포트폴리오 2026 / 개발자

기획부터 운영까지 끝까지 직접 해 보는 개발자입니다.

문제를 찾으면 직접 조사하고 만들어 본 뒤, 아니다 싶으면 빨리 방향을 바꿉니다.
온라인 스토어를 직접 기획하고 주문 자동화 서버로 하루 최대
1,378건을 처리했습니다.
기업 현장실습과 공공기관 협력 프로젝트에서는 기존 업무에 필요한
시스템을 개발했습니다.



숫자로 보면

1,378건
하루 최대 주문 · 혼자 운영

396개
주문 서버 테스트 함수 · 11/03 기준

1,809건
복지사업 RAG 검색 데이터 · 매일 갱신

수상 3회
은상 · 캡스톤 우수상 · 기관 표창

수상

2025 컴퓨터인공지능학부 작품경진대회 은상
2025 캡스톤디자인 우수상
2025 전북특별자치도사회서비스원 성과보고회 표창

배지혁

전북대학교 컴퓨터공학부 / qowlgur121@gmail.com / github.com/qowlgur121 / qowlgur121.github.io

소개

학력 · 경력 · 기술

학력

전북대학교	컴퓨터공학부 공학사 · 2022.03 ~ 2026.02
성적	전체 평점 3.71/4.5 · 전공 평점 3.83/4.5 3학년 2학기 전 과목 A+ · 학과 1등 (4.50/4.5) 4학년 1학기 4.41 · 2학기 4.17

경력

(주)텔로스	백엔드 개발 현장실습 · 2025.07 ~ 2025.08

수상

2025.12	작품경진대회 은상 · 전북대 컴퓨터인공지능학부
2025.12	캡스톤디자인 우수상 · 전북대학교
2025.11	성과보고회 표창 · 전북특별자치도사회서비스원

자격증 · 활동

자격증	정보처리기사 · SQLD · ADsP
JAVICE	SW 창업동아리 · 2023.09 ~ 2025.12
ALPS	알고리즘 동아리 스터디 멘토 · 2022.03 ~ 2023.07

기술

Java	현장실습 · OpenContext · JeonjuEats 백엔드
Spring Boot	JPA · Spring Security · JWT 포함
Python	주문 자동화 시스템 · 운영 스크립트
FastAPI	복지 AI 챗봇 백엔드 · SSE 스트리밍
PostgreSQL	pgvector 벡터 검색 · OpenContext 메타데이터
Docker	Docker Compose · Nginx로 서버 배포

01

주문 자동화 시스템과 온라인 스토어 운영

대표 프로젝트 · 개요와 구조

개인 · 기획부터 운영까지 혼자

2026.09 ~ 2026.10

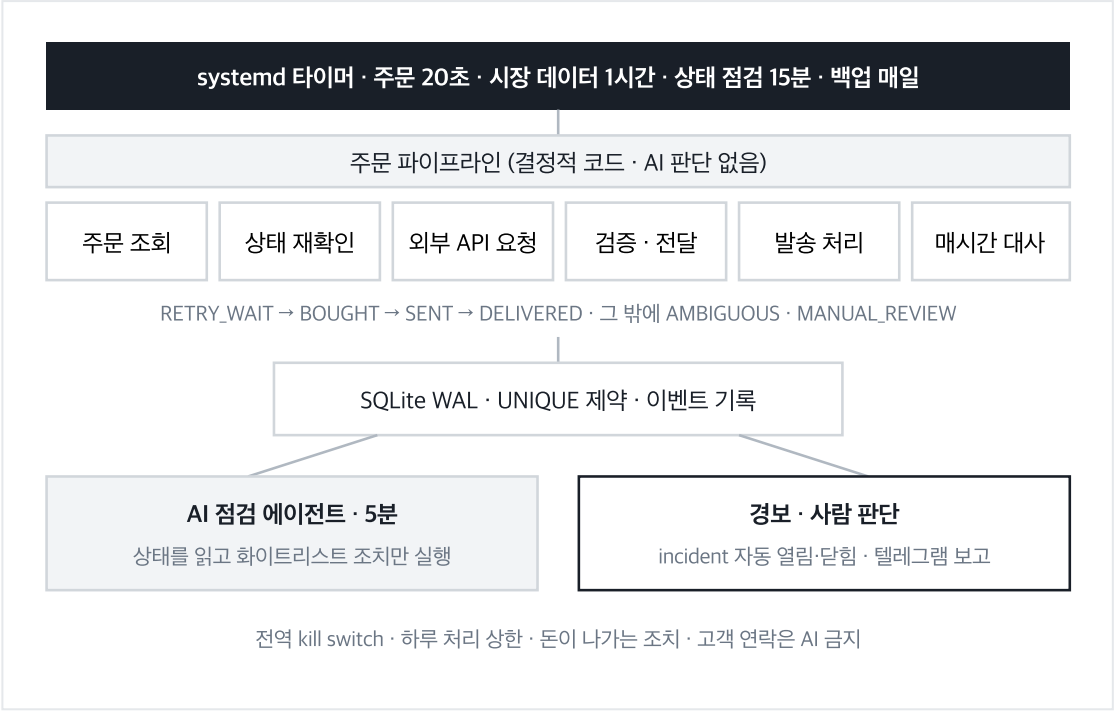
온라인 판매를 직접 기획하고, 주문이 늘어나도 혼자 처리할 수 있는 서버를 먼저 만들었습니다. 결제 확인부터 발주·전달·정산 대조를 자동화해 판매 6일째 하루 1,378건을 처리했습니다.

