Software Design Specification

for

LLM 활용
청년정책 추천 플랫폼

SWE3002_41: Prof. Eun Seok Lee

Team 3: 강민규, 기민성, 김정원, 김희수, 박시온, 조우열

SungKyunKwan University

2025.11.30.

Table of Contents

Table of Figures

Table of Tables

1. Purpose

본 장에서는 문서의 예상 독자, 범위, 목적 및 구조를 설명한다.

1.1 Readership

본 문서는 본 프로젝트의 개발, 평가, 그리고 사용에 관여하는 다음의 독자들을 위해 작성되었다.

1. **시스템 개발자 (Team 3):**
    - Frontend(React), Backend(Node.js), LLM/RAG 개발자 간의 아키텍처 공유 및 구현 가이드라인으로 활용한다.
2. **프로젝트 이해관계자 (평가자):**
    - 소프트웨어공학개론 수업의 담당 교수(이은석 교수님) 및 조교, 그리고 프로젝트의 기술적 타당성을 검토하는 평가자이다.
3. **시스템 목표 사용자 (Target User Classes):**
    - **청년 사용자 (핵심 사용자):** 본인의 조건(나이, 소득, 지역 등)에 맞는 정책을 찾고자 하는 최종 사용자이다. 복잡한 행정 용어에 익숙하지 않으므로, 본 설계 문서는 이들에게 직관적인 React 기반 웹 인터페이스와 RAG 챗봇 경험을 제공하는 데 중점을 둔다.
    - **지자체/정부 담당자 (잠재적 사용자, B2G):** 향후 확장 시 플랫폼에 축적된 데이터(검색 키워드, 인기 정책 등)를 분석하여 정책 사각지대를 파악하고자 하는 관리자이다. 본 설계는 이들을 위한 데이터 리포트 생성이 가능한 DB 구조를 포함한다.

1.2 Scope

본 문서는 'LLM 활용 청년정책 추천 플랫폼'의 기술적 설계(Design)를 다룬다. 이 시스템은 웹(Web) 애플리케이션 환경을 타겟으로 하며, React 기반의 클라이언트, Node.js 기반의 API 서버, LangChain/OpenAI 기반의 RAG(Retrieval-Augmented Generation) 엔진, 그리고 MySQL

데이터베이스를 포함한다. 본 문서는 사용자의 자연어 질의(예시: "수원 사는 대학생인데...")를 처리하여 최적의 정책을 추천하는 알고리즘과 데이터 흐름, 그리고 외부 API(국무조정실, OpenAI 등)와의 연동 구조를 명세한다.

1.3 Objective

본 소프트웨어 디자인 명세서의 주요 목적은 '청년정책 추천 플랫폼'의 아키텍처 및 상세 설계를 정의하는 것이다. 본 문서는 앞서 작성된 Software Requirements Specification 의 요구사항을 기술적으로 구현하기 위한 Frontend 컴포넌트 구조, Backend API 명세, Database Schema, 그리고 AI 모듈 연동 방식을 구체화하여, 개발 단계에서의 혼선을 줄이고 시스템의 신뢰성과 유지보수성을 확보하는 데 있다.

1.4 Document Structure

1. **Purpose**: 문서의 목적, 독자, 범위를 설명한다.
2. **Introduction**: 설계에 사용된 도구, 다이어그램 정의 및 참고 문헌을 기술한다.
3. **System Architecture - Overall**: 시스템의 전체적인 구조를 Context Diagram, Use Case Diagram 등을 통해 거시적으로 조망한다.
4. **System Architecture - Frontend**: React 기반 클라이언트의 컴포넌트 및 클래스 구조를 설명한다.
5. **System Architecture - Backend**: Node.js 서버와 Python AI 엔진의 구조 및 상호작용을 설명한다.
6. **Protocol Design**: 클라이언트-서버, 서버-AI 엔진, 서버-DB 간의 통신 프로토콜을 정의한다.
7. **Database Design**: ER Diagram 및 Relational Schema 를 통해 데이터 구조를 기술한다.
8. **Testing Plan**: 시스템의 기능 및 성능 검증을 위한 테스트 계획을 수립한다.
9. **Development Plan**: 개발 환경, 도구, 제약 사항을 명시한다.

2. Introduction

본 장에서는 '청년정책 추천 플랫폼'의 시스템 설계에 사용된 방법론, 도구, 그리고 프로젝트 범위를 명확히 정의한다. 이는 개발 팀원 간의 아키텍처 이해도를 높이고, 시스템 구현의 일관성을 유지하기 위한 기반 자료로 활용된다.

2.1 Objectives

이 챕터의 목적은 제안된 시스템의 아키텍처와 모듈 설계를 시각화하기 위해 사용된 다이어그램 방법론과 도구를 명시하고, 프로젝트의 기술적 범위를 재확인하는 데 있다. 특히, RAG 기반의 AI 시스템과 웹 애플리케이션 아키텍처가 어떻게 통합되었는지를 설명하는 데 중점을 둔다. 이를 통해 개발자 및 이해관계자는 시스템의 구조적 특징과 데이터 흐름을 명확히 이해할 수 있다.

2.2 Applied Diagrams

본 문서에서는 시스템의 다양한 측면(기능, 동적 흐름, 정적 구조, 상호작용 등)을 명확히 표현하기 위해 다음과 같은 다이어그램을 사용한다.

2.2.1. Used Tools

본 프로젝트의 다이어그램 작성 및 설계를 위해 다음과 같은 도구가 사용되었다.

- Microsoft PowerPoint: 발표 자료 제작 프로그램 PowerPoint 를 통해 기본적인 도형이나 아이콘 도구를 사용하여 다이어그램 작성
- Mermaid Live Editor: 텍스트 코드를 기반으로 시퀀스 다이어그램, 흐름도 등을 신속하게 작성하고 버전 관리를 용이하게 하기 위해 사용
- Figma: 사용자 인터페이스(UI) 및 프로세스 흐름을 시각화하는 데 보조적으로 활용

2.2.2. Use Case Diagram

- **Type:** Behavioral Diagram
- **Purpose:** 시스템이 제공하는 기능과 사용자(Actor) 간의 상호작용을 정의
- **Overall Description:**
    - **Actors:** Unregistered User(미등록 사용자), Registered User(등록된 사용자), User(일반 사용자), System(시스템).
    - **Account Management:** Register(회원가입), Log in/out(로그인/로그아웃) 기능을 포함하며, 미등록 사용자는 가입을 통해 등록된 사용자가 됨
    - **Policy Management:** 등록된 사용자는 맞춤 정책 추천, 정책 확인, 정책 상세 조회 등 기능 수행 가능

2.2.3. Sequence Diagram

- **Type:** Interaction Diagram
- **Purpose:** 특정 기능(AI 맞춤 추천)이 실행될 때 객체 간의 메시지 흐름을 시간 순서대로 표현
- **Overall Description:**
    1. 청년(User)이 Chatbot UI 에 자연어 질의(예시. "내 조건에 맞는 지원금 찾아줘") 전송
    2. Platform Server 는 User DB 에서 사용자 프로필(User Profile Data) 조회
    3. 확보된 프로필과 질의를 바탕으로 Policy DB 에서 적합한 정책 검색
    4. 검색된 정책 문서와 질의를 결합하여 최종 답변 생성, 이를 사용자에게 출력

2.2.4. Class Diagram

- **Type:** Structural Diagram
- **Purpose:** 시스템을 구성하는 주요 클래스(객체)와 그들 간의 정적 관계(연관, 의존 등) 정의

- Overall Description:
    - **User Entities:** Youth(청년) 클래스는 User Profile 을 1:1 로 보유, Recommendation 을 생성하고 Policy 스크랩
    - **Admin Entities:** Administrator(관리자)는 시스템의 Report 를 조회(views)
    - **System Entities:** Recommendation 은 여러 Report 에 집계되며, Policy 를 제안하는 관계

2.2.5. Context Diagram

- **Type:** Analysis Diagram
- **Purpose:** 전체 시스템(Policy Recommendation System)과 외부 시스템/서브시스템 간의 인터페이스 및 경계 식별
- Overall Description:
    - **Center:** Policy Recommendation System
    - **Internal Subsystems:** Policy Management System(정책 관리) 및 Account Management System(계정 관리)과 상호작용
    - **External APIs:** 청년 정책 API 로부터 정책 데이터를 수집하고, Open AI API 를 활용하여 지능형 추천 로직 수행

2.2.6. Process Model

- **Type:** Activity/Process Diagram
- **Purpose:** 사용자의 서비스 이용 흐름(Workflow)을 순차적으로 시각화
- Overall Description:
    - **Start:** 서비스 접속 후 분기(Decision) 발생

- **Auth Flow**: 계정이 없으면 Register 과정을 거쳐 Login 을 수행하고, 계정이 있으면 바로 Login
- **Main Flow**: 로그인 후 Find Policy(정책 찾기) 단계로 진입
- **Action**: 사용자는 Recommend(추천 받기)를 통해 맞춤 정책을 확인하거나, 검색된 정책 목록에서 Detailed Policy(상세 정보)를 조회 후, 프로세스 종료

2.2.7. Project Scope

본 프로젝트는 대한민국 청년들이 자신의 상황(나이, 소득, 거주지, 관심사 등)에 맞는 정책을 쉽고 정확하게 찾을 수 있도록 돕는 'LLM 활용 지능형 청년정책 추천 플랫폼'이다.

- Core Scope:
  - 사용자 프로필 기반의 정밀 필터링 및 RAG 기술을 활용한 맞춤형 정책 추천
  - 자연어 처리를 통한 직관적인 질의응답 인터페이스 제공
  - 국무조정실 등 공공 데이터 API 와의 동기화를 통한 최신 정책 정보 제공
- Limitations:
  - 정책의 실제 신청 및 접수 처리는 해당 기관 사이트로 연결하는 것을 원칙으로 하며, 본 플랫폼 내에서 직접 처리하지 않는다.

2.2.8. References

1. *IEEE Std 1016-2009: IEEE Standard for Information Technology—Systems Design—Software Design Descriptions.*
2. *Software Requirements Specification (SRS): Team 3 - LLM 활용 청년정책 추천 플랫폼 요구사항 명세서 (v1.00).*
3. *Project Proposal: Team 3 - LLM 활용 청년정책 추천 플랫폼 제안서.*
4. *External API Docs: 국무조정실 청년정책 API 가이드, OpenAI API Reference.*
5. *GitHub Repository: Team 3 Source Code & Documentation Archive (https://github.com/skkuse/2025fall_41class_team3).*

3. System Architecture - Overall

3.1 Objectives

이 챕터에서는 '청년정책 추천 플랫폼'의 프론트엔드 설계에서 백엔드 설계에 이르는 프로젝트 애플리케이션의 전체적인 시스템 구성을 설명한다. 사용자 인터페이스(View)와 비즈니스 로직(Controller), 그리고 데이터 저장소(Model)가 어떻게 유기적으로 연결되어 '맞춤형 청년정책 추천'이라는 핵심 가치를 전달하는지, 그 상호작용의 흐름을 기술하는 것을 목적으로 한다. React Client, Node.js, LLM/RAG 서브시스템, 그리고 데이터베이스(MySQL) 등 유기적으로 연결되어 서비스를 제공하는 전체적인 구조와 흐름을 설명한다.

3.2 System Organization

본 서비스는 MVC (Model-View-Controller) 아키텍처 패턴을 기반으로 설계되었으며, 사용자의 요청을 효율적으로 처리하고 데이터의 일관성을 유지한다. 시스템은 크게 사용자 인터페이스(View), 추천 엔진 및 제어 로직(Controller), 그리고 데이터베이스(Model)로 구성된다. 다음 다이어그램은 시스템의 전체적인 구성과 데이터 흐름을 시각적으로 나타낸다.

