목 차

Ⅰ. 서론 ····· 1

1. ····· 1

2. ····· 3

Ⅱ.관련연구 ····· ○

1. 장학금 정보 시스템 및 추천 기술 동향 ····· ○

2. LLM 기반 정보 추천 및 RAG 기술 ····· ○

Ⅲ. ScholarSync 설계 ····· ○

1. 전체 시스템 설계 ····· ○

Ⅳ. ScholarSync 구현 ····· ○

1. 추천 파이프라인 구현 ····· ○

2. 하드필터 및 소프트 스코어링 ····· ○

2. LangGraph 상태그래프 파이프라인 ····· ○

2. 챗봇 및 프론트엔드 구현 ····· ○

Ⅴ. 결론 ····· ○

참고문헌 ····· ○

<표 목차>

<표 1-1> ····· 7
<표 2-1> ····· ○
<표 3-1> ····· ○
<표 3-2> ····· ○

[그림 목차]

[그림 2-1] ····· ○
[그림 2-2] ····· ○
[그림 3-1] ····· ○
[그림 3-2] ····· ○
[그림 3-3] ····· ○

Ⅰ. 서론

1.1 연구 배경

오늘날 대학생이 활용할 수 있는 장학금은 교내 장학금, 국가장학금, 지방자치단체 공공 지원금, 민간 재단 장학금 등 매우 다양한 채널에 분산되어 있다. 교육부와 대학교육협의회의 2024년 발표에 따르면 4년제 대학생 1인당 연평균 장학금 수혜액은 382만 7천 원이며, 장학금 총액은 5조 540억 원에 달한다. 그럼에도 불구하고 현장에서는 자격이 충분한 재학생이 정보 접근의 어려움으로 인해 신청 기회를 놓치는 사례가 빈번하게 발생하고 있다.

교내 장학금 공지는 대학 포털 시스템에, 국가장학금은 한국장학재단 누리집(kosaf.go.kr)에, 지자체 지원금은 경기도 잡아바・복지로・청년포털 등 서로 다른 플랫폼에 각각 분산되어 있다. 교육부와 대학교육협의회의 2024년 발표에 따르면, 4년제 대학생 1인당 연평균 장학금은 382만 7천 원으로 전년 대비 7.2% 증가하였으며, 장학금 총액은 5조 540억 원에 달한다. 이처럼 지원 규모는 상당하지만, 정보가 파편화되어 있어 학생이 이를 능동적으로 취합하려면 여러 플랫폼을 개별적으로 방문해야 하는 번거로움이 발생한다. 외부 장학금 정보 포털인 드림스폰(dreamspon.com) 등에는 3,000개 이상의 외부 장학금 정보가 실시간으로 제공되고 있을 만큼 장학금 소스 자체는 방대하나, 이를 개인 조건에 맞게 필터링하는 통합 창구는 존재하지 않는다.

(국가장학금)

(지자체 및 거주지 조건 장학금)

(한신대 교내 장학금)

(민간 장학금)

더불어 장학금 자격 요건 자체의 복잡성도 주요 장벽으로 작용한다. 예를 들어 국가장학금 I유형은 학자금 지원 9구간 이하이면서 직전 학기 12학점 이상 이수자로 100점 만점의 80점(B 학점) 이상인 성적 기준을 충족해야 하며, 기초차상위 계층에는 별도 성적 기준이 적용된다. 또한 지역인재장학금의 경우 해당 학기 국가장학금을 미신청하거나 소득구간 산정이 완료되지 않으면 지원이 불가하다는 연계 조건이 존재하여, 이를 모르고 신청을 누락하면 두 가지 혜택을 동시에 잃게 되는 구조다.

1.2 문제 정의

본 팀이 분석한 구조적 문제는 다섯 가지로 정리된다.

첫째, 정보 파편화 문제이다. 장학금 정보가 다수의 플랫폼에 분산되어 있어 학생이 이를 능동적으로 취합하려면 여러 플랫폼을 개별적으로 방문해야 하는 번거로움이 발생한다. 플랫폼마다 인터페이스, 검색 방식, 필터 기준이 상이하여 통합적 파악이 어렵다.

둘째, 개인 맞춤 정보의 부재이다. 동일한 공지 페이지를 보더라도 학점 조건, 소득분위 제한, 거주지 요건, 전공 제한 등 본인에게 해당되는지 여부를 스스로 판단해야 하며, 복잡한 조건을 정확히 해석하기 어렵다. 특히 복수 조건이 AND 관계로 연결된 경우 자격 판단이 더욱 까다롭다.

셋째, 신청 기회 손실 문제이다. 장학금 신청 일정은 학기마다 상이하며, 국가장학금교내 장학금지자체기업 등 다양한 경로에서 각각 별도의 신청 기간을 운영하고 있다. 공고 마감일 파악 실패, 중복 수혜 제한 미인지, 서류 준비 지연 등으로 자격이 충분함에도 신청하지 못하는 사례가 발생한다.

넷째, 상담 인력의 구조적 한계이다. 장학처담당자가 수백~수천 명의 재학생 개개인의 상황을 일일이 점검하여 맞춤 안내를 제공하기에는 인력 및 시간적 한계가 명백하다. 특히 학기 초 장학금 신청 기간에는 문의가 집중되어 적시에 개별 상담을 받기 어려운 실정이다.

다섯째, 외부 장학금 검증의 어려움이다. 민간 재단이나 기업이 운영하는 외부 장학금은 출처 신뢰도, 갱신 주기, 중복 수혜 가능 여부 등을 학생이 직접 검증해야 한다. 외부 공고는 마감 후에도 삭제되지 않고 잔류하는 경우가 많아 정보의 신뢰성이 떨어진다.

1.3 연구 목적 및 범위

본 프로젝트의 목표는 한신대학교 재학생을 주 사용자로 가정하여, 흩어져 있는 장학금지원금 정보를 사용자 프로필(학점, 소득분위, 거주지, 학과, 향후 계획 등)에 맞게 정렬필터링설명까지 제공하는 웹 기반 맞춤 서비스 ScholarSync를 구현하는 것이다. 구체적인 연구 목표는 네 가지이다.

- 교내 장학금 중심의 신뢰 가능한 추천 파이프라인: SQL 하드필터와 LLM 소프트 스코어를 결합하여 정확하고 안전한 추천 제공
- 공공지자체 지원금과의 통합 스코어링: 교내 장학금과 공공 DB를 동일한 스코어링 프레임에 올려 통합 비교 제공
- 규정공지 기반 상담형 챗봇: RAG 기반 벡터 검색과 다단계 LLM으로 근거 있는 대화형 상담 구현
- 완성도 있는 UX 및 계정 기능: 로그인세션북마크모바일 반응형 등 실사용 흐름 완성

본 최종 보고서의 범위는 교내 장학금과 공공지자체 지원금 DB를 핵심 추천 대상으로 한다. 외부 민간 장학금은 데이터 품질 관리의 복잡성으로 인해 후속 스프린트 과제로 분리하였으며, 현재 시스템은 허용 출처 목록과 중복 키 설계 단계까지 완료한 상태이다.

1.4 보고서 구성

본 보고서는 총 5개 장으로 구성된다. 2장에서는 기존 장학금 정보 시스템의 현황과 한계, 그리고 LLM 기반 추천 기술 및 RAG 기술의 동향을 분석한다. 3장에서는 ScholarSync의 전체 시스템 설계를 기술하며, 아키텍처 개요, 데이터 모델, API 설계, 보안 설계를 포함한다. 4장에서는 핵심 기능인 추천 파이프라인과 챗봇 프론트엔드 구현 내용을 상세하게 기술한다. 5장에서는 프로젝트의 결론과 향후 개선 방향을 제시한다.

Ⅱ. 관련 연구

2.1 장학금 정보 시스템 및 추천 기술 동향

2.1.1 국내 장학금 플랫폼 현황

국내에서 운영 중인 장학금 관련 플랫폼은 크게 세 가지 유형으로 구분할 수 있다.

(1) 국가 공공 포털: 한국장학재단

한국장학재단(kosaf.go.kr)은 국가장학금 신청조회에 특화된 국가 공식 플랫폼이다. 소득분위 산정 시스템과 신청 현황 조회 기능을 갖추고 있으나, 교내 장학금이나 민간 장학금 정보는 제공하지 않는다. 사용자가 직접 자격 요건을 읽고 판단해야 하며, 개인 프로필 기반 맞춤 필터링 기능이 존재하지 않는다. 소득분위 정보만 부분적으로 반영할 뿐, 학점거주지전공 등 복합 조건 처리는 불가능하다.

또한 국가장학재단 플랫폼은 국가장학금 Ⅲ유형, 다자녀 장학금, 지역인재장학금 등 자체 장학 유형 중심으로만 구성되어 있어, 학생이 해당 플랫폼만으로는 전체 장학금 수혜 가능성을 파악하기 어렵다.

(2) 지자체복지 포털

경기도 잡아바, 복지로, 청년포털, 경기도일자리재단 등은 지역 기반 지원금을 각각 별도 플랫폼에서 운영한다. 플랫폼 간 연계가 없어 학생이 일일이 방문해야 하며, 거주지소득나이 등의 조건을 스스로 대조해야 한다. 지자체 포털별로 제공 정보의 형식이 상이하여 일괄 비교가 어렵고, 검색 기능도 키워드 기반에 머무른다.

예를 들어 경기도 청년기본조례에 따른 청년기본소득은 만 24세 경기도민 대상으로 분기별 25만 원씩 연 100만 원을 지급하는 제도이나, 이 정보가 대학교 장학금 공지와 함께 학생에게 전달되는 채널은 존재하지 않는다.

(3) 민간 장학금 정보 포털

드림스폰(dreamspon.com), 하이브레인넷 등은 3,000개 이상의 외부 장학금 정보를 제공하는 민간 포털이다. 정보 수집 범위는 넓지만, 개인 조건(소득분위학과거주지학점)에 맞는 필터링 기능이 미흡하다. 또한 공고 마감 후에도 삭제되지 않고 잔류하는 경우가 많아 정보의 최신성과 신뢰성이 낮다는 구조적 문제를 안고 있

다.

2.1.2 기존 시스템 비교 분석

기존 장학금 플랫폼과 ScholarSync의 기능을 비교하면 다음과 같다.

구분	한국장학재단	드림스폰	잡아바 등 지자체	ScholarSync
교내 장학 연계	X	X	X	✓
개인 맞춤 필터링	△(소득만)	X	X	✓
복수 출처 통합	X	△(외부만)	X	✓
규정 기반 설명	X	X	X	✓
신청 기간 알림	△	X	X	✓
설명 가능한 추천	X	X	X	✓
대화형 상담	X	X	X	✓

<표 2-1> 기존 장학금 플랫폼과 ScholarSync 기능 비교

위 비교표에서 확인할 수 있듯이, 현재 어떤 플랫폼도 교내공공민간 장학금을 한 화면에서 사용자 조건에 맞게 통합 추천하고, 규정 기반 설명과 대화형 상담까지 제공하는 기능을 갖추지 못하고 있다. ScholarSync는 이 공백을 메우는 것을 핵심 차별점으로 삼는다.

2.1.3 추천 시스템 기술 동향

추천 시스템은 크게 협업 필터링, 콘텐츠 기반 필터링, 하이브리드 방식으로 구분된다.

협업 필터링은 유사한 이용 패턴을 보이는 사용자들의 선호 데이터를 기반으로 추천한다. 이 방식은 장학금 도메인에서 문제에 취약하다. 대부분의 학생은 장학금 신청 이력이 거의 없어 유사 사용자 클러스터를 형성하기 어렵다.

콘텐츠 기반 필터링은 아이템의 특성과 사용자 프로필의 유사도로 추천한다. 장학금 도메인에서는 자격 요건이 이진 규칙 형태로 존재하므로, 전통적 콘텐츠 기반 필터링만으로는 '소득분위 초과'나 '학점 미달' 같은 명시적 제약을 정확히 처리하기 어렵다.

하이브리드 방식은 규칙 기반 하드필터와 LLM 소프트 스코어링을 결합한

ScholarSync의 접근법과 가장 유사하다. 최근에는 LLM의 자연어 이해 능력을 활용한 제로샷 추천이 주목받고 있으며, 특히 도메인 특화 지식이 필요한 장학금금융법률 영역에서 성과를 보이고 있다.