Stack	Python · 외부 의존성 최소화 · SQLite WAL · unittest
Infra	Oracle Cloud ARM VM · Ubuntu · systemd 타이머 · 전용 계정과 ProtectSystem=strict
연동	네이버 커머스·검색광고 API · 외부 주문 API · Telegram Bot
AI	AI 점검 에이전트 (5분) · 운영 조회 에이전트 (읽기 전용)
확장	단일 서버 기준 설계. 부하 근거가 생기면 PostgreSQL로 옮기는 기준을 문서로 정해 둠

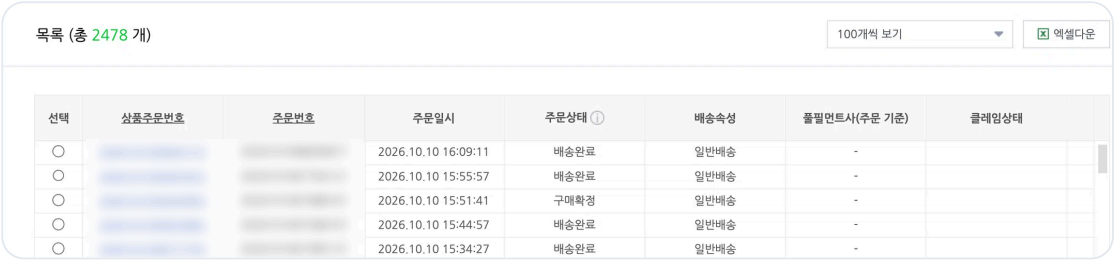
1,378건
하루 최대 주문 (10/07)

2,467건
9일간 누적 주문

396개
테스트 함수



시스템 구조. 회색 상자는 AI 에이전트 · 테두리 상자는 사람 판단



판매자센터 주문 목록 (주문번호는 가림)

01

판매 전략부터 운영까지 직접 판단하고 개선했습니다

시장 조사 · 실험 · 실제 사용자가 있는 운영

주문 자동화 · 서비스 운영

2026.10.02 ~ 10.10

문제 → 확인한 데이터 → 실행한 변경

단계	문제	결정과 결과
출시 전	판매가 늘어도 주문을 혼자 처리할 수 있을까	처리 서버부터 만들고 주문·전달·정산 대조를 자동화해 출시했다
유입	검색 노출과 구매 전환 중 어디가 문제인가	검색 순위·유입 변화를 기록하고 상품 정보·광고를 실제 데이터에 따라 수정했다
광고	클릭 수가 늘어도 이익이 늘어난다는 보장은 없다	광고 ON/OFF 시간대를 비교하고 주문당 광고비와 마진을 기준으로 예산을 조정했다
가격	판매량만 보고 가격을 바꿔도 될까	2시간 단위로 가격·시간당 이익을 비교하고 유리한 근거가 있을 때만 변경했다
지표	시장 수요와 내 판매 변화를 어떻게 구분할까	자체 주문·유입 추이와 관측 가능한 판매 데이터를 비교해 가격 판단 근거를 정했다
고객	구매 전후 반복 문의가 늘어나면	반복 질문은 안내·상담 챗봇으로 묶고 직접 확인해야 할 문의에 집중했다
AI 운영	AI가 운영 판단을 어디까지 맡아야 할까	시장·이상 징후 분석은 AI에 맡기고 구매·전달은 규칙 코드로 분리했다
배포	운영 중 코드 변경으로 기존 주문이 멈추지 않으려면	주문 처리의 핵심 경로는 안정적으로 유지하고 회귀 테스트 확인 뒤 반영했다

결과

1,378건

판매 6일째 하루 최대 · 9일간 2,467건

서비스 경험

실제 사용자의 문의도 운영 데이터였습니다

자주 묻는 질문을 안내·상담 챗봇으로 묶고, 개별 확인이 필요한 문의만 직접 보도록 구분했습니다.

문의와 리뷰에서 반복되는 질문은 안내 문구와 운영 흐름에 반영했습니다.