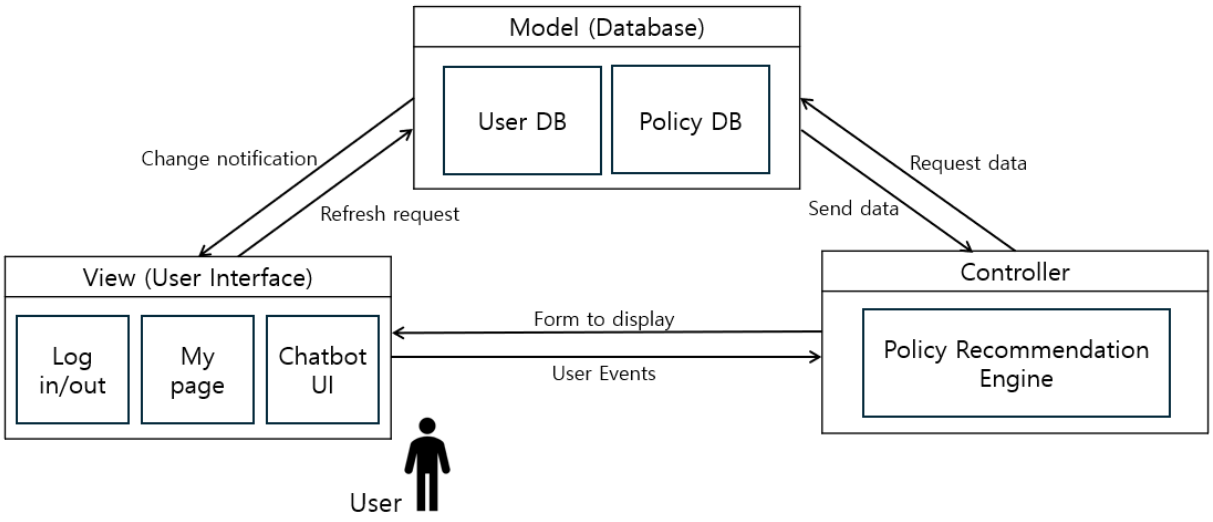

Figure 3.1: Overall System Architecture (MVC)

