여정훈.

AI 엔지니어 · 백엔드

LLM 추출·검증 파이프라인부터 클라우드 운영까지 — 끝까지 직접 만드는 AI 엔지니어

통신·지자체 과제에서 데이터 파이프라인과 예측 AI를 개발했고, 지금은 LLM 데이터 추출·검증 파이프라인과 백엔드, React 대시보드를 설계·개발·운영하는 AI 엔지니어입니다. 프롬프트 세대 관리·모델 실측 평가·환각 방어·비용 최적화 등 프로덕션 LLMOps를 실무로 다루며, 법인을 세워 실서비스를 직접 배포·운영합니다.

데이터 파이프라인

이전 재직 · 다임즈 — 통신 · 지자체 과제의 네트워크 장비 자동화, 트래픽 예측 AI, 신재생에너지 실시간 수집 · ETL 개발

LLM 플랫폼

현재 재직 · 비드코칭연구소 — 입찰공고 문서를 추출 · 검증하는 AI 플랫폼 설계 · 운영 (검증 건당 비용 −82.9%), 사내 MCP 서버 · AI 교육 3부작

앱 2개

법인 자체 서비스 — 말로 기록하는 AI 피트니스 앱 · AI와 MCP로 공지하는 모임 앱, App Store · Google Play 출시 · 운영

AI 스터디

개인 · 신규 모델 · 도구를 직접 돌려 보고 벤치마크 결과를 공개

PART 1 재직 경력 — 2020.11 — 현재
회사에 재직하며 수행한 일입니다. 보안 SW, 데이터 파이프라인 · AI, 그리고 지금의 LLM 플랫폼까지 세 회사.

PART 2 법인 운영 — 주식회사 이엘디 — 2024.05 — 현재
직접 세운 법인에서 대표 · 풀스택으로 수행한 일입니다. 스토어에 낸 자체 서비스 2종, Conoti AI 데스크톱 앱, 자체 앱 2개 (CopyZoom · 실시간 점수 집계 앱), 외주 개발 1건.

PART 3 개인 활동 — 2026.08 — 현재
회사 · 법인 업무 밖에서 공개한 기술 리서치입니다.

메일 **aofhd0003@gmail.com** 웹 **portfolio.eldsoft.com** GitHub **github.com/YeoJeongHun1**

요약 경력 요약 — 지금 하는 일부터 2020.11 — 현재

비드코칭연구소 AI 엔지니어 (재직 중) 2025.04 – 현재

공공조달 입찰공고 문서(HTML · HWP · PDF · 이미지)에서 핵심 항목을 뽑아내고 AI로 검증하는 사내 플랫폼 — 수집 · 변환 · 추출 · 검증 · 적재와 개발 체계까지 **프로젝트 12건**을 설계 · 구축 · 운영했고, **사내 AI 교육 3부작**도 기획 · 강의했습니다.

−82.9%	−33.5%	99.6%	0→80%
검증 건당 비용 (평균 응답 −80.2%)	검증 비용 (응답 −40.4%) — "AI는 추출만, 판정은 워커" 책임 분리	상위 모델을 부르지 않고 기본 모델만으로 끝난 비율 · 일반 트래픽 기준	공고문 자동검수 비율 — 시설 분야 · 사내 내부 지표

01 공고 수집 서비스 재구축 2025.04 — 2025.12

노후 수집 시스템을 API 서버 + 큐 워커 구조로 새로 구축하고, 신 · 구 결과를 공고 단위로 대조하는 전환 검증 도구로 교체를 검증

→ **구 시스템 대비 안정성 · 처리 속도 개선**

02 사내 MCP 서버 · 코딩 에이전트 개발 체계 2025.12 — 2026.09

로그 · DB · 큐 · 컨테이너를 다루는 사내 MCP 서버(조회 읽기 전용 · 실행 화이트리스트)와 에이전트 표준 · 워크트리 · 요청/배포 추적 체계를 구축

→ **팀원들이 실업무에서 쓰는 개발 환경**

03 하이브리드 데이터 추출 파이프라인 2026.01 — 2026.07

규칙 추출을 모듈화하고 텍스트 없는 PDF · 이미지 공고용 LLM 추출 경로, 2단 큐, 지연 시에만 보조 모델을 띄우는 race 폴백을 설계

→ **추출 대상 확장 · 복수공고 중복 AI 호출 3.2% 제거**

04 공고 점검 · 자동검수 시스템 2026.01 — 2026.04

사람이 한 건씩 보던 검수를 4단계 자동검수 엔진(조건은 DB 관리)으로 바꾸고, 권한 기반 점검 콘솔과 승인형 재수집 · 롤백을 구축

→ **시설 분야 자동검수 비율 0% → 80%(내부 지표)**

05 문서 변환 파이프라인 2026.03 — 2026.09

디지털 PDF는 오픈소스 파서로 직접 추출하고 스캔 문서만 OCR로 보내는 분기, AI 변환의 누락을 막는 결정적 품질 게이트 5종을 설계

→ **해당 경로 변환 API 비용 0 · 평균 1.24초/건**

06 AI 보완검증 시스템 2026.03 — 2026.09

규칙이 뽑은 값을 LLM이 다시 확인하는 워커를 설계 · 운영 — 값이 원문에 실제로 있는지 먼저 대조하고, 최상위 모델 대신 같은 표본 A/B로 고른 가벼운 모델로 운영

→ **검증 건당 비용 −82.9% · 평균 응답 −80.2%**

07 레거시 DB 무중단 이관 2026.04 — 2026.05

구버전 · EUC-KR 레거시 DB를 MySQL 8.4 · utf8mb4로 dual-write 병행 이관(Airflow 증분)하고, 운영을 건드리지 않는 DB 샌드박스 4계층을 구성

→ **레거시 정본 유지 · 무중단 병행 운영**

08 정제 규칙 자동학습 루프 2026.06 — 2026.07

원문과 사람이 정리한 결과의 차집합으로 정제 규칙을 제안하고, 영향도 · 과대 제거를 계측한 뒤 사람이 승인하는 학습 루프를 구현

→ **규칙 1,898개를 승인 루프로 전환 · 유사 패턴 67개 → 1개**

09 면허 판정 규칙 검증 플랫폼 2026.07 — 2026.10

운영 판정은 그대로 두고 섀도 트랙에서 새 면허 판정을 전 모수로 채점하는 검증 플랫폼, 정규식 병목은 상주 매칭 · n-gram 프리필터로 해소

→ **운영 매칭 5.7초 → 0.4~0.5초 · 판정 9.2배 단축**

10 워커 파이프라인 재설계 2026.08 — 2026.09

워커가 하나씩 잠기며 추출이 멈춘 장애를 로그로 재구성해 처리 상한 · 격리 · 알람으로 막고, 단계 9개가 같은 계약을 따르는 구조로 재설계

→ **재발 경로 차단 · 단계별 재처리 가능**

11 자동검수 데이터 기반 · 판정 시뮬레이터 2026.10 — 2026.10

자동검수 입력을 공고 단위로 모으는 저장소와 Airflow 적재 DAG(큐 소비자 이관 1호), 조건을 바꿨을 때의 결과를 미리 보는 판정 시뮬레이터를 구축

→ **조건 변경 효과를 운영 반영 전에 검증**

12 사내 AI 챗봇 2026.10 — 2026.10

질문을 분류해 고정 안내와 LLM 분석을 나누는 2단 챗봇 — 읽기 전용 조회 계정 · DB 재확인 인증 · 질문 한도 · 비용 화면 · 골든셋 회귀 시험

→ **사내 점검 웹 전 화면에 상시 위젯으로 배포**

프로젝트별 문제 → 해결 방식 → 결과는 6–11쪽 기술서에 있습니다.

주식회사 이엘디 대표 · 풀스택 — 기획 · 앱 · 서버 · AI · 인프라 · 출시 · 운영 2024.05 – 현재

직접 세운 법인에서 **자체 서비스 2종**(ELANFIT · Conoti AI)을 기획 · 개발부터 App Store · Google Play 출시 · 운영까지 맡고, 모바일 앱과 연동되는 데스크톱 앱과 자체 앱 2개(CopyZoom · 실시간 점수 집계)를 만들었습니다. 외주는 1건입니다.

ELANFIT AI 음성 코치 피트니스 앱 2026.02 출시

React Native 레거시를 Flutter + FastAPI로 재구축하고 GCP 인프라까지 직접 운영. 말로 기록하는 음성 코치 · 사진 식단 분석 · 체크인 맞춤 루틴 · 기억하는 AI 코치

→ **App Store · Google Play 운영 — App Store 4.0.3**

Conoti AI AI가 공지를 쓰고 보내는 모임 운영 앱 2026.06 출시

AI와 대화만으로 공지를 작성 · 예약 · 발송. MCP · OAuth 2.1 인가 서버를 직접 구현해 외부 AI · 에이전트도 공지 등록, 폰에서 PC 코딩 에이전트에 목소리로 지시

→ **App Store · Google Play 운영 — 현재 1.7.0+120 · 패치 1 · 15개 언어**

Conoti AI 데스크톱 앱 모바일 앱 연동 데스크톱 앱 2026.09

Claude Code 훅 · Codex 기록으로 에이전트 세션을 대화방처럼 모으는 Tauri 2 · Rust 앱. 종단간 암호화(Noise) 중계로 폰에서 이어 받고, 서버는 내용을 저장 안 함

→ **Claude Code · Codex 세션 — 0.14.0 공개 배포(macOS · Windows)**

CopyZoom 자체 앱 · 카메라 OCR 묶음 복사 2026.10 출시

카메라에 비친 글씨를 연락처 · 계좌 · 주소 · 금액 등 묶음으로 나눠 탭 한 번에 복사. 온디바이스 OCR · 순수 Dart 분류기 · 주민번호 가림

→ **App Store · Google Play 출시 — 대한민국**

실시간 점수 집계 앱 자체 앱 2024.12

Firebase Realtime Database로 점수를 실시간 동기화하는 Flutter 크로스플랫폼 앱 (Android · iOS)과 FastAPI 서버(AWS EC2)를 직접 설계 · 개발

→ **배포 후 고도화까지 운영**

유어라운드 교육 서비스 웹앱 외주 2024.07 — 10

요구사항명세서 · DB 설계(ERD) · 화면설계서까지 전체 설계하고 FastAPI 백엔드 · Vue 3 프론트엔드를 개발

→ **클라이언트 니즈 분석 · 커뮤니케이션 · 인력 관리까지 프로젝트 총괄**

다임즈 데이터 파이프라인 · AI 개발자 — 이전 재직 2021.09 – 2024.04

통신 · 지자체 과제에서 **네트워크 장비 자동화**, **트래픽 예측 AI**(LSTM · GRU · Prophet 비교로 운영 모델 선정), **신재생에너지 플랫폼의 실시간 수집 · ETL 파이프라인**(Airflow · Kafka · Redis · LoRa IoT)을 개발했습니다.

