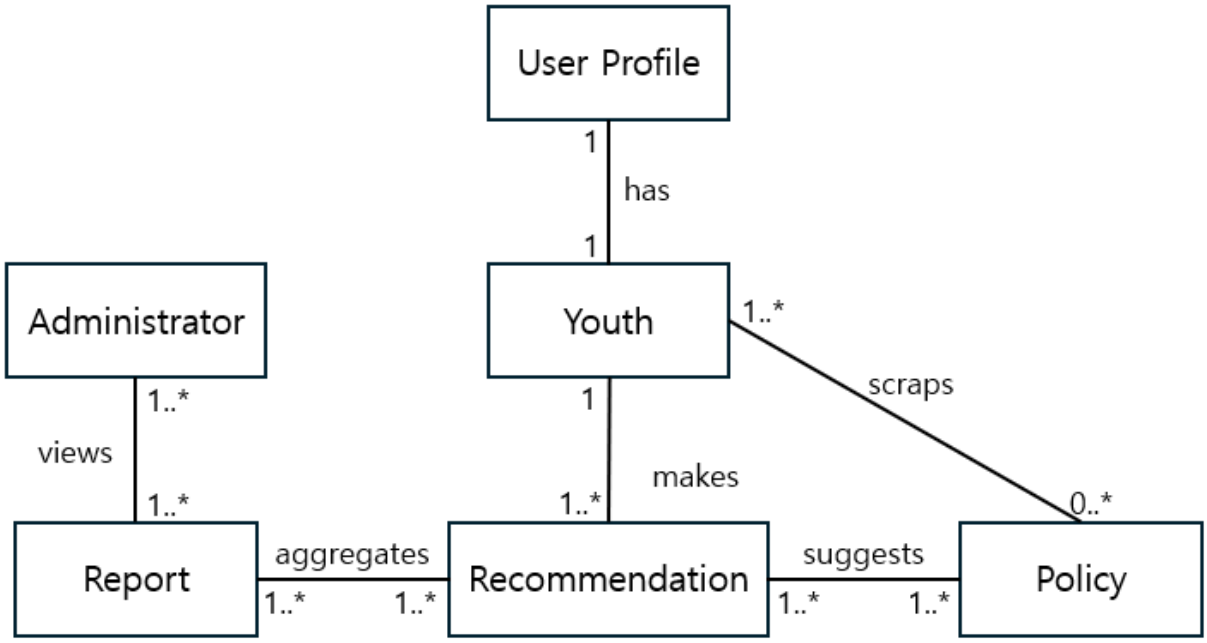


Figure 3.4: Class Diagram

- 1. **청년 (Youth):** 시스템의 핵심 사용자 클래스.
    - **특성:** 고유한 User Profile 을 1:1 로 가진다. 이 프로필은 회원가입 시 수집된 상세 정보(소득, 거주지, 학력, 관심 분야 등)를 포함
    - **주요 요구사항:**
        - 자신의 프로필을 기반으로 맞춤형 Recommendation 을 생성해야 한다.
        - Policy(정책)를 스크랩하여 관심 정책으로 관리할 수 있어야 한다.

- 2. **관리자 (Administrator)**: 시스템을 유지보수 하는 관리자 클래스.
    - **특성**: 시스템 운영에 필요한 데이터 리포트(Report)를 조회할 수 있다.
    - **주요 요구사항**:
        - 플랫폼 내 사용자 데이터(인기 정책, 검색 키워드 등)를 집계한 Report 를 생성 및 조회할 수 있어야 한다.
        - '청년정책 API'의 최신 데이터를 Policy DB 로 동기화할 수 있어야 한다.

3.7.3 Objects

본 시스템의 요구사항은 Class Diagram 에 정의된 주요 데이터 객체(Object)를 중심으로 구성된다.

- **User Profile / Youth**: 사용자의 계정 정보, 인증 토큰, 개인 프로필, 관심 정책 목록, 남은 추천 횟수를 관리하는 핵심 객체
- **Policy**: 외부 API 로부터 수집되어 Policy DB 에 저장되는 정책 원본 데이터 객체. 정책명, 지원 내용, 신청 방법, 자격 요건 등의 속성을 가짐
- **Recommendation**: 사용자의 User Profile 과 Policy 객체를 입력 받아, Open AI API 를 통해 생성되는 맞춤형 추천 결과 객체. 추천된 정책 목록과 추천 사유를 포함한다.

3.7.4 Feature

본 시스템의 기능 요구사항은 사용자에게 제공되는 주요 기능을 기준으로 다음과 같이 분류할 수 있다.

