

이채현 | Chaehyeon Lee

Software Engineer

Product-minded Full-stack (Frontend 중심)

Ich50484084@gmail.com | github.com/chlee1001

요약 (Professional Summary)

React와 TypeScript로 웹과 앱 서비스를 만들고 운영해 온 5년 경력의 프론트엔드 중심 풀스택 엔지니어입니다. 지금은 출결과 근태 서비스를 개발하고 운영하는 6인 Platform Experience(PX) 팀을 이끌고 있습니다.

반복되는 고객 문의나 장애는 원인을 끝까지 추적해 고치고 반복 작업을 줄이는 도구를 만들어 팀과 함께 씁니다. AI 코딩 에이전트를 쓰면서 네이티브 모듈이나 서버처럼 주력 밖 영역의 문제도 직접 재현하고 확인해 해결합니다.

- GPS 인증 실패 개선:** 위치 인증 실패 원인을 셋으로 나눠 추위 방식과 복구 화면을 바꾼 뒤, 2024.01 배포부터 2026.02 인계까지 제가 응대한 고객 문의에서 GPS 문의 0건
- 규칙이 많은 1만 행 편집 테이블:** 3인 FE 팀의 테크 리드로 행 상태마다 편집 규칙이 다른 1만 행 자산 테이블을 TanStack Table과 Virtual로 직접 만들고 JSON으로 선언한 규칙 1,208개를 TypeScript로 바꾸는 컴파일러를 설계해 5개월 일정대로 마침
- AI 코딩 에이전트와 만든 출퇴근 서명 검증 테스트:** 실제 DB에서 출퇴근 흐름을 돌리는 테스트를 에이전트와 함께 만들어 단위 테스트로는 보이지 않던 서명 결함 2건을 찾고 이 테스트를 CI에 넣음

경력 개요 (Career Overview)

주식회사 엘핀은 위치 인증 기반 출결과 근태 제품을 운영하고 수탁 개발을 하는 직원 20명 규모의 회사입니다.

기간	소속 / 역할	주요 업무 및 대표 성과
2026.07 ~ 현재	주식회사 엘핀 PX 팀 리더	- 출결과 근태 서비스 4개를 나눠 맡는 6인 팀 리딩, 매주 서비스 리뷰 - 사내 LLM 시험 환경 구축
2025.03 ~ 현재	주식회사 엘핀 워크플러스 개발 리더 (3인 팀)	- 신규 B2B 근태 솔루션 풀스택 기능 설계 및 관리자 웹 개발 - 인증과 개인정보 저장 방식 무중단 전환 - 사내 공통 디자인 시스템 및 프로젝트 생성 CLI 배포
2022.10 ~ 2026.02	주식회사 엘핀 아임히어워크 운영 담당	- 위치 인증 근태 모바일 앱(RN) 및 관리자 웹 개발과 운영 - GPS 인증 개선과 레거시 CMS의 점진적 Vite 전환
2024.02 ~ 2025.11	주식회사 엘핀 수탁 프로젝트 FE 리드	- 대형 통신사 자회사 자산관리 웹 (FE 테크 리드) - 대형 통신사 임직원 쇼핑몰 (FE 단독 1인, React SPA 재구축)
2021.09 ~ 2023	주식회사 엘핀 Software Engineer	빌딩챗 웹/CMS, 노닥노닥 앱, 완성차 그룹 PoC 등 신규 서비스 개발

기술 스택

React 19, TypeScript, Vite, TanStack Router/Query/Table/Virtual, Mantine, React Native, Kotlin/Spring Boot (설계 및 협업 검증), AWS (ECS, IAM, RDS), Terraform, Proxmox, Jenkins, Docker, Nginx, Gitea Actions (CI/CD)

주요 프로젝트 상세 (Key Projects)

1. 대형 통신사 자회사 자산관리 시스템

- 기간: 2025.07 ~ 2025.11 (5개월)

- **역할:** 프론트엔드 테크 리드 (3인 팀). 1만 행 테이블은 본인이 만들고 규칙 컴파일러는 본인이 설계. 팀원 두 명은 자산 밖 메뉴 화면 담당
- **기술 스택:** React, TypeScript, TanStack Router/Query/Table/Virtual, Vite

배경 (Context)

고객사가 엑셀로 관리하던 통신 장비와 부동산 자산 대장을 웹으로 옮기는 프로젝트였습니다. 자산 분류 하나만 1만 행이 넘었고 행 상태와 권한에 따라 셀마다 편집 가능 여부가 바뀌며 종속 선택이 여러 단계로 얹혀 있었습니다.