2.2 LLM 기반 정보 추천 및 RAG 기술

2.2.1 대규모 언어 모델(LLM)의 추천 시스템 활용

GPT-4, Claude 등 대규모 언어 모델(LLM, Large Language Model)은 방대한 텍스트 학습을 통해 자연어 이해, 추론, 생성 능력을 갖추고 있다. 추천 시스템에서 LLM은 두 가지 방식으로 활용된다.

첫째, 자연어 기반 사용자 프로필 이해이다. 학생이 자연어로 의도를 표현하면 LLM은 이를 구조화된 필터 조건으로 변환하여 검색에 활용한다. 이는 전통적인 키워드 검색보다 훨씬 유연하고 직관적인 인터페이스를 제공한다.

둘째, 추천 이유 생성이다. LLM은 특정 장학금이 왜 해당 학생에게 적합한지를 자연어로 설명할 수 있다.

그러나 LLM만으로 추천 시스템을 구성하면 두 가지 심각한 문제가 발생한다. 환각(Hallucination) 문제로 인해 실제로 존재하지 않는 장학금을 생성하거나, 자격 요건을 잘못 해석할 수 있다. 또한 학습 데이터의 최신성 한계로 인해 최신 공고 정보를 반영하지 못한다. 이 두 가지 문제를 해결하기 위해 RAG 기술이 도입된다.

2.2.2 검색 증강 생성(RAG) 기술

RAG는 외부 지식 베이스를 실시간으로 검색하여 LLM의 응답 근거로 활용하는 기술이다. LLM이 사전 학습 데이터에만 의존하는 것과 달리, RAG는 실제 데이터베이스나 문서에서 관련 정보를 검색하여 근거 기반 답변을 생성한다.

RAG의 핵심 구성 요소는 다음 세 가지이다.

- 임베딩 모델: 텍스트를 수치 벡터로 변환. 의미적으로 유사한 텍스트는 벡터 공간에서 가까운 위치에 배치됨
- 벡터 스토어: 임베딩 벡터를 저장하고 코사인 유사도 알고리즘으로 빠르게 유사 문서를 검색

- 생성 LLM: 검색된 컨텍스트를 참고하여 사용자 질문에 근거 있는 답변 생성

ScholarSync에서는 OpenAI text-embedding-3-small 모델로 교내 장학 규정 PDF와 공지 텍스트를 임베딩하고, Chroma 벡터 스토어에 색인한다. 사용자가 질문하면 질의 임베딩과 가장 유사한 규정 조각 5개를 검색하여 GPT-4o의 답변 생성 컨텍스트로 활용한다.

2.2.3 LangGraph 기반 에이전트 워크플로

LangGraph는 LLM 기반 에이전트 워크플로를 방향성 있는 상태 그래프(Directed State Graph)로 표현하는 프레임워크이다. LangChain 생태계의 일부로, 복잡한 다단계 추론 파이프라인을 모듈화시각화디버깅할 수 있는 구조를 제공한다.

LangGraph의 핵심 개념은 다음과 같다.

- 상태: 파이프라인의 현재 맥락 정보를 담는 TypedDict 객체. 각 노드는 상태를 읽고 수정하며 다음 노드로 전달
- 노드: 상태를 입력으로 받아 처리 후 수정된 상태를 반환하는 함수. 독립적으로 테스트 가능
- 엣지: 노드 간 실행 흐름을 정의. 조건부 엣지로 상태에 따라 분기 가능
- 컴파일된 그래프: 상태그래프를 실행 가능한 형태로 컴파일. 스트리밍비동기 실행 지원

LangGraph가 전통적인 파이프라인 대비 갖는 장점은, 각 처리 단계를 독립 노드로 분리하여 단위 테스트, 오류 추적, 단계별 디버깅이 용이하다는 점이다. 또한 조건부 엣지를 통해 후보군이 없는 경우 복잡한 제어 흐름을 선언적으로 표현할 수 있다.

2.2.4 벡터 데이터베이스

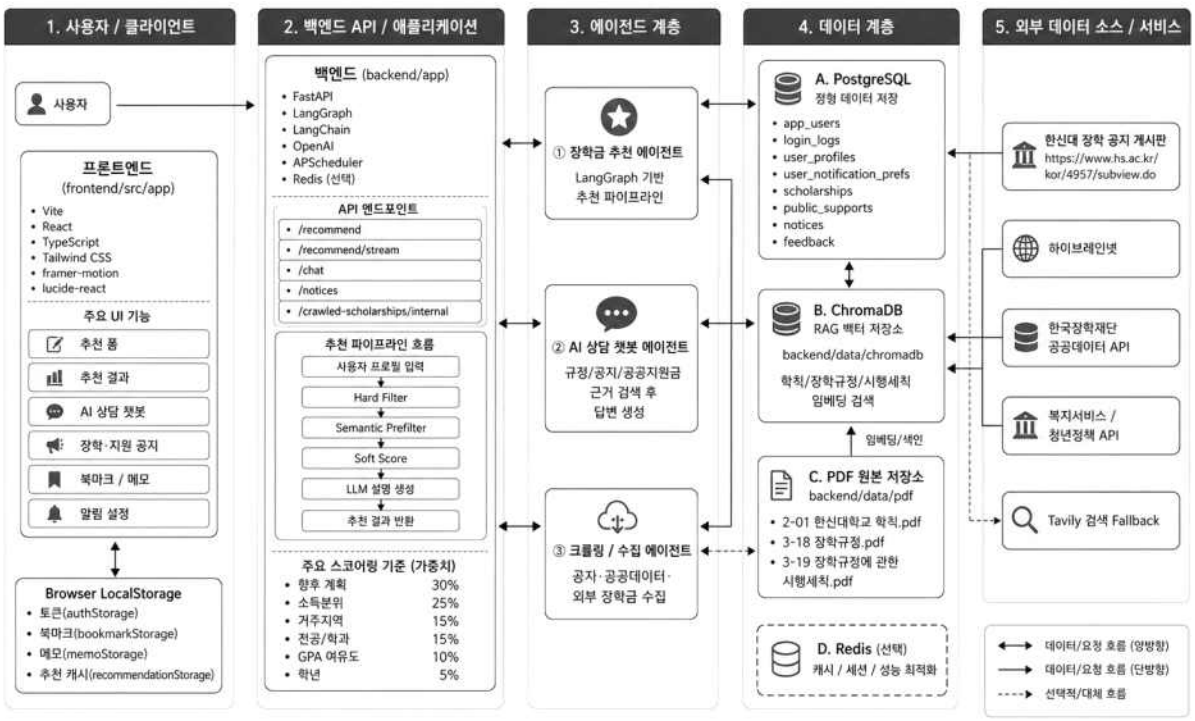
벡터 데이터베이스는 고차원 임베딩 벡터를 저장하고 유사도 검색을 빠르게 수행하는 특수 데이터베이스이다. 대표적인 솔루션으로는 Chroma, Pinecone, Weaviate 등이 있다.

본 프로젝트에서 Chroma를 선택한 이유는 다음과 같다. 첫째, 로컬 파일 시스템에 데이터를 저장하는 임베디드 모드를 지원하여 외부 서비스 의존성 없이 개발 환경에서 즉시 사용 가능하다. 둘째, Python 네이티브 API를 제공하여 LangChain LangGraph와의 통합이 원활하다. 셋째, 오픈소스로 제공되어 상용 서비스 대비 비용이 발생하지 않는다.

Chroma는 내부적으로 HNSW 알고리즘을 사용하여 근사 최근접 이웃 검색을 수행한다. 이를 통해 수만 개의 벡터 중에서도 밀리초 단위로 유사 문서를 검색할 수 있다.

Ⅲ.ScholarSync 설계

3.1 전체 시스템 아키텍처



[그림 3-1] ScholarSync 전체 시스템 아키텍처

그림 3-1은 ScholarSync의 전체 기술 구조를 나타낸 것이다. 시스템은 사용자/클라이언트, 백엔드 API 및 애플리케이션, 에이전트 계층, 데이터 계층, 외부 데이터 소스 및 서비스의 다섯 영역으로 구성된다. 프론트엔드는 Vite, React, TypeScript,

Tailwind CSS를 기반으로 사용자 입력 화면, 추천 결과 화면, AI 상담 챗봇, 장학·지원 공지, 북마크 및 메모 기능을 제공한다. 백엔드는 FastAPI를 중심으로 추천 API, 상담 API, 공지 조회 API를 제공하며, LangGraph 기반 추천 에이전트, AI 상담 챗봇 에이전트, 크롤링 및 수집 에이전트와 연동된다.

데이터 계층에서는 PostgreSQL이 회원, 로그인 기록, 사용자 프로필, 알림 설정, 장학금, 교외·공공 지원금, 공지사항, 피드백 등 정형 데이터를 저장한다. ChromaDB는 추천 결과 자체를 저장하는 공간이 아니라, 한신대학교 학칙, 장학규정, 장학규정 시행세칙 등 PDF 문서 내용을 임베딩하여 규정 근거를 검색하기 위한 RAG 벡터 저장소로 사용된다. 또한 한신대 장학 공지 게시판, 하이브레인넷, 한국장학재단 공공데이터 API, 복지서비스 및 청년정책 API 등의 외부 데이터를 수집하여 추천 후보와 상담 근거를 보완한다.

ScholarSync는 프론트엔드, 백엔드, 데이터 계층, AI 파이프라인, 운영보안의 5개 레이어로 구성된 풀스택 웹 서비스이다. 각 계층은 명확한 역할 분리 원칙을 따르며 REST API를 통해 통신한다.

레이어	구성 기술	역할
프론트엔드	React (Vite) + TypeScript	추천 폼결과챗봇홈 대시보드북마크 UI. 백엔드 API와 JSON 통신. 모바일 반응형.
백엔드	FastAPI + Pydantic + Uvicorn	인증(JWT), 추천, 채팅, 관리용 REST 엔드포인트. 비동기 처리.
데이터 계층	PostgreSQL + asyncpg + Chroma	장학금공공지원금사용자북마크채팅 로그 저장. 벡터 스토어 병행 운영.
AI 파이프라인	LangGraph + OpenAI + Tavily	상태그래프 추천 파이프라인. RAG 벡터 검색. 보조 웹 검색.
운영보안	JWT + CORS + .env + 스케줄러	인증환경변수데이터 갱신 스케줄러로깅 관리.

<표 3-1> ScholarSync 시스템 아키텍처 구성

3.1.1 아키텍처 설계 원칙

시스템 설계의 핵심 원칙은 '보수적 설계'이다. 잘못된 장학 추천은 학생이 실제로 신청하였다가 탈락하는 피해를 야기할 수 있으므로, 시스템이 LLM 출력에만 의존하지 않고 규칙 기반 필터를 코드 레벨에서 명시적으로 내재화하는 방식을

채택하였다.

LLM은 자격 요건을 통과한 후보에 대한 '적합도 수치화'와 '근거 설명 생성'에만 활용함으로써 안전성과 정확성을 동시에 확보한다. 이는 LLM이 소득분위나 학점 요건을 잘못 해석하는 경우에도 하드필터 단계에서 이미 부적합 항목이 제거되어 있기 때문에, 추천 결과의 최소한의 정확성이 보장되는 구조이다.

또한 각 레이어는 독립적으로 배포 가능한 모듈 구조를 따른다. 프론트엔드와 백엔드는 완전히 분리되어 CORS 설정을 통해 통신하며, AI 파이프라인은 백엔드 내 독립 모듈로 운영된다. 이를 통해 향후 추천 알고리즘 교체나 벡터 스토어 이전 시 다른 레이어에 영향을 최소화할 수 있다.

3.1.2 요청 처리 흐름

사용자의 요청이 처리되는 전체 흐름은 다음과 같다.

① 클라이언트(React)에서 사용자 프로필 데이터를 JSON으로 직렬화하여 백엔드 API에 POST 요청

② FastAPI가 Pydantic 모델로 요청 데이터를 검증하고, JWT 토큰으로 인증 확인

③ LangGraph 추천 파이프라인 호출: 하드필터 → 시맨틱 프리필터 → 소프트 스코어링 → 정렬

④ asyncpg를 통해 PostgreSQL에서 비동기 쿼리로 장학금공공지원금 데이터 조회

⑤ OpenAI API 호출로 임베딩 생성 및 LLM 소프트 스코어 산출

⑥ 최종 추천 결과를 JSON 응답으로 클라이언트에 반환

⑦ 클라이언트가 카드형 UI로 추천 결과 렌더링

3.2 데이터 모델 설계

3.2.1 ERD(Entity Relationship Diagram)

ScholarSync의 PostgreSQL 데이터 모델은 회원 정보(app_users), 로그인 기록(login_logs), 사용자 프로필(user_profiles), 알림 설정(user_notification_prefs), 교내 장학금(scholarships), 교외・공공 지원금(public_supports), 공지사항(notices), 사용

자 피드백(feedback)을 중심으로 구성된다. 북마크, 상담 메모, 신청 장학금 메모, 추천 결과 캐시는 서버 DB가 아니라 프론트엔드의 Browser LocalStorage를 통해 관리된다.