KT Dr.IP | **KT IPaSDN 장비 자동화** | **KT 트래픽 예측 AI** | **보령시 에너지 플랫폼** | **천안시 에너지 플랫폼** | **KTOA 해지회선 관리**

더안전한세상 2020.11 – 2021.05 보안 SW 개발 — 보안관제 시스템

01 커리어 타임라인 — 보안 SW에서 LLM 엔지니어링까지

2017 — 현재

2017 — 2019 대학에서 AI를 접해 관련 대회에 나가고 대학원 진학을 준비했지만, 원하던 학교에 가지 못하고 개발자의 길을 택했습니다.

2020.11 다만 방향까지 바꾸지는 않았습니다. 전자제어공학 전공을 바탕으로 백엔드 · 데이터 엔지니어로 실무를 시작했고, 맡은 업무 안에서 AI 과제를 찾아 맡았습니다.

2021 — 2024 네트워크 트래픽 예측 모델을 Airflow 파이프라인 위에서 개발 · 운영했고, GPT-3 공개 직후부터는 새 모델과 기법이 나올 때마다 제 업무에 적용할 수 있는지 직접 테스트했습니다.

그때 든 습관 **써 보는 데 그치지 않고 정확도 · 비용 · 지연을 정량 평가해 도입 여부를 결정하는 습관을 그때 들였습니다.**

그리고 지금 지금 다루는 LLM 파이프라인은 그렇게 쌓아 온 결과이지, 유행을 따라 옮겨온 자리가 아닙니다.

대회 · 수상 · 학력

2017 RC 페스티벌 은상
전국 · IT융합인력양성사업단

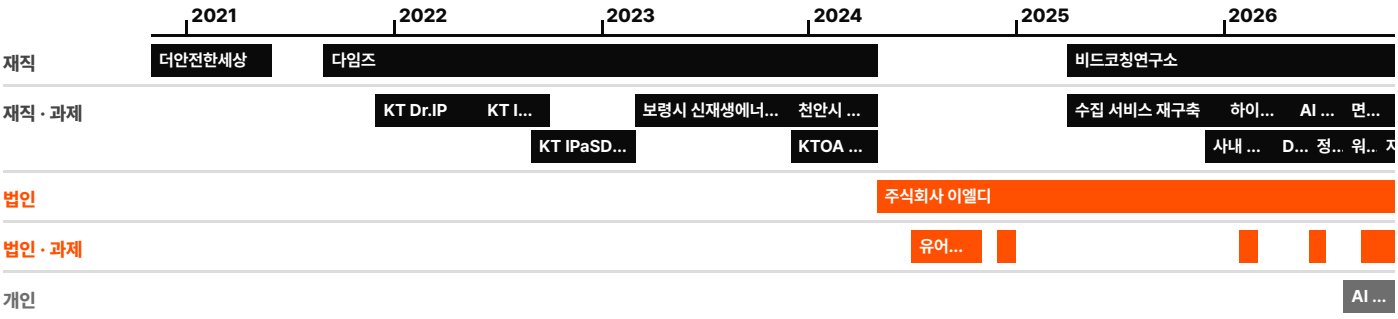
2018 RC 페스티벌 금상
전국 · IT융합인력양성사업단

2019 프로보노 공모전 은상
전국 · 정보통신기획평가원

2019 정보기술대학 작품전시회 동상
교내 · 국립한밭대학교

국립한밭대학교 전자제어공학과 졸업 (2020.02)
정보처리기사 · PC정비사 2급 · ITQ (한글 · 엑셀 · 파워포인트) · 워드프로세서 1급

재직 · 법인 · 개인을 다른 줄로 — 한 칸이 한 달 (2020.11 — 현재)



더안전한세상

2020.11 – 2021.05 · 보안 SW 개발

보안관제 시스템

02 다임즈 — 데이터 엔지니어링 · 시계열 예측 · IoT 파이프라인

2021.09 – 2024.04 · 데이터 파이프라인 · AI 개발자

01 KT · 2021.12 — 2022.05

Dr.IP — 업무 웹 서비스

프론트엔드 · 백엔드

- React 기반 프론트엔드와 Node.js 기반 백엔드 서버 개발
- 문서 다운로드 · 엑셀 자동 생성 기능 구현
- 고객정보 처리 자동화 기능 개발

React | Node.js

02 KT · 2022.06 — 2022.09

IPaSDN 1차 — 네트워크 장비 OS 업그레이드 자동화

백엔드

- Spring 기반 백엔드 개발
- OLT · L2 네트워크 장비의 OS 업그레이드 절차 자동화
- 수동 작업 대비 오류와 작업 시간을 줄이는 방향으로 절차 설계

Spring

03 KT · 2022.09 — 2023.02

IPaSDN 2차 — 트래픽 예측 AI 모델

AI 모델 · 데이터 파이프라인

- SFTP로 들어오는 RAW 트래픽 데이터 수집을 Airflow로 자동화
- 시계열 예측 모델 LSTM · GRU · Prophet을 개발하고 3종 비교 분석으로 운영 모델 선정
- KT대덕연구센터 연구원 대상 사내 AI · ML 세미나 진행

Airflow | TensorFlow | LSTM | GRU | Prophet

04 보령시 · 2023.03 — 2023.11

신재생에너지 플랫폼

백엔드 · 데이터 엔지니어

- FastAPI 기반 REST 수집 서버, 기상청 API 실시간 날씨 · IoT 센서 데이터 수집
- Airflow ETL/ELT 워크플로 설계, Kafka + Logstash 실시간 스트리밍, Redis 캐싱으로 조회 성능 개선
- 발전량 성능평가 알고리즘 개발, 발전량 예측 모델은 과업 외로 추가 개발
- 실시간 장애 감지와 다채널 알람(SMTP · FCM · SMS), Docker 기반 서비스 분리 · NGINX 리버스 프록시

FastAPI | Airflow | Kafka | Logstash | Redis | Docker | NGINX

05 천안시 · 2023.12 — 2024.04

신재생에너지 플랫폼

데이터 엔지니어

- LoRa 저전력 통신 프로토콜로 IoT 데이터 수집
- 공공데이터포털 API로 전력거래 · 날씨 데이터 수집
- Airflow 워크플로 자동화 — 보령시 플랫폼 구조를 재사용해 개발 기간 단축

LoRa | Airflow | FastAPI

06 KTOA · 2023.12 — 2024.04

공중케이블 해지회선 통합관리 시스템

설계 · 데이터 수집

- 시스템 아키텍처 설계 주도 — 프로그램 목록 · DB · ERD
- Crontab 스케줄링으로 OSS 데이터 수집 자동화
- Python 수집 스크립트 최적화

Python | cron | DB 설계

03 공공조달 입찰공고 데이터 자동 추출 · AI 검증 플랫폼

입찰공고 문서(HTML · HWP · PDF · 이미지)에서 자격조건 · 기초금액 같은 핵심 항목을 뽑아내고, 그 결과를 AI로 보완검증하는 7개 서비스짜리 데이터 플랫폼입니다. 프롬프트 세대 관리 · 모델 실측 평가 · 환각 방어 · 비용 최적화까지 프로덕션 LLMOps 전 주기를 설계부터 운영까지 맡고 있습니다.

문서 한 건이 지나가는 길

01	02	03	04	05	06	07
수집	**분배**	**변환**	**추출**	**검증**	**적재**	**활용**
크롤러 → API	RabbitMQ	포맷별 HTML	규칙 + LLM	환각 방어	dual-write	대시보드 · MCP
공고문 자동검수	중복 제거	변환 품질게이트	지연-트리거 병렬 race	검증 이슈코드 41종	dual-write	정제 규칙 자동학습 루프

실측으로 확인한 변화 — 산출 기준 함께

−82.9 %
검증 건당 비용 (평균 응답 −80.2%)
운영 모델을 가벼운 모델(3.1 Flash-Lite)로 바꾸기 전과 후 — 운영 로그 기준

−33.5 %
검증 비용 (응답 −40.4%) — "AI는 추출만, 판정은 워커" 책임 분리
재설계 전후, 컨텍스트 캐시를 적용한 상태를 기준선으로

−14.2 %
검증 비용 — 명시적 컨텍스트 캐시 도입
캐시 적용 전후 — 운영 로그 기준

99.6 %
상위 모델을 부르지 않고 기본 모델만으로 끝난 비율
일반 트래픽 기준

0→80 %
공고문 자동검수 비율 — 시설 분야 · 사내 내부 지표
사람이 직접 하던 검수를 조건 기반 자동 통과로 전환한 뒤 · 사내 내부 지표, 시설 분야 기준

최상위 모델과 가벼운 모델, 같은 문제로 비교했습니다



일치율 차이는 0.6%p, 비용은 8.7배 — 가벼운 Flash-Lite로 운영 모델을 정했습니다.

반대로 벤치마크 점수가 더 높던 새 세대 모델은, 같은 입력에도 실행할 때마다 답이 달라져 생긴 오탐 5건을 모두 확인하고 도입하지 않았습니다.

당시 Gemini 최상위는 막 나온 3.5 Flash였습니다(3.5 Pro는 아직 없었습니다). 이 모델과 가벼운 3.1 Flash-Lite에 같은 표본 · 같은 프롬프트(해시로 같음을 확인)를 넣고, 상위 모델이 항목마다 만든 정답지와 얼마나 일치하는지 쟀습니다. 사람 검수 범위 · 표본 수는 공개 자료에 없어 절대 정확도로 읽지 않습니다. 비용은 상대값입니다.

사내 AI 교육 3부작 — 개발자와 비개발 직군이 함께

43장
1차 · AI 기초
퍼셉트론 → XOR → 다층 신경망 · 역전파. 청중이 브라우저에서 신경망을 직접 학습시키고 손글씨를 그려 인식시키는 MNIST 라이브 데모로 끝납니다.

51장
2차 · 산업 속의 AI
CNN · RNN · LLM · 생성 · 음성 · 코딩 AI. 웹캠 얼굴 탐지, LLM 추론 스트리밍, TTS, GAN · Diffusion을 강의 중에 라이브로 돌립니다.

24장
3차 · AI에게 일을 시키다
에이전트 · MCP · 스킬을 역사 서사로 풀고, 텔레그램 에이전트가 실서비스(Conoti AI)를 사람 승인 하에 제어하는 라이브 데모와 커서 MCP 시연으로 마무리합니다.

업무 자산이라 서버 · 접속정보 · 고객 데이터 · 물량 같은 내부 세부는 비공개입니다. 아키텍처 · 기법 · 역량 수준으로만 정리했습니다.

프로젝트마다 문제 → 해결 방식 → 결과 순서로 적었습니다. 사내 형상관리의 커밋 · 요청 · 배포 기록을 근거로 했고, 수치는 확정 실측값만 적었습니다.

