

이재신

Software Engineer

010-5472-4279
ijaesinii@gmail.com
github.com/ijaesin
ijaesinii.dev ↗

AI를 기존 학습 관리 시스템(LMS)과 연동해 콘텐츠 제작과 질의응답 기능을 개발했습니다. 인증과 권한, 기관별 배포를 맡았으며 장시간 생성 작업과 지속 연결의 상태 관리, 실패 복구, 공급자별 스트리밍과 정산을 개선했습니다.

경력

자이닉스 대학 LMS / AI 교육 제품

2024.11 ~ 현재

개발 TypeScript, NestJS, React, PHP/Laravel, Python

데이터 / AI PostgreSQL, MongoDB, Redis, BullMQ, OpenAI, Claude, Gemini, ElevenLabs

주요 프로젝트

AI Commons

2025.08 ~ 현재

프로젝트 소개 교수자의 강의 콘텐츠 제작 부담을 줄이기 위한 AI 플랫폼입니다. 강의 설계부터 슬라이드, 음성, 아바타 영상 제작까지 지원하며 기존 학습 관리 시스템과 연동합니다. 생성물은 콘텐츠 관리 시스템(CMS)에 보관합니다.

담당 역할 초기 PoC는 고객 및 이사진과 직접 소통하며 기획과 개발을 맡았습니다. 출시부터 현재까지 고객 소통을 담당하는 기획자 1명과 2인 체제로 개발과 운영을 이어가고 있습니다.

주요 기술 NestJS, React, PostgreSQL, Redis/BullMQ, OpenAI, ElevenLabs TTS, FFmpeg

강의계획서 공통 API로 제품마다 반복하던 고객사별 조회 로직을 통합하여 시스템 확장성을 개선하였습니다.

학교마다 강의계획서를 HTML, PDF, 외부 링크, 임베드 뷰어, iframe 등 서로 다른 형태로 제공했습니다.

LMS에 주소나 뷰어 코드만 저장된 학교는 실제 본문을 가져올 수 없어, 과목 개요와 주차별 계획을 LLM에 전달하기 어려웠습니다.

기존에는 새 고객사가 생길 때마다 Tutor와 AI Commons에 학교별 조회 모듈을 따로 추가했습니다. 같은 학사 데이터를 쓰면서도 제품마다 학교별 형식과 코드 변환을 중복해서 관리해야 했습니다.

이 중복을 줄이기 위해 **LMS에 강의계획서 공통 뷰와 API**를 만들었습니다. 학교별 학사 데이터를 공통 뷰에 맞춰 연동하고, 과목 연결, 언어별 주차 선택, Markdown 변환은 한곳에서 처리하도록 했습니다.

초기 설치·업데이트 때 고객사별 연동과 매핑을 설정하면 각 AI 제품이 같은 API로 Markdown 문서를 받도록 했습니다. AI Tutor를 포함한 여러 AI 제품에서 이 API로 강의계획서를 조회하고 있습니다.

자동 복구로 생성 작업의 재시도 준비시간을 46분에서 3분으로 단축하였습니다.

작업을 DB에 저장한 뒤 큐에 넣지 못하면 화면에는 계속 대기 중으로 표시됐고, 재시도도 할 수 없었습니다. 그렇다고 오래 걸린 작업을 모두 실패로 처리할 수는 없었습니다. 아바타 영상은 정상적으로 생성 중이어도 수시간이 걸리기 때문입니다.

워커가 재시작할 때만 복구하던 방식을 바꿔, **5분마다 DB의 작업 기록과 큐 상태를 비교**했습니다. 큐에서 사라진 작업과 실행 중이지만 진행이 멈춘 작업을 구분했습니다. 후자는 작업별 최대 실행시간과 마지막 진행 시각을 함께 확인했고, 큐 조회 자체가 실패한 경우에는 작업이 사라졌다고 판단하지 않았습니다.

상태를 확인하는 동안 작업이 정상 진행될 수 있어, 조회 당시의 상태와 갱신 시각이 그대로인 작업만 변경했습니다. 실패 처리한 작업의 결과가 뒤늦게 도착하면 현재 상태를 다시 확인해, 결과 저장과 사용량 반영에서 제외했습니다.