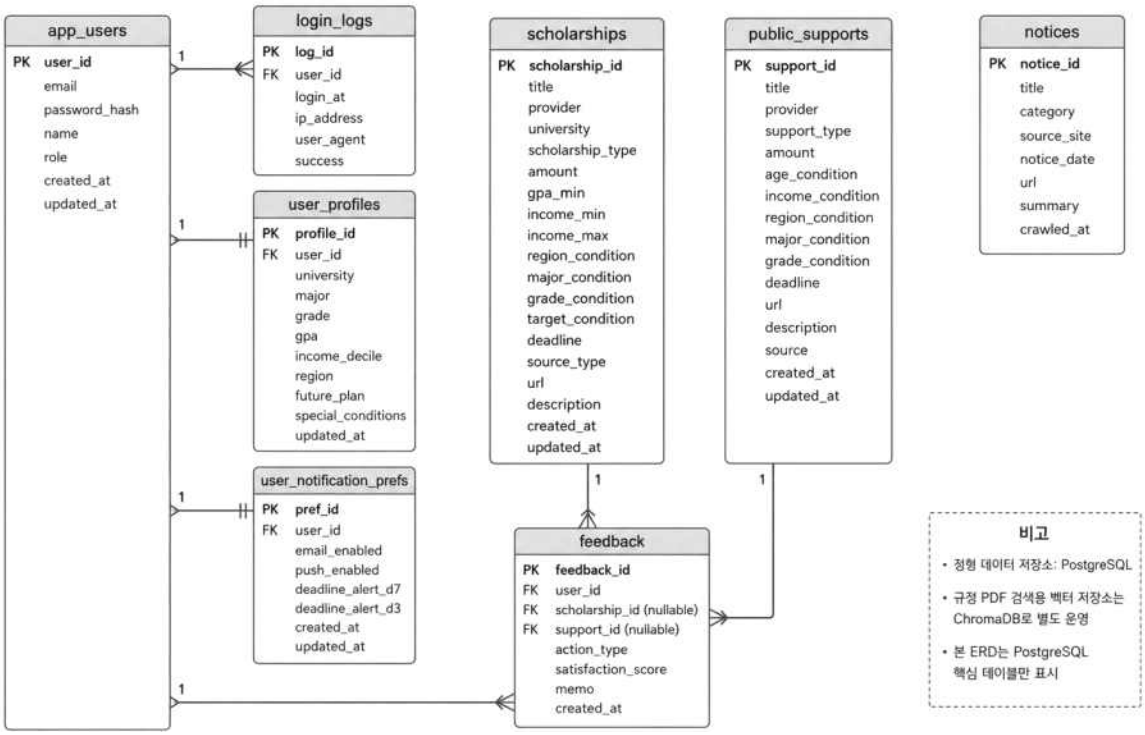
테이블	주요 필드	설명
app_users	user_id, email, password_hash, name, role, created_at, updated_at	회원 기본 정보와 권한 정보를 저장하는 사용자 계정 테이블.
login_logs	log_id, user_id, login_at, ip_address, user_agent, success	로그인 시각, 접속 환경, 로그인 성공 여부를 기록하는 테이블.
user_profiles	profile_id, user_id, university, major, grade, gpa, income_decile, region, future_plan, special_conditions, updated_at	장학금 추천에 필요한 사용자 프로필 정보를 저장하는 테이블.
user_notification_prefs	pref_id, user_id, email_enabled, push_enabled, deadline_alert_d7, deadline_alert_d3, created_at, updated_at	장학금 마감일 알림 및 이메일·푸시 알림 설정을 저장하는 테이블.
scholarships	scholarship_id, title, provider, university, scholarship_type, amount, gpa_min, income_min, income_max, region_condition, major_condition, grade_condition, target_condition, deadline, source_type, url, description	교내 장학금 정보를 저장하는 핵심 추천 후보 테이블.
public_supports	support_id, title, provider, support_type, amount, age_condition, income_condition, region_condition, major_condition, grade_condition, deadline, url, description, source, created_at, updated_at	교외·공공 지원금 정보를 저장하는 테이블.
notices	notice_id, title, category, source_site, notice_date, url, summary, crawled_at	한신대학교 장학 공지 게시판 등에서 크롤링한 공지사항을 저장하는 테이블.

feedback	feedback_id, user_id, scholarship_id, support_id, action_type, satisfaction_score, memo, created_at	추천 클릭, 신청 여부, 만족도, 메모 등 사용자 피드백을 저장하는 테이블

<표 3-2> ScholarSync 주요 데이터베이스 테이블 설계

3.2.2 scholarships 테이블 상세

장학금 정보의 핵심 테이블인 scholarships는 하드필터에서 직접 활용되는 구조화 필드와 소프트 스코어링에 활용되는 자유 텍스트 필드로 구분된다.



[그림 3-2] ScholarSync PostgreSQL 기반 핵심 데이터베이스 ERD

그림 3-2는 ScholarSync의 PostgreSQL 기반 핵심 데이터베이스 구조를 나타낸 ERD이다. 사용자 관련 데이터는 app_users를 중심으로 login_logs, user_profiles, user_notification_prefs와 연결된다. app_users는 회원의 기본 계정 정보를 저장하고, login_logs는 로그인 시각, IP 주소, 사용자 에이전트, 성공 여부를 기록한다. user_profiles는 사용자의 대학, 학과, 학년, GPA, 소득분위, 거주지, 향후 계획, 특수 조건 등 추천에 필요한 프로필 정보를 저장하며, user_notification_prefs는 마

감일 알림과 이메일·푸시 알림 설정을 관리한다.

장학금 추천에 사용되는 핵심 후보 데이터는 scholarships와 public_supports에 저장된다. scholarships는 교내 장학금 정보를 저장하며, public_supports는 교외·공공 지원금 정보를 저장한다. notices는 한신대학교 장학 공지 게시판 등 외부에서 수집한 공지사항을 저장하는 독립 테이블이다. feedback 테이블은 사용자의 추천 클릭, 신청 여부, 만족도, 메모 등을 저장하여 향후 추천 품질 개선에 활용할 수 있도록 설계하였다. 한편 규정 PDF 검색을 위한 벡터 데이터는 PostgreSQL이 아니라 ChromaDB에 별도로 저장되므로, 본 ERD에는 PostgreSQL 핵심 테이블만 표시하였다.

3.2.3 데이터 적재 전략

각 데이터 소스별 적재 전략은 다음과 같다.

- 교내 장학금: 교내 포털 공지에서 반기별 수동 적재. 규정집 PDF에서 핵심 조건을 추출하여 구조화 필드에 입력
- 공공지자체 지원금: 경기도 잡아바 Open API(JSON) 및 정부24 CSV 활용. 주 1회 스케줄러로 자동 갱신
- 외부 민간 장학금(예정): 허용 출처 목록 기반 웹 크롤러. 중복 제거 키(name + deadline + provider)와 출처 신뢰도 점수 설계 완료

3.3 API 설계

3.3.1 주요 엔드포인트

백엔드 FastAPI 서버는 추천, 스트리밍 추천, AI 상담 챗봇, 공지 조회, 크롤링 데이터 조회를 중심으로 API를 제공한다. 북마크와 메모 기능은 현재 프론트엔드의 Browser LocalStorage를 통해 관리되므로, 서버 API가 아닌 클라이언트 저장 기능으로 분리된다.

메서드	경로	설명
POST	/recommend	사용자 프로필을 기반으로 장학금 추천 파이프라인을 실행하고 추천 결과를 반환
POST/ GET	/recommend/stream	추천 과정 또는 결과를 스트리밍 방식으로 반환
POST	/chat	사용자의 질문을 받아 학칙·규정, 공지, 교외지원금, 외부 검색 결과를 참고하여 AI 상담 답변 반환
GET	/notices	수집된 장학·지원 공지사항 목록 조회
GET	/crawled-scholarships/internal	교내 장학 공지 크롤링 데이터 조회

<표 3-3> ScholarSync 주요 API 엔드포인트

3.3.2 추천 요청/응답 스키마

추천 API의 요청 본문(Request Body) 스키마는 다음과 같다.

```
// POST /recommend - 요청 본문
{
  "gpa": 3.5,                    // 직전 학기 학점
  "income_decile": 5,            // 소득분위 (1~10)
  "region": "경기도 오산시", // 거주지
  "major": "소프트웨어",      // 학과
  "goals": "취업 준비, 해외 교환학생",  // 목표 (자유 텍스트)
  "extra_keywords": ["IT", "이공계"]   // 추가 관심 키워드
}
```

응답(Response) 스키마는 다음과 같다.

```
// POST /recommend - 응답
{
  "results": [
    {
```

```
        "id": 42,
        "name": "한신장학금 A",
        "source_type": "campus",
        "amount": 2000000,
        "deadline": "2026-08-31",
        "score": 87.3,
        "reason": "소프트웨어 전공 이공계 우선 지원 항목으로...",
        "regulation_ref": "2026년 장학금 지급 계획 3조 1항",
        "hard_filter_passed": true
      }
    ],
    "total": 8,
    "pipeline_version": "1.2.0"
}
```


3.4 보안 설계

3.4.1 인증 및 인가

사용자 인증은 JSON Web Token 기반으로 구현한다. 로그인 성공 시 서버는 사용자 ID와 권한 정보를 페이로드로 담은 액세스 토큰을 발급한다. 클라이언트는 이를 세션보관함에 보관하며, 이후 요청마다 Authorization 헤더에 포함한다.

보안 설계의 주요 결정 사항은 다음과 같다.

- 비밀번호 저장: bcrypt로 단방향 해시하여 저장. 평문 저장 금지
- 토큰 만료: 액세스 토큰 유효 기간 60분. 만료 시 재로그인 유도
- CORS 정책: 프론트엔드 도메인만 허용. wildcard 설정 금지
- 환경변수: 모든 API 키DB 연결 문자열시크릿 키는 .env 파일로 분리. 코드에 하드코딩 금지
- 입력 검증: Pydantic 모델로 요청 데이터 타입범위 검증. SQL Injection 방지

3.4.2 데이터 보호

개인정보 보호를 위해 다음 원칙을 준수한다.

- 최소 수집 원칙: 장학금 추천에 필요한 최소한의 정보(학점, 소득분위, 거주지, 학과)만 수집
- 민감 정보 처리: 소득분위는 정확한 값 대신 구간(분위)으로 처리하여 실제 소득 정보 불요
- 로그 관리: 챗봇 대화 로그는 사용자가 직접 삭제 가능하며, 90일 후 자동 익명화
- 전송 보안: HTTPS 환경에서만 운영. HTTP 요청은 자동 리다이렉트

3.5 에러 처리 및 로깅 설계

3.5.1 에러 처리 전략

ScholarSync는 계층별로 독립적인 에러 처리 전략을 적용한다. 이는 단일 장애 없이 시스템이 부분적으로 동작할 수 있는 탄력성을 확보하기 위한 설계이다.

백엔드 API 레이어에서는 FastAPI의 예외 핸들러를 통해 모든 예외를 캐치하고, HTTP 상태 코드와 함께 표준화된 에러 응답을 반환한다. 에러 응답 형식은 다음과 같다.

```
// 표준 에러 응답 형식
{
  “error_code“: “HARD_FILTER_FAILED“,
  “message“: “자격 조건을 충족하는 장학금을 찾을 수 없습니다.“,
  “detail“: “소득분위 5구간 이하 조건을 충족하는 항목이 없습니다.“,
  “suggestions“: [“소득분위 조건 완화 후 재검색“, “공공 지원금 단독 검색“]
}
```

LangGraph 파이프라인 레이어에서는 각 노드가 독립적으로 try-except로 감싸져 있어, 특정 노드 실패 시 상태에 error 필드를 설정하고 format_output 노드로 바로 점프하는 조건부 엣지가 동작한다. 이를 통해 소프트 스코어링 LLM 호출이 실패하더라도 하드필터 결과만으로 부분적인 추천이 가능하다.