3.3 Use Case Diagram

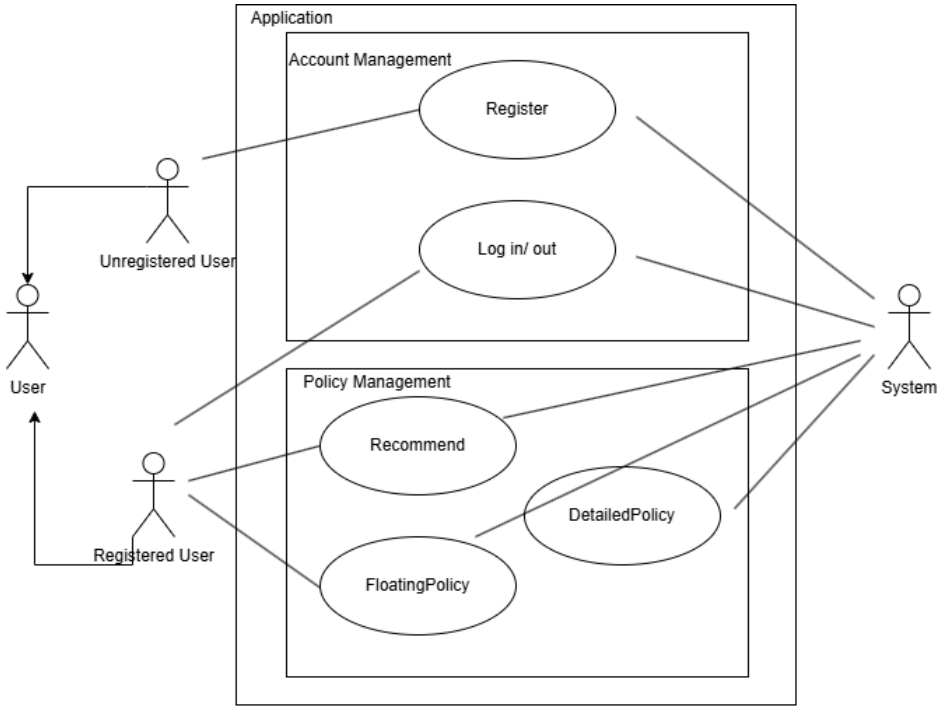

Figure 3.2: Use Case Diagram

3.4 Sequence Diagram

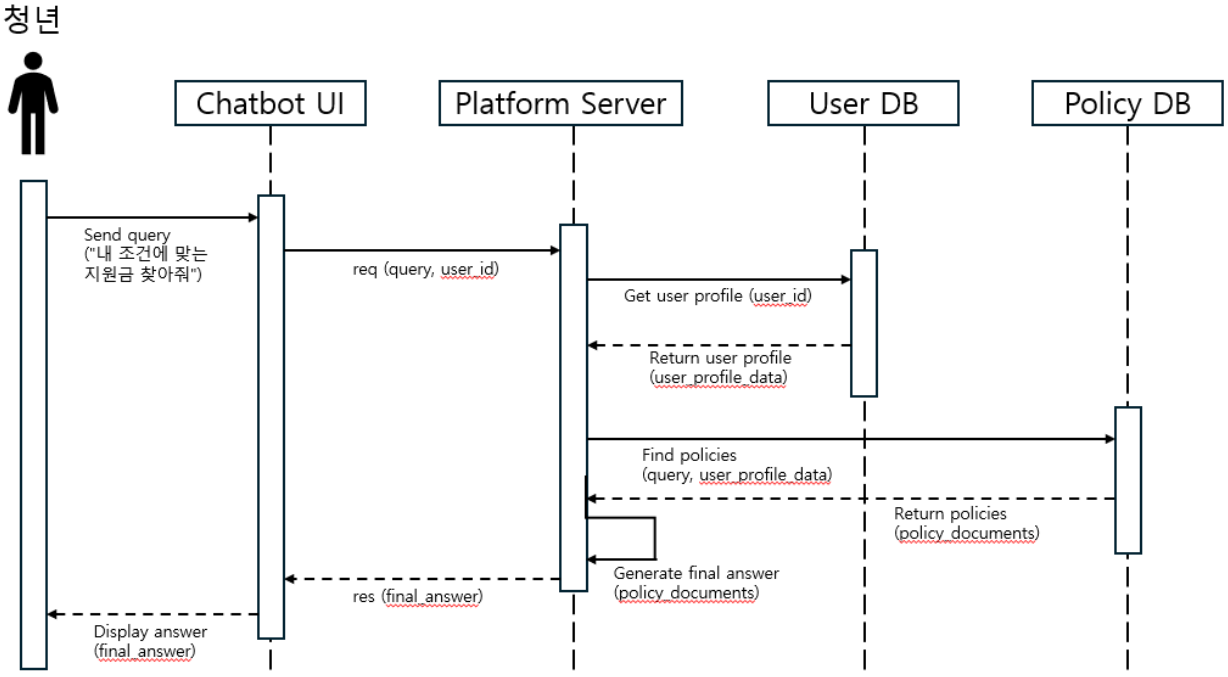

Figure 3.3: Sequence Diagram

3.5 Class Diagram

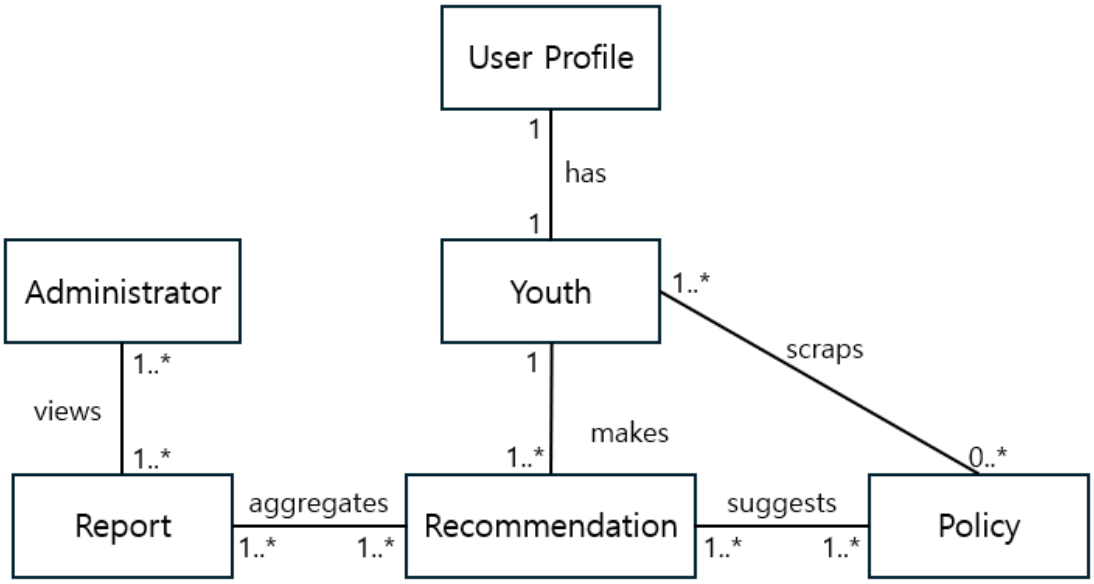

Figure 3.4: Class Diagram

3.6 Context Diagram

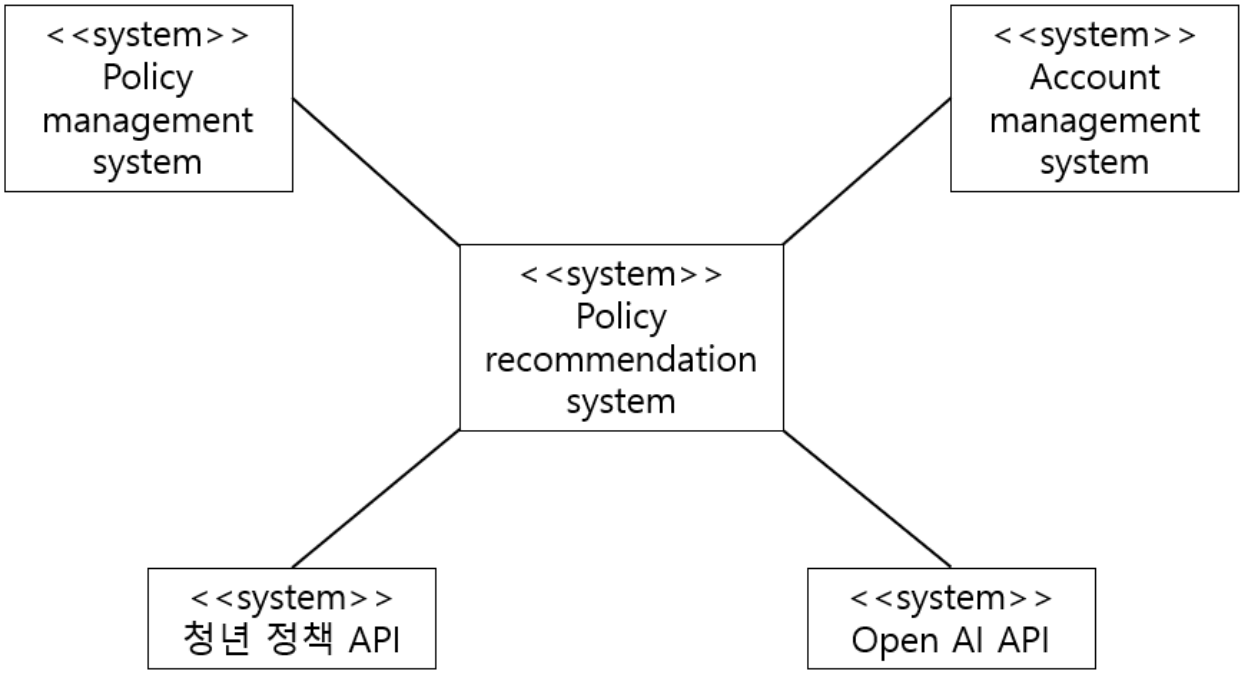

Figure 3.5: Context Diagram

3.7 Process Model

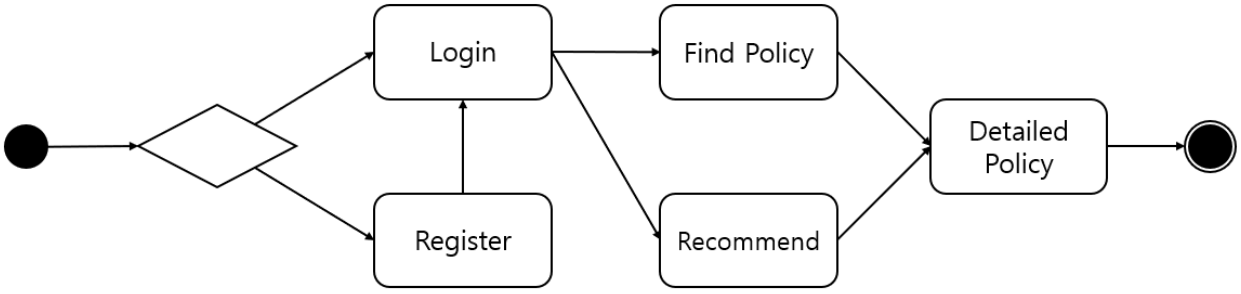

Figure 3.6: Process Model

4. System Architecture - Frontend

4.1 Objectives

이 챕터에서는 프론트엔드 시스템의 구조, 속성 및 기능을 설명하고, 청년정책 추천 플랫폼(React Application) 내 각 구성 요소(Component/Module)의 관계를 설명한다. 특히 사용자 경험(UX)의 핵심인 '개인화 프로필 관리'와 'RAG 기반 정책 추천' 프로세스를 중심으로 설계를 기술한다. React 기반의 SPA 구조를 따르며, MVVM(Model-View-ViewModel) 패턴을 차용하여 UI 화면 설계와 비즈니스 로직(API 호출 및 상태 관리)을 분리하는 것을 목표로 한다.

4.2 Components

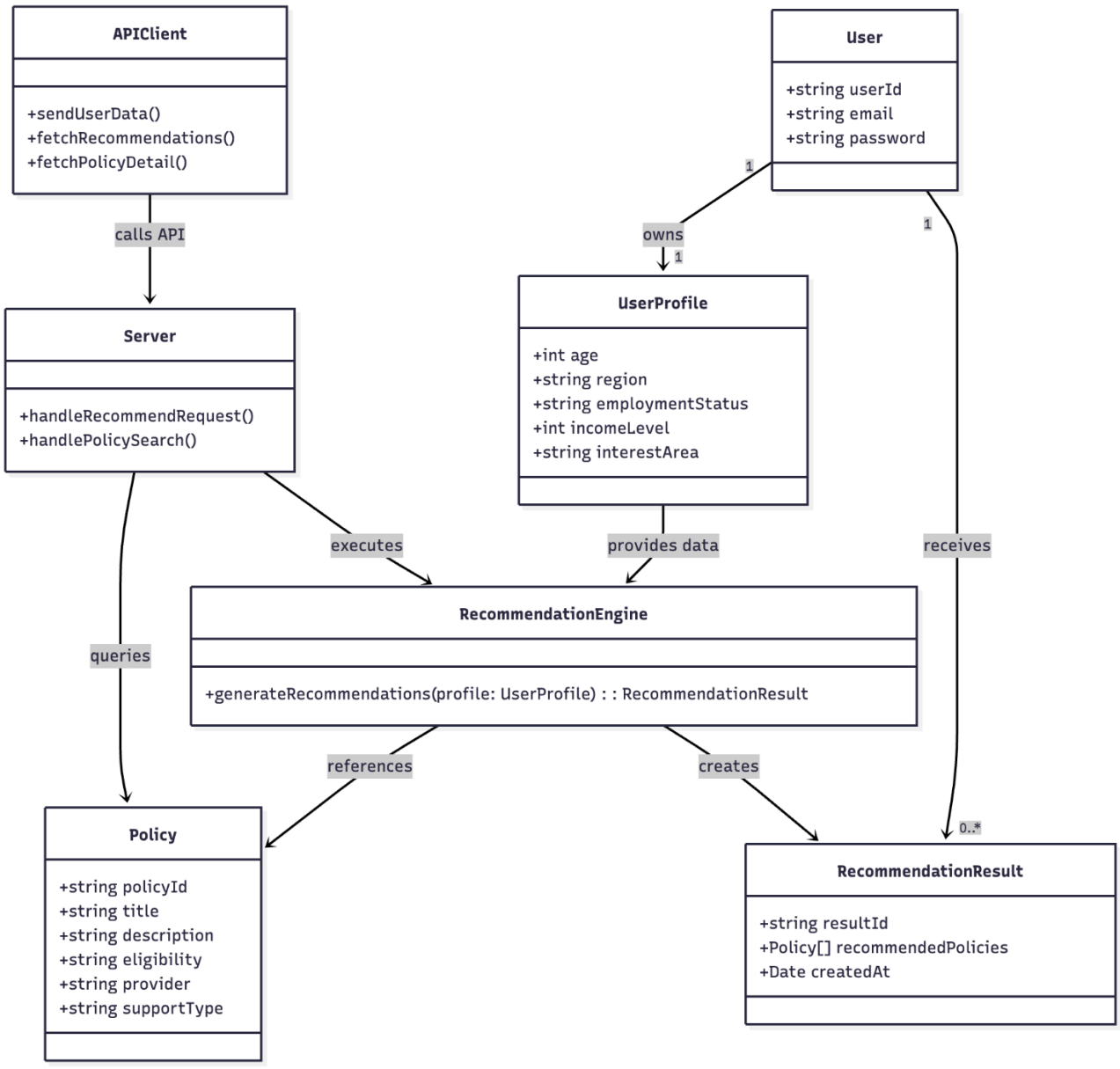

Figure 4.1: Service Class Diagram

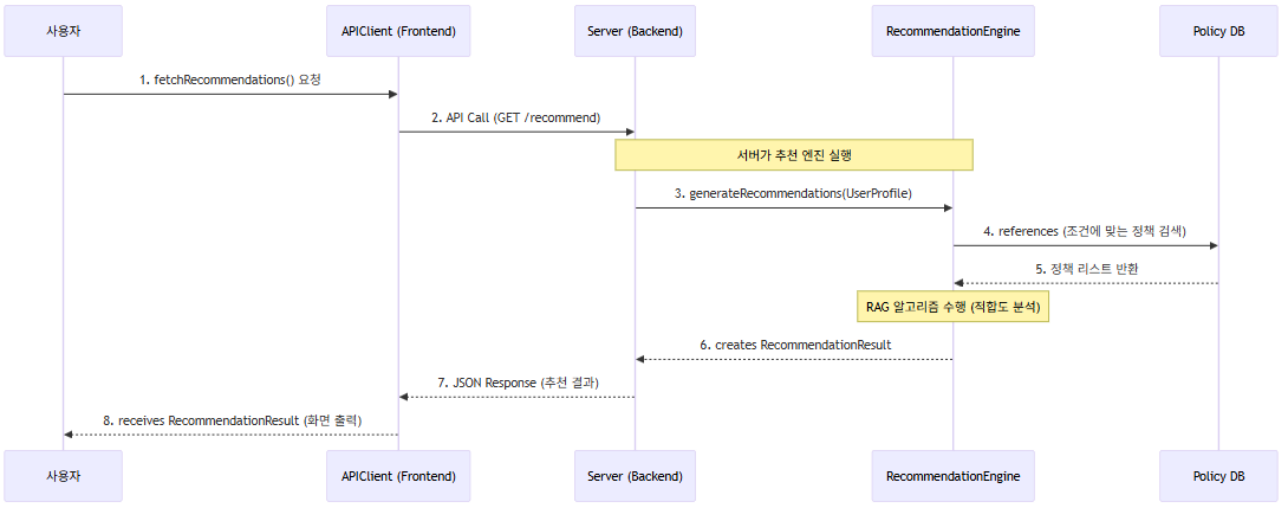

Figure 4.2: Service Sequence Diagram

4.2.1. User Profile Management

사용자의 로그인 정보와 정책 추천을 위한 민감 정보를 분리하여 관리하는 프로필 클래스 설계이다. User 는 계정 인증을 담당하고, UserProfile 은 추천 알고리즘의 입력 데이터로 사용된다.

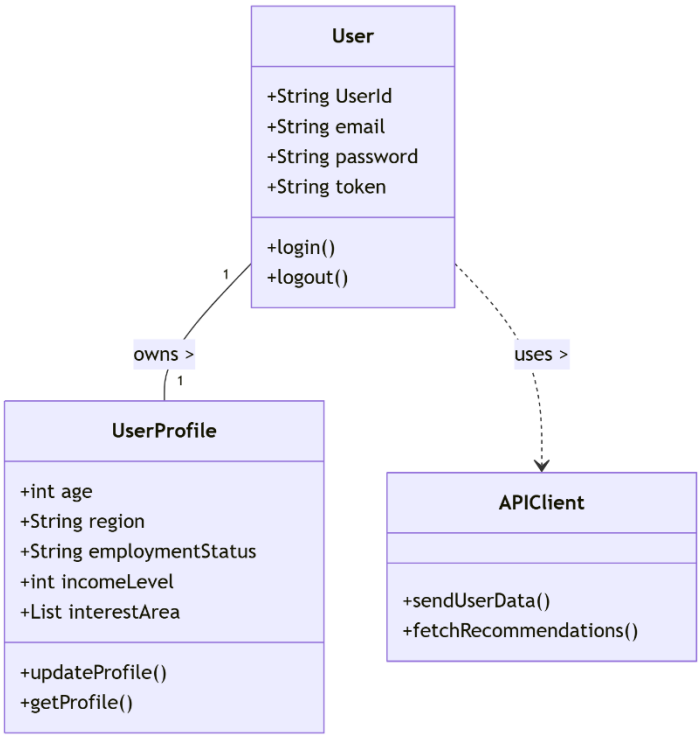

Figure 4.3: Class Diagram - User & Profile Structure

<Attributes>

- User (Account):
    - userId: 사용자 ID
    - email: 사용자 이메일
    - password: 암호화된 비밀번호
    - token: JWT 인증 토큰 (Access/Refresh)
- UserProfile (Survey Data):
    - age: 나이
    - region: 거주 지역 (시/도, 시/군/구)
    - employmentStatus: 취업 상태 (재직, 미취업 등)
    - incomeLevel: 소득 구간
    - interestArea: 관심 정책 분야 (배열)

<Methods>

- login(email, password): 서버 인증 요청 및 토큰 저장
- updateProfile(profileData): 사용자 상세 정보 수정
- getProfile(): 마이페이지 진입 시 상세 정보 로드

4.2.2. Policy Recommendation (RAG Engine Interaction)

사용자가 자연어 질문(Prompt)을 입력하면, UserProfile 정보와 결합하여 서버의 추천 엔진(Recommendation Engine)에 요청을 보내고, 결과를 받아 화면에 표시하는 구조이다. 사용자는 여러 번 추천을 받을 수 있다. Recommendation Engine 에 UserProfile 을 입력하면, Policy 리스트에서 적합한 정책을 추천한다.