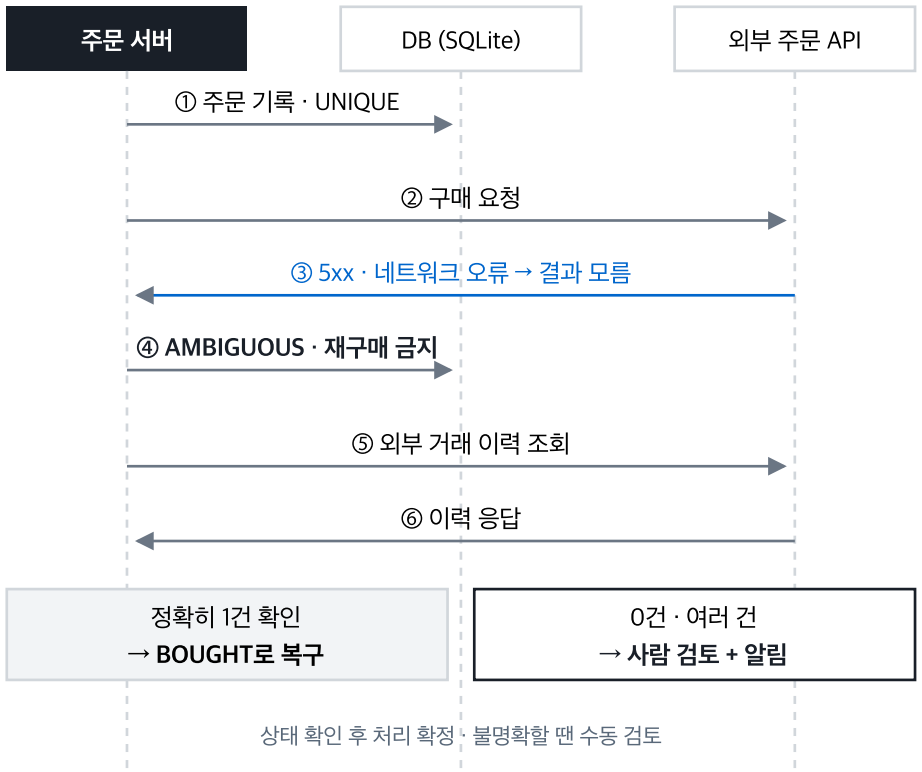
설계 원칙

AI를 쓰되 거래의 책임은 코드에 남겼습니다

시장 관측과 이상 징후 정리는 AI가 돕도록 했고, 실제 결제·전달은 상태 검사와 명시적 규칙으로 처리했습니다. 결과가 불명확한 작업은 재실행하지 않고 확인하도록 설계했습니다.

외부 API가 응답하지 않아도 중복 주문을 막는 상태 관리

문제 해결 1



구매 결과를 모를 때의 처리 순서

문제

외부 주문 API의 응답이 끊기면 요청의 처리 여부를 알 수 없었습니다. 재시도만으로 복구하면 하나의 주문이 중복 처리될 수 있어, 확인 가능한 상태 관리가 필요했습니다.

설계 판단

자동 재시도	동일 요청이 중복 실행될 수 있음
즉시 실패	실제 처리 여부와 내부 상태가 어긋날 수 있음
선택한 방식	**상태 보류 → 거래 이력 확인 → 결과 확정**

처리 결과가 불명확하면 **AMBIGUOUS** 상태로 보류했습니다. 외부 거래 이력에서 일치하는 처리 내역을 찾은 뒤 복구하고, 판단이 어려운 건은 수동 확인으로 넘겼습니다.

주문번호 **UNIQUE** 제약과 조건부 UPDATE로 동일 주문의 상태 전이를 보호했습니다.