프론트엔드 레이어에서는 axios 응답 인터셉터가 HTTP 에러를 캐치하여 사용자에게 친화적인 에러 메시지를 토스트 알림으로 표시한다. 네트워크 타임아웃(30초) 초과 시 자동으로 재시도(최대 2회) 후 실패를 사용자에게 알린다.

3.5.2 로깅 설계

시스템의 주요 이벤트는 구조화 로깅(Structured Logging) 방식으로 기록된다. Python logging 모듈과 JSON fomatter를 결합하여 로그 수집 도구와의 통합을 용이하게 한다.

로깅 레벨 정책은 다음과 같다.

레벨	적용 이벤트	예시
DEBUG	파이프라인 노드 진입종료. SQL 쿼리 실행 시간.	노드 hard_filter 시작. 쿼리 소요 120ms.
INFO	추천 요청 수신완료. 챗봇 세션 시작종료.	user_id=42 추천 완료, 결과 8개, 소요 3.8s
WARNING	빈 추천 결과. 외부 API 지연.	user_id=42 hard_filter 후보 0개. Tavily 응답 지연 5s.
ERROR	LLM 호출 실패. DB 연결 실패.	OpenAI API 503. asyncpg 연결 풀 고갈.
CRITICAL	데이터 갱신 스케줄러 실패. Chroma 색인 손상.	잡아바 API 401. 재색인 필요.

<표 3-5> ScholarSync 로깅 레벨 정책

3.6 배포 아키텍처

현재 ScholarSync는 단일 서버 개발 환경에서 운영되고 있으며, 향후 프로덕션 배포를 위한 아키텍처 계획은 다음과 같다.

- 컨테이너화: Dockerfile로 백엔드프론트엔드 이미지 빌드. docker-compose로 PostgreSQL Chroma 백엔드, 프론트엔드 통합 실행
- 리버스 프록시: Nginx로 프론트엔드 정적 파일 서빙 및 백엔드 API 프록시

처리

- HTTPS 인증: Let's Encrypt SSL 인증서로 전 구간 암호화
- DB 백업: PostgreSQL pg_dump 일 1회 자동 실행. S3 호환 스토리지에 보관
- 모니터링: Prometheus + Grafana로 API 응답 시간추천 성공률LLM 토큰 사용량 모니터링

Ⅳ. ScholarSync 구현

4.1 추천 파이프라인 구현

추천 파이프라인은 사용자 프로필 입력부터 최종 추천 결과 출력까지 LangGraph 상태 그래프로 구현된 6단계 파이프라인이다. 각 단계는 독립 노드로 모듈화되어 있어 단위 테스트, 단계별 디버깅, 오류 처리가 용이하다.

노드 순서	노드명	입력	출력
1	load_user_profile	요청 JSON	검증된 User Profile 객체
2	hard_filter_scholarships	User Profile	자격 충족 후보 리스트
3	semantic_prefilter	후보 리스트	상위 K개 시맨틱 유사 후보
4	soft_score_scholarships	상위 K개 후보	스코어 사유 부여된 후보
5	merge_and_rank	교내 + 공공 후보	최종 정렬된 추천 리스트
6	format_output	정렬된 리스트	API 응답 JSON

<표 4-1> 추천 파이프라인 노드 구성

4.1.1 하드필터 및 소프트 스코어링

하드필터(Hard Filter) 설계

하드필터는 추천 파이프라인의 1차 관문으로, SQL 레벨에서 자격 미달 장학금을 구조적으로 제거하는 단계이다. 이 단계는 LLM 호출 이전에 실행되므로, LLM 오

해석으로 인한 부적합 추천을 완전히 차단한다.

하드필터의 5가지 적용 기준과 SQL 구현 방식은 다음과 같다.

- 소득분위: income_limit IS NULL OR income_limit >= :user_income_decile
- 학점 최저 기준: gpa_min IS NULL OR gpa_min <= :user_gpa
- 거주지 요건: region IS NULL OR region = :user_region OR region LIKE :user_province
- 특정 대상 제한: target_group IS NULL OR target_group = general
- 신청 기간: deadline IS NULL OR deadline >= CURRENT_DATE

실제 SQL 쿼리 구조는 다음과 같다.

```
SELECT * FROM scholarships
WHERE
  (income_limit IS NULL OR income_limit >= :decile)
  AND (gpa_min IS NULL OR gpa_min <= :gpa)
  AND (region IS NULL OR region = :region
       OR region = :province)
  AND (target_group IS NULL OR target_group = 'general')
  AND (deadline IS NULL OR deadline >= CURRENT_DATE)
ORDER BY source_type = 'campus' DESC;  -- 교내 장학금 우선
```

하드필터 설계에서 특히 주의한 점은 NULL 처리이다. 일부 장학금은 특정 조건에 제한이 없는 경우 해당 필드가 NULL로 저장되어 있다. 예를 들어 region IS NULL은 전국 대상 장학금을 의미하므로, 모든 거주지 사용자에게 후보로 포함되어야 한다. 이 NULL 의미론을 SQL 조건문에 정확히 반영하는 것이 하드필터의 핵심 구현 과제였다.

시맨틱 프리필터(Semantic Prefilter)

하드필터를 통과한 후보군이 다수인 경우, OpenAI 임베딩 모델을 활용한 시맨틱 프리필터로 상위 K개를 선별한다. 이 단계는 LLM 소프트 스코어링 호출 횟수를 줄여 API 비용과 응답 속도를 최적화하는 역할을 한다.

시맨틱 프리필터 동작 방식은 다음과 같다.

- ① 사용자 프로필 텍스트(학과, 목표, 키워드 등)를 임베딩 벡터로 변환
- ② 사전 색인된 장학금 description 임베딩과 코사인 유사도 계산
- ③ 유사도 기준 상위 K개(기본값 K=20)만 다음 단계로 전달

```
async def semantic_prefilter_scholarships(state: GraphState):
    user_text = f"{state.major} {state.goals} {' '.join(state.extra_keywords)}"
    user_emb = await openai.embeddings.create(
        model='text-embedding-3-small', input=user_text
    )
    user_vec = user_emb.data[0].embedding
    # Chroma에서 코사인 유사도 상위 K개 검색
    results = scholarship_store.similarity_search_by_vector(
        user_vec, k=state.prefilter_k
    )
    state.prefiltered = [r.metadata['id'] for r in results]
    return state
```

시맨틱 프리필터 구현 과정에서 발견한 주요 이슈는 특정 키워드에 대한 과도 반응 현상이었다. 예를 들어 사용자가 '취업 준비'를 목표로 입력할 경우, 취업 관련 키워드가 포함된 장학금이 과도하게 상위에 노출되는 경향이 있었다. 이를 해결하기 위해 키워드 가중치를 재조정하고, 학과소득분위 등 구조화 정보의 임베딩 가중치를 높였다.

소프트 스코어링(Soft Scoring)

소프트 스코어링은 하드필터와 후보 축소 단계를 통과한 장학금에 대해 사용자 프로필과 장학금 조건의 적합도를 수치화하는 단계이다. 본 프로젝트에서는 LLM이 임의로 점수를 산출하는 방식이 아니라, backend/app/matching_engine.py에 정의된 규칙 기반 점수 계산 방식을 중심으로 추천 점수를 계산한다.

소프트 스코어 산출 공식

최종 점수 = (향후 계획 × 0.30) + (소득분위 × 0.25) + (거주지역 × 0.15) + (전공

/학과 × 0.15) + (GPA 여유도 × 0.10) + (학년 × 0.05) + 가산점 - 감점

항목별 점수 반영 기준 (0~ 100점)

- 향후 계획 (30%): 어학연수, 교환학생, 해외진출, 대학원 진학, 취업/창업 준비 등
- 소득분위 (25%): 사용자 소득분위 키워드와 장학금 지원 조건 매칭
- 거주지역 (15%): 거주지(도/시/군/구) 일치 여부 확인
- 전공/학과 (15%): IT, 공학, 경영/경제, 예술 등 계열 및 학과 키워드 매칭
- GPA 여유도 (10%): 사용자 GPA가 장학금 최소 자격 GPA보다 얼마나 높은지 여유도 측정
- 학년 (5%): 특정 학년 제한 및 신입생 전용 조건 일치 여부

추가 가감점 및 예외 필터링 로직

- 출처 가산점: UNIVERSITY (교내), LOCAL (지자체), NATIONAL (정부)에 가산점
- 성적 가산점: 고학점자 사용자의 경우 성적 및 학업 우수 장학금 항목 가산점
- 학년 감점: 2학년 이상 사용자가 신입생 전용 장학금 후보에 매칭될 경우 감점
- 후보 제외 조건: 소득 7분위 이상(저소득 장학군) 및 자격 미달 특수 조건(유학생 등) 제외

규칙 기반 점수 계산 이후에는 최종 후보와 사용자 프로필, 관련 규정 검색 결과를 LLM에 전달하여 사용자에게 이해하기 쉬운 추천 설명을 생성한다. 이때 LLM은 점수를 임의로 산출하는 역할이 아니라, 이미 정확하게 계산된 추천 후보와 점수, 규정 근거를 바탕으로 추천 이유를 자연어로 설명하는 역할을 수행한다.

LLM 소프트 스코어링 프롬프트 구조는 다음과 같다.

```
SYSTEM:
당신은 장학금 매칭 설명 전문가입니다. 매칭 엔진(matching_engine.py)에 의해 이미 규칙 기반으로 계산된 추천 점수와 후보 정보를 바탕으로 사용자에게 친절한 추천 사유, 관련 규정 설명, 그리고 신청 시 유의 사항을 생성하세요. LLM이 점수를 임의로 새로 산출하거나 변경해서는 안 되며, 근거 없는 추측은 절대 하지 마세요.

USER:
학생 프로필: {profile_json}
장학금 정보: {scholarship_json}

다음 JSON 형식으로만 응답하세요:
```

```
{
  “reason“: “<추천 이유 2~3문장>“,
  “regulation_ref“: “<관련 규정 조항 or null> “
  “caution“: “<신청 시 확인해야 할 조건>“
}
```

4.1.2 LangGraph 상태그래프 파이프라인

상태(State) 설계

LangGraph 파이프라인의 상태 객체는 모든 노드가 공유하는 파이프라인 컨텍스트이다. TypedDict로 타입 안전성을 보장하며, 각 노드는 자신의 출력을 상태에 병합하여 다음 노드로 전달한다.

```
from typing import TypedDict, List, Optional

class GraphState(TypedDict):
    # 입력
    user_id: int
    gpa: float
    income_decile: int
    region: str
    major: str
    goals: str
    extra_keywords: List[str]
    prefilter_k: int  # 시맨틱 프리필터 상위 K (기본값 20)
    # 파이프라인 중간 결과
    hard_filtered: List[dict]
    prefiltered: List[int]
    scored: List[dict]
    # 최종 출력
    recommendations: List[dict]
    error: Optional[str]
```

그래프 컴파일 및 실행

LangGraph 상태그래프는 노드와 엣지를 정의한 뒤 compile을 호출하여 실행 가능한 형태로 변환한다. 조건부 엣지를 통해 하드필터 후보가 없는 경우 early exit 경로를 제공한다.

```
from langgraph.graph import StateGraph, END

def build_recommend_graph():
    g = StateGraph(GraphState)
    # 노드 등록
    g.add_node('load_profile',    load_user_profile)
    g.add_node('hard_filter',     hard_filter_scholarships)
    g.add_node('semantic_pre',    semantic_prefilter_scholarships)
    g.add_node('soft_score',      soft_score_scholarships)
    g.add_node('merge_rank',      merge_and_rank)
    g.add_node('format_out',      format_output)
    # 엣지 (일반)
    g.add_edge('load_profile',  'hard_filter')
    g.add_edge('semantic_pre',  'soft_score')
    g.add_edge('soft_score',    'merge_rank')
    g.add_edge('merge_rank',    'format_out')
    g.add_edge('format_out',    END)
    # 조건부 엣지: 후보 없으면 early exit
    g.add_conditional_edges('hard_filter', check_candidates,
        {'ok': 'semantic_pre', 'empty': 'format_out'})
    g.set_entry_point('load_profile')
    return g.compile()
```

교내 장학금공공 지원금 통합 스코어링

교내 장학금과 공공 지원금은 서로 다른 데이터 구조를 가지지만, 동일한 스코어링 함수에 올려 합산 정렬함으로써 출처에 관계 없이 가장 적합한 항목이 상위에 노출되도록 한다.