01 공고 수집 서비스 재구축 — 신 · 구 전환 검증

2025.04 — 2025.12

수집 · 전환 · 설계 · 백엔드 · 전환 검증

문제

노후화된 기존 수집 시스템은 안정성과 처리 속도에 한계가 있었습니다. 새 시스템으로 바꾸는 동안에는 두 시스템의 결과가 같다는 것을 공고 단위로 보여 줄 근거가 필요했습니다.

해결 방식

- 수집 서비스를 새로 구축 — 수집 요청을 받는 API 서버와 다운로드 · 변환 · 추출을 나눠 맡는 큐 워커로 분리
- 신 · 구 시스템의 추출 결과를 공고 단위로 대조하는 전환 검증 도구 — 개발 · 운영 환경 데이터를 나란히 놓고 처리 상태 · 필드별 정확도로 비교
- 불일치를 유형별 오류 코드로 분류해 전환 판단 기준으로 삼고, 이후 명칭형 이슈코드로 바꿔 검증 단계 전체의 공통 분류 체계로 확장

결과

구 시스템 대비 안정성 · 처리 속도 개선

전환 검증 화면은 이후 통합 점검 콘솔의 바탕이 됨

Python | FastAPI | RabbitMQ | React | TypeScript | MySQL

02 사내 MCP 서버 · 코딩 에이전트 개발 체계

2025.12 — 2026.09

개발 체계 · 설계 · 구축 · 팀 보급

문제

코딩 에이전트가 실제 서버의 로그 · DB · 큐를 보지 못하면 한 번의 요청으로 끝낼 일이 사람의 복사 · 붙여넣기에 막히고, 권한을 그대로 주면 운영 시스템이 위험해집니다. 여러 세션이 동시에 작업하며 변경이 충돌하는 문제도 있었습니다.

해결 방식

- 에이전트가 로그 · DB · 큐 · 컨테이너를 다루는 사내 MCP 서버 — API 키 인증 · IP 허용 목록 · 접근 로그, 조회는 읽기 전용 강제, 배포 같은 실행은 화이트리스트로 제한
- 운영 DB 조회에는 무거운 JOIN · LIKE 차단과 실행 시간 상한, 도구 레지스트리를 재구성해 세션을 다시 열지 않고 도구 목록이 갱신되게 함
- 서비스 4개를 통합 저장소로 모으고 도구 중립 에이전트 표준(AGENTS.md) · 작업 단위 임시 워크트리 → PR → 병합 · 요청 / 배포 식별 코드로 변경 이력을 추적
- 요청 · 종료 훅이 작업 이력을 DB에 남겨 보고서를 자동 작성, 개발 · 운영 DB 쓰기는 사람이 실행하는 SQL로만 허용
- 감시 대상 코드를 건드리지 않고 외부 호출만으로 장애를 판정하는 헬스체크 서비스
- 환경별 DB 4곳의 스키마 차이를 매일 대조해 알리는 드리프트 감시기 · 화면, 코딩 에이전트 토큰 사용량을 계정 · PC · 세션별로 모으는 수집 훅과 대시보드

결과

팀원들이 실업무에서 쓰는 개발 환경 — 에이전트 작업과 배포가 요청 단위로 추적된다

MCP | Python | FastAPI | Claude Code 훅 · 스킬 | git worktree | Docker

03 하이브리드 데이터 추출 파이프라인 — 규칙 + LLM

2026.01 — 2026.07

데이터 추출 · AI 파이프라인 설계 · 구현

문제

입찰공고는 금액 · 자격조건 표기가 서술형 · 표 · 혼합형으로 제각각이고 합본 공고도 많아 정규식 추출이 자주 실패했습니다. 텍스트가 없는 PDF · 이미지 공고는 규칙으로 읽을 방법이 없었고, AI를 붙이자 느린 건이 큐 전체를 막았습니다.

해결 방식

- 규칙 추출을 모듈로 분리하고 표 · 인라인 패턴(병합 셀 · 합계 행 · 자간 공백)을 확장, 실패와 불일치를 구분하는 이슈코드 · 등급과 공고 유형별 패턴으로 재구성
- PDF · 이미지 공고용 LLM 추출 경로 — 프롬프트를 DB에서 버전으로 관리하고 호출마다 토큰 · 비용 · 응답 시간을 기록, 필드별 AI 적용을 설정으로 제어하며 기존 값은 덮어쓰지 않게 보호
- 느린 AI 작업을 2단 큐(접수 · 변환 / AI 추출 · 저장)로 분리해 빠른 건이 기다리지 않게 함 — 첫 운영 반영에서 분기 결함을 찾아 롤백 후 수정 재배포
- 장애 로그를 실측해 실패 대부분이 타임아웃 · 쿼터 초과임을 확인하고, 기본 모델이 늦을 때만 이종 벤더 보조 모델을 병렬로 띄우는 지연-트리거 race를 설계해 운영 중
- 같은 공고문을 공유하는 복수공고는 DB 기본키 충돌의 원자성으로 대표 1건을 정해 결과를 재사용 — 타이밍에 기대던 첫 설계를 폐기하고 동시성 결함(스냅샷 고정 · 같은 밀리초 중복 삽입)까지 수정

결과

텍스트 없는 PDF · 이미지 공고까지 추출 대상으로 확장

복수공고 중복 AI 호출 3.2% 제거

Python | 정규식 | Gemini | RabbitMQ | MySQL | Docker Compose

04 공고 점검 · 자동검수 시스템

2026.01 — 2026.04

점검 · 자동검수 · 검수 엔진 · 웹 콘솔

문제

추출된 공고는 사람이 한 건씩 눈으로 검수했고, 검수 조건은 코드에 묻혀 비개발 직군이 볼 수 없었습니다. 잘못 수집된 공고를 되돌려 다시 처리하는 공식 절차도 없었습니다.

해결 방식

- 자동검수 엔진을 대상 → 필수 → 제외 → 면허 4단계로 구조화 — 조건은 DB로 관리해 화면에서 조회, 대표 사유는 우선순위가 가장 높은 단계로 결정, 통과 · 대상외 사유 코드를 필드별로 세분화
- 새 분야로 넓힐 때는 운영에서 상태 변경을 건너뛰는 관찰 모드로 먼저 검증하고, 통계 API · 차트로 성공 · 실패 사유를 추적
- 점검 콘솔 — JWT 슬라이딩 세션 · 레벨 기반 RBAC(권한은 DB에서 동적 관리), 환경별 AI 호출 현황(토큰 · 비용 · 응답 시간), SVG로 그린 파이프라인 시각화
- 승인형 재수집 · 롤백 — 외부 점검 시스템이 요청하고 책임자가 승인하면 실행, 테이블별 롤백 동작은 설정 테이블로 관리, 동시 실행 방지 · 데드락 재시도 · 트랜잭션 원자성
- 화면 설계를 웹 페이지로 먼저 만들어 비개발 직군과 합의한 뒤 DB → API → 웹 → 워커 → 통합 순으로 구현

결과

사람이 하던 공고문 검수를 조건 기반 자동 통과로 전환 — 시설 분야 자동검수 비율 0% → 80%(사내 내부 지표)

Python | FastAPI | React | TypeScript | JWT | MySQL

05 문서 변환 파이프라인 — 무과금 추출 · AI 변환 품질 게이트

2026.03 — 2026.09

문서 변환 · 변환 설계 · 장애 분석

문제

HWP · PDF · 이미지로 오는 공고문을 모두 HTML로 정규화해야 했는데 변환 API 비용이 문서마다 들었습니다. AI로 HTML을 만들게 하자 모델이 본문을 요약하거나 붙임을 빠뜨리는 일이 확률적으로 생겼고, 에러 없이 글자만 깨지는 PDF도 걸러지지 않았습니다.

해결 방식

- 텍스트 레이어가 있는 디지털 PDF는 오픈소스 파서(Open Data Loader)로 직접 추출하고 스캔 문서에만 OCR을 남기는 하이브리드 분기, 파일을 한 번만 읽어 HTML 변환과 데이터 추출을 병렬 호출
- AI HTML 변환을 운영에 켠 당일 누락을 확인해 즉시 되돌리고, 원인이 입력이 아니라 모델의 확률적 조기 종료임을 분석
- 파서가 뽑은 원문 텍스트를 정답지로 삼는 결정적 게이트 5종(종료 사유 · 축약 문구 · 분량 비율 · 붙임 전수 · 꼬리 누락) — 탈락하면 이종 벤더 2차 모델, 초대형 문서는 페이지 조각별 변환 후 이어붙임
- 모든 경로가 실패하면 기존 이미지 HTML로 돌아가 불완전한 본문은 저장될 수 없게 설계, 2차 모델 · 프롬프트는 DB 행으로 관리해 코드 배포 없이 교체
- 글자만 깨지는 PDF는 폰트 경고 신호로 탐지해 바이트 단계에서 교정 · 재렌더(실패하면 원래 결과 유지), 화면이 겹치던 일부 발주처 엑셀 공고는 AI 없이 표를 보존해 직접 변환, 한글 문서 변환 엔진 4종 비교 도구 구축

결과

디지털 PDF 경로의 변환 API 비용 0 · 평균 1.24초/건

AI 변환 품질 게이트는 테스트 환경에서 라이브 검증 · PDF 폰트 교정과 엑셀 직접 변환은 운영 반영

Open Data Loader | Cloud Vision OCR | Gemini | 이종 벤더 LLM | PyMuPDF | Docker

06 AI 보완검증 시스템 — 설계부터 비용 · 정확도 최적화까지

2026.03 — 2026.09

AI 보완검증 · 설계 → 비용 분석 → 구현 → 운영 → 고도화

문제

규칙으로 뽑은 값이 틀려도 알아챌 방법이 없었습니다. LLM으로 다시 확인하려면 LLM이 원문에 없는 값을 지어내는 문제(환각)부터 막아야 했고, 검증 호출이 AI 비용의 대부분을 차지하게 되자 정확도를 지키면서 비용을 줄일 근거가 필요했습니다. AI API에 장애가 나도 검증이 멈추면 안 됐습니다.