구현과 판단

- **TanStack으로 만든 1만 행 편집 테이블:**
 - 이 프로젝트처럼 행 상태마다 편집 규칙이 다르면 완성형 그리드에서는 규칙을 컬럼 설정마다 나눠 적어야 해서 헤드리스(headless) TanStack Table에 TanStack Virtual을 조합해 보이는 행만 그리는 편집 테이블을 만들고 규칙은 데이터 모델 한곳에 둬.
 - 감수한 것: 그리드가 기본으로 주는 변경 내역 확인을 직접 만들어야 했음. 편집 규칙이 단순한 프로젝트라면 완성형 그리드를 고르겠음.
- **규칙 컴파일러 설계:**
 - 컬럼별 검사, 종속 선택, 조건부 편집 규칙 1,208개를 화면 코드에 흠뻑 써면 기획이 바뀔 때마다 여러 파일을 손으로 고쳐야 했음.
 - 그래서 규칙을 JSON으로 선언하고 컴파일러가 이를 읽어 TypeScript 코드를 생성하게 설계함.
 - 기획 변경으로 처리 모드 순서가 바뀌었을 때 컬럼 규칙 파일 39개를 손으로 고치지 않고 컴파일러로 다시 생성함.
- **변경 내역 관리:**
 - Chrome 개발자 도구 Performance 탭에서 가상화 스크롤과 셀 편집 때 프레임이 떨어지는 원인이 편집 변경 내역을 배열에서 매번 찾는 데 있다는 것을 찾고 변경 내역을 행 키로 바로 찾는 Map에 두게 바꿈.

트러블슈팅 (Troubleshooting)

- **문제:** 안정화 기간에 결함 두 가지가 나옴. 보기 모드로 바뀌고 고치던 셀의 노란 편집 표시가 그대로 남았고 행을 고친 뒤 필터나 페이지를 바꾸면 변경 내역 화면에서 그 행의 나머지 값이 비어 보였음.
- **원인 분석:** 보기 모드 버튼은 데이터만 보기 모드로 바꾸고 화면에 열려 있던 셀 편집은 닫지 않았음. 변경 내역에는 고친 칸의 값만 남아 있어서 필터나 페이지가 바뀌어 그 행이 화면 데이터에서 빠지면 나머지 칸을 채울 원래 값이 없었음.
- **해결:** 모드를 바꾸기 전에 셀 편집부터 닫게 함. 행을 처음 고칠 때 원래 행 전체를 따로 저장해 두고 변경 내역 화면은 이 값으로 채움. 변경 내역 화면에서 되돌린 값이 다시 들어가지 않도록 이 화면이 열린 동안에는 고치던 행을 잠금.
- **검증:** 모드 전환, 필터 변경, 페이지 이동을 섞어 가며 화면에서 직접 확인함.

팀 리딩 및 결과

- 프로젝트 초기에 기획 인력이 바뀌어 요구사항이 흐려지자 PM과 직접 화면 기준을 정하고 공통 컴포넌트와 레이아웃 규칙을 먼저 정해 팀원끼리 같은 화면을 두 번 만들지 않게 함.
- 2025년 10월 통합 테스트와 11월 안정화까지 일정대로 마치고 유지보수를 팀원에게 인계함.

2. 사내 디자인 시스템과 릴리스 파이프라인 구축

- **기간:** 2026.01 ~ 현재 (진행 중)
- **역할:** 초기 2인 태스크포스(TF). 컴포넌트는 동료와 나눠 만들고 아키텍처 설계, 개발 도구, 릴리스 파이프라인, 도입 가이드는 본인 담당
- **기술 스택:** React 19, TypeScript, Mantine, Storybook, Vitest, Nx, Gitea Actions/Packages

배경 (Context)

새 관리자 웹을 시작할 때마다 로그인, 레이아웃, 공통 품을 다시 붙이느라 며칠씩 들었고 프로젝트마다 구조가 달라 다른 사람이 말으면 처음부터 다시 익혀야 했습니다. 공통 패키지를 여러 개 배포하면서 파이프라인끼리 충돌하거나 일부 패키지가 배포에서 빠지는 일도 있었습니다.