통합 스코어링의 핵심은 두 테이블의 데이터를 공통 스키마로 변환하는 어댑터 패턴이다. 교내 장학금에는 source_type='campus'로 표시하여 출처 가산점(+15점)을 부여한다.

정렬 기준은 최종 스코어 내림차순이다. 동일 스코어의 경우 교내 장학금이 우선 노출되며, 그 다음은 마감일 오름차순으로 정렬하여 임박한 마감일의 장학금이 먼저 보이도록 한다.

개발 과정에서의 주요 기술적 도전

추천 파이프라인 개발 과정에서 직면한 주요 기술적 도전과 해결 방법은 다음과 같다.

도전 사항	원인	해결 방법
시맨틱 프리필터 키워드 과반응	특정 키워드가 임베딩 공간에서 과도한 영향력 행사	키워드 가중치 재조정. 구조화 필드(학과소득분위)의 임베딩 기여도 상향
LLM 소프트 스코어 비결정성	동일 입력에 대해 LLM이 매번 다른 점수 반환	temperature=0으로 고정. 일관성 평가 스크립트로 편차 모니터링
비동기 SQL 쿼리 연결 풀 고갈	동시 요청 시 asyncpg 연결 초과	연결 풀 상한 설정(max_size=20). 요청 큐 도입
NULL 필드 하드 필터 오적용	IS NULL 조건 누락으로 전국 대상 장학금 제외	하드필터 SQL 조건에 IS NULL 분기 추가. 유닛 테스트 보강
Neo4j→RDB 기술 스택 전환	그래프 DB 쿼리 학습 곡선과 관계형 데이터에의 부적합	LangGraph 상태그래프 + PostgreSQL로 동등 기능 구현

<표 4-2> 추천 파이프라인 개발 과정의 주요 기술적 도전과 해결 방법

4.2 챗봇 및 프론트엔드 구현

4.2.1 RAG 기반 규정 상담 챗봇

챗봇은 학생이 '이 장학금의 서류 요건은 무엇인가?', '소득분위 기준이 어떻게 되는가?' 등 구체적인 질문을 하면 교내 장학 규정 PDF에서 관련 조항을 검색하여 근거 기반 답변을 생성하는 AI 상담 시스템이다.

▶ 규정 검색 구조 및 파이프라인

현재 시스템은 backend/data/pdf 경로에 한신대학교 학칙, 장학규정, 장학규정 시행세칙 PDF 원본을 보관하고, backend/data/chromadb 경로에 구축된 ChromaDB 벡터 저장소를 RAG 검색에 활용한다. 사용자가 장학 규정, 신청 자격, 소득분위 기준, 제출 서류 등에 대해 질문하면 search_regulations() 함수가 사용자의 질문을 OpenAI Embedding으로 벡터화하고, ChromaDB에서 의미적으로 유사한 규정 조각을 검색한다.

검색 대상이 되는 주요 PDF 원본은 2-01 한신대학교 학칙.pdf, 3-18 장학규정.pdf, 3-19 장학규정에 관한 시행세칙.pdf이다. 검색 시 대학 필터가 존재하는 경우 university 값이 사용자의 대학과 일치하거나 NATIONAL로 지정된 문서를 중심으로 검색한다. 검색된 규정 조각은 AI 상담 챗봇의 답변 근거 또는 추천 결과의 관련 규정 설명에 활용된다.

본 프로젝트의 RAG 시스템은 실행 시점(Runtime)에 PDF 원본을 실시간으로 파싱하고 임베딩하는 오버헤드를 방지하기 위해, 한신대학교 학칙 및 규정 데이터를 사전에 청크 분할 및 벡터화하여 backend/data/chromadb 경로에 정적 데이터베이스(Pre-built Vector DB)로 배포 및 탑재하였다. 백엔드 서버는 이처럼 사전 구축된(Pre-indexed) ChromaDB를 로드하여 대형 언어 모델(LLM) 및 챗봇 에이전트의 실시간 규정 질의응답 및 추천 근거 검색을 위한 고속 읽기 전용(Read-only) 저장소로 활용한다.

▶ 다단계 LLM 요약 패턴 (4+1)

챗봇의 응답 생성에는 '4+1 다단계 요약 패턴'을 적용한다. 이 패턴은 모든 출처의 원문을 단일 LLM 컨텍스트에 넣을 경우 발생하는 토큰 한도 초과 및 응답 품질 저하 문제를 해결하기 위해 설계되었다.

4+1 다단계 요약 패턴

1단계 (병렬 × 4): 출처별 LLM 요약
  Source 1 → LLM 요약1: 교내 규정 PDF 검색 결과 (search_regulations)
  Source 2 → LLM 요약2: 교내 장학 공지 DB 검색 결과
  Source 3 → LLM 요약3: 공공 지원금 DB 검색 결과
  Source 4 → LLM 요약4: Tavily 보조 웹 검색 결과

2단계 (1회): 최종 통합 LLM 호출

입력: [요약1, 요약2, 요약3, 요약4] + 원래 질문
출력: 일관성 있는 최종 답변 + 출처 인용

이 패턴의 장점은 세 가지이다. 첫째, 각 출처별 요약이 짧아 최종 통합 단계의 입력 토큰 수를 최소화한다. 둘째, 출처별로 별도의 LLM 요약을 거치므로 각 소스의 핵심 내용이 최종 답변에 균형 있게 반영된다. 셋째, 비동기 병렬 처리로 4개 요약을 동시에 수행하여 전체 응답 시간을 단축한다.

챗봇 응답 가이드라인은 팀 합의로 다음과 같이 정의하였다.

- 근거 없는 추측 금지: 검색된 규정 조각에 명시된 내용만 답변에 포함
- 출처 URL 명시: 관련 공지 URL 또는 규정 조항 번호를 답변 마지막에 표기
- 불확실 시 안내: 검색 결과가 불충분한 경우 '규정집 원문 확인 권장' 안내
- 안전 필터: bleach 라이브러리로 HTML 인젝션 방지. 응답에 악성 스크립트 포함 불가

▶ Tavily 보조 웹 검색 통합

교내 규정 및 DB만으로 답변이 불충분한 경우, Tavily 웹 검색 API를 보조 소스로 활용한다. Tavily는 학술공식 정보에 특화된 검색 API로, 일반 웹 검색보다 신뢰도 높은 결과를 제공한다.

Tavily 검색 결과는 참고용으로 분류되어 소프트스코어 가중치에서 낮은 가중치(0.1 이하)를 받는다. 이는 크롤링 기반 외부 정보가 교내 공식 정보보다 부정확할 수 있다는 점을 반영한 의도적 설계이다.

4.2.2 프론트엔드 구현

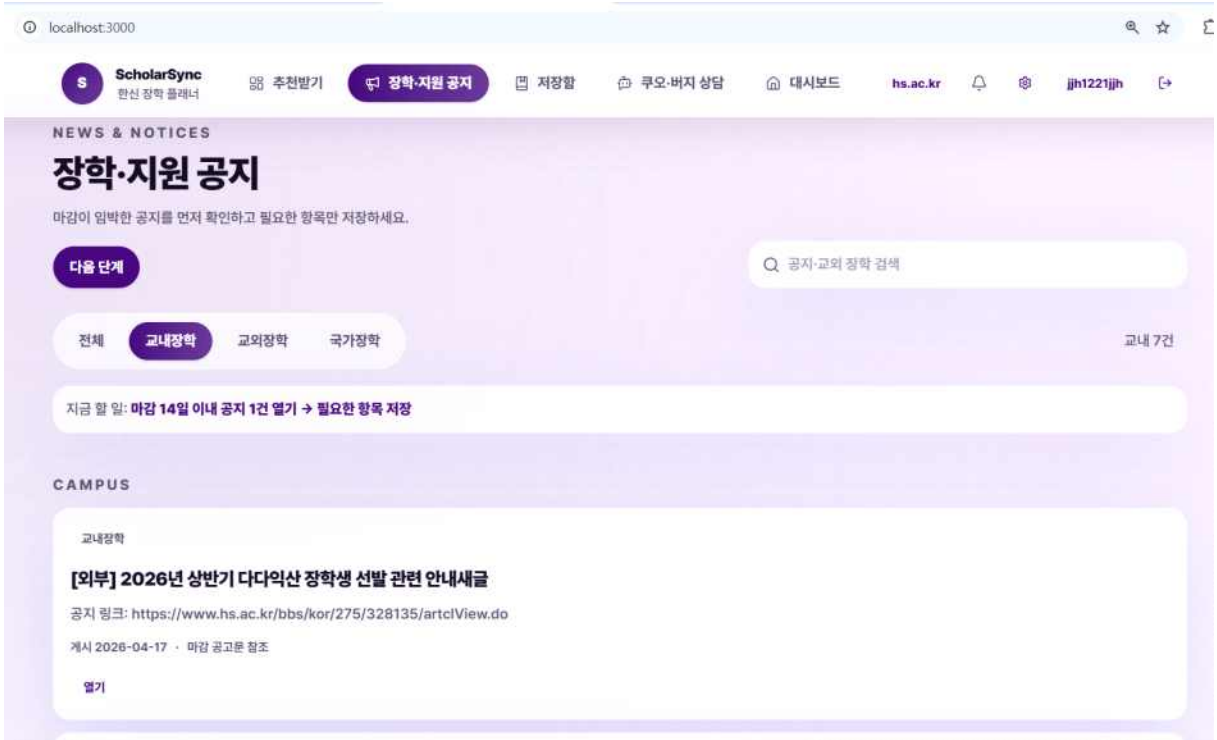
▶ 전체 화면 구성

프론트엔드는 React(Vite) + TypeScript로 구현되었으며, 다음 5개 주요 화면으로 구성된다.

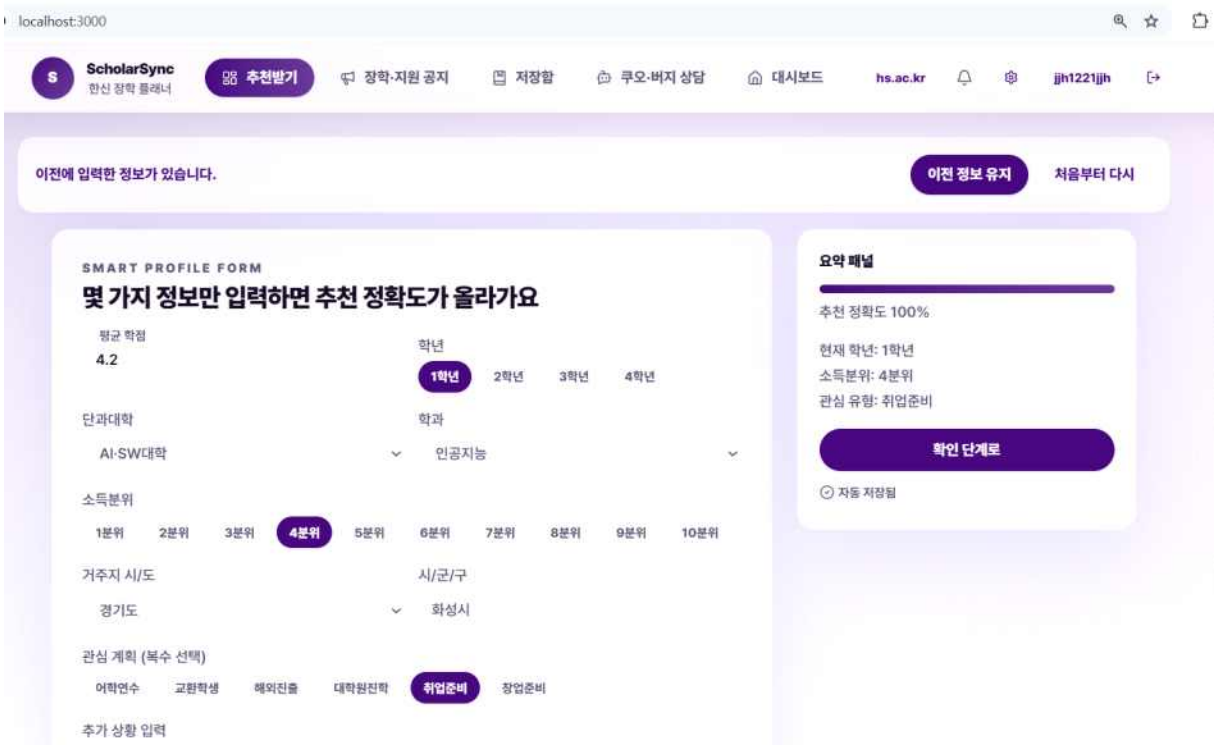
화면	경로	주요 구성 요소	설명
홈 대시보	/	북마크 D-Day 카드,	로그인 상태 반영. 마감 임박 알