해결 방식

- 규칙이 뽑은 값을 별도 LLM이 한 번 더 확인하는 검증 워커 — 대안 9가지를 비교해 큐에서 한 건씩 꺼내 처리하고 실패하면 다시 시도하는 구조로 정하고, 여러 모델을 같은 정답지로 비교(정확도 · 환각 · 틀린 값끼리의 일치)해 운영 모델 선정
- 값이 원문에 진짜 있는지부터 확인(원문 근거 게이트) — 없으면 지어낸 값으로 보고 한 번 다시 추출하고, 근거로 든 문구가 원문과 어긋난 항목만 다른 회사의 상위 모델에 다시 맡기는 2단 구성
- 역할 나누기 — LLM이 값을 바로 고치던 구조를 "LLM은 값만 뽑고, 원문 대조 · 판정은 워커 프로그램이"로 바꾸고, 불일치는 검증 이슈코드 41종으로 등급 관리 · 결과 화면과 환경별 켜기/끄기(웹 토글 + 워커 이중 가드) 제공
- 최상위 모델을 당연하게 쓰지 않음 — 당시 Gemini 최상위였던 3.5 Flash와 가벼운 3.1 Flash-Lite에 같은 표본 · 같은 프롬프트를 넣어 비교(A/B)하고, 정확도가 거의 같아 Flash-Lite로 운영 모델을 정함
- 벤치마크 점수가 더 높은 새 세대 모델은 같은 입력을 세 번씩 다시 돌려 모델 문제인지 우연한 흔들림인지 가렸고, 실행마다 답이 달라지는 문제를 확인해 도입하지 않음 · 프롬프트 구조(한 번에 묻기 대 나눠 묻기)도 입력을 고정한 실험 전용 서비스에서 같은 채점기로 비교하고, 모델 · 프롬프트 변경 회귀를 막는 골든셋 게이트까지 구축해 운영
- 반복되는 지시문을 모델 쪽에 저장해 다시 쓰는 컨텍스트 캐시와, 모델에 넣기 전에 HTML을 걷어 내는 전처리로 호출 비용 · 입력량 절감
- AI API 장애에 대비한 레이싱 체계 도입 — API에 장애가 나도 검증이 멈추지 않고 처리되도록 설계

결과

컨텍스트 캐시로 검증 비용 −14.2% · HTML 전처리로 검증 입력 −92.2%(문자 수 기준)

역할 나누기로 검증 비용 −33.5% · 응답 −40.4%(캐시를 켠 상태와 비교)

A/B(상위 모델이 만든 정답지와 항목별 대조) 일치율 3.5 Flash 91.4% vs 3.1 Flash-Lite 90.8% — 차이는 0.6%p, 비용은 8.7배라 가벼운 모델을 택함 · 운영 모델을 바꾼 전후 운영 로그 기준 검증 건당 비용 −82.9% · 평균 응답 −80.2%

일반 트래픽의 99.6%는 상위 모델을 부르지 않고 기본 모델만으로 끝남 · 새 세대 모델은 오탐 5건을 확인하고 도입하지 않음(판정 일치율 88.0%)

Gemini 3.1 Flash-Lite | 다른 회사 상위 모델 | RabbitMQ | 컨텍스트 캐시 | FastAPI | React

07 레거시 DB 무중단 이관 — dual-write · 4계층 샌드박스

2026.04 — 2026.05

DB 이관 · 단독 설계 · 구축

문제

구버전 · EUC-KR 레거시 운영 DB를 멈추지 않고 최신 버전 · utf8mb4로 옮겨야 했고, 코딩 에이전트가 운영 DB에 직접 닿지 않는 개발 · 검증 환경이 필요했습니다.

해결 방식

- 운영 미러와 개발 · 테스트 DB를 새 서버(MySQL 8.4)로 분리하고 역할별 계정을 나눔
- Python 인코딩 변환 이관 — 덤프 방식의 한글 깨짐을 피하고 실패 행을 격리, 공고 단위 멱등 적재
- Airflow로 일 단위 증분 이관(catchup · backfill)
- 운영 워커의 모든 SQL을 단일 진입점으로 모아 신규 DB에 이중 기록 — 미러 실패는 경고만 남기고 조회 시점에 다시 미러링
- DB 샌드박스 4계층 — 운영 조회 전용(JOIN · LIKE 제약) · 신규 운영(dual-write) · 개발(운영 데이터로 검증) · 테스트(자유)

결과

레거시를 정본으로 둔 dual-write 병행 운영으로 무중단 이관 진행 중(cutover 미실시)

MySQL 5.5 → 8.4 | Apache Airflow | Python | Docker Compose

08 정제 규칙 자동학습 루프 — 제안 → 영향도 → 사람 승인

2026.06 — 2026.07

규칙 자동학습 · 설계 · 구현

문제

자격조건 원문에서 불필요한 문구를 지우는 제거 규칙과 종목 판별 규칙을 사람이 하나씩 쌓아 관리했습니다. 규칙이 늘수록 무엇이 필요하고 무엇이 과하게 지우는지 판단할 근거가 없었습니다.

해결 방식

- AI 추출 원문과 사람이 정리한 결과의 차이가 곧 제거 대상이라는 점을 실데이터로 확인 — 줄 단위 차집합을 공고 빈도로 통계화해 새 규칙 제안
- 규칙마다 영향도와 과대 제거 여부를 계산해 운영 담당자에게 후보 제시
- 매시간 증분 학습 · 규칙 병합(대체 관계 기록), 자동 승인은 금지 — 커버리지 유지와 과대 제거 비증가를 통과할 때만 병합
- 엔진은 MCP 도구로 옮겨 에이전트가 다루게 하고, 쓰기는 테스트 DB에서만

결과

수동 관리하던 정제 · 판별 규칙 1,898개(종목 판별 1,166 + 본문 제거 732)를 제안 → 영향도 계측 → 사람 승인 루프로 전환

유사 패턴 67개 → 1개 병합 실증

Python | 정규식 | MCP 도구 | FastAPI | React

09 면허 판정 규칙 검증 플랫폼 — 채점 · 버전관리 · 판정 성능

2026.07 — 2026.10

면허 판정 · 설계 · 풀스택 · 검증 방법론

문제

업종 재정의로 면허를 다시 판정해야 했지만 규칙은 운영 표 하나뿐이라 고쳐도 효과를 잴 수 없었고, 운영 전환을 판단할 근거가 없었습니다. 규칙이 늘면서 공고마다 반복되는 정규식 컴파일이 처리 시간 대부분을 차지했습니다.

해결 방식

- 운영 판정은 그대로 두고 병행(섀도) 트랙으로 새 판정을 적재하는 독립 검증 플랫폼 — 전용 API · 웹 · MCP
- 스키마를 원천 · 사실 · 미러 · 규칙 · 파생 층으로 나누고 규칙은 버전으로 고정 — 화면 수치는 언제든 다시 만들 수 있는 파생 층에서만 읽고, 외부 기준 데이터와의 종목 매핑은 확실한 1:1만 쓰며 버전관리
- 코딩 에이전트가 자격조건 원문을 읽어 전수 판정하고 채점은 집합 비교로 결정적으로 — 비개발 직군용 사례 채점 화면, 반려 사유를 받아 다시 제안하는 루프
- 개선 라운드마다 도달 가능한 상한을 먼저 재고 전 모수로 채점 — 통합 업종명 하나에 계열 전체가 발화하던 오탐을 주력 분야 판별 · 앞뒤 조건 가드로 교정
- 정규식 컴파일 병목은 컴파일 결과를 들고 있는 상주 매칭 보관소(응답이 없으면 로컬 매칭으로 폴백)와, 구문 트리에서 필수 리터럴을 뽑아 n-gram 색인과 대조하는 프리필터로 해소 — 놓치는 발화가 없도록 한 방향으로만 틀리게 설계
- 운영자 코멘트 창구(대화 스레드 · 알림 · 진행 상태)로 받은 오탐 · 미탐을 새 초안 판에 모아 전 모수로 채점 · 재판정한 뒤 켜는 개선 루프 — 전용 콘솔은 사내 점검 웹으로 통합

결과

개선 라운드를 섀도 트랙에서 검증한 뒤 운영에 단계적으로 반영

매칭 보관소로 운영 공고당 면허 매칭 5.7초 → 0.4~0.5초 · 프리필터로 판정 시간 9.2배 단축(전 기간 재판정 놓침 0)

FastAPI | React | MySQL 8.4 | Airflow 3 | 정규식 AST · n-gram | Playwright

10 워커 파이프라인 재설계 — 장애 분석에서 구조 개편으로

2026.08 — 2026.09

파이프라인 · 원인 분석 · 재설계

문제

본문 전체가 인코딩된 이미지인 공고가 정제 정규식의 이차 시간 복잡도에 걸려, 워커가 메시지 재배달 주기마다 하나씩 잠기며 추출이 멈춘 장애가 있었습니다. 컨테이너는 실행 중이라 감시에 걸리지 않았고, 한 함수에 묶인 처리 흐름은 단계별 재처리가 불가능했습니다.

해결 방식

- 로그로 장애 타임라인을 재구성해 원인을 확정하고 처방 — 정제 앞 이미지 제거 가드, 자식 프로세스 처리 상한 + 강제 종료, 재시도 없이 확인 응답하는 격리 원장(재배달 확산 차단 · 사람 승인), 재시작 없이 바꾸는 알람 설정
- 테스트 환경에서 타임아웃 → 격리 → 알람 도착까지 전 구간을 훈련으로 확인한 뒤 운영 배포
- 드러난 구조 결함을 재설계로 연결 — 단계 9개가 같은 계약(run(ctx) → StageResult)을 따르고, 순서는 코드가 아니라 단계 마스터 표가 가지며 기동할 때 상호 검증
- 추출(raw)과 정제를 워커 경계로 분리하고 표본 비교 게이트로 회귀 없음을 확인, 오케스트레이터는 Hatchet PoC와 비교한 뒤 Airflow 3(공고 1건 = DagRun) 채택

결과

장애 재발 경로 차단(처리 상한 · 격리 · 알람)

단계별 재처리가 가능한 구조로 개편 게이트 통과

RabbitMQ | multiprocessing | Airflow 3 | Python | Docker Compose

11 자동검수 데이터 기반 · 판정 시뮬레이터 — Airflow 적재 DAG

2026.10 — 2026.10

자동검수 데이터 · 설계 · 구현 · 이관

문제

자동검수 조건을 바꾸면 결과가 어떻게 달라질지 미리 계산하려면, 판정에 쓰인 입력을 공고 단위로 모아 둔 저장소가 필요했습니다. 그 적재를 메시지 큐 소비자에서 오케스트레이터로 옮기면서 실패한 지점부터 다시 돌릴 수 있어야 했습니다.