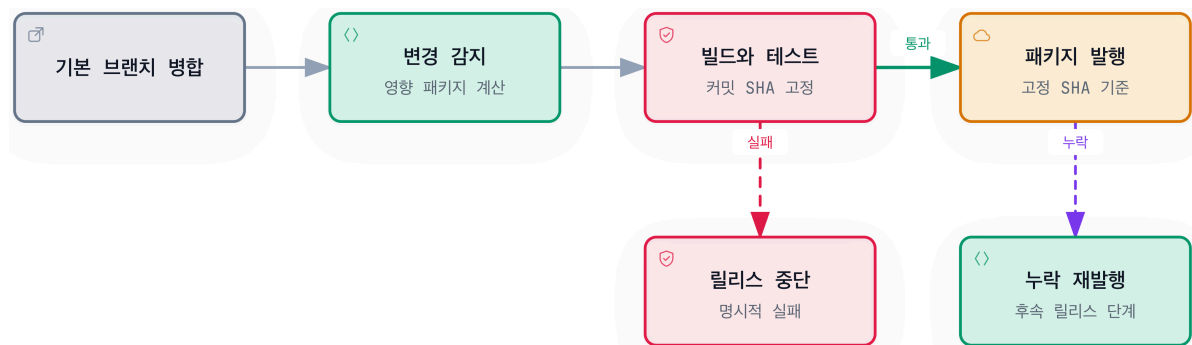
실행

- **프로젝트 생성 CLI:** 컴포넌트만 배포하지 않고 폴더 구조, 라우팅, 공통 UI가 미리 들어간 프로젝트를 만드는 CLI(create-lfin-app)를 만듦. 전에는 초기 세팅에 며칠이 걸렸는데 CLI로 만들면 첫날 바로 화면 작업에 들어감.
- **모노레포 구성:** 패키지 여러 개를 Nx 모노레포 하나에 두고 바뀐 패키지만 골라 빌드하고 배포함.

- **AI 코딩 도구용 MCP 서버와 스킬:** AI 코딩 도구가 사내 컴포넌트와 디자인 토큰을 찾아 쓰도록 MCP(Model Context Protocol) 서버를 만들. 동료와 컴포넌트를 나눠 만들어도 같은 모양이 되도록 컴포넌트 유형(입력, 탐색, 선택)마다 Figma 토큰 적용, 테스트, Storybook 문서를 같은 규격으로 만드는 스킬을 만들어 함께 씀.
- **다른 팀 도입:** 2026년 10월 기준 사내 저장소 12곳이 공통 UI 패키지를 쓰고 그중 운영 중인 서비스는 워크플러스 관리자 웹을 포함해 3곳. 다른 팀 개발자가 제 도움 없이 CLI와 패키지 문서만 보고 기존 관리자 웹을 이 디자인 시스템으로 다시 만든 사례도 있음.

릴리스 파이프라인

- **배경:** 패키지 배포 파이프라인이 동시에 돌면 버전 태그 커밋을 서로 밀어 넣다 충돌했고 검증하지 않은 코드가 배포될 수도 있었음.
- **개선:**
 - 실패를 재시도로 덮지 않고 검증을 통과한 커밋(SHA)만 빌드하고 배포하게 바꿈.
 - 버전 태그 커밋이 충돌하면 리베이스한 뒤 다시 올리고 배포에서 빠진 패키지는 다음 파이프라인이 찾아 배포하게 함.
 - 높은 권한으로 PR 코드를 실행하던 `pull_request_target` 트리거를 없애 PR 코드가 배포용 비밀값에 닿을 수 있는 경로를 막음.