배포 전후 각 7일의 운영 로그에서 복구 대상 18건과 21건을 비교했습니다. 복구 조건을 만족한 뒤 다시 시도할 수 있을 때까지의 중앙값은 **46분에서 3분**으로 줄었습니다. 변경 후에는 정상 진행 중인 작업을 실패로 처리한 사례가 없었습니다.

서버 전체의 AI 호출량을 함께 조절해 429 발생 작업 비율을 18%에서 4%로 개선하였습니다.

Vertex AI 이미지 생성에서 한 워커가 429 오류를 받고 대기해도 다른 서버는 계속 요청을 보내 같은 오류가 반복됐습니다. 서버를 늘리면 전체 요청 수도 늘어, 워커마다 동시 실행 수를 제한하고 재시도 간격을 늘리는 방법만으로는 해결할 수 없었습니다.

서버별 제한을 유지하는 방법, 워커 하나에서 순서대로 처리하는 방법, Redis로 전체 호출량을 함께 관리하는 방법을 비교했습니다. 순차 처리는 정상 상황의 처리량도 떨어뜨리고, 고정된 호출 상한은 공급자 용량 변화에 대응하기 어려웠습니다. 그래서 **호출 허가**와 **429 이후 대기 상태를 Redis에서 공유**하도록 했습니다.

실행 수 확인과 호출 허가 발급은 Lua로 묶어 원자 처리했습니다. 429가 발생하면 전체 호출 상한을 낮추고, 소수의 요청으로 정상 응답을 확인한 뒤 다시 늘렸습니다. 같은 요청 묶음에서 늦게 돌아온 429 때문에 상한을 여러 번 낮추지 않도록, 요청 당시의 세대 번호도 확인했습니다.

동일한 모델과 해상도로 생성한 운영 이미지 작업을 변경 전후 1,200건씩 비교했습니다. 429를 한 번 이상 겪은 작업은 **216건에서 48건(18%에서 4%)**으로 줄었고, 요청 접수부터 완료까지의 p95는 **96초에서 69초**로 줄었습니다.

2025.04 ~ 현재

프로젝트 소개 대학 구성원의 학습과 업무를 지원하는 AI 서비스입니다. Chat은 여러 LLM 기반 대화와 웹 검색, 파일 분석, 코드 실행, 이미지 생성을 제공합니다. Tutor는 과목 자료와 교수자가 설정한 답변 지침을 바탕으로 학생의 수업 질문에 답합니다.

담당 역할 기획자 1명, 개발자 2명으로 PoC부터 출시까지 진행했습니다. 현재는 핵심 AI 기술 개발과 개선에 집중하고 있습니다.

주요 기술 NestJS, React, MongoDB, Redis/BullMQ, SSE, OpenAI, Claude, Gemini

AI 응답 처리를 통합해 이벤트 중복과 인용 누락 비율을 15.6%에서 0%로 개선하였습니다.

Claude를 추가하면서 답변 텍스트뿐 아니라 웹 검색 인용, 코드 실행 결과, 사용량 이벤트의 형식도 기존 모델과 달랐습니다. 화면에서 공급자별로 처리하지 않고, **서버의 공통 스트리밍 계층에서 공급자별 이벤트를 같은 SSE와 저장 형식으로 변환**했습니다.

도구 실행이 pause_turn 상태로 멈추면 원본 응답 블록을 보관해 같은 요청을 이어갔습니다. 다음 사용자 질문에는 저장된 텍스트로 대화 이력을 재구성했습니다. 도구 실행을 재개할 상태와 다음 질문에 전달할 이력을 따로 관리했습니다.

재개할 때 생성 이벤트가 다시 전송되거나 인용이 사라지지 않도록, 최초 이벤트 전송 여부와 누적 결과를 재개 반복문 밖에서 관리했습니다. 새 공급자를 추가해도 화면에서는 같은 생성과 완료 이벤트를 처리할 수 있게 했습니다.