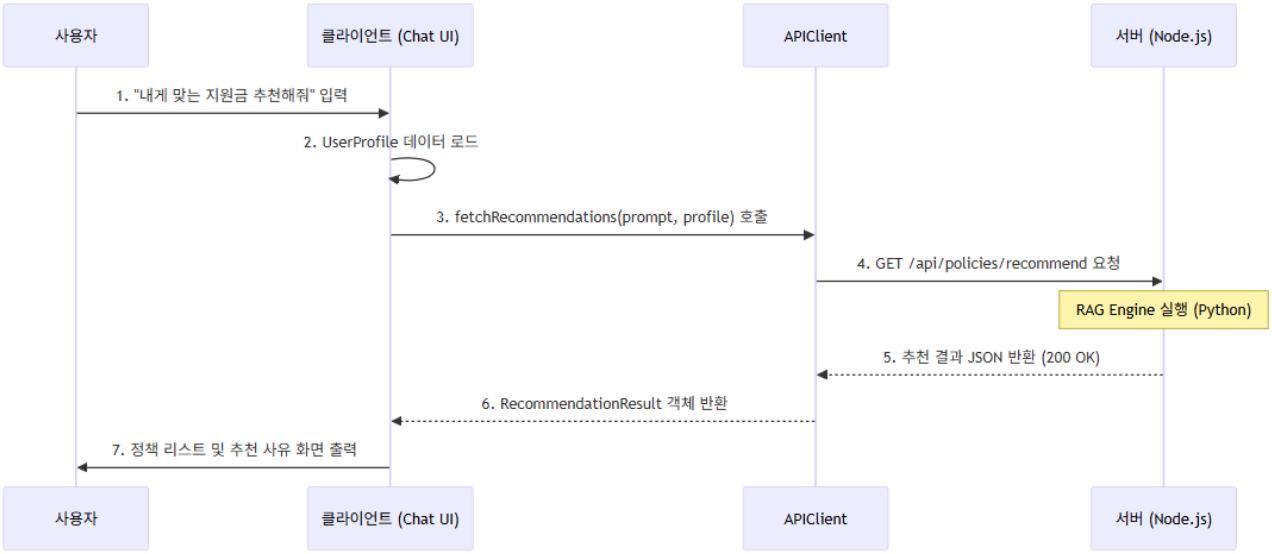

Figure 4.4: Sequence Diagram - Policy Recommendation

Policy

- Description: 실제 정책 정보를 담고 있는 객체로, 사용자가 조회하는 최종 콘텐츠
- Attributes:
    - policyId: 정책 고유 ID
    - title: 정책명
    - description: 정책 요약
    - eligibility: 신청 자격 요건
    - provider: 정책 주관 기관
    - supportType: 지원 유형 (예: "현금지원")

RecommendationResult

- Description: 특정 시점에 요청된 추천 결과를 저장하는 클래스
- Attributes:
    - resultId: 추천 결과 식별자
    - recommendedPolicies: 추천된 Policy 객체의 배열
    - createdAt: 생성 일시

4.2.3. API Client Structure

프론트엔드와 백엔드 간의 통신을 전담하는 모듈로, 모든 HTTP 요청을 캡슐화하여 관리한다.

APIClient:

- 모든 API 호출의 진입점
- Base URL 및 Header(Authorization) 설정 관리

Methods

- **sendUserData(userData)**: 회원가입 및 정보 수정
- **fetchRecommendations(userId, prompt)**: 정책 추천 요청
- **fetchPolicyDetail(policyId)**: 정책 상세 정보 요청

5. System Architecture - Backend

5.1 Objectives

이 챕터에서는 백엔드 시스템 아키텍처를 정의한다. 백엔드 시스템은 클라이언트(Next.js)의 요청을 처리하고, 데이터의 영속성을 보장한다. 또한, Python 기반의 AI 엔진을 제어하여 사용자에게 맞춤형 정책 정보를 제공하는 것을 목표로 한다. 주요 설계 목표는 다음과 같다.

- **Hybrid Runtime Orchestration**: 메인 웹 서버인 Node.js 가 Python 스크립트를 자식 프로세스(Child Process)로 실행하여, 웹 서비스의 반응성과 AI 모델의 연산 능력을 동시에 확보한다.
- **Stateless Authentication**: JWT 기반의 인증 방식을 채택하고, Access Token 만료 시 Refresh Token 을 통한 갱신 메커니즘을 지원하여 보안성과 사용자 편의성을 극대화한다.
- **Data Integrity & Synchronization**: 외부 국무조정실 API 와 주기적으로 동기화를 수행하여 정책 데이터의 최신성을 유지한다.

5.2 Overall Architecture

시스템은 Layered Architecture 를 기반으로 하며, 프론트엔드 요청을 중계하는 Proxy Layer, 비즈니스 로직을 처리하는 Application Layer, 그리고 AI 연산을 담당하는 Worker Layer 로 구성된다.

5.2.1. System Diagram

다음 다이어그램은 Next.js(Client Proxy)에서 시작하여 Node.js 서버, Python 서브시스템, 데이터베이스로 이어지는 전체적인 데이터 흐름을 나타낸다.

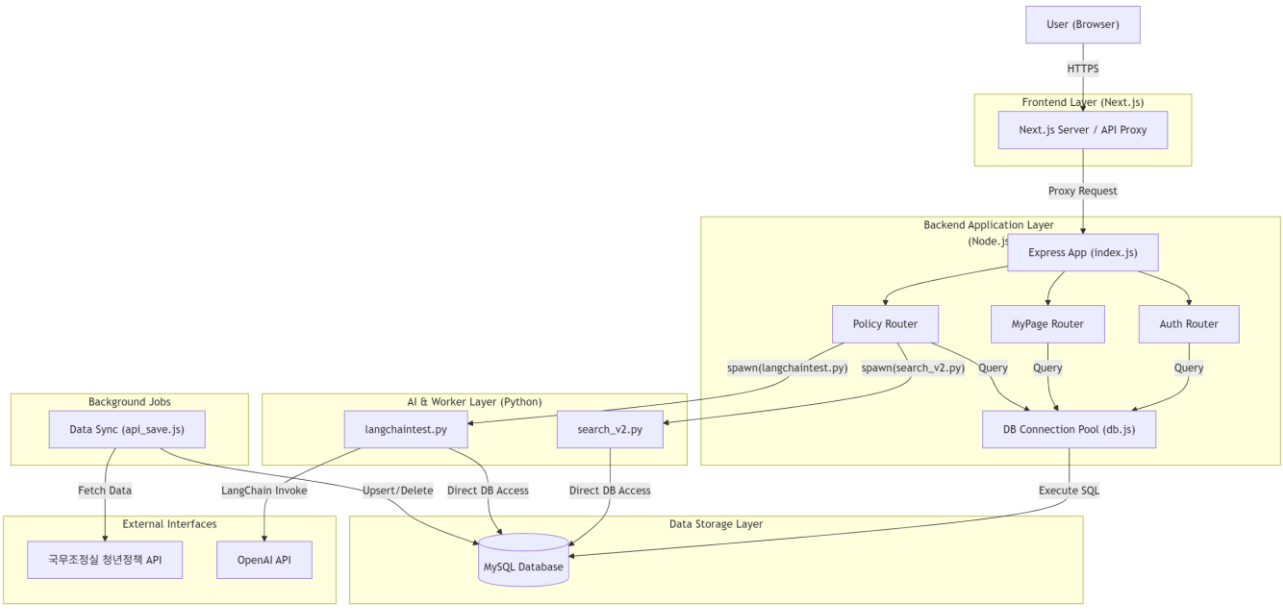

Figure 5.1: System Diagram

5.3 Subcomponents

백엔드 시스템은 기능적 역할에 따라 User Management, Policy Management, Recommendation System 의 세 가지 핵심 서브 컴포넌트로 분류된다.

5.3.1. User Management System

사용자의 회원가입, 로그인, 토큰 관리 및 프로필 정보(JSON 필드 포함)를 처리한다. 로그인 성공 시 Access Token 과 Refresh Token 을 발급하고, DB 에 Refresh Token 을 저장한다.