검증한 커밋만 배포하고 실패하면 멈추는 릴리스 경로

3. 아임히어워크 (B2B 근태 관리 서비스) 운영 및 현대화

- **기간:** 2022.10 ~ 2026.02 (3년 5개월)
- **역할:** 모바일 앱(RN) 및 관리자 웹 운영 엔지니어 (2~3인)
- **기술 스택:** React Native, Redux Saga, CodePush, Fastlane, React, TypeScript, Vite

GPS 출퇴근 인증 실패 해결

- **문제:** “사무실인데 집 위치로 찍힌다”, “인증을 믿을 수 없다”는 출퇴근 실패 문의가 계속 들어옴.
- **원인 분석:** 사용자 로그와 출퇴근 패턴으로 실패를 재현해 원인을 셋으로 나눔.
 1. 기기에 남아 있던 직전 위치 좌표가 그대로 반환됨
 2. 사용자가 OS 위치 권한을 거부함
 3. 건물이 밀집한 도심에서 GPS 측위 오차가 커짐
- **해결:**
 - **측위 방식 변경:** 버튼을 누를 때 한 번 좌표를 받던 방식을 지도 앱처럼 위치를 계속 받는 방식으로 바꾸고 Android에서는 GPS에 기지국(Cell)과 Wi-Fi 측위를 합친 Fused Location Provider를 씀. 모의 위치 조작도 막음.
 - **원인별 복구 화면:** 모든 실패를 오류 화면 하나로 보여 주던 것을 권한 문제는 OS 설정 화면으로 이동하게 하고 측위 오차는 바로 다시 시도하게 나눔.
- **결과:** 2024.01 개선 배포부터 2026.02 인계까지, **제가 직접 응대한 고객 문의 중 GPS 관련 문의 0건.**

레거시 관리자 CMS를 배포를 멈추지 않고 점진적으로 현대화

- **배경:** 인수한 관리자 웹은 React 16.6과 클래스형 컴포넌트로 되어 있어 화면 하나를 고칠 때도 라이브러리 버전부터 올려야 했고 화면 상태까지 담은 MobX 전역 스토어 탓에 상태가 초기화되지 않는 오류가 반복됐음. Webpack 빌드도 느렸음.
- **전환 순서:**
 - 실행 동작에 영향이 없는 빌드 도구부터 Vite로 옮겨 CI 빌드를 1분 30초에서 30초로 66% 줄임.
 - 전체를 다시 쓰지 않고 새 화면과 고치는 화면부터 TypeScript와 Recoil을 적용함. 기존 MobX 스토어는 남겨 두고 한동안 두 상태 관리를 함께 씀.
 - 타입을 붙이며 화면 사이에 암묵적으로 오가던 데이터를 인터페이스로 적어 둬.

장애 모니터링 인프라 구축

- 회사에 오류 추적 도구가 없어 장애를 늦게 알았기 때문에 온프레미스 Docker로 Sentry 인스턴스를 직접 구축함. 사용자 신고 전에 크래시와 API 오류를 알고 스택으로 원인을 찾게 되자, 사내 다른 프로젝트도 같은 인스턴스를 쓰게 됨.

4. 워크플러스 (신규 B2B 근태 솔루션)