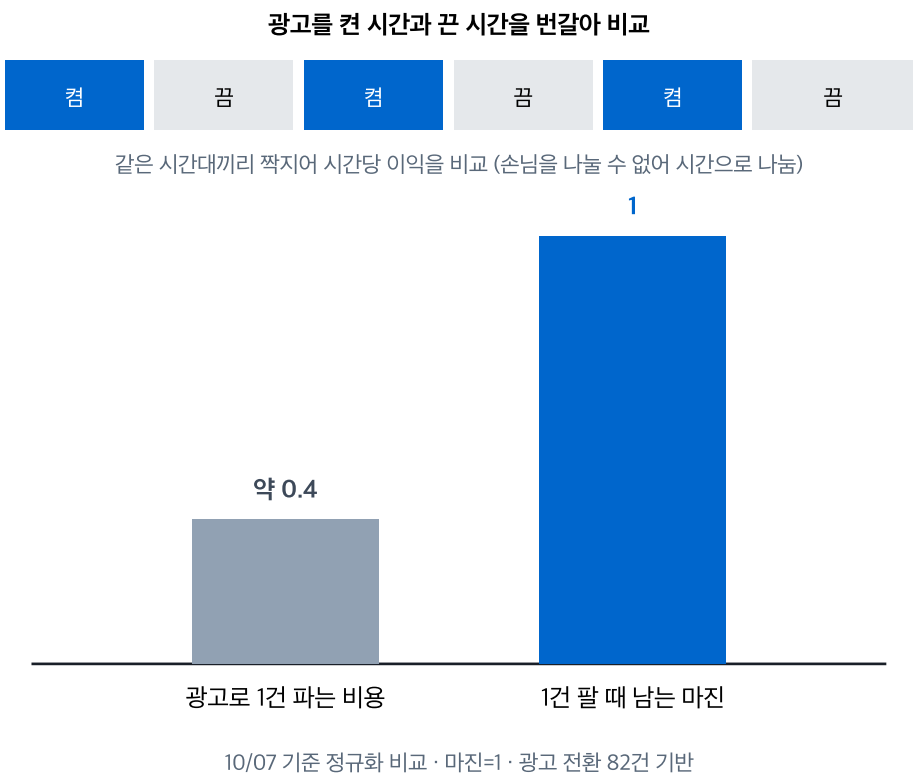
결과

불명확한 주문을 무조건 재실행하지 않고 **대조 후 복구하는 경로**를 마련했습니다. 판매 6일째 하루 최대 1,378건을 자동 처리했고, 중복 처리 관련 예외는 회귀 테스트에 반영했습니다.

운영 환경 · Oracle Cloud ARM VM 1대 · SQLite WAL · 20초 주기 조회
테스트 함수 396개는 이후 개선을 포함한 11월 3일 기준
핵심 로직 예제 코드 · github.com/qowlgur121/order-fulfillment-guard

광고를 켜야 할 시점을 주문당 이익으로 판단했습니다

문제 해결 2



광고 판단 방식과 하루 기준 비교

문제

광고 클릭 수나 노출량이 늘어도 실제로 이익이 남는지는 별개의 문제였습니다. 예산을 늘리기 전에 **주문당 비용과 마진**을 함께 확인할 기준이 필요했습니다.

해결

광고를 켠 시간과 끈 시간을 번갈아 기록하고, 비슷한 시간대의 **주문당 광고비·마진·시간당 이익**을 비교했습니다.

손익 기준에 미달하면 광고를 끄도록 규칙을 두고, 주문 유입경로와 판매 흐름을 함께 수집해 예산 판단에 반영했습니다.

> **핵심 선택** · 클릭 수 최적화보다 실제 주문 수익성을 우선

확인한 결과

10월 7일 광고 전환 주문 82건에서 주문당 광고비가 마진의 약 40% 수준임을 확인했습니다. 이를 예산 판단 근거로 활용했습니다.

조건 · 10월 7일 광고 전환 주문 기준 · 판매자센터/광고 API 집계
같은 날 전체 주문 1,378건과 광고 전환 주문은 별도 지표

02

공공 복지 AI 챗봇 고도화 · RAG

대표 프로젝트 · 개요와 구조

산학 프로젝트 · 3인 팀 팀장

2025.09.22 ~ 2025.12.05

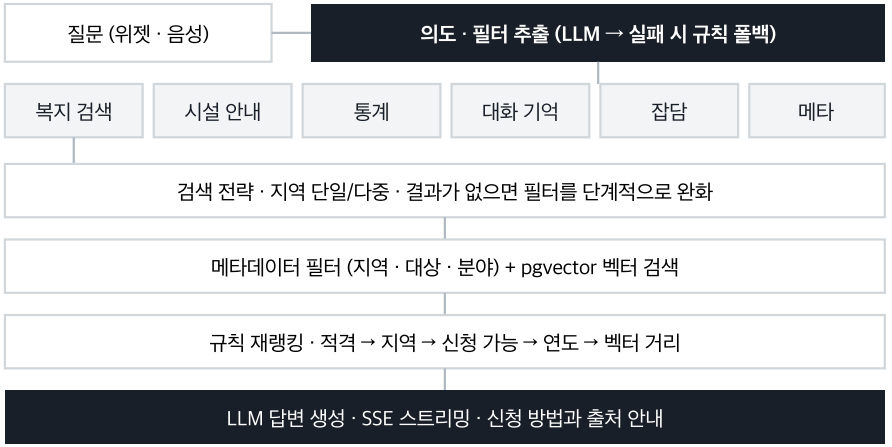
표현이 조금만 달라도 답을 못 찾던 기관 챗봇을 의도 판단과 RAG 검색으로 다시 만들었습니다. 팀장으로 기관과 요구사항을 맞추고 개발을 맡았습니다.

협력	전북특별자치도사회서비스원
Backend	FastAPI 0.115 · SQLAlchemy 2 · APScheduler · SSE · Docker Compose · Nginx
AI · 검색	LangChain 0.3 · bge-m3 임베딩 · PostgreSQL + pgvector · LLM 클라우드/로컬 선택
Frontend	관리자 React 19 · Vite · Tailwind · Recharts / 위젯 Vanilla JS Custom Element
역할	**내 담당** 요구사항 협의 · 설계 · 백엔드 · 검색 · 관리자 · 위젯 · 발표 **팀원** 자료 조사 · 발표 자료 · 문서