해결 방식

- 자동검수 입력 전용 스키마 — 운영 원천 하나에서 읽어 전부 갱신(upsert)하고, 값이 바뀔 때마다 변경 이력을 남겨 처음 판정 때의 입력을 재현
- 메시지 큐 소비자를 Airflow DAG로 옮긴 첫 사례 — 공고 1건 = DagRun, 표마다 태스크 · 트랜잭션을 나눠 실패한 표부터 재시작, 같은 공고의 요청은 순서행 잠금으로 직렬화
- 어느 API 서버로 요청해도 같은 동작을 하는 공용 접수 창구, 새벽 점검 DAG가 누락 · 실패 공고를 다시 보냄 — Airflow가 응답하지 않을 때 대기 표에 쌓았다 재전송하는 경로는 테스트 환경까지
- 원천 DB 사이의 표현 차이(빈 값 · 구분자 · 날짜 오차)가 가짜 변경 이력으로 찍히지 않게 정규화 규칙을 두고, 신규 공고 동시 적재의 교착은 첫 버전에 잠금을 걸지 않는 방식으로 해소
- 과거 3개월 백필은 실시간 소비자 코드를 그대로 쓰는 배치로 — 일 · 항목 단위 관리표로 진행과 오류를 추적
- 판정 엔진을 브라우저로 옮긴 시뮬레이터 — 조건을 끼우고 빼며 단계별 유입 · 통과 · 제외를 퍼널로 보고, 시나리오는 서버에 저장해 팀이 함께 검토 · 대시보드에는 기존 산식과 객관 지표를 나란히

결과

시뮬레이터가 운영 판정을 99.96% 재현(배포 시점 대조) — 조건 변경 효과를 반영 전에 검증

적재 DAG는 테스트 → 개발 → 운영 순으로 반영

Airflow 3 | FastAPI | MySQL 8.4 | React | TypeScript | Playwright

12 사내 AI 챗봇 — 업무 화면 안내 · 공고 분석

2026.10 — 2026.10

사내 AI · 설계 · 구현 · 보안

문제

점검 웹의 화면과 기능이 계속 늘어, 사용법 · 화면 위치 · 공고 처리 결과를 바로 물어볼 창구가 요구됐습니다. 사내 데이터를 읽는 도구인 만큼 권한 · 비용 · 오남용 방어가 먼저 설계돼야 했습니다.

해결 방식

- 질문을 먼저 가볍게 분류해 바로가기 · 사용법은 고정 안내와 문서로 답하고, 공고 분석처럼 판단이 필요한 질문만 LLM에 넘기는 2단 구조 — 사용법 문서는 단계별 트리로 필요한 만큼만 펼쳐 입력 토큰을 줄임
- 조회 도구는 읽기 전용 권한만 가진 전용 계정으로 API 서버 안에 내장(외부 도구 서버 경유 폐기) — 공고 요약 · 처리 상태 · 규칙 판 · 정규식 진단을 정해진 양식으로만 조회
- 공용 인증이 토큰만 믿지 않고 DB의 현재 활성 · 등급을 다시 확인하도록 보강, 하루 질문 한도(파트장급 이상 면제)와 사용자별 일 · 주 · 월 실제 비용 화면
- 보고 있는 화면의 맥락을 함께 넘기고, 시스템 규칙을 고칠 때마다 고정 질문 묶음(골든셋)으로 회귀를 확인 — 분류기 시간 상한도 시간 초과 실측으로 조정
- 별도 서비스로 시작한 챗봇을 API 서버 안으로 통합해 중계 · 서명 단계와 컨테이너를 줄임

결과

사내 점검 웹 전 화면에 상시 위젯으로 배포

설계 → 보안 검토 → 단계별 개방 순서로 진행

FastAPI | React | TypeScript | LLM | MySQL | pytest

04 ELANFIT

App Store · Google Play · elanfit.app · 서비스 구조 도식

AI 음성 코치가 붙은 피트니스 앱

운동 · 식단 · 인바디 기록과 AI 코치, 커뮤니티를 한 앱에 담았습니다. React Native 레거시를 Flutter + FastAPI로 전면 재구축하고 GCP 인프라까지 직접 설계 · 배포 · 운영합니다. 4.0부터는 말로 기록하고 말로 코칭받는 앱이 됐습니다.

출시	2026.02 첫 출시 · App Store 4.0.3 · Google Play 운영
역할	기획 · 모바일 · 백엔드 · 어드민 · 인프라
스택	Flutter · Swift · FastAPI · PostgreSQL · React · GCP · Terraform
모델	gpt-live-1 · gemini-3.8-live (음성) · gpt-6-luna · gpt-6-sol

버전

3.1	2026.09 초 — 문의 · 푸시 · 체크인 · AI 코치 방어막
3.2.0	2026.09.12 — HealthKit · Health Connect, 오프라인 기록 큐, 계측
4.0.0	2026.09.23 — 음성 코치 · 에너지 과금 · SHOP 탭 · 호출어 · Live Activity
4.0.2	2026.09.28 — 식단 목표 계산 통합 · 코드 푸시(Shorebird) 도입
4.0.3	2026.10.01 — 호출어 응답 · 오탐 개선, 식단 한글 입력, 제미나이 엔진 음량 · 즉시 말하기

App Store · Google Play · elanfit.app · 서비스 구조 도식

실제 사용 화면



말로 기록 | 사진 한 장으로 식단 | 체크인 → 오늘의 루틴 → 운동 | 실시간 음성 코치 | 인바디 추세

ELANFIT 4.0 분해 — 음성 모델이 바뀌어도, 앱은 그대로. 엔진은 서버가 고르고, 앱은 그리기만 합니다.

01 · 화면 **Flutter 앱 + Swift 네이티브**	02 · 음성 엔진 **gpt-live-1 · gemini-3.8-live**	03 · 판단 서버 **FastAPI — 같은 음성 세션에 따로 붙는다**
04 · 모델 배분 **gpt-6-luna · gpt-6-sol**	05 · 과금 **에너지 — 재화 하나**	06 · 인프라 **GCP · Terraform · 롤링 배포**

사용자가 실제로 쓰는 기능을 쉬운 말로 적었습니다 — AI가 맡는 부분과 서버 · 규칙이 맡는 부분을 나눠서.

01

라이브 음성 코치

운동하면서 말로 기록하고 코칭받습니다. 음성 모델은 듣고 말하기만 하고, 기록 · 조회 · 사진 분석 같은 실제 일은 서버가 넘겨받아 처리한 뒤 결과를 읽어 줍니다. 음성 엔진은 gpt-live-1과 gemini-3.8-live 중에서 고르고, "엘란!" 하고 부르면 바로 음성 모드로 이어집니다.

02

사진 · 말 한마디로 식단 분석

음식 사진을 올리거나 먹은 것을 말하면 칼로리 · 영양소를 추정해 식단에 기록합니다. 사진은 비전 모델이 읽고, "괜찮음 · 주의 · 초과" 판정과 남은 칼로리는 모델이 아니라 서버가 하루 목표로 계산합니다. 목표는 신체 정보 · 활동량 · 그날 운동으로 정합니다.

03

인바디 결과지 판독

인바디 결과지 사진에서 체성분 숫자를 읽어 추세로 이어 붙입니다. 기록이 많으면 CSV로 한꺼번에 가져옵니다.

04

나에게 맞춘 운동 추천

오늘의 체크인(컨디션 · 시간 · 부위)을 받으면 규칙 엔진이 운동 목록에서 루틴을 짜고, 최근 완료율 · 수면 · 피로에 맞춰 세트와 강도를 조정합니다. 이 부분은 LLM이 아니라 규칙 엔진이라 같은 입력이면 같은 루틴이 나오고, AI 코치와 PT 사전 상담을 하면 코치가 제안한 계획을 사용자가 동의할 때 저장합니다.

05

기억하고 찾아보는 AI 코치

코치가 기록 · 조회 · 정정 · 되돌리기 도구를 직접 부르고(한 턴의 호출 수에 상한), 지난 상담을 요약한 장기 메모리와 운동 · 영양 지식 DB 검색으로 답을 보강합니다. 앱과 무관한 질문은 모델을 부르기 전에 규칙 분류기가 거릅니다.

06

모델을 효율적으로 고르는 구조

대화는 가벼운 gpt-6-luna, 사진 · 결과지 판독은 gpt-6-sol로 나눴습니다. 어려운 턴만 상위 모델로 올리는 승급은 만들었지만 실제 원가가 커서 꺼 두었고, 판단만 돌려주는 모델 제브(Jev)가 턴마다 가벼운 모델로 충분한지 고르는 선택기를 개발 버전에서 시험하고 있습니다(운영 미적용).

AI와 함께 개발한 방식

앱 · 서버 · 어드민 · 인프라를 코딩 에이전트와 함께 만듭니다. Claude Code · Codex · Gemini CLI · Cursor가 같은 작업 규칙 문서(AGENTS.md)를 읽고 일하고, 나눠 한 결과를 합쳐 배포 · 스토어 제출하는 일은 본인이 맡습니다.

- API 계약 파일을 먼저 고치고 앱과 서버가 그 계약을 따르게 해, 에이전트가 나눠 일해도 둘이 어긋나지 않게 합니다. 계약 검사 · 테스트 · 작업 기록을 맡는 에이전트를 따로 둡니다.
- 작업마다 검증 수단(테스트 · 시뮬레이터 캡처 · 실기기 · 운영 실측)을 미리 정하고, 증거가 기록에 없으면 완료로 보지 않습니다.
- 체크리스트를 따라 구현 → 검증 → 커밋을 반복하는 무인 실행 도구를 직접 만들어 씁니다.

실시간 음성 · gpt-live-1

말로 기록하는 음성 코치

문제 운동 중에는 화면을 보고 입력할 수 없다.

결정 음성 입출력은 모델이 맡고 판단은 우리 서버가 합니다. 서버가 같은 음성 세션에 따로 붙어 도구 호출을 가로채고, 권한 · 확인 · 상한을 전부 서버에 둡니다. 별도 화면 없이 채팅창 안에서 이어집니다.

엔진 교체 구조

음성 모델이 바뀌어도 앱은 그대로

문제 음성 모델은 몇 달마다 새로 나온다. 바꿀 때마다 앱을 고치면 심사가 병목이 된다.

결정 엔진 사이의 차이를 "전송 방식"과 "도구 경로" 두 축으로 보고 서버 한 곳에 모았습니다. 서버가 엔진 목록을 내려주고 앱은 그리기만 합니다. 앱이 모르는 전송 방식이 오면 조용히 넘어가지 않고 거절합니다. gpt-live-1과 gemini-3.8-live가 같은 구조로 돕니다.

모델 배분 · gpt-6

실측으로 찾은 운영 결함

문제 사진 분석 호출에 추론 설정이 빠져 기본 추론이 돌았고, 추론이 출력 상한을 다 써 버려 "분석 실패"가 나오고 있었다.

결정 모델 교체를 앞두고 세 모델 × 추론 설정을 운영 코드 경로로 실측하다가 이 결함을 찾았습니다. 추론 기본값을 none으로 바꾸고, 사진 식별이 크게 떨어지는 경량 모델은 사진에서 빼 대화는 gpt-6-luna, 사진 · 인바디 · 음성 기록 도구는 gpt-6-sol로 나눴습니다. 대화 승급 기능은 만들었지만 실제 턴의 원가가 벤치마크의 10배라 꺼 두었습니다.