- 기간: 2025.03 ~ 현재 (진행 중)
- 역할: 본인 포함 3인 개발팀의 개발 리더 (모바일 앱/백엔드 기능 설계, 관리자 웹 구현)
- 기술 스택: React 19, TypeScript, Vite, React Native, Kotlin/Spring Boot
- 출퇴근 서명 검증 테스트: 단위 테스트가 모두 통과해도 실제 출퇴근이 막히는 문제가 있어서 실제 DB를 쓰는 테스트를 에이전트와 함께 만듦. 앱이 서명할 수 없는 값이 서명 대상에 들어 있던 결함과 서버가 기록한 시각과 앱이 서명한 시각이 어긋나던 결함까지 2건을 찾아 고치고 이 테스트를 CI에 넣음.
- 인증과 개인정보 저장 방식 무중단 전환:
 - 관리자 웹 토큰을 localStorage에서 스크립트가 읽을 수 없는 HttpOnly 쿠키로 옮기고 탭 사이 동기화에도 토큰 원문을 신지 않음. 오래 열어 둔 관리자 화면은 쿠키로 세션을 되살림.
 - 한국인터넷진흥원(KISA) 권고 알고리즘 기준에 맞추려고 비밀번호 해시를 bcrypt에서 새 해시 방식(PBKDF2)으로 바꿈. 사용자가 로그인할 때 새 방식으로 다시 저장해 서비스를 멈추지 않고 옮기고 bcrypt로 새 비밀번호를 만드는 코드는 CI에서 막음.
 - DB의 개인정보 컬럼도 평문에서 암호문으로 단계를 나눠 무중단으로 바꾸고 되돌리기 스크립트를 함께 만듦.
- 탈퇴자와 계약 종료 회사의 개인정보 파기: 탈퇴자 정보는 바로 지울 것과 2년 보존할 것으로 나누고 계약이 끝난 회사는 가명으로 보존한 뒤 최종 소거하는 흐름을 판정 기준 하나로 묶어 서버와 관리자 화면에 만듦. 2026년 10월 릴리스에 반영함.
- 멀티 채널 빌드 격리: 한 코드베이스에서 고객용 번들과 사내 관리자용 번들을 함께 만드는 구조라 CI에서 빌드 산출물을 검사하는 스크립트를 넣어 고객용 번들에 사내 관리 코드가 섞이면 빌드가 멈추게 함.
- 공간이미지 인증의 서버 쪽 정책: 사진으로 위치를 인증하는 기능에서 회사 수집 정책과 직원 동의를 서버에서 둘 다 확인해야 사진을 수집하고 하나라도 아니면 출근은 막지 않고 수집만 건너뛰게 함.
- 얼굴 인식 모듈 이전과 실기기 문제 해결: 레거시 얼굴 인식 네이티브 모듈을 React Native 새 아키텍처로 옮김. 특정 기기에서 카메라가 멈추는 문제는 실기기에서 재현해 시스템 로그를 모아 분석해 GPU 가속 호환성 문제까지 좁힘. 상위 오픈소스 라이브러리에 이슈를 올리고 GPU가 실패하면 CPU로 처리하는 대체 경로를 넣음.

5. PX (Platform Experience) 팀 리딩

- 기간: 2026.07 ~ 현재 (진행 중)
- 역할: PX 팀 리더 (6인 팀)
- 주요 내용: 출결 관리(아임히어, 체크히어)와 근태 관리(아임히어워크, 워크플러스) 서비스 4개를 팀원과 나눠 유지보수하고 업무 우선순위, 기술 결정, 코드 리뷰를 맡음

매주 서비스 리뷰와 종료 서비스 이관

- 배경: 출결과 근태 모두 기존 서비스와 개선 버전이 함께 운영되고 있고 기존 서비스인 아임히어와 아임히어워크는 2026년 말 종료 예정.
- 실행:
 - 매주 서비스별 CS, 유입, 계약 현황, 다음 계획을 리뷰하고 CS가 몰린 기능은 다음 작업 우선순위로 올림.
 - 예: 체크히어에서 위치 인증 출석 문의가 물리자 다음 작업으로 올림. 안드로이드 앱이 위치 권한만 보고 기기의 위치 서비스가 꺼진 것은 확인하지 않아 예전 Wi-Fi 스캔 결과로 출석이 처리될 수 있다는 원인을 짚고 위치 서비스까지 확인하고 강제 업데이트로 반영하는 방향을 정함. iOS는 위치 인증이 실패하면 접속한 Wi-Fi로 임시 출석을 남기고 관리자가 승인하는 방식을 정하고 서버와 관리자 화면을 팀원에게 나눠 맡겨 반영을 준비하고 있음.
 - 종료 서비스의 고객사를 개선 버전으로 옮기는 이관 작업의 전체 흐름을 정리함.
 - 보안 기준 결정: 계약이 끝난 고객사의 개인정보는 가명처리하되 법적 증빙은 그대로 남기기로 정하고 이 기준을 워크플러스의 파기 흐름으로 만들어 10월 릴리스에 넣음(위 4장 워크플러스).

워크플러스: 보험사 실증 사업과 인증시험 일정 분리

- 배경: 2026년 8월, 한국정보통신기술협회(TTA) 인증시험을 준비하던 중에 개발 마감일이 가까운 대형 보험사 실증 사업이 갑자기 들어와 두 일이 겹침.