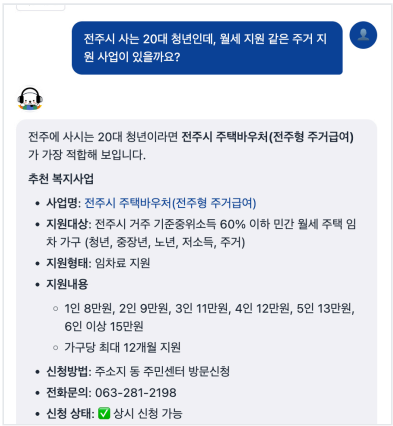
1,809건
복지사업 · 매일 03시 갱신

18,318곳
복지시설 위치 데이터

우수상
캡스톤 우수상 · 기관 표창



질문 하나가 답이 되기까지의 흐름





조건을 말하면 받을 수 있는 사업과 신청 방법 안내 / 증상을 말하면 가까운 의료기관을 거리순 안내

새 사업을 넣으려면 재학습이 필요했던 파인튜닝 챗봇을 RAG로 전환

문제 해결 3



기존 방식과 새 방식 비교

문제

기관의 기존 챗봇은 파인튜닝 기반이라 학습된 Q&A 범위 안에서만 답했습니다. 새 복지사업을 반영하려면 다시 학습해야 해서 신청 기간이 지난 사업을 안내하기도 했습니다. 결국 포털 안에 숨겨진 채 운영되고 있었습니다.

해결

재학습 주기화 학습 비용이 들고 그 사이 정보는 계속 낡음

채택 **모델은 그대로 두고 매일 갱신되는 데이터를 찾아 답하는 RAG**

게시판에서 바뀐 사업만 매일 수집하고 500자 이하 조각으로 나눠 bge-m3로 임베딩했습니다. 벡터 DB를 따로 두지 않고 PostgreSQL에 pgvector를 붙여 조건 필터와 의미 검색을 한 쿼리에서 처리했습니다.

전용 벡터 DB의 확장성보다 기관이 관리할 시스템을 하나로 줄이는 쪽을 골랐습니다. 사업과 시설 2만 건 규모와 기관의 운영 인력을 기준으로 한 판단입니다.

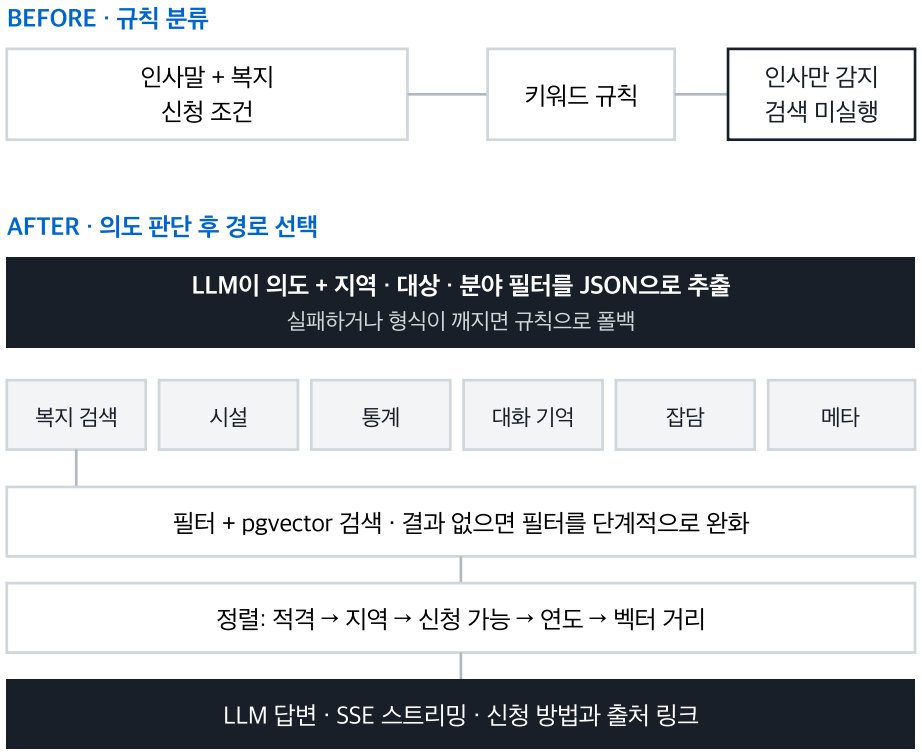
결과

재학습 없이 복지사업 1,809건이 매일 새벽 자동으로 반영됩니다. 찾은 원문만 근거로 답하고 공식 게시글 링크를 함께 보여 줍니다. 단순 키워드 검색으로는 0건이던 "치매" 관련 질문에서도 관련 사업을 찾아 안내했습니다. 시군 질문에 도·중앙부처 사업도 함께 찾도록 지역 필터를 고쳐 "남원시 청년 일자리" 검색은 0건에서 13건이 됐습니다.