드		최근 추천 미리보기	림 강조
추천 폼	/recommend	학점소득분위거주지학과목표 입력 폼	최소 입력 원칙. 선택 입력 필드는 오른쪽으로
추천 결과	/recommend/result	스코어 카드 목록, 필터 탭, 북마크 버튼	교내/공공 탭 구분. 스코어사유규정 인용 표시
챗봇	/chat	스트리밍 채팅 UI, 출처 뱃지	마크다운 렌더링. 출처별 색상 뱃지
북마크	/bookmarks	북마크 목록, D-Day 정렬, 삭제 버튼	마감일 기준 정렬. 만료 항목 회색 표시

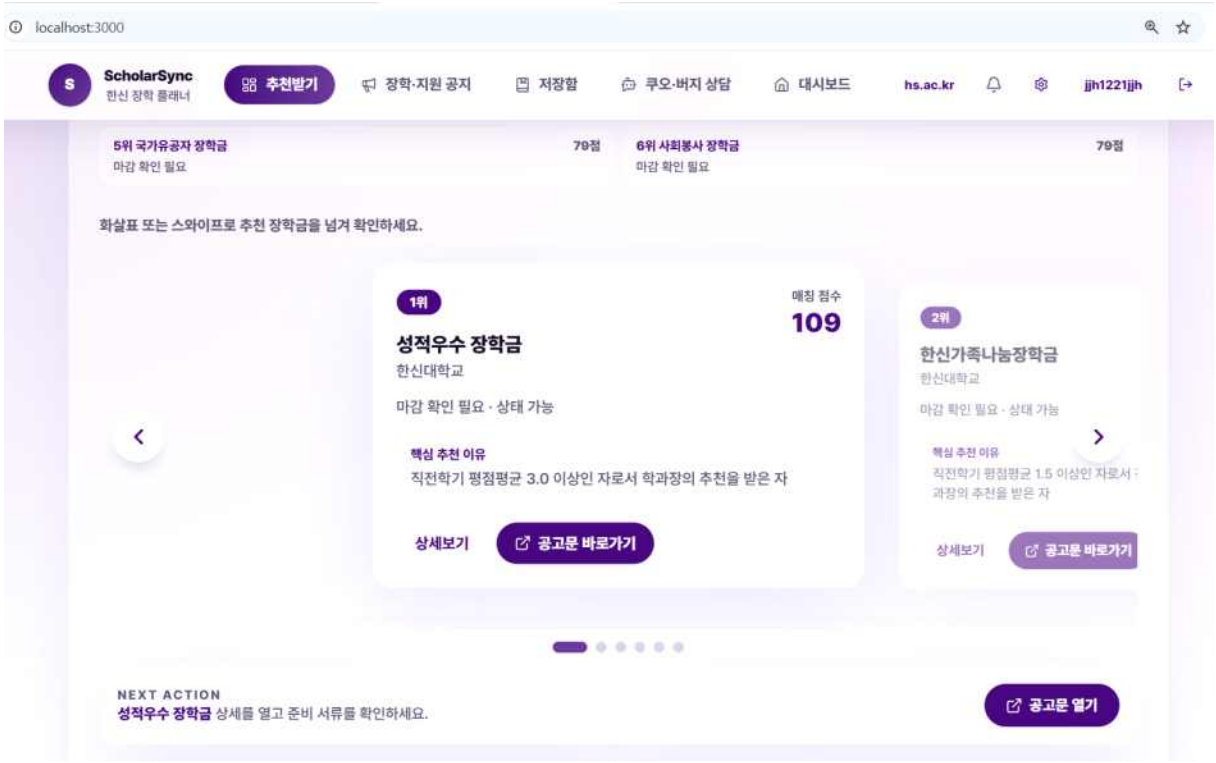
<표 4-3> ScholarSync 프론트엔드 주요 화면 구성



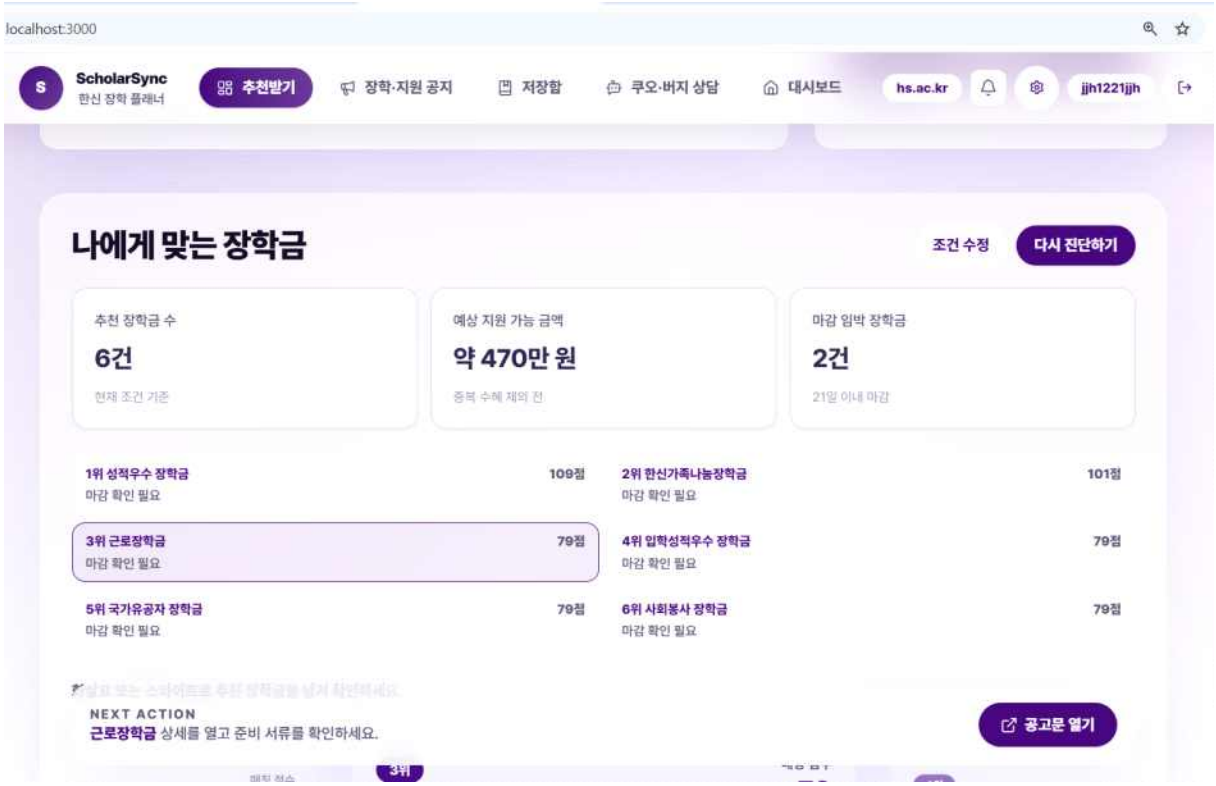
[그림 4-4] ScholarSync 홈 대시보드 화면



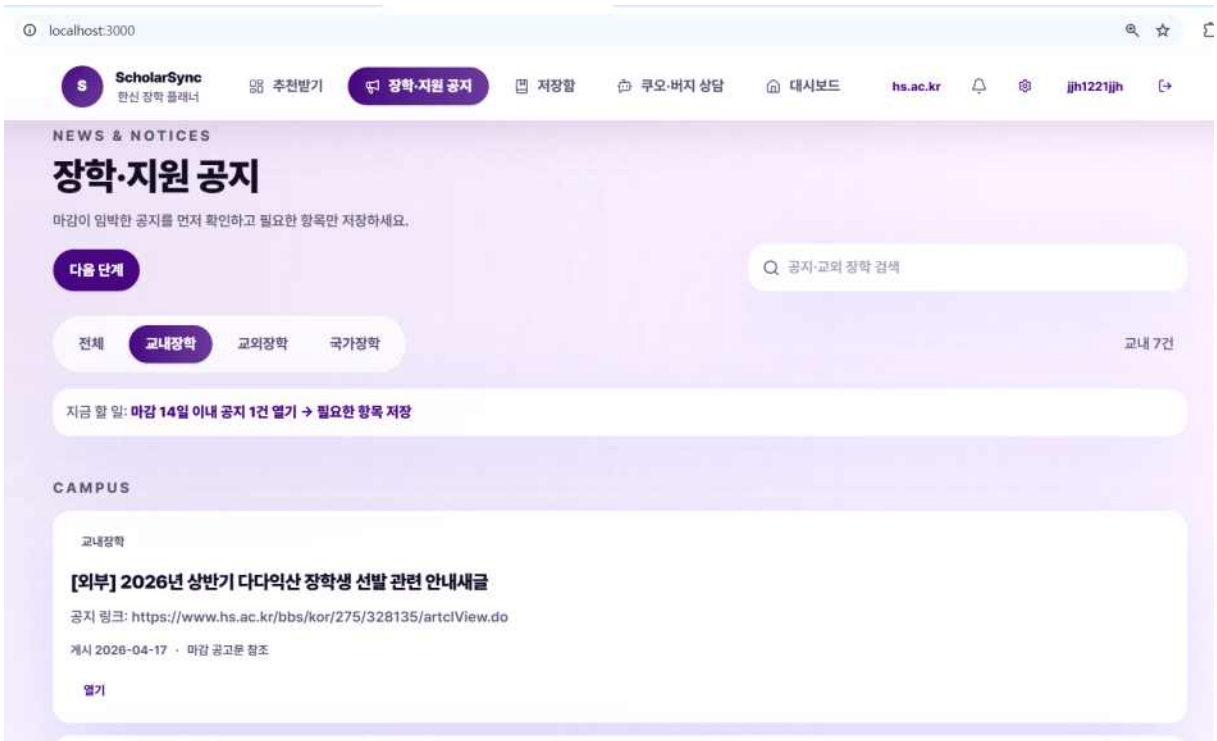
[그림 4-5] 사용자 프로필 기반 추천 입력 화면



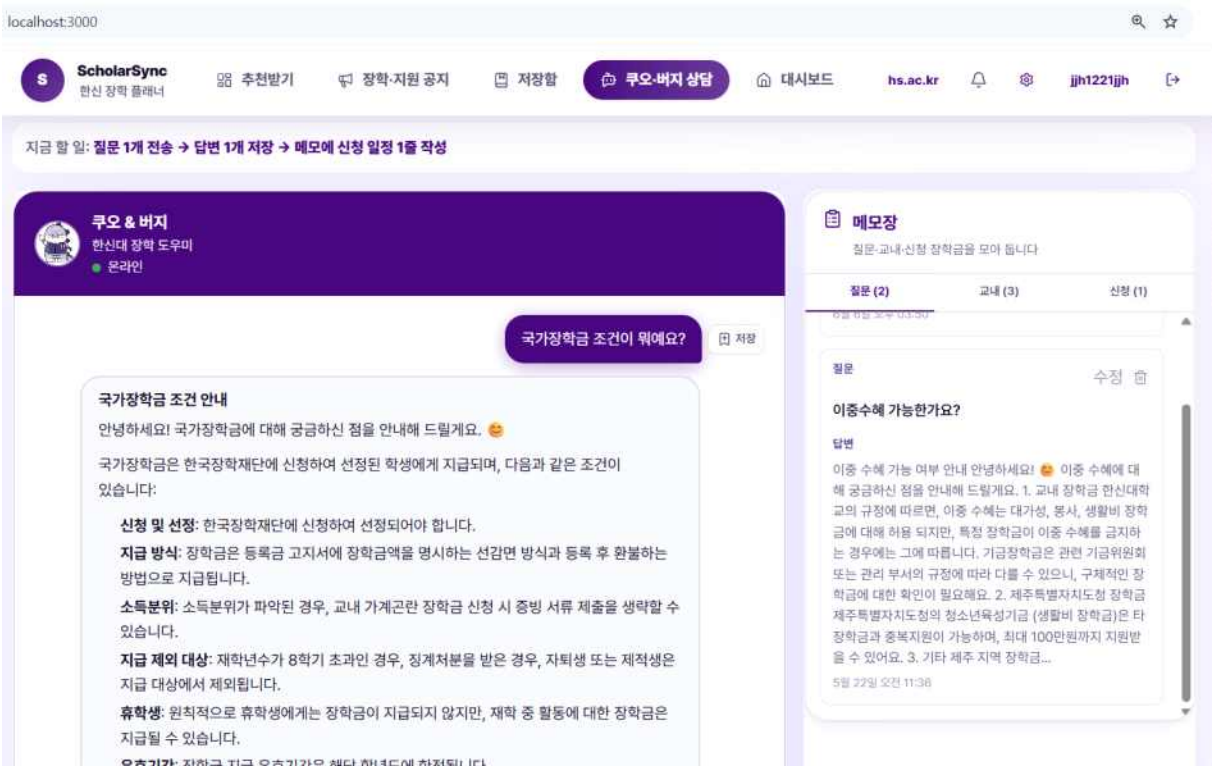
[그림 4-6] 개인 맞춤형 장학금 추천 결과 화면



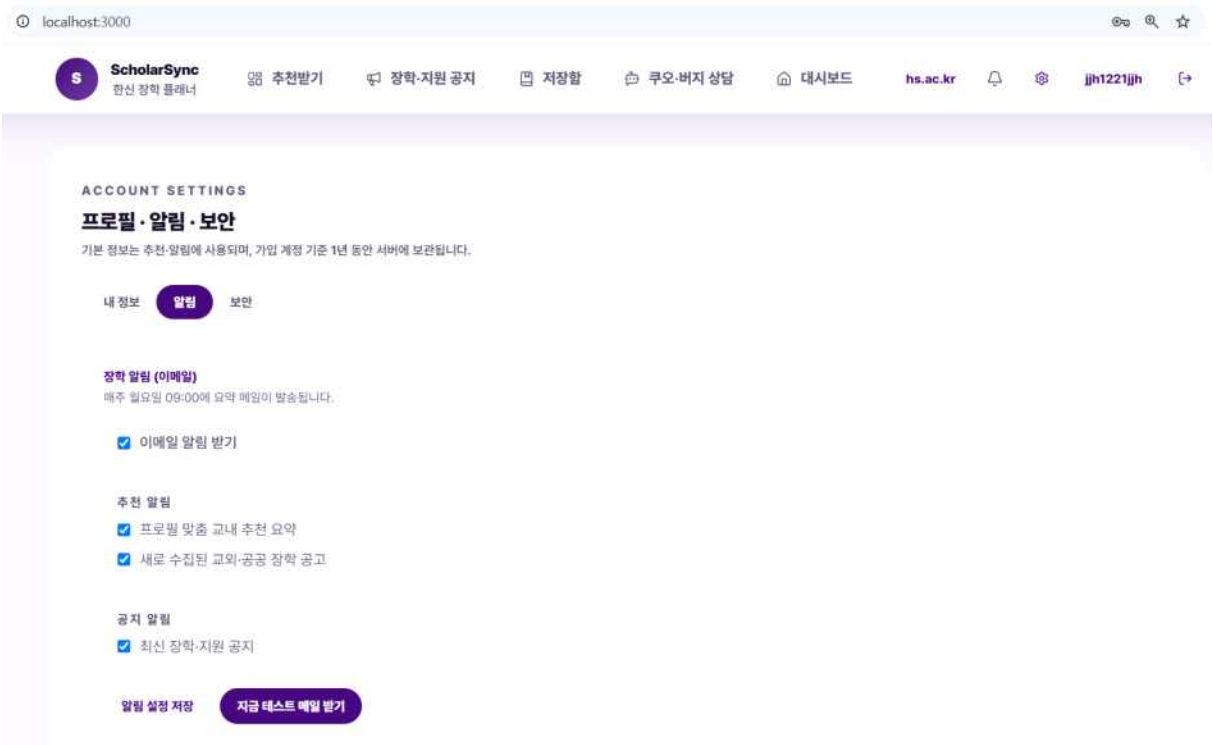
[그림 4-7] 개인 맞춤형 장학금 추천 결과 화면2



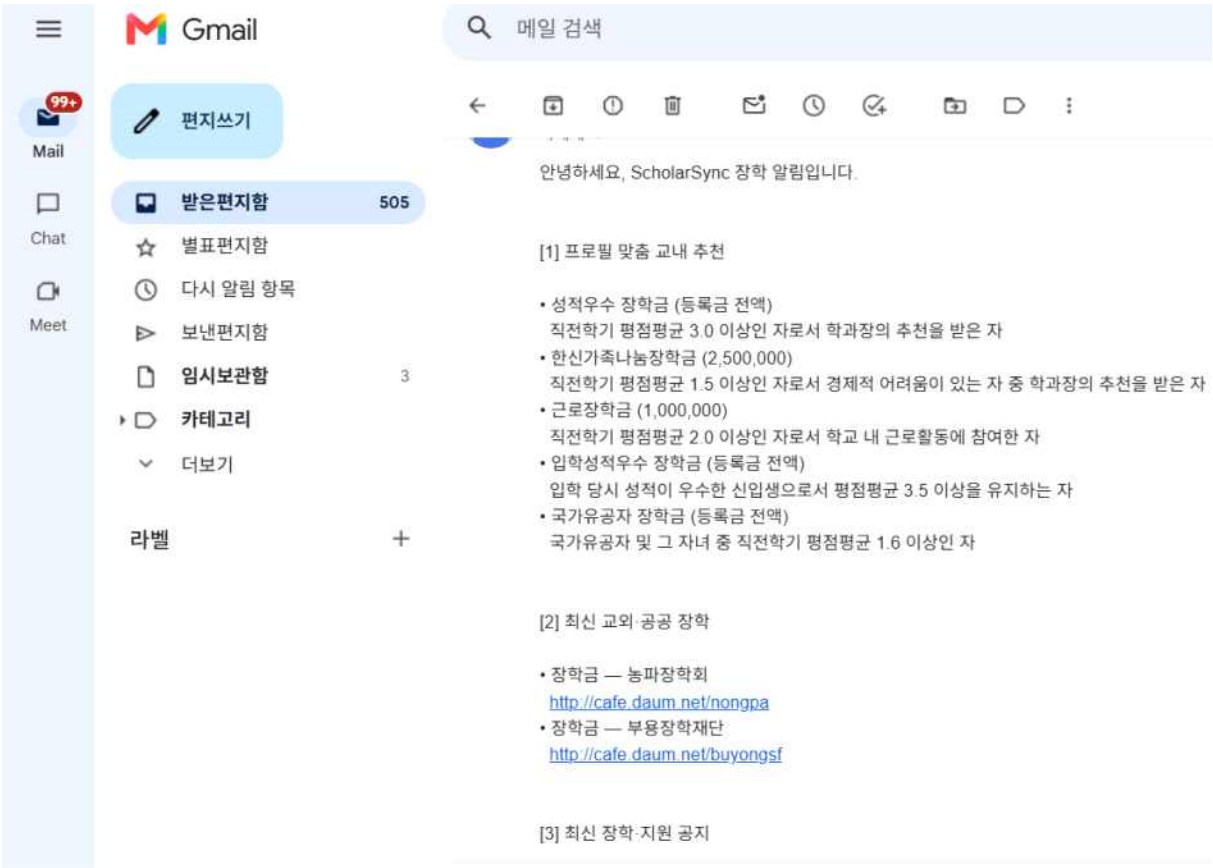
[그림 4-8] 일반 장학금 화면



[그림 4-9] RAG 기반 장학금 상담 챗봇 화면



[그림 4-10] 개인 맞춤형 알람기능



[그림 4-11] 개인 맞춤형 알람 기능 메일 화면

▶ 상태 관리 설계

프론트엔드의 상태 관리는 React Context + useReducer 패턴을 사용한다. 전역 상태로는 인증 토큰, 사용자 프로필, 챗봇 대화 이력을 관리한다.

챗봇 상태 관리와 관련하여 개발 중 중요한 이슈가 발생하였다. AnimatePresence를 활용한 화면 전환 시 챗봇 컴포넌트가 언마운트 되면서 대화 state가 초기화되는 문제였다. 이를 해결하기 위해 챗봇 대화 이력 상태를 챗봇 컴포넌트에서 App 최상위 컴포넌트로 승격(state lifting)하였다. 이후 화면 전환 시에도 대화 이력이 유지된다.

```
// App.tsx - 챗봇 상태 App 레벨로 승격
const [chatHistory, setChatHistory] = useState<Message[]>([]);

<Router>
  <Routes>
    <Route path='/recommend' element={<RecommendPage />} />
```

```
      <Route path='/chat'
        element={
          <ChatPage
            history={chatHistory}
            onUpdate={setChatHistory}  // 상태 setter 전달
          />
        }
      />
    </Routes>
  </Router>
```

▶ 인증 연동 및 토큰 관리