• **판단과 조율:**

- 마감이 가깝다고 고객사 전용 분기 코드를 넣지 않고 고객사 차이를 테넌트 설정으로 나누는 공통 기반을 먼저 만들기로 함. 다음 고객사도 설정만 더해 붙일 수 있게 하기 위해서임.
- TTA 준비는 PX 팀원 세 명에게 나눠 맡기되 할 일과 목표를 정해 주고 중간에 검토하며 피드백함. 본인은 공통 기반과 고객사 화면을 직접 만듦.

• **결과:**

- 공통 기반과 보험사 고객사 기능 10건을 마감보다 11일 앞선 8월 30일에 배포함.
- 기본 설정을 쓰는 기존 고객사의 동작을 회귀 확인하고 다음 고객사별 개발을 위한 가이드를 남김.

6. 사내 GPU 서버 LLM 시험 환경 구축

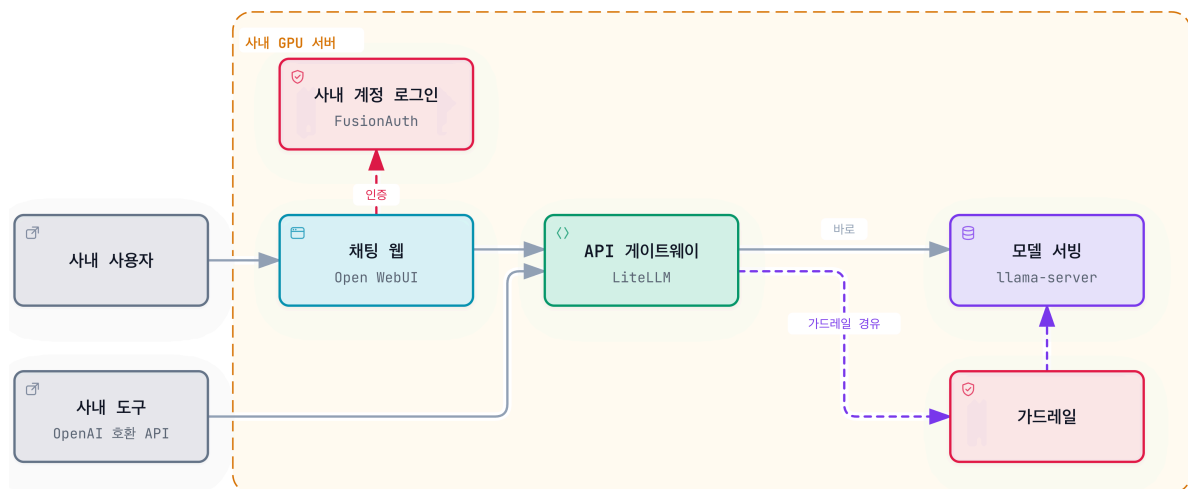
- **기간:** 2026.09 ~ 2026.10 (2개월)
- **역할:** 1인 (구성 설계, 구축, 운영 문서화)
- **기술 스택:** llama-server, LiteLLM, Open WebUI, FusionAuth, Docker Compose, Nginx

배경 (Context)

사내 GPU 서버에서 단일 컨테이너로 돌던 LLM PoC에는 사내 계정으로 로그인해 쓸 경로와 권한 구분이 없었습니다. 회사가 연구하는 가드레일(Guardrails) 모델을 실제 채팅에 붙여 시험해 볼 환경도 없었습니다.

구성과 판단 (Technical Decision)

- **층 나누기:** 단일 컨테이너에 인증만 덧붙이지 않고 화면(Open WebUI), 게이트웨이(LiteLLM), 모델 서빙(llama-server), 가드레일을 따로 나눔. 문제가 생긴 층만 따로 고치고 바꿀 수 있게 하기 위해서임.
- **GPU 한 장의 한계:** 생성 모델 1종과 가드레일 모델 2종을 GPU 한 장에 함께 올리려고 100,000 토큰까지 시험한 문맥 길이를 65,536 토큰으로 되돌리고 GPU가 한 번에 요청 하나만 처리하게 해 메모리 부족(OOM)을 막음. 동시 처리량이 줄어드는 것은 감수함.
- **사내 계정 로그인과 경로 분리:** 사내 SSO(FusionAuth)로 로그인하게 하고 승인된 사용자는 모델을 바로 호출하고 일반 사용자는 가드레일 검사를 거치도록 경로를 나눔.
- **구현과 테스트:** Docker Compose 구성, 연동 코드, 권한별 차단 테스트는 AI 코딩 에이전트가 만들고 돌리게 함. 이 테스트로는 보이지 않던 아래 결함은 직접 써 보다 찾음.