조건 · 사업 1,809건 · 시설 18,318곳 · 치매 비교는 결과보고서 · 남원 비교는 당시 커밋 기록 기준

인사가 섞인 복지 질문도 검색하도록 의도 분류를 개선했습니다

문제 해결 4



질문 처리 흐름 개선 전후

문제

인사말과 복지 신청 조건이 함께 들어간 질문이 단순 인사로 분류되어 검색을 건너뛰는 사례를 발견했습니다. 표현별 예외 규칙을 늘리는 방식만으로는 확장성이 부족했습니다.

설계 판단

규칙만 추가	표현이 달라질 때마다 예외 유지보수 필요
LLM 단독	응답 실패·형식 오류에 대한 별도 처리 필요
선택한 방식	**LLM으로 의도 추출 + 검증 후 규칙 폴백**

의도·지역·대상·분야를 JSON으로 추출해 경로를 선택하고, 오류 발생 시 규칙 기반 분류로 이어지도록 했습니다. 사업 검색은 적격 여부와 신청 가능성을 우선해 정렬했습니다.

답변은 **SSE 스트리밍**으로 전송해 생성 과정을 단계적으로 보여줬습니다.

결과

질문을 **6개 의도 경로**로 분기해 복지 검색으로 이어지게 했습니다. LLM 오류가 발생해도 규칙 기반 분류로 처리하는 대체 경로를 확보했습니다.

구현 · FastAPI · LangChain · JSON 구조화 · 규칙 폴백 · SSE

OpenContext · 사내 문서를 밖으로 보내지 않는 AI 검색

개요와 구조

3인 팀 · 오픈소스 개발자대회

2025.08 ~ 2025.12 · github.com/OpenContextAI/open-context

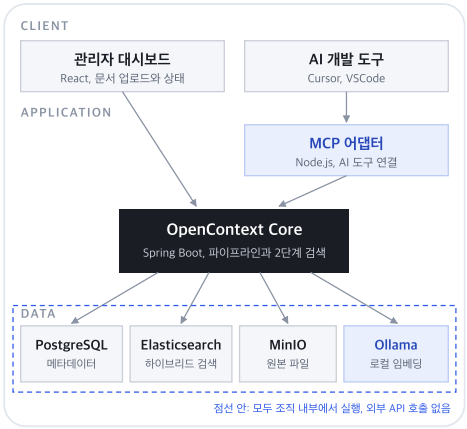
현장실습 때 외부 MCP 문서 서비스를 써 보니 효과가 컸지만 통제권이 밖에 있었습니다. 그래서 회사가 직접 통제하는 사내 버전을 제안해 팀원들과 만들었습니다.

Backend	Java 21 · Spring Boot 3.3 · QueryDSL · Flyway · LangChain4j
검색	Elasticsearch 8.11 · Ollama bge-m3 로컬 임베딩 · MinIO · PostgreSQL 16
연동	MCP (TypeScript SDK) · Cursor · VS Code
Admin	React 19 · Vite · TanStack Query · Zustand · Tailwind
역할	**내 담당** 제안 · 하이브리드 검색 엔진 · 2단계 검색 API · DB 모델링과 테스트 · 관리자 화면 · Docker 배포 · 보안 패치 **팀원** 문서 수집 파이프라인 · MCP 어댑터 · 공통 에러 처리