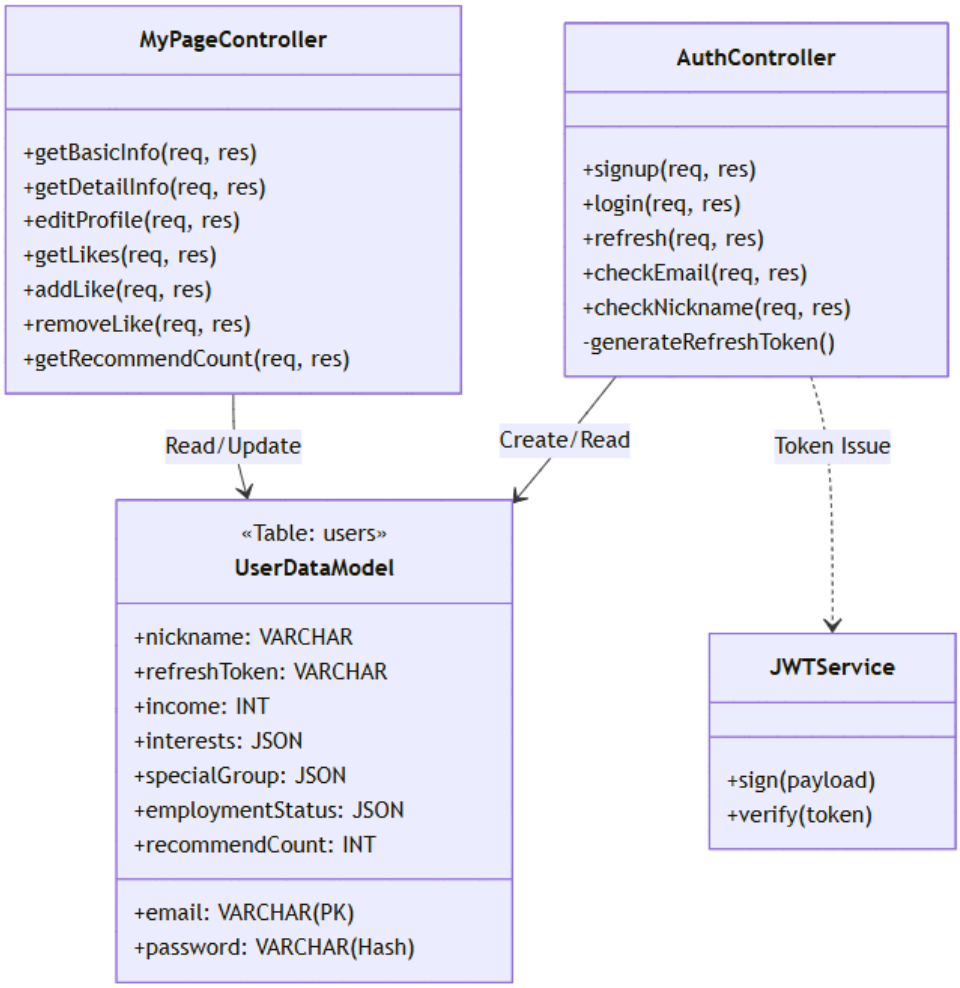

Figure 5.2: Class Diagram - User Management System

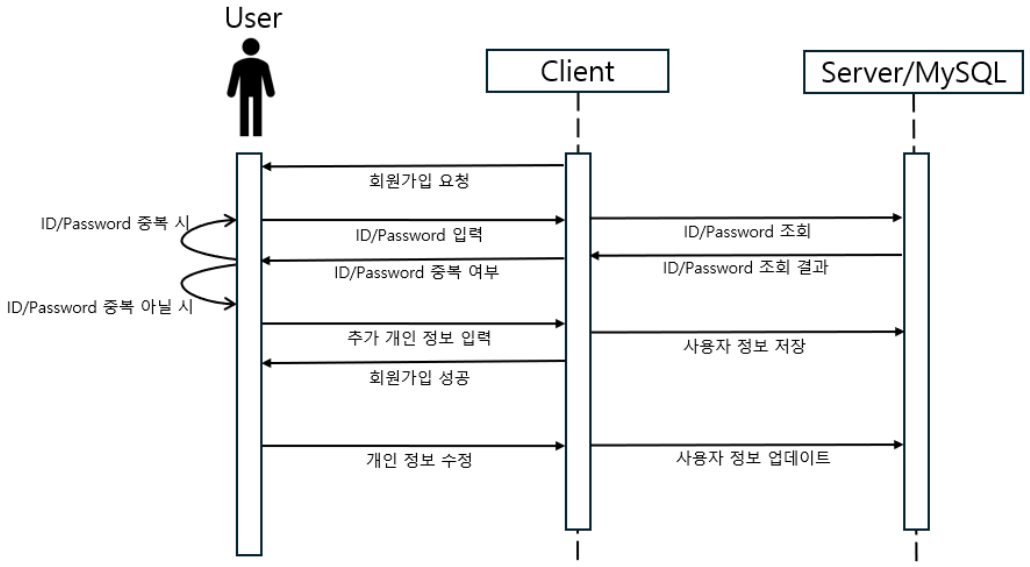

Figure 5.3: Sequence Diagram - User Profile/Register

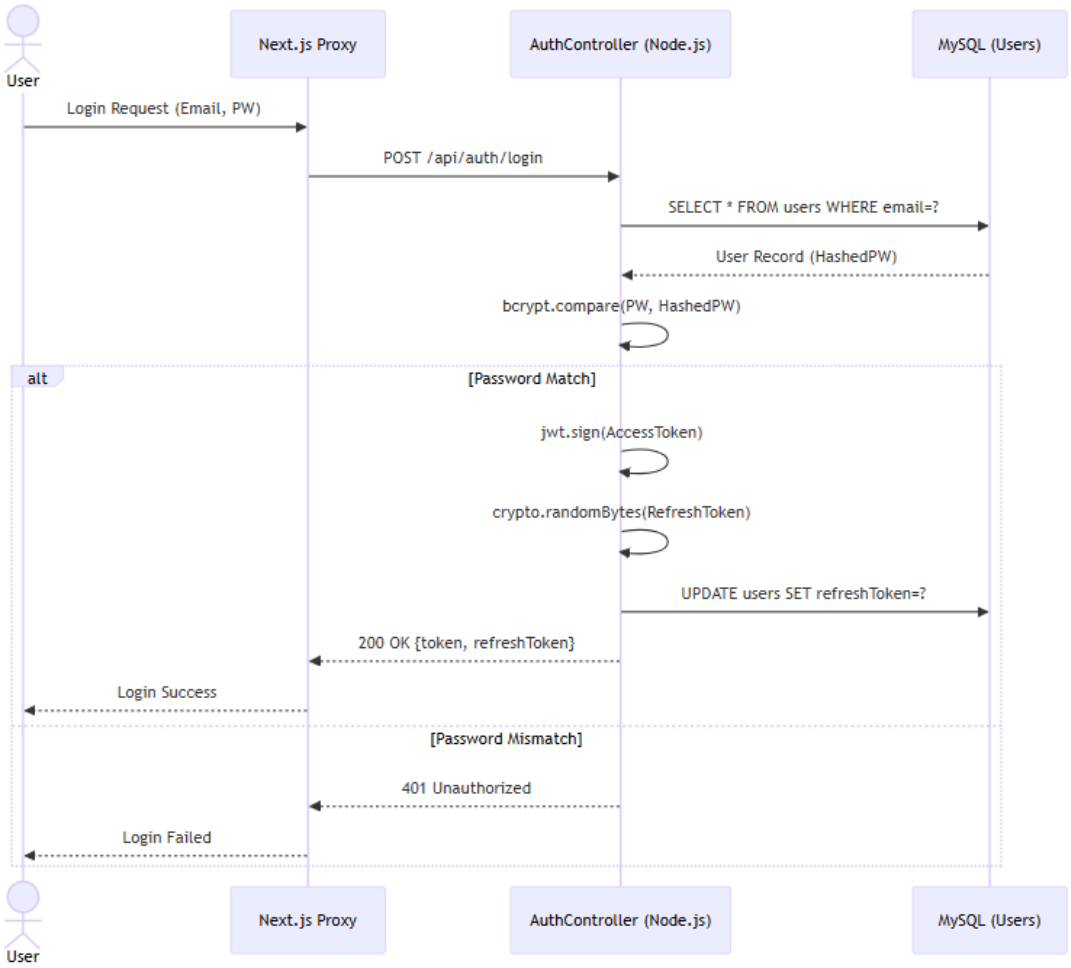

Figure 5.4: Sequence Diagram - User Management System

5.3.2. Policy Management & Search System

정책 데이터의 수집과 조회, 그리고 Python 스크립트를 이용한 필터링 검색을 담당한다.

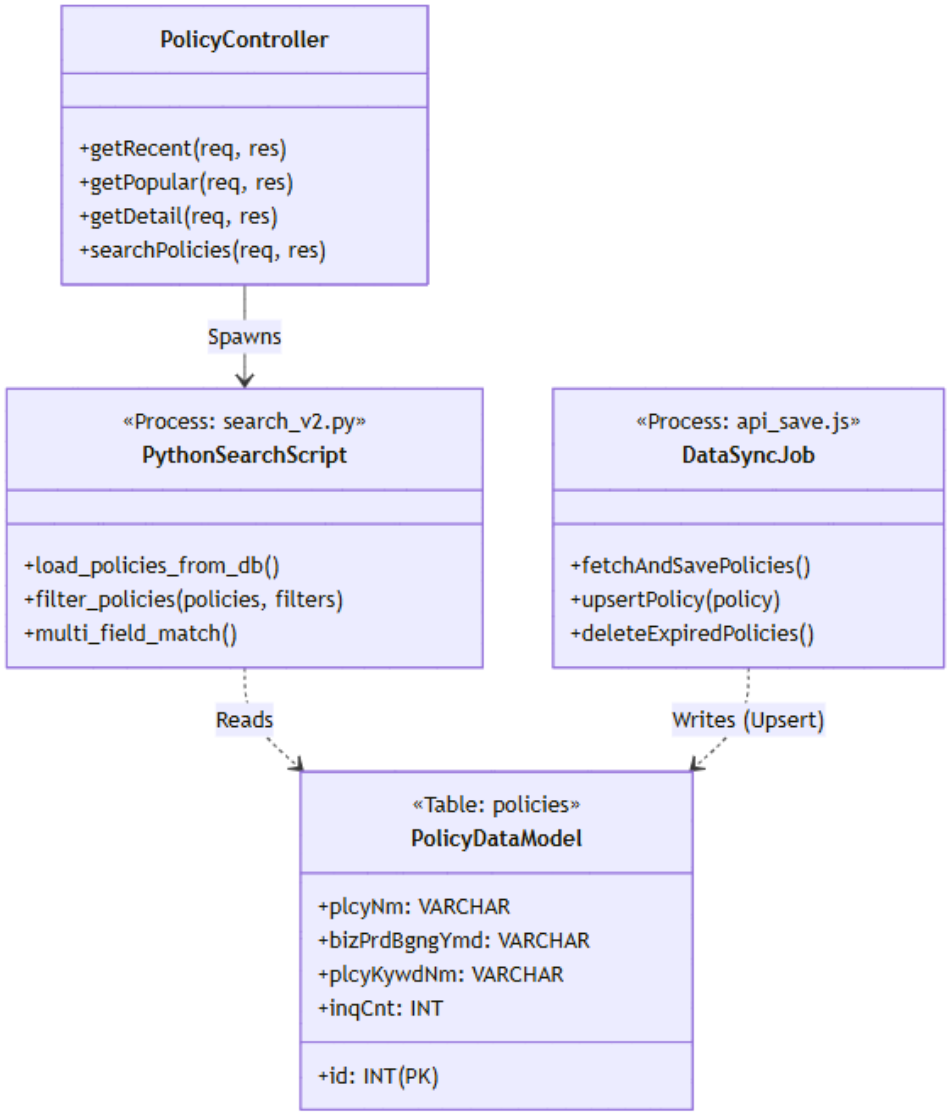

Figure 5.5: Class Diagram - Policy Management & Search System

5.3.3. Recommendation System

사용자의 프로필과 자연어 프롬프트를 기반으로 LangChain RAG 파이프라인을 실행하여 맞춤형 정책을 추천한다. 아래 Sequence Diagram 은 Node.js 가 Python 프로세스를 생성하고, Python 내부에서 RAG 로직(DB 검색 -> LLM 질의)이 수행되는 과정을 상세히 나타낸다.