사내 계정 인증 뒤 게이트웨이에서 모델 서빙과 가드레일로 나뉘는 구조

트러블슈팅 (Troubleshooting)

- **문제:** 가드레일에 한 번 걸리면 그 채팅에서는 이후 질문까지 계속 막혀 대화를 이어 갈 수 없었음.
- **원인:** Open WebUI가 거절된 질문을 대화 기록에 남겨 다음 요청마다 같은 질문이 함께 가드레일 검사로 넘어감.
- **해결:** 거절된 질문은 대화 기록에서 빼고 새로 입력한 메시지만 가드레일 검사를 거치게 해, 차단 뒤에도 같은 채팅을 이어 갈 수 있게 함.

결과 (Impact)

- 아직 전사에 연 서비스가 아닌 시험 환경이며 지금은 가드레일 관련 사업을 검토하는 데 씀. 운영 문서와 인수 문서를 남김.

- 지금 구성으로 바꾸기 전에는 같은 GPU 서버에 판정 전용 오픈소스 모델(Kev)을 올려 고객 문의의 서비스, 유형, 긴급도를 예측해 봄. 원문 전송을 승인한 31건 중 응답 오류 1건을 뺀 30건에서 셋 모두 맞은 것이 질문 방식에 따라 1~4건이라 자동 확정은 하지 않고 사람이 확정할 때 참고하는 힌트로만 둬.

7. 기타 주요 프로젝트

대형 통신사 자회사 임직원 쇼핑물

- 기간: 2024.02 ~ 2024.08 (7개월) | 쇼핑물 프론트엔드 단독 (1인)과 일부 백엔드 API 개발
- 기술 스택: React, TypeScript, Vite, BroadcastChannel
- JSP 쇼핑물을 React SPA로 재구축: 회사 백엔드 팀이 관리자 시스템과 새 백엔드를 만드는 동안 쇼핑물 사용자 화면(PC와 모바일)을 React SPA로 혼자 다시 만들고 새 API가 나오는 대로 붙임. 필요한 백엔드 API 일부는 직접 만듦.
- 문서에 없던 예외 조건 찾기: 소스 실행과 문서 확인이 어려워 운영 화면 동작을 거꾸로 따라가며 포인트 적립 약관에 동의하지 않았을 때만 뜨는 팝업 같은 예외 조건을 찾아 구현함.
- 휴대폰 개통 신청서 연동: 휴대폰 상품은 주문을 만든 뒤 외부 업체의 개통 신청서(TGate)까지 써야 끝나고 신청서는 iframe으로만 넣을 수 있었음. 작성이 끝나면 신청서가 우리 서버를 부르고 서버가 결과와 다음 동작 스크립트를 담은 HTML을 돌려주게 해, 그 페이지가 원래 주문 화면에 결과를 넘겨 완료 화면과 주문 목록이 바뀌게 함.
- 로그인 창 사이 통신: 통신사 통합 계정(T아이디) 로그인 창이 통신사 인증 페이지를 거쳐 돌아오면 `window.opener`가 비어 원래 창에 결과를 보내지 못해 BroadcastChannel로 창 사이에 결과를 주고받게 바꿈.

사내 인프라 (AWS, Proxmox)

- 기간: 2023 ~ 현재
- 기술 스택: AWS (ECS, IAM, RDS), Terraform, Proxmox, Jenkins, Docker
- Azure에서 AWS로 이전: 2023년 위치 플랫폼과 지오로케이션 API를 Azure에서 AWS ECS로 옮기는 일을 맡아 컨테이너 저장소, 로그, 로드밸런서, 상용 RDS와 Redis를 Terraform으로 만듦. 이후 교육 서비스, 설문 서버, 워크플러스 인프라도 Terraform으로 더함.
- IAM 권한 구조 (2026.08 ~ 2026.09): 기능별 IAM 사용자 키를 여러 서비스가 함께 쓰던 방식을 서비스별 권한으로 바꾸고 운영 서버는 정적 키 대신 ECS Task Role을 쓰게 함. 워크플러스부터 바꾸고 아임히어워크, 체크히어, 렉처온으로 넓힘. 개발자 권한은 환경별 Viewer, Operator, Admin 역할과 MFA를 거친 역할 전환으로 나눔.
- 사무실 Proxmox 서버: Proxmox 호스트를 직접 설치하고 네트워크와 스토리지를 구성해 서비스별 개발 서버(VM과 LXC 컨테이너), Jenkins 분산 빌드, Sentry와 파일 공유 같은 사내 도구를 운영함. 2023년에는 한 서비스의 상용 서버를 Azure에서 Proxmox VM으로 옮김.