사진 분석 지연 중앙값 7.7초 → 3.5초, 빈 결과 0 (같은 사진 세트, 적용 후 재측정)

프롬프트 캐시

매 턴 빗나가던 캐시를 되살림

문제 새 모델은 마지막 메시지 앞 전체를 한 단위로 캐시한다. 매 턴 바뀌는 맥락이 시스템 프롬프트 끝에 있어 매번 캐시가 빗나갔다.

결정 고정 지시문만 시스템 프롬프트에 두고, 턴마다 바뀌는 맥락은 마지막 사용자 메시지 바로 앞으로 옮겼습니다.

캐시 적중 93.8% (개발 서버 실측)

과금 설계

서로 다른 과금 단위를 재화 하나로

문제 분당 정액인 음성 엔진과 토큰 과금 엔진이 같은 1분을 다르게 센다.

결정 모든 모델에 공통인 "사용 토큰 → 모델 단가 → 원가" 로 모아 에너지라는 재화 하나로 차감합니다. 국가별 통화 분기를 없앴고, 가격표가 규칙과 어긋나면 서버가 시작 단계에서 멈추게 했습니다.

iOS 네이티브 · Swift

잠금화면에서도 "엘란!"

문제 백그라운드 · 잠금화면에서 호출어가 동작하지 않았다. 원인은 기존 플러그인이 매번 오디오 세션을 내리던 것.

결정 Swift로 호출어 엔진을 따로 만들어 오디오 세션은 유지하고 인식 요청만 교체합니다. 인식은 iOS 26 기기에서는 SpeechAnalyzer로 기기 안에서 합니다. 운동 중 진행 상황은 잠금화면 Live Activity와 Android 진행 알림으로 보여 줍니다.

AI 원가 · 안전

음성 세션의 원가 누수를 막는 장치

문제 음성 세션은 켜 두기만 해도 원가가 샌다.

결정 무음 자동 종료, 워치독, 세션 시간 상한, 연속 실패 차단, 서비스 전체의 하루 원가 킬스위치를 겹쳐 둡니다. 말로 지우거나 고친 기록의 원본은 기기에만 24시간 보관하고 서버에는 남기지 않습니다.

오프라인 · 멱등

끊겨도 기록은 한 번만

문제 헬스장은 전파가 약하다. 재전송이 겹치면 같은 기록이 두 번 생긴다.

결정 기록을 로컬 큐에 먼저 쓰고 복귀 시 멱등키로 재전송합니다. 형식이 틀린 키는 요청을 거부하지 않고 멱등만 포기합니다 — 큐 전체가 막히지 않게.

배포 · 개발 체계

심사 없는 패치와 무중단 배포

문제 Dart 코드만 바뀐 수정도 매번 스토어 심사를 거쳐야 했다.

결정 Shorebird 코드 푸시를 4.0.2부터 도입했습니다(2026.09.28 App Store 출시) — 이제 승인된 릴리스 위 Dart 수정은 심사 없이 패치로, 네이티브 · 권한 · 에셋이 바뀌면 정식 릴리스로. 서버는 롤링 배포로 올립니다. 개발은 앱 · API · 웹 · 인프라 에이전트가 소유 경로를 나눠 병렬로 하고, 검증 증거가 기록에 없으면 완료로 보지 않습니다.

무중단 배포 중 1초 간격 420회 요청 전부 200

04 Conoti AI

App Store · Google Play · conoti.app · 서비스 구조 도식

AI가 공지를 쓰고 보내는 모임 운영 앱

강사 · 학원 · 동호회 같은 모임 운영자가 AI와 대화하는 것만으로 공지를 작성 · 예약 · 발송합니다. ChatGPT · Claude 같은 외부 AI도 MCP로 같은 도구를 씁니다. 1.5.0부터는 PC의 Conoti AI 데스크톱 앱을 폰에서 보는 "AI 작업" 탭이 붙었고, 1.6.0에서 PC 여러 대 · Codex 작업 · 목소리로 지시하는 음성 대화를, 1.7.0에서 잉크 구매 · 15개 언어 · 폰에서 PC 세션을 제어하는 기능까지 넓혔습니다.

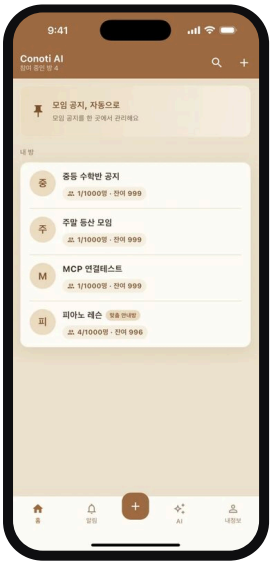
출시	2026.06 App Store · Google Play · 현재 1.7.0+120 · 패치 1
역할	기획 · 앱 · 웹 · 백엔드 · AI · 인프라
스택	Flutter · FastAPI · PostgreSQL · pgvector · React · APScheduler · FCM
언어	15개 — 한국어 · 영어 · 베트남어 · 태국어 · 크메르어 · 라오어 · 미얀마어 · 필리핀어 · 일본어 · 스페인어 · 포르투갈어(브라질) · 프랑스어 · 독일어 · 번체 중국어 · 인도네시아어

버전

1.0	2026.06 — App Store · Google Play 출시
1.4.2	2026.08.10 — 웹 푸시, MCP 중심 랜딩(당시 8개 언어)
웹앱	2026.08.27 — React 웹앱 앱 기능 동등화, 보안 헤더
1.4.3	2026.09.25 — Shorebird 코드 푸시 도입 (같은 주 서버 gpt-6 전환)
1.5.0	2026.09.27 — Conoti AI 데스크톱 앱과 종단간 암호화 중계 · AI 작업 탭
1.6.0	2026.09.29 — AI 작업에 PC 여러 대 · Codex 작업 · 음성 지시 · 밀어서 답장
1.7.0+120	2026.10.15 — 잉크 구매 · 7개 언어 추가(15개) · 음성 대화 인사와 마이크 복구 · 내 데이터 내보내기 · 잉크 이력
패치 1	2026.10.15 — 폰에서 PC 허용 확인 · 질문 답하기, 권한 모드 · 모델 · effort 바꾸기, 구독 사용량 확인, 대화 압축, 선택지 카드

App Store · Google Play · conoti.app · 서비스 구조 도식

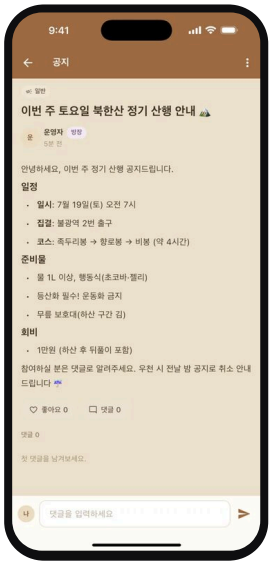
실제 사용 화면



모임별 공지 게시판



AI가 쓰는 공지 초안



참여자는 수신 전용



외부 AI 연동

Conoti AI 모바일 앱 분해 — AI는 초안까지, 보내는 건 사람이. 자율성을 넓히는 설계와 사고를 막는 설계를 한 쌍으로.

01 · 화면 **운영자와 AI의 대화**	02 · 확인 토큰 **한 번만 · 시간 제한 · 내용 해시**	03 · MCP 서버 **도구 21종**
04 · 인가서버 **MCP 표준 OAuth 2.1**	05 · 의미검색 **pgvector 사이드카**	06 · 발송 **스케줄러 → FCM**

사용자가 실제로 쓰는 기능을 쉬운 말로 적었습니다 — AI가 맡는 부분과 서버 · 규칙이 맡는 부분을 나눠서.

01

대화로 쓰는 AI 공지

운영자가 AI에게 말로 부탁하면 제목 · 본문 · 일정 · 반복 규칙을 담은 공지 초안을 씁니다. 지난 공지를 물으면 의미 검색 결과를 근거로 답하고, 읽음 현황을 보고 안 읽은 사람에게만 다시 보내자고 제안합니다. 실제 등록 · 발송은 사람이 확인 버튼을 누를 때만 됩니다.

02

MCP로 공지를 등록하는 서버

ChatGPT · Claude 같은 외부 AI나 코딩 에이전트가 MCP 도구로 공지 초안 · 예약 · 반복 등록 · 발송 상태 확인 · 안 읽은 사람 재발송을 합니다. 아침 날씨 · 환율 요약, 이번 주 시간표 · 준비물, 회의록 요약, 서버 점검 알림처럼 사람이 매번 쓰던 공지를 AI에게 맡깁니다.

03

운영 알림 채널로도 직접 사용

모임 공지용으로 만든 앱을 개발 · 운영 알림 채널로도 씁니다. 서버 상태 점검 결과와 코딩 에이전트의 작업 완료 · 확인 대기 · 오류 알림이 Conoti AI 방으로 올라와 폰 푸시로 옵니다 — 에이전트 알림은 MCP 도구로 올립니다.

04

PC의 코딩 에이전트를 폰에서

PC의 Conoti AI 데스크톱 앱이 Claude Code · Codex의 대화 기록을 읽고 Claude Code 훅으로 완료 · 권한 대기를 바로 받아 PC 안의 파일 하나(SQLite)에 모으고, 폰의 "AI 작업" 탭이 그 세션을 종단간 암호화로 이어 받아 결과를 보고 답장합니다. 서버는 암호문만 넘기고 저장하지 않습니다.

05

목소리로 하는 원격 지시

채팅을 치는 대신 폰에 말로 PC의 코딩 에이전트에게 일을 시키고 결과를 듣습니다. 음성 AI(제미나이 라이브 또는 GPT-Live)는 세션 목록 · 지시 보내기 · 결과 듣기 · 끝내기 네 가지 도구만 쓰고, 그 도구는 폰이 실행해 암호화 통로로 PC에 전합니다. 되돌리기 어려운 지시는 한 번 더 확인하고, 서버는 받아쓴 글을 저장하지 않습니다.

06

폰에서 PC 세션 제어 (1.7.0 패치 1)

PC 세션이 허용 확인이나 질문으로 멈추면 폰에서 답하고, 세션의 권한 모드 · 모델 · effort를 바꿉니다(tmux · IDE 터미널은 바로 적용). Claude · Codex 구독 사용량 확인, 이미지 요청 허용 · 거부, 대화 압축 · 비우기, AI 답 끝의 선택지 고르기도 폰에서 합니다 — PC 쪽은 Conoti AI 데스크톱 앱 0.14.0이 받습니다.

AI와 함께 개발한 방식

앱 · 웹 · 서버를 코딩 에이전트와 함께 만듭니다. 서버 담당 · 앱 담당 · 리뷰 담당 에이전트로 일할 경로를 나누고, 개발은 개발 트리에서만 한 뒤 운영 트리에는 승격 절차로만 반영합니다.