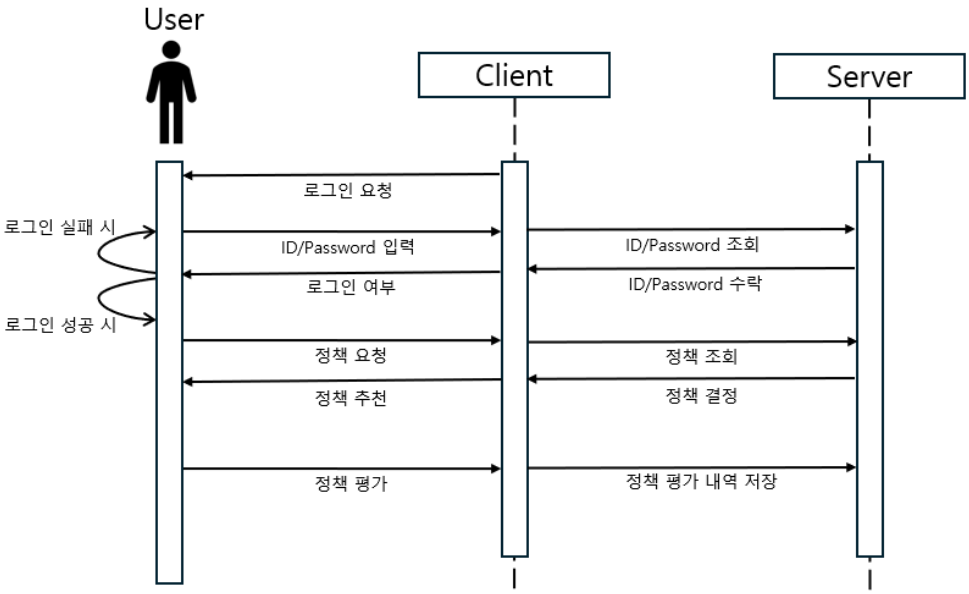

Figure 5.6: Sequence Diagram – login/Policy Recommendation

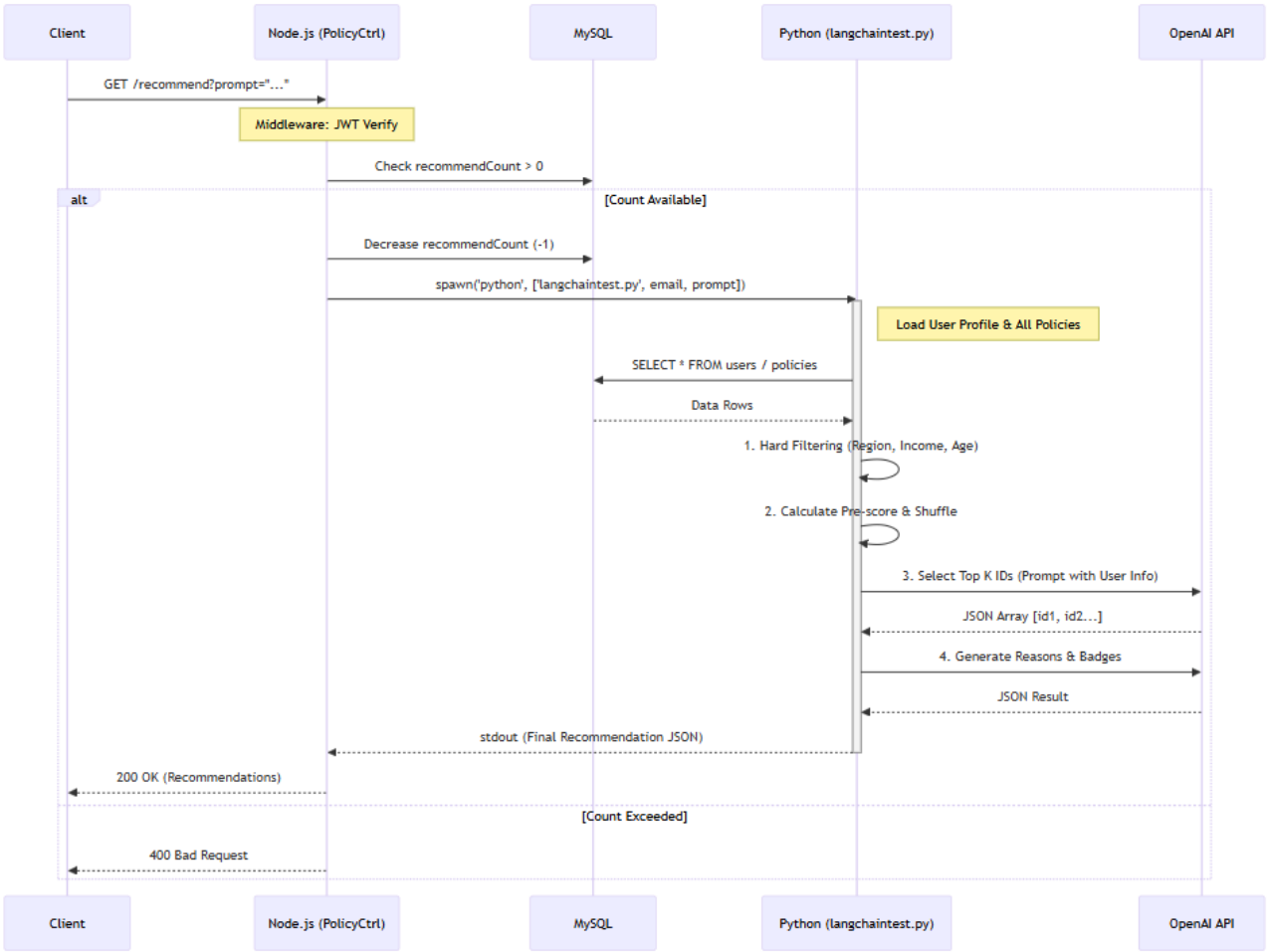

Figure 5.7: Sequence Diagram – RAG Recommendation Pipeline

6. Protocol Design

6.1 Objectives

이 챕터에서는 웹 클라이언트(React Client)와 백엔드, 그리고 외부 시스템 간의 통신 프로토콜을 정의한다. 모든 데이터 교환은 HTTP/1.1 기반의 RESTful API 를 따르며, 데이터 포맷은 JSON 을 표준으로 사용한다. 데이터 포맷, 인증 방식, 그리고 각 기능별 요청/응답 명세 등을 상세히 기술하여 인터페이스의 일관성을 보장하는 것을 목표로 한다.

6.2 RESTful API & JSON

본 시스템은 자원(Resource)을 이름(URI)으로 구분하고, 자원의 상태(State)를 주고받는 RESTful 아키텍처를 따른다.

- **Request:** 클라이언트는 GET, POST, PUT, DELETE 등의 표준 HTTP 메서드를 사용하여 서버에 요청을 보낸다.
- **Response:** 서버는 요청 처리 결과를 JSON 형식으로 반환하여, 프론트엔드(React)에서 유연하게 처리할 수 있도록 한다.

6.3 TLS and HTTPS

사용자의 개인정보(비밀번호, 소득, 거주지 등) 보호를 위해, 클라이언트와 서버 간의 모든 통신은 TLS(Transport Layer Security) 프로토콜을 통해 암호화된 HTTPS 채널 위에서 이루어진다. 이를 통해 데이터 도청 및 변조를 방지한다.

6.4 JWT Authentication

본 시스템은 Stateless 한 인증 처리를 위해 JWT(JSON Web Token) 방식을 사용한다.

- **Access Token:** API 요청 시 HTTP Header 의 Authorization 필드에 포함하여 전송 (유효기간: 1 시간)
- **Refresh Token:** Access Token 만료 시 새로운 토큰을 발급받기 위해 사용되며, 데이터베이스에 저장되어 관리

6.5 Authentication Protocols

6.5.1. Sign Up

사용자의 계정을 생성하고 초기 프로필 정보를 등록한다.

Request

Attribute	Detail
Protocol	HTTPS
Method	POST (/api/auth/signup)
Header	Content-Type: application/json
Request Body	email: 사용자 이메일 (ID) password: 비밀번호 nickname: 닉네임 birthDate: 생년월일 (YYYY-MM-DD) location: 거주 지역 income: 소득 정보 interests: 관심 분야 (Array)

Response

Attribute	Detail
Protocol	HTTPS
Success Code	201 Created
Failure Code	400 Bad Request (필수 값 누락) 409 Conflict (이메일 중복)
Response Body	status: "success" token: Access Token refreshToken: Refresh Token nickname: 사용자 닉네임

Table 6.1: Request-Response: Sign Up

6.5.2. Log In

이메일과 비밀번호를 검증하고 인증 토큰을 발급한다.

Request

Attribute	Detail
Protocol	HTTPS
Method	POST (/api/auth/login)
Request Body	email: 사용자 이메일 password: 비밀번호

Response

Attribute	Detail
Protocol	HTTPS

Success Code	200 OK
Failure Code	401 Unauthorized (비밀번호 불일치 또는 사용자 없음)
Response Body	token: Access Token refreshToken: Refresh Token expires_in: 토큰 만료 시간 (초) nickname: 사용자 닉네임

Table 6.2: Request-Response: Log In

6.5.3. Token Refresh

만료된 Access Token 을 Refresh Token 을 사용하여 갱신한다.

Request

Attribute	Detail
Protocol	HTTPS
Method	POST (/api/auth/refresh)
Request Body	refreshToken: 서버로부터 발급받은 리프레시 토큰

Response

Attribute	Detail
Protocol	HTTPS
Success Code	200 OK
Failure Code	401 Unauthorized (유효하지 않은 리프레시 토큰)
Response Body	token: 새로운 Access Token refreshToken: 새로운 Refresh Token (Rotation 적용)

Table 6.3: Request-Response: Token Refresh

6.6 Policy Protocols

6.6.1. AI Policy Recommendation

사용자의 프로필과 자연어 프롬프트를 기반으로 AI가 추천한 정책 목록을 반환한다.

Request

Attribute	Detail
Protocol	HTTPS
Method	GET (/api/policies/recommend)
Header	Authorization: Bearer <Access Token>
Query Param	prompt: 사용자 입력 자연어 요구사항 (예: "취업 준비 지원금 찾아줘")

Response

Attribute	Detail
Protocol	HTTPS
Success Code	200 OK
Failure Code	400 Bad Request (일일 추천 횟수 초과) 500 Internal Server Error (AI 모델 오류)
Response Body	recommendations: 추천 정책 배열 [{ id: 정책 ID, plcyNm: 정책명, reason: AI 추천 사유, badges: 핵심 키워드 배지 (Array)

	}]

Table 6.4: Request-Response: AI Policy Recommendation

6.6.2. Policy Search

다양한 필터 조건을 적용하여 정책을 검색한다.

Request

Attribute	Detail
Protocol	HTTPS
Method	GET (/api/policies/search)
Query Params	q: 검색 키워드 sido: 지역 필터 employmentStatus: 취업 상태 education: 학력 interests: 관심 분야 (콤마로 구분)

Response

Attribute	Detail
Protocol	HTTPS
Success Code	200 OK
Response Body	Array: 검색된 정책 객체 배열 [{ id: 정책 ID,

	plcyNm: 정책명 }, …]

Table 6.5: Request-Response: Policy Search

6.6.3. Policy Detail

특정 정책의 상세 정보를 조회한다.

Request

Attribute	Detail
Protocol	HTTPS
Method	GET (/api/policies/{id})
Header	Authorization: Bearer <Access Token>
Path Variable	id: 조회할 정책의 고유 ID

Response

Attribute	Detail
Protocol	HTTPS
Success Code	200 OK
Response Body	plcyNm: 정책명 plcyExplnCn: 정책 설명 plcySprtCn: 지원 내용 aplyUrlAddr: 신청 링크 bizPrdBgngYmd: 사업 시작일 bizPrdEndYmd: 사업 종료일

Table 6.6: Request-Response: Policy Detail

6.7 My Page Protocols

6.7.1. Get User Profile

사용자의 상세 프로필 정보를 조회하여 마이페이지에 표시한다.

Request

Attribute	Detail
Protocol	HTTPS
Method	GET (/api/mypage/detail)
Header	Authorization: Bearer <Access Token>

Response

Attribute	Detail
Protocol	HTTPS
Success Code	200 OK

Response Body	email: 이메일
	nickname: 닉네임
	tags: 프로필 태그 객체
	{
	관심 분야: ["창업", "교육"],
	최종 학력: ["대학교 졸업"]
	}

Table 6.7: Request-Response: Get User Profile

7. Database Design

7.1 Objectives

이 챕터에서는 시스템의 데이터 영속성을 보장하기 위한 데이터베이스의 논리적, 물리적 구조를 정의한다. 데이터 모델링은 MySQL 데이터베이스를 기반으로 하며, 사용자의 프로필 정보, 정책 데이터, 그리고 사용자 피드백 간의 관계를 명확히 하여 데이터 무결성과 효율적인 조회를 지원하는 것을 목표로 한다.

7.2 ER Diagram

시스템의 주요 엔티티(User, Policy 등)와 이들의 상호작용을 나타내는 ER 다이어그램(ERD)이다.