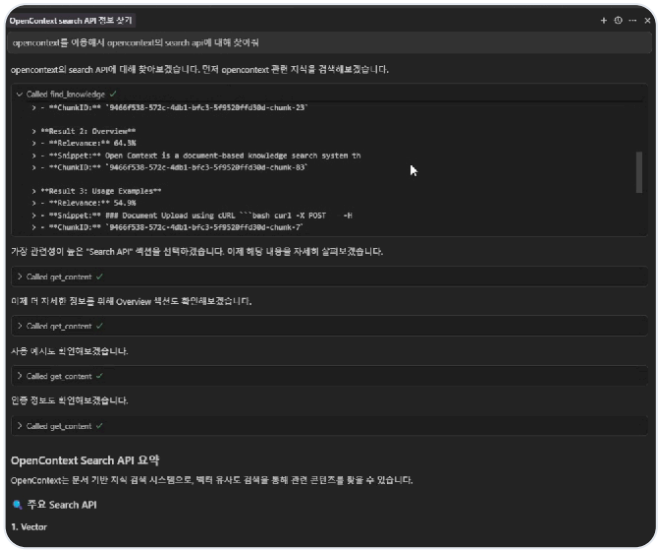
은상
학부 작품경진대회

로컬 처리
문서 처리 · 임베딩을 조직 안에서

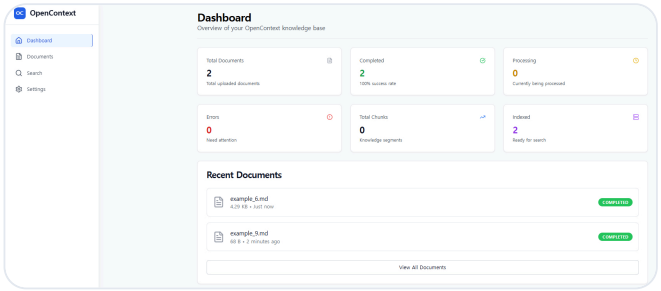
2단계
고르고 원문만 가져오기



전체 구조. 점선 안은 모두 조직 내부에서 실행



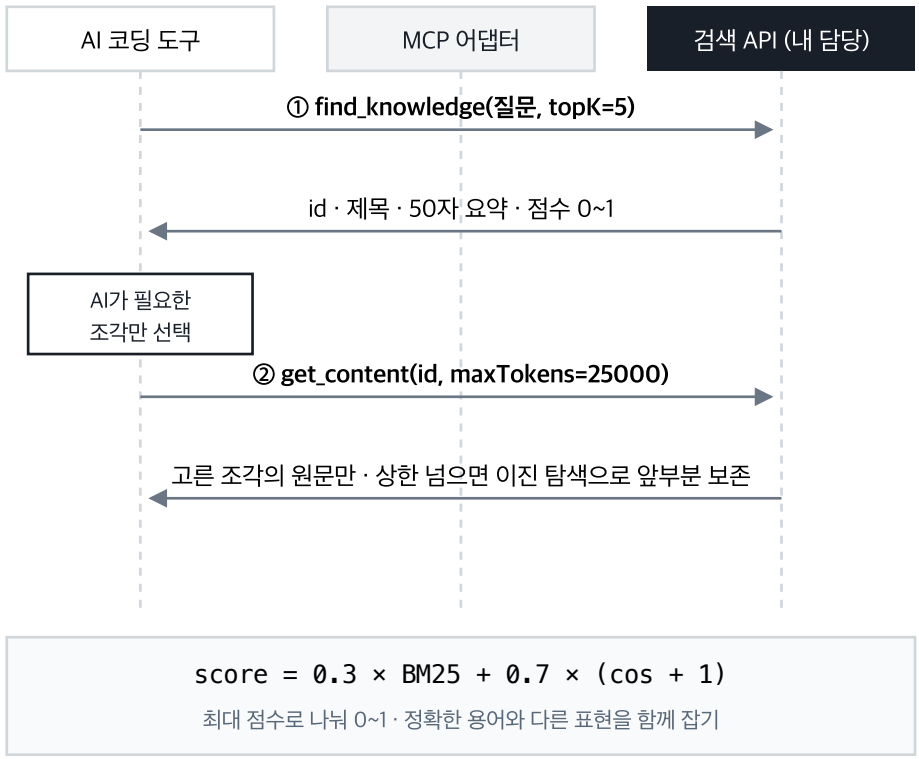
AI 도구가 find_knowledge로 고르고 get_content로 원문만 가져옴



제가 만든 관리자 화면 · 문서 처리 상태 확인

AI에 필요한 문서만 전달하는 2단계 검색 API를 설계했습니다

문제 해결 5



2단계 검색 호출 순서와 점수 계산

문제

사내 문서를 검색한 뒤 모든 원문을 한 번에 전달하면 불필요한 컨텍스트가 커지고, AI가 어떤 자료를 사용할지 선택하기 어려웠습니다.

해결

find_knowledge에서 제목·요약·관련도 목록을 먼저 반환하고, AI가 선택한 문서만 **get_content**로 원문을 가져가도록 나눴습니다.

검색 API에는 BM25 키워드 점수와 벡터 유사도를 결합했습니다. 결과를 단순 순위만으로 보여주지 않고 관련도 점수도 함께 반환하도록 설계했습니다.

> **핵심 선택** · 검색 결과 탐색과 원문 읽기를 분리해 필요한 컨텍스트만 사용

결과

AI 도구가 **관련 자료를 먼저 고르고 원문을 선택적으로 읽는 검색 경로**를 구현했습니다. 문서 처리와 임베딩은 조직 내부 환경에서 동작하도록 구성했고, 팀은 학부 작품경진대회 은상을 받았습니다.

본인 담당 · 하이브리드 검색 API · 데이터 모델링 · 2단계 조회 설계

04

JH Personal OS · AI가 참조하는 정보와 변경 권한을 설계했습니다

개요와 구조

개인 · 매일 사용 중 · 커밋 155개

2026.09 ~ 현재

과거 대화 요약을 AI가 최신 사실처럼 쓰거나, 여러 작업의 수정 내용이 섞이는 문제가 반복됐습니다. 그래서 필요한 원본만 읽고 검증된 변경만 기록하는 로컬 커널을 만들었습니다.