- 완료 기준을 미리 정해 둡니다 — 서버 테스트 통과, 앱 정적 분석 경고 없음, 시뮬레이터 실행 확인, 15개 언어 넘침 검사, 화면 진입 경로 테스트.
- 보안 점검도 에이전트로 합니다 — 여러 에이전트가 취약점을 찾고 다른 에이전트가 반박하며 검증해, 검증을 통과한 지적을 모두 고쳤습니다.
- 만든 제품을 개발 도구로 다시 씁니다 — 에이전트 작업 알림은 Conoti AI 방으로 받고, PC에서 돌던 에이전트 세션은 Conoti AI 데스크톱 앱으로 폰에서 이어 받습니다.

MCP · OAuth 2.1

ChatGPT가 접속하는 인가서버를 직접

문제 ChatGPT 커넥터를 붙이려면 MCP 표준 인증이 필요하다. 그런데 이미 발급키로 쓰는 사용자가 있었다.

결정 MCP Authorization 명세와 OAuth 2.1, RFC 9728 · 8414 · 7591 · 7636 · 8707 · 7009 · 9207을 따라 인가서버를 구현했습니다. 기존 계약을 깨지 않으려고 URL로 인증 모드를 나눴고(발급키 · OAuth · 경로키), 토큰은 키 ID만 담아 호출마다 권한을 다시 검사하게 해 즉시 폐기가 됩니다.

MCP 서버 · 도구 21종

외부 AI에게 어디까지 열 것인가

문제 외부 AI가 공지 작성 · 예약 · 발송 · 검색 · 읽음 현황까지 다룰 수 있어야 한다.

결정 읽기 도구는 권한 없이 열되, 댓글과 읽음 현황은 개인 행동 데이터라 읽기라도 권한을 요구하고 기본값을 꺼 두었습니다. 새 도구는 기존 권한을 재사용해 이미 발급된 키가 그대로 동작합니다. 방 · 도구 단위 범위 제한과 호출 감사를 둡니다.

확인 토큰

AI는 초안까지, 등록은 사람이

문제 AI가 잘못 알아들으면 모임 전체에 잘못된 공지가 나간다.

결정 실제 등록 직전에 한 번만 쓸 수 있고 시간이 제한되며 내용 해시가 묶인 확인 토큰을 발급합니다. 내용이 바뀌면 토큰이 무효가 됩니다. 발송은 파일 잠금 · 상태 CAS · 멱등키 세 겹으로 중복을 막습니다.

종단간 암호화 · Noise

서버가 내용을 볼 수 없는 중계

문제 폰에서 PC의 코딩 에이전트 세션을 보고 답하려면 서버를 거쳐야 한다.

결정 처음 설계한 평문 저장 방식을 취소하고, Noise 프로토콜(IKpsk2_25519_ChaChaPoly_SHA256)로 폰과 PC가 직접 암호화하게 바꿨습니다. 서버는 암호문만 넘기고 저장하지 않습니다. 폰 쪽 Dart 구현은 PC가 뽑은 기준값과 바이트 단위로 대조하고, 페어링은 6자리 확인 코드로 중간자를 막습니다. 운영 배포 전 보안 점검을 두 번 거쳤습니다.

pgvector 사이드카

남의 서비스를 멈추지 않고 의미검색 추가

문제 공유 Postgres는 pgvector가 없고, 이미지를 바꾸면 같은 호스트의 다른 서비스까지 재시작된다.

결정 임베딩 벡터만 전용 pgvector 사이드카에 두고 본문은 메인 DB에 둔 채 ID로 이어 붙였습니다. Gemini 임베딩(768차원). 키나 pgvector가 없으면 키워드 검색으로 자동 전환합니다.

AI 원가 · 단일 재화 잉크

모든 AI 사용을 재화 하나로

문제 요금제마다 한도를 따로 두면 한도에서 사용자를 그냥 막게 되고, 정가표 · 원가 관리도 기능마다 갈라진다.

결정 AI 사용을 단일 재화 "잉크"로 모았습니다. 판매는 Pro 구독 하나와 소모성 잉크 팩 7종이고, 보상형 광고를 보면 그 광고가 실제로 낸 수익에 비례해 잉크를 적립합니다(서버 측 검증). 잉크는 기능별 실제 원가로 차감하고, 쓴 곳과 시각은 잉크 이력으로 보여 줍니다. ELANFIT의 에너지 과금과 같은 구조라 두 서비스의 회계 · 화면 · 코드가 같습니다.

배포 · Shorebird

심사 없는 패치, 섞이지 않는 dev

문제 Dart 수정마다 심사를 기다려야 했고, 같은 앱 ID를 쓰면 dev 패치가 운영 앱에 들어간다.

결정 1.4.3부터 Shorebird 코드 푸시를 도입했습니다(2026.09.25 App Store 출시) — 이제 Dart 수정은 심사 없이 패치로 낼 수 있고, dev는 별도 앱으로 분리했습니다. 한 저장소 안에서 개발 트리와 운영 트리를 나누고 승격 절차로만 운영에 반영합니다.

실패 → 테스트

사람이 겪은 실패를 게이트로

문제 테스트는 통과했는데 사용자 손에서 깨지는 일이 반복됐다.

결정 iOS 실행 직후 크래시로 심사 거절된 뒤 "첫 프레임 전에 외부 SDK를 기다리지 않는다"를 테스트로 고정했습니다. 진입점 없는 화면을 겪은 뒤 도달 가능성 테스트를 넣었고, 통합 테스트는 초록인데 dev 빌드가 네 번 연속 설치 직후 깨진 뒤(원인: 실제와 다른 모양의 테스트 데이터) 실전과 같은 데이터로 도는 시뮬레이터 E2E 시험을 규칙으로 만들었습니다.

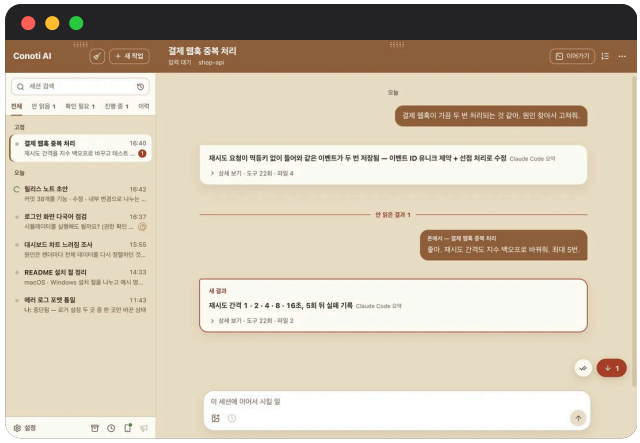
05 Conoti AI 데스크톱 앱

github.com/YeoJeongHun1/conoti-ai/releases

코딩 에이전트 세션을 메신저처럼 — macOS · Windows

여러 세션을 동시에 돌리면 어느 세션이 끝났고 어느 세션이 확인을 기다리는지 놓칩니다. 세션을 대화방처럼 모으고, 안 읽음과 권한 대기를 표시하고, 앱이나 폰에서 실행 중인 세션에 바로 말을 넣습니다. Claude Code 세션에 더해 0.6.0부터 OpenAI Codex 세션도 같은 목록에 모입니다.

- 2026.09 개발 시작 → 0.14.0(2026.10.16 공개) — 0.12.0에서 이름을 AI Inbox에서 Conoti AI로 바꿈
- Claude Code · Codex의 대화 기록 파일을 읽고 Claude Code 훅(세션 시작 · 요청 · 알림 · 완료 등)으로 상태를 바로 받아, PC 안의 SQLite 파일 하나에 모음 — 별도 서버 · 원격 측정 없음
- Conoti AI 모바일 앱의 "AI 작업" 탭에서 결과를 보고 답장하고, 모바일 1.6.0부터는 목소리로도 지시(음성 AI가 고른 도구를 폰이 실행해 암호화 통로로 전달)
- 0.14.0 — 폰에서 보낸 허용 확인 · 질문 답, 권한 모드 · 모델 · effort 변경, 구독 사용량 · 대화 압축 요청을 PC에서 처리(모바일 1.7.0 패치 1과 짝)
- Codex 세션도 같은 목록에서 보고 이어서 시킴 — 열린 세션엔 바로(일하는 중이면 끝난 뒤), 꺼진 세션은 백그라운드로 이어서 실행, 새 작업에서 Claude Code · Codex 선택
- 위협 모델을 보안 문서에 정본으로 두고, 보안 검토 지적을 반영하며 기능을 올림



세션 목록 · 폰에서 온 답 · 안 읽은 결과 — 데모 데이터

여섯 층 — 서버는 내용을 모릅니다. 처음 설계한 평문 저장 방식을 스스로 취소하고 종단간 암호화로 바꿨습니다.

01 · 화면
React 19
세션마다 요청과 결과가 대화처럼 쌓이고, 안 읽은 결과 · 확인 필요 · 진행 중을 나눠 봅니다. 결과는 문서로 펼쳐 요청 · 작업 요약 · 바뀐 파일 · 응답을 한 장에 봅니다.

02 · 앱 코어
Tauri 2 · Rust · SQLite
모든 기록은 로컬 파일 하나. 저장 전에 API 키 · 토큰 같은 비밀값을 가립니다. 업데이트는 앱에 고정된 공개키로 서명을 확인한 뒤, 사용자가 누를 때만 설치합니다.

03 · 훅
Claude Code 훅 + 대기 훅
완료 · 권한 대기를 훅 이벤트로 즉시 받고, asyncRewake 대기 훅으로 실행 중인 세션에 재시작 없이 말을 넣습니다. 세션이 쉴 때만 넘기고 30초 지난 입력은 버립니다.

04 · 암호화
Noise IKpsk2 · 6자리 확인 코드
키는 폰이 PC 화면의 QR(5분 · 1회용)을 직접 찍어 교환합니다. 페어링 때 양쪽에 같은 확인 코드를 보여 중간자를 막습니다.

05 · 중계
Conoti AI 백엔드
방 번호로 짝만 지어 주고 암호문만 넘깁니다. 저장하지 않고, 푸시 알림에도 내용을 싣지 않습니다.

06 · 폰
Conoti AI 모바일 앱 1.7.x · AI 작업 탭
폰에서 온 답은 원격 명령으로 봅니다. 허용한 기기 · 세션별 스위치 · 전체 멈춤 · 데스크톱 확인 · 길이와 제어문자 검사를 겹쳐 두고, 명령은 셸을 거치지 않습니다.