기타 사내 도구 및 플랫폼 기여

- 사내 대용량 파일 공유 서비스 (2024 / 1인): 보안 규정상 외부 드라이브를 쓸 수 없어 온프레미스 파일 공유 서비스를 만듦. 최대 2GB 파일을 나눠 올리는 스트리밍 업로드를 지원하고 사내에서 운영 중.
- 사내 익명 소통 봇 (2025.03 / 1인): 서버리스로 배포하는 Slack 익명 소통 봇을 만들어 사내에서 운영함.
- 수도권 도시철도 운영기관 테그리스 웹 (2024.09 ~ 2024.12 / FE 리드): 따로 배포되는 웹 여러 개가 함께 쓰는 UI와 API 데이터 타입을 pnpm workspace 모노레포 패키지로 묶어 같은 코드를 여러 번 쓰지 않게 함.

오픈소스 기여 및 대외 활동 (Open Source)

쓰는 도구에서 문제를 만나면 우회하기보다 원인을 찾아 원래 저장소에 고친 코드를 보냅니다.

- [Mantine](#): React UI 라이브러리
 - Checkbox와 Chip 컴포넌트 결함 수정 PR 4건 병합 (#8363, #8385, #8460, #8630)
- [Gitea](#): 경량 Git 호스팅 플랫폼
 - 한글 채널명을 거부하던 Slack Webhook 채널명 검사를 없앴 PR 1건 병합 (#38608)
- [Gajae Code](#): AI 코딩 에이전트 도구
 - 실무에서 쓰다 찾은 도구 격리(Sandbox Isolation)와 MCP 세션 비정상 종료 등 결함 수정 PR 13건 병합
- [react-native-iphone-screen-helper](#): React Native npm 라이브러리
 - 노치와 다이내믹 아일랜드에 대응하는 라이브러리를 포크해 2022년부터 혼자 유지보수
 - 주간 다운로드 5만 건 안팎 (2026-09-21~27에 53,958건, 2026-09-28~10-04에 63,182건)

- [react-native-simple-heic2jpg](#): React Native npm 라이브러리
 - iOS와 Android에서 HEIC 이미지를 JPEG로 바꾸고 EXIF를 남길지 지울지 고르게 한 라이브러리를 2023년에 만들어 혼자 유지보수함
 - 2026-09-28~10-04 한 주에 4,030건 다운로드

개인 프로젝트

- [Chostty](#): macOS 터미널 앱 (Zig)
 - Ghostty를 포크해 창 안에 워크스페이스와 탭 구조를 더하고 코드 서명, 공증, 자동 업데이트(Sparkle)를 갖춰 2026년 10월 v1.0.0을 배포함
 - [BarNook](#): macOS 메뉴 막대 관리 앱 (Swift)
 - Apache-2.0 프로젝트를 바탕으로 숨겨진 메뉴 막대 항목을 고정하고 배치하는 앱을 만들고 원본 고지를 지키며 서명과 공증을 거친 릴리스 9개를 배포함
 - [Kabin](#): 오프라인 칸반 데스크톱 앱 (Tauri 2, React 19, SQLite)
 - 인터넷 없이 쓰는 칸반 보드를 만들어 v1.0.0을 배포함
 - [excalidash-mcp](#): MCP 서버 (TypeScript)
 - 셀프호스팅 Excalidash 드로잉을 AI 코딩 에이전트가 조회하고 만들고 고칠 수 있게 함
-

학력 및 병역

- 가천대학교 소프트웨어학과 학사 졸업 (2016.03 ~ 2022.02)
- 대한민국 육군 병장 만기 전역 (2018.02 ~ 2019.10)