- Feature1: **사용자 계정 관리 (Account Management)**
    - Feature1.1: (회원가입) 사용자는 AI 맞춤 추천에 필요한 상세 프로필 정보(소득, 거주지, 학력, 관심 분야 등)를 입력하여 회원가입을 할 수 있다.

- Feature1.2: (로그인) 사용자는 이메일과 비밀번호로 로그인하며, 시스템은 접근(Access) 토큰과 갱신(Refresh) 토큰을 발급한다.
- Feature1.3: (정보 수정) 사용자는 마이페이지에서 닉네임, 소득, 관심 분야 등 자신의 프로필 정보를 수정할 수 있다.

- Feature2: AI 맞춤 정책 추천 (AI Recommendation)
  - Feature2.1: (추천 요청) 사용자는 자연어 프롬프트를 입력하여 자신에게 맞는 정책 추천을 요청할 수 있다.
  - Feature2.2: (RAG 기반 추천) 시스템은 사용자의 프로필 정보를 기반으로 Policy DB 를 1 차 필터링하고, AI 모델(Open AI API)을 통해 프롬프트와 가장 일치하는 정책을 최종 추천한다.
  - Feature2.3: (추천 사유 제공) 시스템은 AI 가 생성한 구체적인 추천 사유(예: "관심 키워드(교육지원)와 일치...")와 핵심 배지(예: "보조금")를 결과와 함께 제공한다.
  - Feature2.4: (추천 횟수 제한) AI 추천 기능은 사용자별로 일일 사용 횟수가 제한되며, 요청 시 1 회 차감된다.
- Feature3: 정책 탐색 (Policy Exploration)
  - Feature3.1: (필터 검색) 사용자는 키워드, 지역, 취업 상태 등 상세 필터를 적용하여 정책을 검색할 수 있다.
  - Feature3.2: (상세 조회) 사용자는 정책 목록에서 특정 항목을 클릭하여 정책 설명, 지원 내용, 신청 방법, 신청 기간 등 상세 정보를 조회할 수 있다.
  - Feature3.3: (홈페이지) 사용자는 메인 페이지에서 '최신 정책'과 '인기 정책' 목록을 조회할 수 있다.

3.7.5 Stimulus / Response

본 시스템의 핵심 상호작용인 'AI 맞춤 정책 추천'의 자극(Stimulus)과 반응(Response)은 Sequence Diagram 에 정의된 바와 같이 다음과 같은 순서로 진행된다.