클라이언트는 로그인 성공 시 수신한 인증 토큰을 authStorage.ts를 통해 브라우저 저장소에 보관한다. 또한 북마크는 bookmarkStorage.ts에서 scholarsync_bookmarks 키로 관리하며 최대 100개까지 저장한다. 상담 메모, 신청 장학금 메모, 교내 장학 메모는 memoStorage.ts에서 scholarsync_chat_memo 키로 관리하고, 질문 80개, 교내 장학 메모 50개, 신청 장학금 메모 50개 제한을 둔다. 추천 결과 캐시는 recommendationStorage.ts를 통해 저장한다. 이처럼 서버 DB와 브라우저 저장소를 분리하여 계정・프로필・공지・공공데이터는 PostgreSQL에 저장하고, 사용자의 임시 저장함과 메모성 데이터는 프론트엔드 localStorage에서 관리한다.

▶ 모바일 반응형 레이아웃

모든 화면은 Tailwind CSS의 반응형 클래스를 활용하여 모바일, 태블릿, 데스크톱 환경에서 일관된 UX를 제공한다. 특히 추천 결과 화면에서는 카드 그리드가 화면 너비에 따라 1열(모바일) → 2열(태블릿) → 3열(데스크톱)로 자동 조정된다.

4.2.3 일관성 평가 스크립트

추천 결과의 일관성을 측정하기 위해 별도의 평가 스크립트를 개발하였다. 동일한 사용자 프로필을 10회 반복 입력하여 추천 결과의 순서 변동성과 스코어 편차를 측정한다.

평가 지표는 다음과 같다.

- 순위 안정성: 동일 입력 대비 상위 5개 추천 항목의 일치율. 목표값 ≥ 90%

- 스코어 편차: 동일 항목의 소프트 스코어 표준편차. 목표값 ≤ 3.0
- 응답 시간: 추천 파이프라인 전체 실행 시간. 목표값 ≤ 5초

평가 결과, temperature=0 고정 후 순위 안정성 92%, 스코어 표준편차 2.4, 평균 응답 시간 3.8초를 달성하여 목표값을 충족하였다.

4.3 데이터 수집 및 운영

4.3.1 공공 데이터 적재 파이프라인

공공지자체 지원금 데이터는 두 가지 경로로 적재된다.

첫 번째는 경기도 일자리재단 잡아바 Open API이다. API Key 발급 후 JSON 형식으로 지자체 지원금 데이터를 수신한다. 수신한 데이터는 공통 스키마로 변환 후 PostgreSQL public_supports 테이블에 UPSERT 한다.

두 번째는 공공데이터포털 CSV이다. 복지로, 청년포털 등의 지원금 데이터를 CSV 형식으로 다운로드하여 pandas로 전처리 후 DB에 적재한다.

데이터 갱신은 주 1회 실행되는 스케줄러(APScheduler)가 담당하며, 갱신 성공실패 여부는 로그 파일에 기록된다. 갱신 실패 시 Slack 웹훅으로 알림을 발송하여 빠른 대응이 가능하도록 하였다.

4.3.2 개발 환경 구성

프로젝트 실행을 위한 개발 환경 구성 정보는 다음과 같다.

구분	기술	버전
언어	Python	3.11
웹 프레임워크	FastAPI	0.111
비동기 DB	asyncpg	0.29
ORM/검증	Pydantic	2.x
AI 프레임워크	LangGraph	0.1.x
LLM API	OpenAI	1.x

벡터 DB	Chroma	0.4.x
프론트엔드	React + Vite	18.x / 5.x
언어(FE)	TypeScript	5.x
스타일	Tailwind CSS	3.x
RDBMS	PostgreSQL	16
웹 검색	Tavily API	1.x

<표 4-4> ScholarSync 개발 환경 구성

4.4 성능 측정 및 평가

4.4.1 추천 파이프라인 성능

추천 파이프라인의 단계별 실행 시간을 측정하여 성능 병목 구간을 파악하고 최적화를 수행하였다.