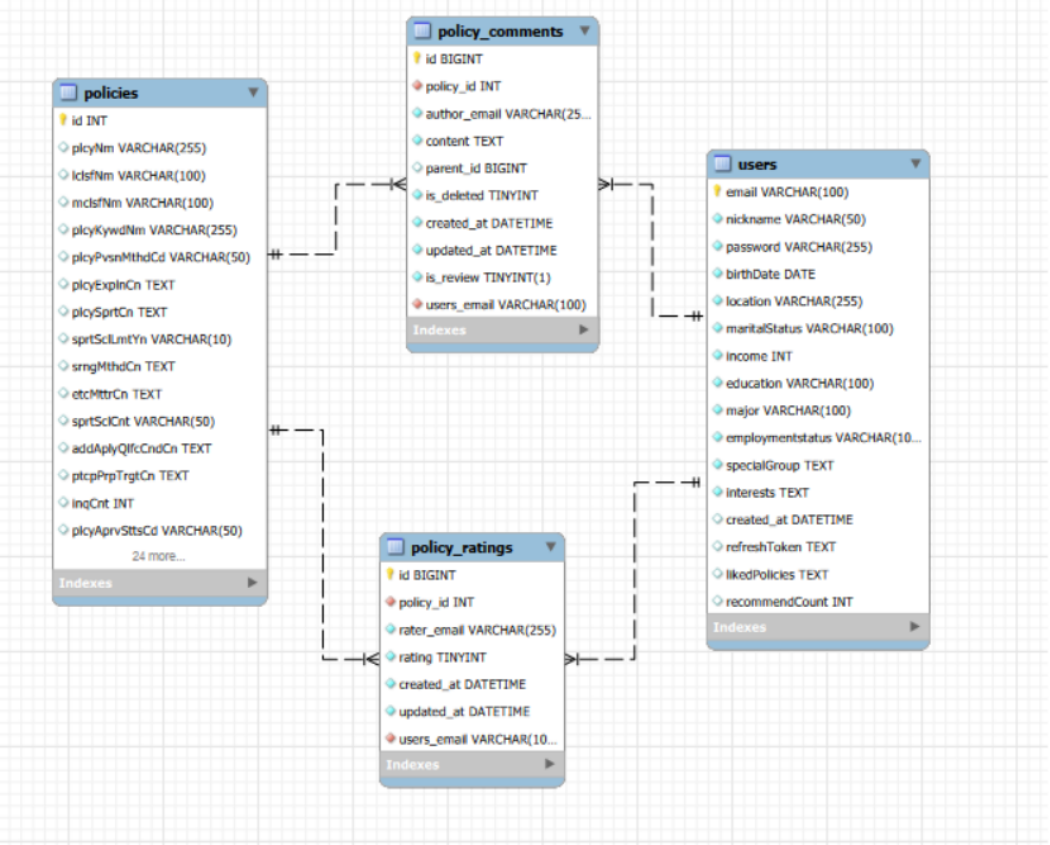

Figure 7.1: ER diagram - Schema

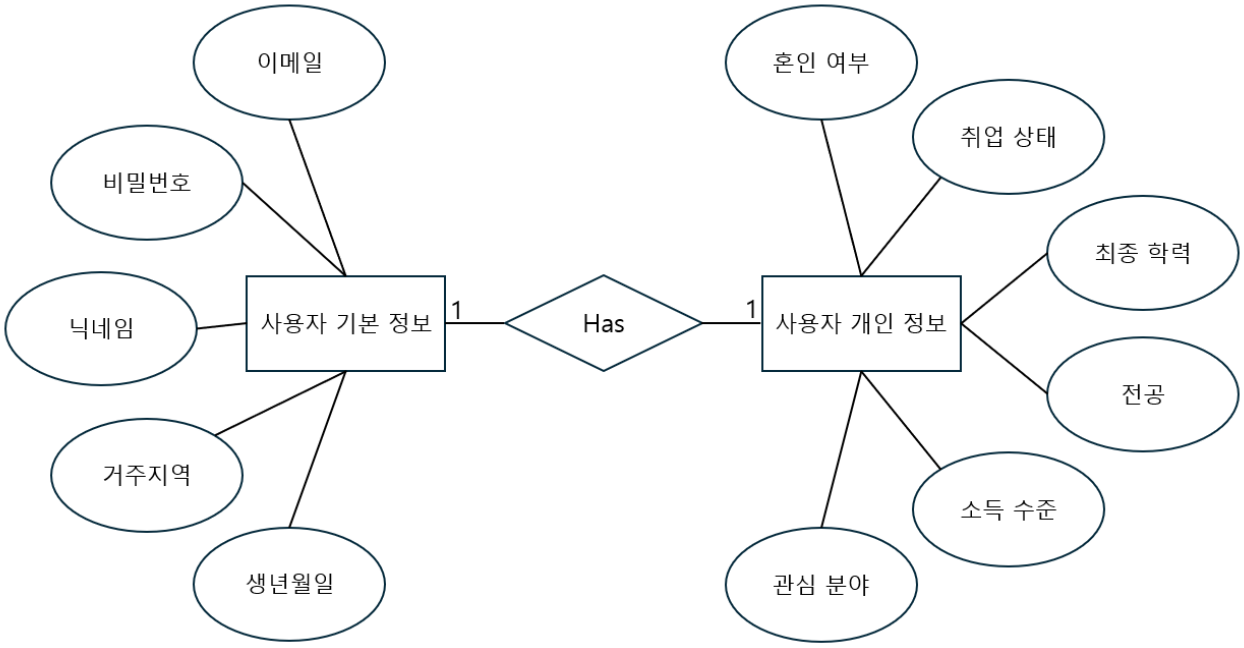

Figure 7.2: ER diagram - User

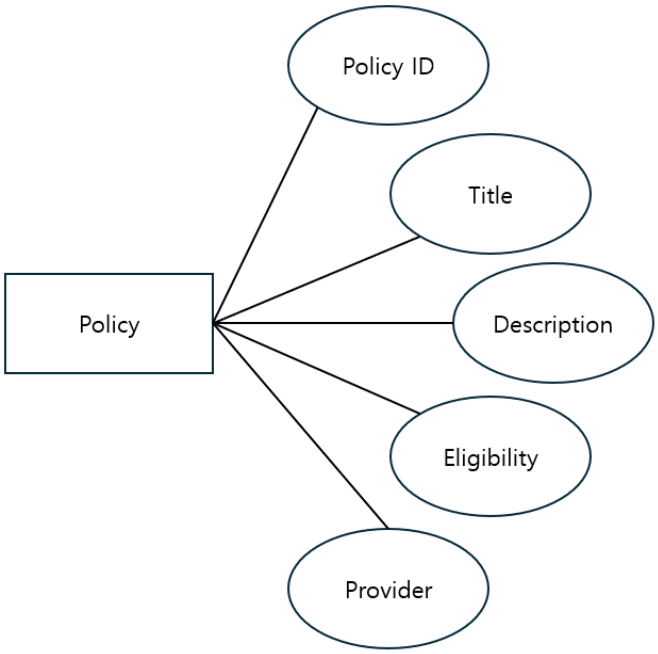

Figure 7.3: ER diagram - Policy

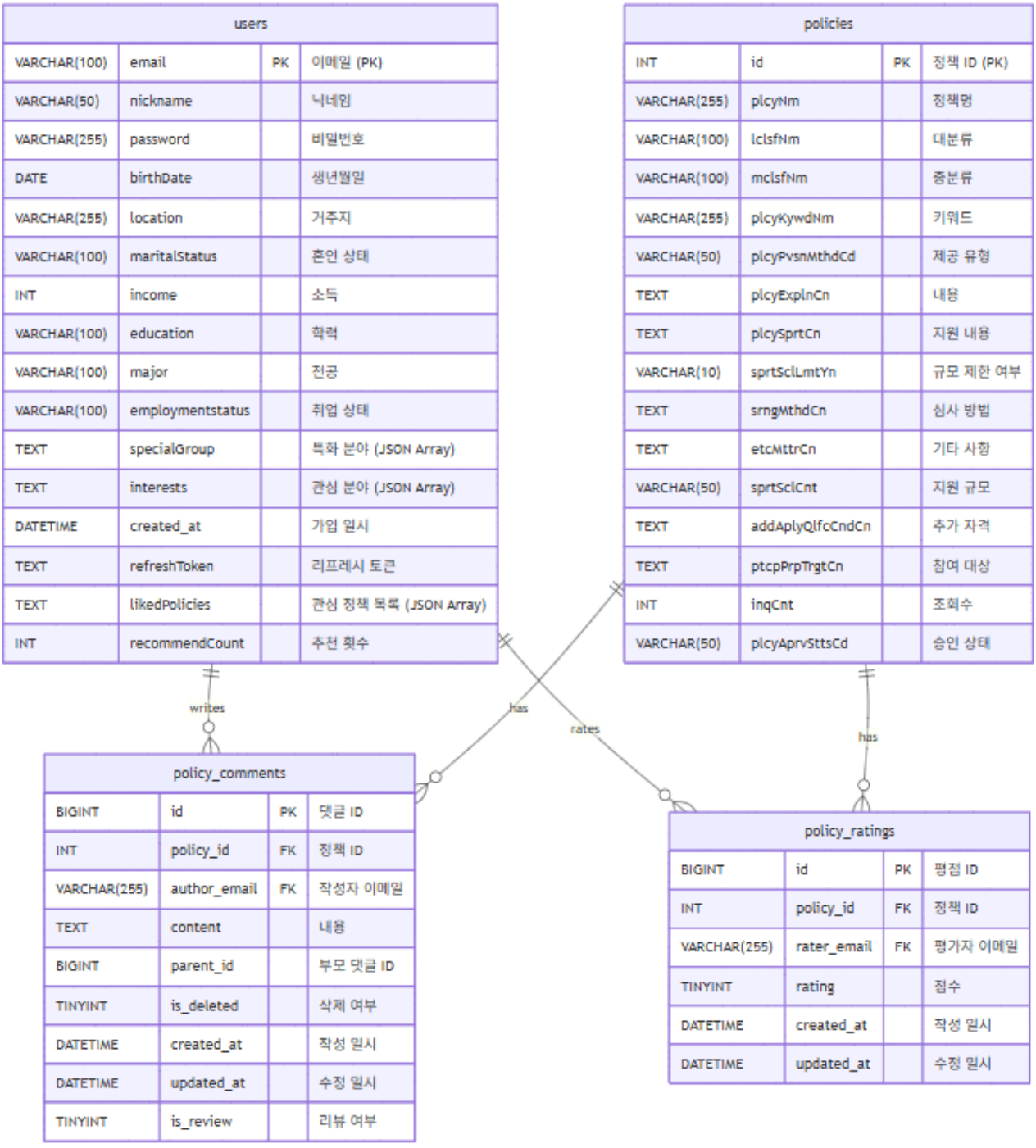

Figure 7.4: ER diagram – Relational Schema

7.2.1. Entity Descriptions

데이터베이스를 구성하는 4 가지 핵심 엔티티의 역할과 주요 속성은 다음과 같다.

1. Users

시스템의 회원 정보를 통합 관리하는 테이블. 개인화된 추천 알고리즘(RAG)의 정확도를 높이기 위해 상세한 프로필 정보 저장

핵심 속성:

- **프로필 정보:** maritalStatus(혼인 상태), major(전공), employmentStatus(취업 상태), income(소득) 등 정책 매칭에 필수적인 상세 조건을 컬럼으로 관리
- **JSON 데이터:** specialGroup(특화 분야), interests(관심 분야), likedPolicies(관심 정책)는 사용자에 따라 선택 항목이 유동적이므로, 확장성을 위해 TEXT 타입으로 정의하고 내부적으로 JSON 배열 문자열 저장
- **보안:** password 는 평문이 아닌 Bcrypt 로 해싱된 문자열을 저장하며, refreshToken 을 통해 세션 보안 강화

2. Policies

국무조정실 등 외부 API 로부터 수집(ETL)된 청년 정책 원본 및 가공 데이터 저장

핵심 속성:

- **분류 정보:** lclsfNm(대분류)뿐만 아니라 mclsfNm(중분류), plcyPvsnMthdCd(제공 유형)를 포함하여 사용자의 다각적인 필터링 검색 지원
- **RAG 최적화 데이터:** plcyExplnCn(정책 설명), plcySprtCn(지원 내용) 외에도 srngMthdCn(심사 방법), etcMttrCn(기타 사항) 등 풍부한 텍스트 정보를 저장. 이는 LLM 이 정책을 이해하고 추천 사유를 생성할 때 활용되는 핵심 컨텍스트(Context).
- **생명주기:** bizPrdBgngYmd, bizPrdEndYmd 를 통해 정책의 유효 기간 관리

3. Policy_Comments

정책에 대한 사용자의 정성적 피드백(텍스트) 저장

핵심 속성:

- **구분**: is_review (TINYINT) 플래그를 사용, 단순 문의성 댓글이 포함된 정식 피드백을 구분하여 관리
- **데이터 보존**: is_deleted (TINYINT) 컬럼을 사용, 사용자가 삭제 요청 시 DB 에서 즉시 제거하지 않고 플래그만 업데이트하는 Soft Delete 방식 수행. 이를 통해 데이터 복구 가능성을 열어두고 참조 무결성 유지

4. Policy_Ratings

정책에 대한 사용자의 정량적 만족도 저장

핵심 속성:

- **관계 해소**: Users 테이블과 Policies 테이블 간의 N:M(다대다) 관계를 1:N 관계로 풀어내는 교차 테이블(Associative Entity) 역할 수행
- **선호도 데이터**: 사용자별 피드백 데이터는 추후 협업 필터링(Collaborative Filtering)이나 추천 알고리즘의 가중치로 활용되어 추천 품질을 고도화하는 데 사용

7.3 SQL Example

```
// 정책을 한글 변환해서 DB에 저장
async function upsertPolicy(policy) {
   // outputSpec 순서대로 의미 변환 적용
  const keys = outputSpec.map(({ key }) => key);
  const fields = keys.join(',');
  const placeholders = keys.map(() => '?').join(',');
  const updateFields = keys.map(k => `${k}=VALUES(${k})`).join(',');
  // 각 필드를 변환해서 저장!
  const values = keys.map(k => convertCodeToMeaning(k, policy[k] || ''));

  const sql = `
    INSERT INTO policies (${fields}) VALUES (${placeholders})
    ON DUPLICATE KEY UPDATE ${updateFields}
  `;
  await pool.query(sql, values);
}
```

```
const criteria = `
  (aplyPrdSeCd IN ('마감', '0057003'))
  OR (
    aplyPrdSeCd IN ('특정기간', '0057001')
    AND aplyYmd IS NOT NULL AND aplyYmd <> ''
    AND STR_TO_DATE(
      REGEXP_REPLACE(TRIM(SUBSTRING_INDEX(aplyYmd, '~', -1)), '[^0-9]', ''),
      '%Y%m%d'
    ) < CURDATE()
  )
`;

const [rows] = await conn.query(`SELECT id FROM policies WHERE ${criteria}`);
```

Figure 7.5, 7.6: SQL Example

8. Testing Plan

8.1 Objectives

본 챕터에서는 'LLM 활용 청년정책 추천 플랫폼'의 안정성과 신뢰성을 검증하기 위한 포괄적인 테스트 계획을 수립한다. Development Testing, Release Testing, User Testing 3 단계로 나누어 수행되는 테스트에 대한 계획을 설명한다. 테스트의 주된 목적은 웹 애플리케이션의 기능적 무결성을 확인하고,

특히 Node.js-Python 하이브리드 아키텍처와 RAG 기반 AI 추천 시스템이 사용자에게 정확하고 시의적절한 정보를 제공하는지 보증하는 데 있다. 이를 위해 단위 테스트, 통합 테스트, 시스템 테스트, 그리고 AI 특화 테스트를 수행한다.

8.2 Testing Policy

8.2.1. Development Testing

개발 단계에서 발생하는 오류를 실시간으로 감지하고 수정하기 위한 테스트 정책이다. Frontend (Next.js)와 Backend(Node.js/Python) 간의 데이터 흐름 검증에 중점을 둔다.

1. Unit & Integration Testing Strategy

- **API Testing:** Postman 및 Swagger UI 를 활용하여 auth, mypage, policies 의 모든 REST API 엔드포인트에 대한 Request/Response 형식을 검증한다.
- **RAG Pipeline Testing:** langchaintest.py 스크립트의 단위 테스트를 통해, 입력된 프롬프트에 대해 의도한 정책이 Retrieval 되는지, 생성된 답변이 정책 원본 데이터와 일치하는지 검증한다.

2. Performance Metrics

SRS(요구사항 명세서)에 정의된 동적 요구사항을 준수하기 위해 다음 지표를 테스트 기준으로 삼는다.

- **RAG 응답 시간:** 사용자 질문 후 추천 결과 생성까지 15 초 이내 (OpenAI API 지연 포함).
- **일반 API 응답:** 로그인, 리스트 조회 등 DB 기반 작업은 3 초 이내 완료.
- **동시 접속:** Node.js 의 비동기 처리 능력을 활용하여 동시 접속자 100 명 상황에서도 에러율 1% 미만 유지.

3. Security Testing

- **SQL Injection:** search_v2.py 및 Node.js 의 SQL 쿼리에 악의적인 구문 삽입 시도 시 방어 여부 확인.

- **JWT Integrity**: 만료된 Access Token 사용 시 401 Unauthorized 반환 및 refresh 로직의 자동 수행 여부 검증.

8.2.2. Release Testing

실제 배포 환경에서의 시스템 동작을 검증하는 단계이다.

- **Environment Parity**: 로컬 개발 환경과 배포 환경 간의 환경 변수 설정(DB 호스트, API 키) 일치 여부를 확인한다.
- **Process Management**: Node.js 메인 서버가 Python 자식 프로세스를 정상적으로 생성하고, 프로세스 종료 시 좀비 프로세스가 남지 않는지 확인한다.
- **Data Sync Test**: api_save.js 스케줄러를 수동 실행하여 외부 국무조정실 API 로부터 대량의 데이터를 수집할 때, MySQL DB 에 중복 없이 Upsert 되는지 검증한다.

8.2.3. User Testing

베타 버전 배포 후 실제 청년 사용자를 대상으로 진행하는 테스트이다.

- **Usability Testing**: 사용자 모바일 및 데스크톱 환경에서 반응형 UI 가 깨지지 않고 의도대로 표시되는지 확인한다.
- **Relevance Feedback**: 사용자가 추천받은 정책이 실제 본인의 상황(나이, 거주지, 취업 상태)에 부합하는지 '정성적 평가'를 수집하여 프롬프트 엔지니어링을 개선한다.

8.2.4. Testing Case

주요 시나리오별 테스트 케이스(TC)는 다음과 같다.

TC ID	시나리오	입력 데이터 예시	예상 결과	비고
TC-01	회원가입	이메일 중복 시도	409 Conflict 응답 및 "이메일 중복" 메시지 출력	필수
TC-02	토큰 갱신	Access Token 만료 후 API 요청	클라이언트가 자동으로 /refresh 호출 후 원본 요청 재수행 (사용자 로그아웃 없음)	핵심
TC-03	RAG 추천	"서울 사는 취준생 지원금 찾아줘"	서울 지역, 미취업자 대상의 현금성 지원 정책 5 개 추천 및 사유 생성	핵심
TC-04	필터 검색	지역: 부산, 분야: 창업	zipCd 가 부산이고 plcyKywdNm 에 창업이 포함된 정책 리스트 반환	기능

Table 8.1: Test Cases

9. Development Plan

9.1 Objectives

이 챕터에서는 시스템 구현을 위해 선택된 하드웨어 및 소프트웨어 개발 환경, 사용 도구, 그리고 개발 제약 사항을 상세히 기술한다. 개발 팀원 간의 환경 일치성을 보장하고, 시스템의 배포 및 운영에 필요한 요구사항을 명확히 하는 것을 목표로 한다. 특히, 최신 웹 트렌드를 반영한 Next.js + Node.js/Python 스택을 통해 고성능의 확장 가능한 시스템을 구축하는 것을 목표로 한다.

9.2 Frontend Environment

9.2.1. TypeScript & React (Next.js)

- **Framework:** Next.js 14 (App Router)를 사용하여 서버 사이드 렌더링(SSR)을 통한 초기 로딩 속도 최적화와 SEO(검색 엔진 최적화)를 달성한다.
- **Language:** TypeScript 를 채택하여 컴파일 단계에서 타입 오류를 사전에 방지하고, 코드의 안정성을 높인다.
- **Styling:** Tailwind CSS 를 사용하여 유틸리티 클래스 기반의 신속하고 일관된 반응형 디자인을 구현한다.

9.3 Backend Environment

9.3.1. Node.js (Express) & Python

- Dual-Process Architecture:
    - **Node.js (Express):** 비동기 I/O 처리에 강점이 있는 Node.js 를 메인 API 서버로 사용하여 높은 동시성 처리
    - **Python:** 풍부한 AI 생태계(LangChain, Pandas 등)를 활용하기 위해, RAG 및 데이터 분석 로직은 Python 별도 프로세스로 구동

9.3.2. MySQL

- **RDBMS:** 정형화된 사용자 프로필과 정책 데이터를 관계형 데이터베이스인 MySQL 8.0 에 저장
- **JSON Support:** MySQL 의 JSON 컬럼 타입을 활용하여 가변적인 정책 속성(특화 분야 등)을 유연하게 저장

9.3.3. Git & GitHub

- **VCS:** 소스 코드 형상 관리를 위해 Git 을 사용하며, GitHub 를 통해 팀원 간의 협업, 코드 리뷰, 이슈 트래킹을 수행. 이때, main, develop, feature/* 브랜치 전략 사용

9.4 Constraints

본 시스템은 다음의 제약 사항을 준수하며 개발된다.

1. **응답 시간 제약:** 사용자의 이탈을 방지하기 위해 일반적인 페이지 로딩 및 API 응답은 3 초 이내를 목표로 한다. 단, RAG 생성은 외부 LLM 연산 시간을 고려하여 15 초 이내로 제한하며 로딩 인디케이터를 필수로 제공한다.
2. **비용 효율성:** OpenAI API 호출 비용을 최소화하기 위해, RAG 프로세스 전 MySQL 기반의 1 차 필터링을 반드시 수행하여 LLM 에 전달되는 토큰 양을 최적화한다.
3. **데이터 최신성:** 정책 데이터는 매일 1 회 이상 api_save.js 스케줄러를 통해 동기화되어야 하며, 신청 기간이 만료된 정책은 즉시 검색 결과에서 제외되어야 한다.
4. **표준 호환성:** Chrome, Safari, Edge 등 모던 브라우저의 최신 버전을 타겟으로 개발한다.

9.5 Assumptions and Dependencies

- **OpenAI API 가용성:** 시스템의 핵심 기능인 추천 사유 생성은 OpenAI API 의 정상 동작을 전제로 한다. API 장애 발생 시, 시스템은 LLM 생성 단계 없이 '키워드 매칭 결과'만을 제공하는 Fallback 모드로 동작한다.
- **공공데이터 포털 API:** 정책 원본 데이터의 정확성과 무결성은 국무조정실 API 에 의존한다. 해당 API 의 스키마 변경 시 api_save.js 의 수정이 필요하다.
- **운영 환경:** 시스템은 AWS(EC2, RDS)와 같은 클라우드 환경에 배포됨을 가정하며, Node.js 와 Python 런타임이 모두 설치된 OS 환경(Linux Ubuntu 등)이 필요하다

Document History

소프트웨어 디자인 명세서 IEEE 권장사항(IEEE Standard for Information Technology—Systems Design—Software Design Descriptions, IEEE Std 1016-2009)에 따라 작성되었다.

Name	***Date***	***Reason For Changes***	***Version***
강민규	2025-11-30	1. Purpose 2. Introduction	V 1.00
김정원	2025-11-30	3. System Architecture - Overall	V 1.00
기민성, 김희수	2025-11-30	4. System Architecture - Frontend	V 1.00
조우열	2025-11-30	5. System Architecture - Backend	V 1.00
강민규	2025-11-30	6. Protocol Design	V 1.00
강민규, 김정원	2025-11-30	7. Database Design	V 1.00
강민규	2025-11-30	8. Test Plan 9. Development Plan	V 1.00