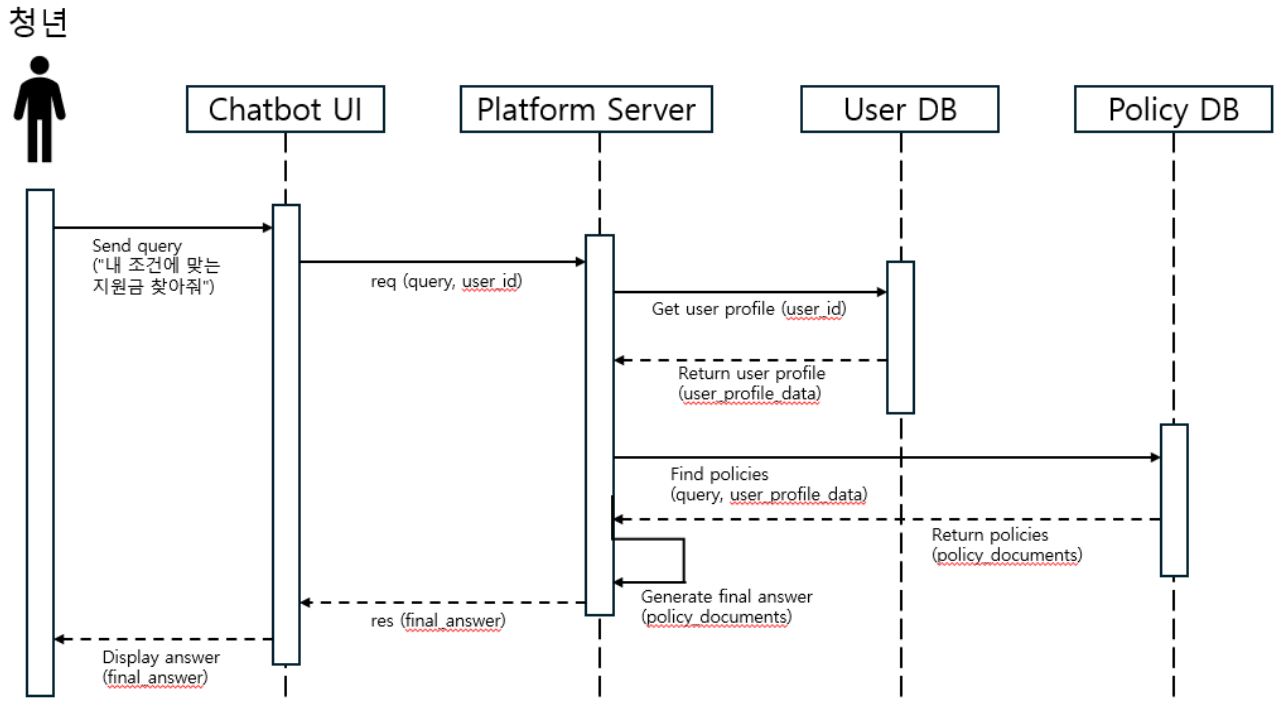


Figure 3.5: Sequence Diagram

- **Stimulus:** 청년 사용자가 Chatbot UI (웹 추천 페이지)에서 "내 조건에 맞는 지원금 찾아줘"와 같은 쿼리를 입력하고 '추천 받기' 버튼을 클릭한다.
- **Response:** 시스템은 다음 순서로 반응한다.
    1. **UI → Server:** req(query, user_id) API 요청(GET /api/policies/recommend)을 Platform Server 로 전송한다.
    2. **Server → User DB:** Get user profile (user_id) 요청을 User DB 로 전송한다.
    3. **User DB → Server:** Return user profile (user_profile_data) 응답을 반환한다. (이 과정은 recommend.py 스크립트 내부에서 실행)

4. **Server → Policy DB:** Find policies (query, user_profile_data) 요청을 Policy DB 로 전송한다. (RAG 1 차 필터링)
5. **Policy DB → Server:** Return policies (policy_documents) 응답을 반환한다.
6. **Server (Self-call):** Generate final answer (policy_documents)를 수행한다. (RAG 2/3 차, Open AI API 를 호출하여 랭킹 및 사유 생성)
7. **Server → UI:** res(final_answer) 최종 응답을 Chatbot UI 로 전송한다.
8. **UI (Self-call):** Display answer (final_answer)를 통해 사용자 화면에 추천 사유가 포함된 정책 목록을 표시한다.

3.7.6 Functional Hierarchy

본 시스템의 기능은 다음과 같은 계층 구조를 가진다.

```
1.0 사용자 계정 관리
    1.1 회원가입
        1.1.1 기본 정보 입력 (이메일, 비밀번호)
        1.1.2 상세 프로필 입력 (소득, 거주지, 관심사 등)
        1.1.3 이메일/닉네임 중복 확인
    1.2 로그인
        1.2.1 토큰 발급 (Access, Refresh)
    1.3 로그아웃

2.0 정책 탐색
    2.1 메인 (홈)
        2.1.1 최신 정책 목록 조회
        2.1.2 인기 정책 목록 조회
        2.1.3 AI 추천 바로가기
    2.2 AI 맞춤 추천
        2.2.1 자연어 프롬프트 입력
        2.2.2 추천 결과 조회 (추천 사유 포함)
    2.3 정책 필터 검색
        2.3.1 키워드 검색
        2.3.2 상세 필터 적용 (지역, 학력, 취업 상태 등)
    2.4 정책 상세 조회
        2.4.1 정책 기본 정보 조회

3.0 마이페이지 (개인화)
    3.1 내 정보 관리
        3.1.1 상세 프로필 조회
        3.1.2 상세 프로필 수정
    3.2 관심 정책 관리 (스크랩)
        3.2.1 관심 정책 추가/삭제
        3.2.2 관심 정책 목록 조회
```

3.7.7 Additional Comments

본 시스템을 추가적으로 표현하는 Unified Modeling Language(UML) 기반 Diagram 은 본 문서의 부록(Appendix) 섹션에 기술되어 있다.

4. Appendix: Analysis Models

본 섹션에서는 Unified Modeling Language(UML) 기반의 그래픽 표기법을 사용하여 시스템 모델을 설명한다. 시스템 모델은 시스템, 서브 시스템 간의 관계를 설명한다.