05 자체 앱 · 외주 개발

법인 자체 앱 · 클라이언트 과제

자체 앱 · 기획 · 앱 · 2026.10 –
CopyZoom
- 글씨를 연락처 · 계좌 · 주소 · 금액 등 묶음으로 나눠 탭 한 번에 복사
- 온디바이스 OCR · 주민번호 가림 · 계좌 2분 만료 복사 — 2026.10 출시(대한민국)

Flutter | Swift · VisionKit | Kotlin · ML Kit | AdMob

자체 앱 · 설계 · 서버 · 앱 · 2024.12 –
실시간 점수 집계 앱
- Firebase Realtime Database로 실시간 점수 동기화
- Flutter 크로스플랫폼 앱(Android · iOS)
- FastAPI 서버 · AWS EC2 호스팅, 배포 후 고도화

Flutter | Firebase | FastAPI | AWS EC2

외주 · 프로젝트 총괄 · 2024.07 – 2024.10
유어라운드 교육 서비스 웹앱
- 요구사항명세서 · DB 설계(ERD) · 화면설계서까지 전체 설계
- FastAPI 백엔드 · Vue 3 프론트엔드 개발
- 클라이언트 니즈 분석, 고객 커뮤니케이션, 인력 관리까지 프로젝트 총괄

FastAPI | Vue 3

06 AI 스터디 — 신규 모델 · 도구 벤치마크 공개 자료

portfolio.eldsoft.com/ai_edu

31
공개 주제
2026.08.25 — 09.22

558
슬라이드 (덱 27개)
+ 스크롤 문서 4편

34
해설 · 플레이 영상
대본 → 로컬 TTS → 자동 합성

실측 · 검증 결과가 있는 주제

09.22 **0.6초 늦는 AI에게 실시간 게임을 시켰다**
AI 64.3초 · 규칙 66.0초 — 천장은 정책이 아니라 난이도 곡선

09.22 **미사일 피하기 실험실**
"최고 142초는 실력이 아니라 운" — 자기 기록을 재실행으로 반박

09.21 **판단만 하는 AI에게 게임을 시켰다**
응답 0.56초 중 서버 계산은 0.06초 — 나머지는 오가는 시간

09.15 **GPT-Live-1 — 우리 앱에 붙여 봤다**
같은 시험을 바깥에서 재니 18점이 빠졌다

09.14 **구글 ARTEMIS — 말로 시키는 안드로이드 테스트**
앱 안은 놀렀고, 앱 밖은 못 나갔다 · 99.1%의 측정 조건 공백 지적

09.12 **표식 그 다음 — 지우는 도구, 저작권, 학교와 관공서**
탐지기 13종이 같은 원고에 0~100%로 답한 연구

09.07 **코드로 그린 그림 — 생성 AI 만큼 나오는가**
30장 중 12장에서 자기 평가가 독립 심사보다 후했다

09.03 **코딩 점수표를 읽는 법**
점수와 같이 적어야 할 여섯 가지

09.03 **코딩 모델 점수판 — 같은 문제로 줄 세우기**
1판 9개 모델 전원 만점 → 2판은 "얼마나 싸게 해내나"로 재설계

08.26 **가장 게으른 시니어처럼 — ponytail**
여섯 번 다 짧아졌다 — 61줄에서 11줄

08.26 **클로드가 쓴 글에 표식이 남는다**
출처가 갈린 곳 5 · 아직 확인 못 한 것 11을 따로 적음

08.25 **디자인 스킬 하나가 결과를 얼마나 바꾸나**
결과로 쓰던 스킬 여섯 개 중 다섯 개를 지웠다

출처 3표시

문장마다 실측 / 내가 보탠 것 / 확인 필요를 가릅니다. 색이 아니라 채움으로 구분해 흑백에서도 읽히고, "확인 필요"를 "실측"으로 올리는 것은 금지입니다.

AI 스터디 도우미 (챗봇)

BM25 검색 → 모델이 JSON으로 답 → 서버가 실제로 넘긴 조각 id만 출처로 인정합니다. 요청 속도 · 길이 · 하루 비용 상한, 규칙 기반 즉시 거절(지시문 공개 · 탈옥 · 개인정보), 범위 판정을 겹쳐 둡니다. 시험지 45문항 전부와 대화 흐름 5단계를 통과한 뒤 공개.

해설 영상 자동 제작

대본이 정본입니다. 대본 → 점검 → 캡처 → 오픈 웨이트 TTS를 로컬에서 합성 → 받아쓰기로 발음 검증 → 합성. 5분 상한, 첫 30초는 결론 요약.

고정 무대 덱

1920×1080 무대를 창 크기에 맞춰 통째로 줄입니다. 글자를 줄이지 않고 장을 쪼개고, 폰 세로에서만 세로 문서로 바뀝니다. 기본 청중은 비전문가 — 전문용어는 그 자리에서 한 줄로 정의합니다.

01 모르는 것을 아는 척하지 않는다

개발자로서 가장 위험한 태도는 모르는 것을 아는 척하는 것이라고 봅니다. AI를 붙이면 이 위험은 몇 배가 됩니다 — 모델은 모른다고 말하는 대신 그럴듯하게 지어내기 때문입니다. 경험이 없는 영역은 없다고 적고, 대신 가장 가까운 경험과 지금 준비하고 있는 것을 함께 적습니다. 예를 들어 컨테이너 오케스트레이션은 Docker Compose 까지가 제 실무 범위이고, Kubernetes 는 운영해 본 적이 없어 이 사이트 어디에도 경험으로 적지 않았습니다.

02 이견이 있으면 상대의 배경부터 묻는다

제 생각과 맞지 않는 제안을 받으면 먼저 왜 그렇게 제안했는지 설명을 요청합니다. 같은 조직에 있어도 각자 봐온 환경이 달라서, 제가 보지 못한 것을 보고 있을 수 있다고 생각합니다. 상대의 근거를 듣고 제 근거도 함께 꺼낸 다음 이 프로젝트에 무엇이 최선인지 같이 정합니다. 제 말을 관철하는 것이 목적이 아닙니다.

03 막히면 안 되는 이유 대신 방법을 바꾼다

기획 · 운영이나 프론트엔드 팀과 부딪힌 적이 여러 번 있습니다. 서로 해내려는 마음이 확인되면 핑계를 찾는 대신 소통 방식 자체를 바꿔서 끝냈습니다. 1인 법인을 운영하며 회사 입장과 클라이언트의 니즈를 동시에 봐야 했던 경험이 여기에 쓰입니다.

04 조율은 요구사항보다 사람을 먼저 본다

법인을 운영하며 디자이너와 프리랜서 개발자 사이에서 중간 조율을 맡았다가 실패한 적이 있습니다. 일정과 산출물을 맞추는 데 집중하느라, 함께 일하던 분이 개인적으로 감당하기 어려운 상황에 있다는 것을 충분히 살피지 못했습니다. 조율은 요구를 맞추는 일이기 이전에 사람의 상태를 살피는 일이라는 걸 그때 알았습니다.

05 도메인은 아는 것보다 익히는 방법이 자산이다

공공조달 문서 도메인도 처음에는 몰랐습니다. 규칙과 예외를 데이터로 하나씩 확인하며 익혔고, 그 과정에서 나온 실패 케이스를 규칙으로 축적하는 방식으로 정확도를 올렸습니다. 도메인을 아는 것보다 도메인을 익히는 방법을 아는 것이 옮겨 다니는 자산이라고 봅니다.

AI

AI에 대한 폭넓은 지식과 경험

시계열 예측에서 LLM 추출 · 검증, 실시간 음성 모델, MCP 에이전트까지 — 새 모델은 실측으로 평가하고 도입합니다.

INFRA

인프라를 직접 세우는 능력

큐 · 배치 파이프라인, dual-write 이관, GCP · AWS · Terraform, 종단간 암호화 중계까지 운영 가능한 형태로 만듭니다.

PRODUCT

제품을 끝까지 완성하는 힘

기획 · 설계 · 개발 · 스토어 출시 · 운영 · 원가 통제까지 — App Store · Google Play에 앱 두 개를 출시해 운영합니다.

다양한 기술 스택이 아니라, AI에 대한 폭넓은 지식과 경험 · 인프라 구축 · 끝까지 완성해 낸 제품이 곧 실력인 시대입니다.

웹 **portfolio.eldsoft.com** GitHub **github.com/YeoJeongHun1**

이 문서는 portfolio.eldsoft.com 의 인쇄판입니다(2026.10.20). 회사 업무는 아키텍처 · 역할 수준으로만 적었고, 서버 · 접속정보 · 고객 데이터 · 물량은 싣지 않았습니다.

성과 수치를 읽는 기준

운영 전후 비교 · 동일 입력 A/B는 서로 다른 측정

비용 절감률은 각 변경 전후의 운영 로그 기준입니다. 서로 다른 기준선의 절감률을 더하거나, 동일 조건의 누적 절감률로 환산하지 않습니다.

지표별 비교 대상과 해석 범위

지표	비교 대상·단위	해석 범위
비용 −14.2%	명시적 캐시 도입 전후 · 검증 호출 비용	캐시 적용 효과
비용 −33.5% **응답 −40.4%**	캐시 적용 상태에서 책임 분리 재설계 전후	LLM 추출과 워커 판정 분리 효과
건당 비용 −82.9% **평균 응답 −80.2%**	가벼운 모델(3.1 Flash-Lite)로 바꾼 전후 · 운영 로그	운영 관측값. 동일 입력 A/B 비용비 8.7배와 별개
91.4% / 90.8%	3.5 Flash 대 3.1 Flash-Lite · 같은 표본·프롬프트(해시 일치) · 항목 단위 대조	상위 모델 참조 정답지 기준 일치율. 절대 정확도와 구분
기본 모델 처리 99.6%	일반 트래픽에서 상위 모델을 호출하지 않은 비율	처리 경로 지표이며 정답률이 아님
자동검수 0 → 80% (시설 분야)	수동 검수에서 조건 기반 자동 통과로 전환한 뒤 · 사내 내부 지표, 시설 분야	시설 분야의 자동 통과 비율이며 전체 분야 대표값 · 업무시간 절감률이 아님

측정 조건과 공개 범위

공개된 자료에는 정확한 측정 기간, 문서·항목별 표본 수, 참조 정답지의 사람 검수 범위가 포함되어 있지 않습니다. 따라서 통계적 유의성이나 전체 문서에 대한 정확도를 주장하지 않습니다. 비용 비교에는 입력량·캐시·모델별 단가가 영향을 주므로 운영 전후 변화와 고정 입력 실험을 구분합니다.

원문 근거를 확인하는 흐름

원리를 설명하기 위한 가상 예시 · 실제 업무 문서·고객 데이터 아님

입력 문서

"기초금액은 100,000원입니다."

구조화 추출

기초금액: 100000
근거: "100,000원"

프로그램 검증

정규화한 값·근거를 원문과 대조하고, 근거가 맞지 않는 항목만 재추출합니다.

책임 분리 전후

변경 전

LLM이 추출과 판정을 함께 수행

변경 후

LLM 추출 → 워커의 원문 근거 대조 → 필요 항목만 재추출

확인한 결과

캐시 적용 기준선 대비 검증 비용 −33.5% · 응답 −40.4%