웹 검색, 코드 실행, 도구 재개를 포함한 30개 시나리오에서 생성 이벤트가 중복되거나 인용이 빠진 흐름은 **15.6%에서 0%로** 줄었습니다.

캐시를 조정해 4턴 사용량 비교에서 입력 비용 지수 37.1% 감소를 확인하였습니다.

Claude 요청에는 매번 지시문과 대화 이력이 포함됩니다. 캐시를 오래 유지해도 다음 질문이 없거나 입력 앞부분이 바뀌면 캐시 쓰기 비용만 늘 수 있어, 실제 대화의 길이와 질문 간격을 먼저 확인했습니다.

한 기관의 최근 90일 대화 **15,232개 중 10턴 이하는 83.3%, 모든 턴 간격이 5분 이내인 대화는 71.7%**였습니다. 당시 캐시 쓰기 단가는 일반 입력 대비 5분 유지 시 1.25배, 1시간 유지 시 2배였습니다. 질문 간격과 쓰기 비용을 고려해 기본 TTL을 5분으로 정했습니다.

대화가 길어져 최근 20개 메시지만 전달하면 입력 앞부분이 매번 달라집니다. 이때는 대화 이력에 캐시를 적용하지 않고 고정 지시문에만 유지했습니다. 실제로 다시 사용할 수 있는 입력에 캐시를 적용해 불필요한 쓰기 비용을 줄이도록 했습니다.

별도 4턴의 사용량 기록에 당시 요율을 적용해 비교한 결과, 입력 비용 지수는 **5,348에서 3,364.3으로 37.1% 감소**했습니다.

Night Owl ↗

프로젝트 소개 반복 테스트와 코드 검토를 예약하고, CLI 에이전트의 실행 기록과 변경 내용을 확인하는 사내 자동화 도구입니다.

반복 테스트와 코드 검토를 예약 실행하고, 팀이 실행 기록과 변경 내용을 확인하도록 구현하였습니다.

반복 테스트와 코드 검토를 예약 실행하는 사내 도구를 개발했습니다. 작업별 지시문·사용 도구·비용 한도를 정하고 CLI를 사용자 응답 없이 실행하는 자식 프로세스로 만들었습니다.

CLI 출력을 메시지·도구 호출·종료 결과로 나눠 SQLite에 저장하고 원문은 JSONL로 남겼습니다. 진행 기록은 SSE로 화면에 전달하며 종료 후에도 조회할 수 있습니다. 완료 이메일에 실행 기록을 담아 결과와 변경 내용을 검토하도록 했습니다.

Xinics Skill Plugin ↗

프로젝트 소개 AI 개발에 쓰는 공통 절차와 제품별 개발 및 배포 지침을 Claude Code와 Codex에서 재사용하도록 구성한 사내 플러그인입니다.

공통 개발 절차와 제품별 지식을 정리해 익숙하지 않은 제품에서도 작업을 진행하도록 지원하였습니다.

모든 제품의 스킬을 함께 설치하면 해당 작업과 무관한 설명도 문맥에 포함돼 다른 제품의 지침이 선택될 수 있었습니다. 공통 개발 절차는 xinics 플러그인으로, 제품별 개발, 릴리스, 배포 지침은 해당 제품 플러그인으로 분리했습니다. 제품에 익숙하지 않은 개발자도 필요한 지침을 참고해 작업을 진행할 수 있도록 했습니다.

공통 개발 절차에는 OpenSpec과 Superpowers를 활용한 요구사항 정리, 구현, 검증 흐름을 담고, Jira 이슈와 Confluence 문서를 사용하는 사내 업무 방식에 맞췄습니다.

제품별 플러그인에는 개발 지침과 함께 사업·기획·영업 자료를 찾는 경로도 추가했습니다. 제품 배경과 업무 맥락을 참고하는 사내 지식 기반인 '자이닉스 브레인'으로 활용하도록 했습니다.

학력 및 자격

한양대학교 원자력공학과 학사

2017 ~ 2022

정보처리기사 한국산업인력공단