4.1 Context Model

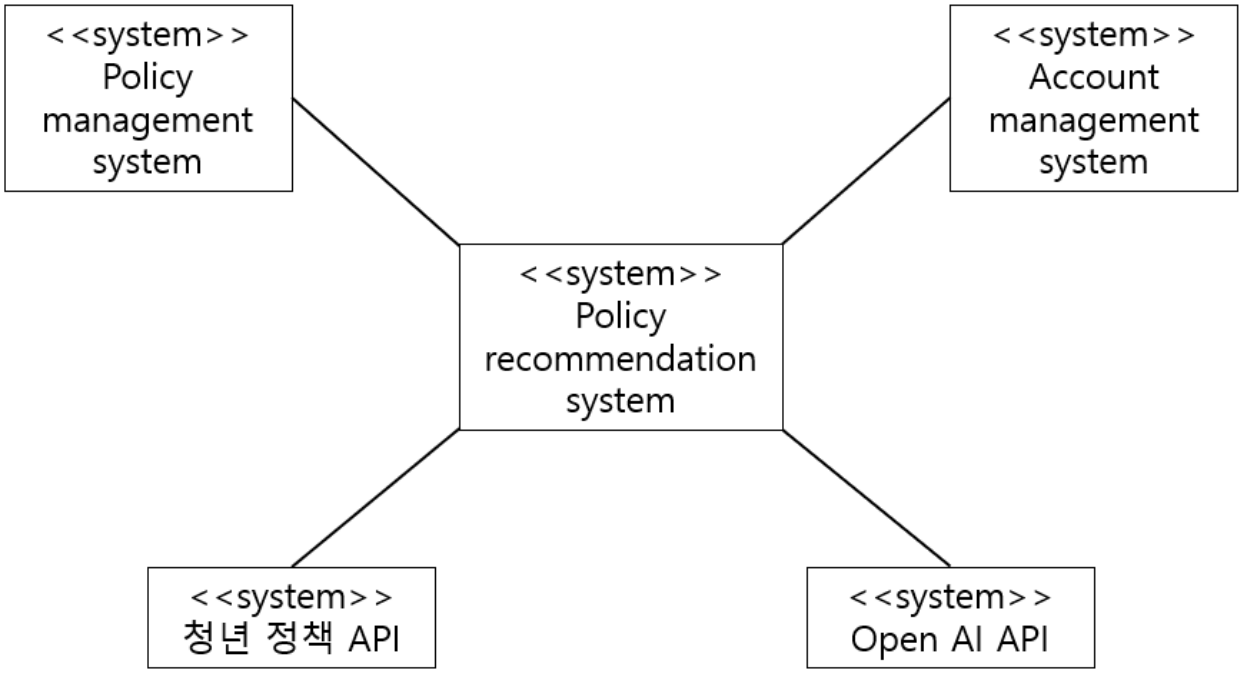


Figure 4.1: Context Model

4.2 Process Model

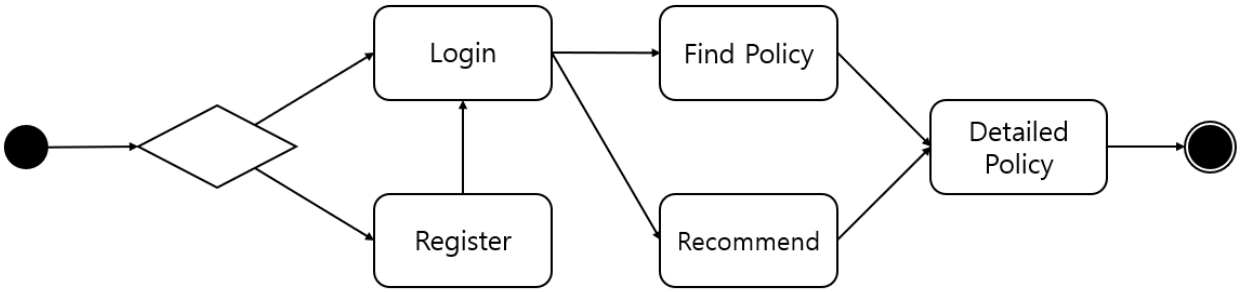


Figure 4.2: Process Model

Document History

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

Name	***Date***	***Reason For Changes***	***Version***
강민규, 김정원	2025-11-09	1. Introduction (Purpose, Scope) 2. Overall Description	V 1.00
김희수, 기민성	2025-11-09	User Interface (UI) Requirements	V 1.00
조우열	2025-11-09	Software Interface Requirements	V 1.00
강민규, 김정원	2025-11-09	3.2 Functional Requirements 3.3 Performance Requirements	V 1.00
조우열	2025-11-09	3.4 Logical Database Requirements 3.5 Design Constraints	V 1.00
강민규, 박시온	2025-11-09	3.6 Software System Attributes	V 1.00
김정원	2025-11-09	3.7 Organizing the Specific Requirements 4. Appendix	V 1.00