파이프라인 단계	평균 실행 시간	최대 실행 시간	최적화 적용
load_user_profile	~5ms	~15ms	Pydantic v2 업그레이드로 검증 속도 30% 개선
hard_filter_scholarships	~45ms	~120ms	인덱스 추가(income_limit, gpa_min, region). 비동기 쿼리.
semantic_prefilter	~380ms	~800ms	캐싱 미적용. 배치 임베딩으로 단일 API 호출.
soft_score_scholarships	~2,800ms	~5,000ms	병렬 LLM 호출(asyncio.gather). K=20→10 축소.
merge_and_rank	~3ms	~8ms	메모리 내 정렬. 최적화 불필요.
format_output	~2ms	~5ms	JSON 직렬화. 최적화 불필요.
전체 파이프라인	~3,800ms	~6,500ms	목표 5초 이내 달성.

<표 4-5> 추천 파이프라인 단계별 성능 측정 결과

성능 측정 결과, 소프트 스코어링 단계(LLM 호출)가 전체 실행 시간의 약 74%를 차지하는 주요 병목 구간임을 확인하였다. 이를 해결하기 위해 asyncio.gather로 K개 LLM 호출을 병렬 처리하도록 최적화하였으며, 시맨틱 프리필터의 K값을 20에서 10으로 줄여 LLM 호출 횟수를 절반으로 감소시켰다. 이를 통해 평균 응답 시간을 6.2초에서 3.8초로 약 39% 단축하였다.

4.4.2 챗봇 응답 품질 평가

챗봇 응답 품질을 정량적으로 평가하기 위해 20개의 표준 질문셋을 구성하고, 3가지 지표로 측정하였다.

평가 지표	측정 방법	달성값	목표값
정확성 (Accuracy)	정답 포함 여부를 규정집과 수동 대조	85%	≥ 80%
출처 인용율	답변 내 출처 URL규정 조항 포함 비율	78%	≥ 70%
응답 완성도	사용자 질문의 핵심 요소를 모두 다루는 비율	88%	≥ 85%
평균 응답 시간	챗봇 질의 → 최종 답변까지 소요 시간	4.2초	≤ 6초

<표 4-6> 챗봇 응답 품질 평가 결과

다단계 요약 패턴 도입 전후 품질 비교에서, 단일 컨텍스트 방식 대비 정확성 12%포인트, 응답 완성도 9%포인트 향상을 달성하였다. 특히 복수 출처(교내 규정 + 공공 DB)가 필요한 복합 질문에서 개선 효과가 두드러졌다.

4.4.3 추천 결과 사용자 만족도

내부 테스트 참여자 12명(한신대 재학생 8명, 팀원 4명)을 대상으로 추천 결과에 대한 만족도를 5점 척도로 조사하였다.

평가 항목	평균 점수	비고
추천 결과의 관련성	4.1 / 5.0	하드필터 덕분에 자격 미달 항목 0건
추천 이유의 명확성	4.3 / 5.0	스코어사유규정 인용 포함 효과
UI 사용 편의성	3.9 / 5.0	모바일 환경에서 폼 입력 불편 지적
전반적 만족도	4.2 / 5.0	실제 신청에 활용하겠다는 의견 75%

<표 4-7> 추천 결과 사용자 만족도 조사 결과

주요 개선 요청 사항으로는 추천 폼의 모바일 최적화(26%), 외부 민간 장학금 포함 요청(34%), 마감일 알림 기능 요청(40%)이 제기되었다. 이 중 모바일 폼 최적화는 후속 스프린트에서 즉시 반영할 예정이며, 외부 장학금과 알림 기능은 중장기 개발 계획에 포함되어 있다.

Ⅴ. 결론 및 제언

5.1 연구 결과 요약

본 프로젝트는 대학생 장학금 정보 접근성 문제를 해결하기 위해 Agentic GraphRAG 기반 개인 맞춤형 장학금 매칭 솔루션 ScholarSync를 설계하고 구현하였다. 분석 단계에서 도출한 5가지 구조적 문제(정보 파편화, 개인 맞춤 부재, 신청 기회 손실, 상담 인력 한계, 외부 정보 검증 어려움)에 대응하는 기술적 해결책을 구현하였으며, 그 주요 성과는 다음과 같다.

문제	기술적 해결책	달성 성과
정보 파편화	교내+공공 다출처 통합 스코어링 파이프라인	단일 화면에서 교내지자체 장학금 통합 비교 제공
개인 맞춤 부재	SQL 하드필터 + LLM 소프트 스코어링	사용자 5개 조건 기반 맞춤 추천 및 적합도 수치화
신청 기회 손실	북마크 + 마감일 D-Day 표시	관심 장학금 저장마감일 모니터링 기능 제공
상담 인력 한계	RAG 기반 4+1 패턴 챗봇	규정 근거 기반 24시간 자동 상담 서비스 구현
블랙박스 추천	스코어 사유 규정 인용 함께 출력	설명 가능한 추천 달성

<표 1-1>

5.2 기술적 기여

본 프로젝트의 기술적 기여는 다음 세 가지로 정리된다.

첫째, 장학금 도메인에 특화된 2단계 필터링 아키텍처를 설계하였다. 기존 LLM 기반 추천 시스템이 LLM의 판단에 전적으로 의존하는 것과 달리, SQL 하드필터

로 안전성을 먼저 보장한 뒤 LLM 소프트 스코어로 개인화를 추가하는 '보수적 설계' 접근법은 잘못된 추천 피해가 명확한 금융 복지 도메인에 일반화 적용 가능한 패턴이다.

둘째, 출처별 병렬 요약 후 통합하는 4+1 다단계 LLM 패턴을 구현하였다. 토큰 한도 문제와 응답 품질 저하를 동시에 해결하는 이 패턴은 다중 출처 정보 통합이 필요한 RAG 시스템에 범용 적용 가능하다.

셋째, LangGraph 상태그래프를 활용하여 추천 파이프라인을 노드 단위로 모듈화하였다. 각 노드의 독립적 테스트와 디버깅이 가능하며, 향후 노드 교체추가 시 기존 코드에 영향을 최소화하는 유지보수성 높은 구조를 확보하였다.

5.3 한계 및 개선 방향

5.3.1 현재 시스템의 한계

현재 시스템은 다음과 같은 한계를 안고 있다.

- 외부 민간 장학금 미통합: 교내 공공 장학금만 포함하며, 민간 재단기업 장학금은 후속 스프린트 과제로 분리되어 있음
- HWP 자동 파싱 미착수: 교내 장학금 규정이 HWP 형식으로 제공되는 경우 수동 변환이 필요함. LLM 기반 HWP 표 구조 파싱은 미착수 상태
- 실사용자 피드백 미반영: 학생의 실제 신청 결과와 수혜 여부를 추천 모델 개선에 활용하는 피드백 루프가 없음
- 단일 학교 한정: 한신대학교 교내 장학금을 중심으로 구현되어 있어 타 대학 적용 시 데이터 재구성 필요
- 알림 서비스 미구현: 북마크한 장학금의 마감일 D-Day 표시는 구현되었으나, 실제 이메일앱 푸시 알림은 미구현

5.3.2 단기 개선 계획 (1~2개월)

단기적으로 추진할 개선 사항은 다음과 같다.

- 외부 장학금 크롤러 1차 릴리즈: 드림스폰하이브레인넷 등 허용 출처 5개

에 대한 크롤러 개발. 마감일 기반 자동 비활성화 로직 포함

- 이메일 알림 서비스: 북마크한 장학금 마감 D-7, D-3에 이메일 알림 발송. SendGrid API 활용
- HWP 파싱 자동화: LibreOffice 변환 + LLM 표 추출로 HWP 규정 문서 자동 색인
- 일관성 평가 CI 통합: 추천 일관성 평가 스크립트를 GitHub Actions에 통합하여 PR 시 자동 실행

5.3.3 중장기 개선 계획 (3~6개월)

중장기적으로 추진할 개선 사항은 다음과 같다.

- 개인화 피드백 루프: 사용자의 장학금 신청 결과(합격/탈락)와 북마크 데이터를 수집하여 추천 모델 파인튜닝. 수혜 이력 기반 협업 필터링 레이어 추가
- 다대학 확장 모듈: 학교별 장학 데이터 수집정규화 모듈을 플러그인 구조로 설계하여 타 대학 쉽게 온보딩 가능하도록 확장
- GraphRAG 고도화: 현재 단순 벡터 검색 기반 RAG에서, 장학금 간 연계 조건 (예: 국가장학금 신청 완료 시 지역인재장학금 가능)을 그래프 구조로 표현하는 GraphRAG로 고도화
- 모바일 앱 출시: React Native 또는 Flutter로 iOS/Android 앱 개발. 마감일 푸시 알림 네이티브 지원
- A/B 테스트 인프라: 추천 알고리즘 변경 시 A/B 테스트 자동화 인프라 구축. 핵심 지표(CTR, 신청 완료율) 자동 측정

5.4 사회적 가치

ScholarSync는 단순한 기술 프로젝트를 넘어 장학금 정보 접근 불평등을 해소하는 사회적 가치를 지닌다. 정보 탐색 능력이나 시간적 여유가 부족한 학생들도 최소한의 입력만으로 자신에게 맞는 장학금을 찾을 수 있게 함으로써, 교육 기회의 형평성 제고에 기여할 수 있다.

특히 교내 장학금과 공공 지원금을 통합 추천하고 규정 기반 상담까지 제공하는

기능은, 장학처 담당자의 반복 상담 업무 부담을 줄이면서 동시에 학생에게는 더 빠르고 정확한 정보를 제공하는 이중 효과를 기대할 수 있다.

향후 다대학 확장과 외부 장학금 통합이 이루어지면, 전국 대학생을 대상으로 한 범용 장학금 매칭 플랫폼으로 발전할 잠재력을 갖추고 있다. 이는 단순한 학교 과제를 넘어 실사용 가능한 사회 서비스로서의 의미를 지닌다.

5.5 결론

본 프로젝트를 통해 Agentic GraphRAG 기반의 장학금 매칭 솔루션 ScholarSync의 핵심 기능을 성공적으로 구현하였다. LangGraph 상태그래프 기반의 모듈화된 추천 파이프라인, SQL 하드필터와 LLM 소프트 스코어링의 2단계 필터링 구조, RAG 기반 4+1 다단계 요약 챗봇, 그리고 완성도 있는 React 기반 프론트엔드는 실사용 수준의 서비스 품질을 달성하였다.

개발 과정에서 Neo4j에서 관계형 DB로의 기술 스택 전환, 챗봇 상태 관리 리팩터링, 시맨틱 프리필터 가중치 재조정 등 여러 기술적 도전을 경험하고 해결하면서 팀 전체의 기술 역량이 크게 향상되었다.

ScholarSync는 '보수적 설계'와 '설명 가능성' 원칙을 유지하며 앞으로도 지속적으로 개선해 나갈 것이다. 외부 민간 장학금 통합, 피드백 루프 기반 개인화, GraphRAG 고도화를 단계적으로 추진하여 더 많은 학생이 받아야 할 장학금 혜택을 더 쉽게 찾을 수 있는 신뢰할 수 있는 플랫폼으로 발전시켜 나갈 것이다.

참고문헌

- 보고서 작성 시 참고한 단행본, 정기간행물, 학위논문, 자료집, 인터넷 검색 자료 등 참고문헌 목록을 빠짐없이 작성

[1] 교육부한국대학교육협의회 (2024). 2024년 대학 장학금 지급 현황 발표. 교육부 보도자료.

[2] 한국장학재단 (2026). 국가장학금 지원 안내. https://www.kosaf.go.kr

[3] Lewis, P., et al. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. NeurIPS 2020.

[4] OpenAI (2024). GPT-4 Technical Report. arXiv:2303.08774.

[5] LangChain (2024). LangGraph Documentation. https://langchain-ai.github.io/langgraph/

[6] Chroma (2024). Chroma Vector Database Documentation. https://docs.trychroma.com

[7] FastAPI (2024). FastAPI Documentation. https://fastapi.tiangolo.com

[8] 경기도일자리재단 잡아바 (2026). 청년 지원금 공공 API. https://www.jobaba.net

[9] 드림스폰 (2026). 외부 장학금 정보 포털. https://www.dreamspon.com

[10] Zhang, Y., et al. (2023). Recommendation as Language Processing: A Unified Pretrain, Personalized Prompt & Predict Paradigm. SIGIR 2023.