Stack	Python · Git · Markdown과 JSON · launchd · MCP
요청 분류	공부·취업·프로젝트별 manifest로 작업 영역 선택. 애매하면 임의 결정 대신 후보 제공
정보 조회	필요한 도메인의 원본·현재 상태·근거만 묶어 AI에 전달하는 context compiler
변경 통제	읽기→변경 계획→승인된 액션 실행을 분리. revision 검사·중복 요청 차단
정확성	AI 추론은 사용자 사실로 승격 금지. 출처·현재성·계약을 검증 규칙 31개로 검사
복구	병렬 작업 전에 Git 스냅샷 생성. 수정 충돌을 방지하고 작업 이력 보존

AI 도구가 바뀌어도 동일한 원본을 조회하고 변경 이력을 확인할 수 있도록 설계했습니다.



요청이 들어와서 기록되기까지의 흐름

운영하며 실제로 고친 문제 · 커밋 기록 기준

작업 소실	병렬 세션이 남긴 미커밋 파일 105개(3.5MB)가 정리 명령 한 번에 사라질 수 있었다 → 커밋과 분리된 스냅샷으로 먼저 보존
조용한 오답	한글 파일명 2,382개 중 518개가 다른 유니코드 형식이라 경로 비교가 조용히 틀렸다 → 정규화와 상시 검사 추가
응답 속도	메뉴를 만들 때 쓰지 않을 일정 조회까지 실행해 11.1초 걸렸다 → 필요할 때만 실행해 0.3초
백업	원격 저장소 없이 디스크 하나에 이력이 모여 있었다 → Git 번들 백업과 실제 복구 검증

05

기존 사내 출입 데이터를 새 관리 서비스에 연결했습니다

백엔드 현장실습 · 데이터 연동과 협업

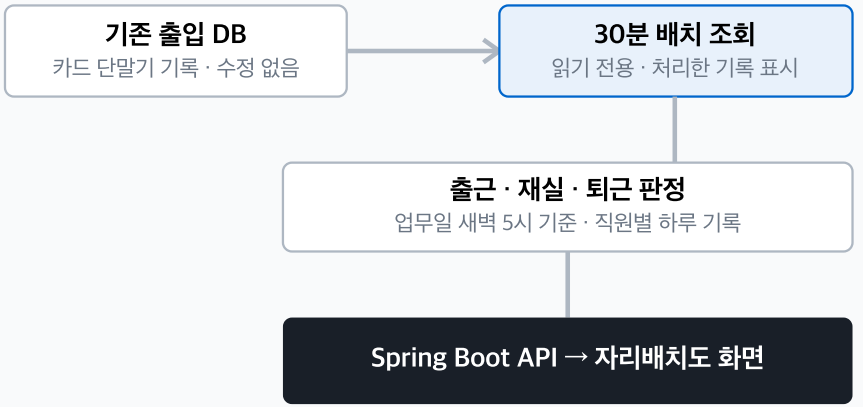
(주)텔로스 · 백엔드 담당

2025.07 ~ 2025.08

직원·좌석·출퇴근 데이터를 다루는 사내 시스템의 백엔드를 맡아 2주 반 동안 89개 커밋으로 API 설계부터 회사 서버 배포까지 했습니다.

기술	Java 17 · Spring Boot 3.5 · JPA · SQL Server · Spring Security · JWT · Docker Compose · Nginx
협업	Swagger로 API를 먼저 정리해 프론트엔드 선임과 공유, 백엔드 전담
연동	기존 출입 DB 기록을 30분마다 읽어 직원별 출근·재실·퇴근 판정
응답	공통 실패 형식과 상황별 오류 코드 26개, 성공·실패 응답 구조 통일
인증	Access 30분 · Refresh 90일(HttpOnly 쿠키) 이중 토큰

기존 시스템과의 연결 구조



기존 출입 시스템을 변경하지 않는 데이터 연동

데이터 연동

원본은 읽기만, 처리한 기록은 표시

운영 중인 출입 시스템은 수정하지 않고 읽기만 했습니다. 처리한 기록에는 표시를 남겨 다음 주기에 다시 읽지 않게 했고, 업무일은 새벽 5시 기준으로 나눠 판정했습니다.

배포 후 버그

서버 시간대로 어긋난 출근 현황

회사 서버에 올리자 결석 기록과 출근 현황이 어긋났습니다. 원인은 컨테이너 시간대였고, 컨테이너·스케줄러·코드의 현재 시각을 한국 시간으로 맞춰 9개 파일을 고쳤습니다.

협업 방식

API 문서를 먼저, 응답 모양은 쓰는 쪽에 맞췄습니다

Swagger 문서로 API를 먼저 정리해 프론트엔드 선임께 드렸습니다. 선임께서 응답 인터페이스를 정한 다음 날 성공·실패 응답을 같은 구조로 통일했고, 실패도 200으로 내려가던 상태 코드를 400·401·403·404·409·500으로 표준화했습니다.

다른 개발 경험

JeonjuEats · Spring Boot/Vue 기반 주문 서비스, AWS 배포
Calelog · 4인 팀 일정 기반 활동일지 자동화, 백엔드·CI/CD

기존 시스템을 깨지 않고 붙이는 법과, 응답 약속은 쓰는 쪽 기준으로 정한다는 것을 배웠습니다.

배지혁 github.com/qowlgur121