


김명재 이력서

■ 기본사항

	성명	(한글) 김명재 (남) (영문) KIM MYEONG JAE
	생년월일	1994.07.30
	연락처	010-7212-0959
	이메일	mkmj730@gmail.com
	주소	서울특별시 양천구 신정동 876-3

■ 학력 및 경력사항 (개발 총 경력 : 6년)

학 력 사 항	학 교 명	성적/등수(기준)	전 공	입학년도	졸업년도	졸업여부
	호계고등학교	전교 2등/250명	인문계(문과)	2010	2013	졸업
	충남대학교	학점 3.26/4.5	자유전공학부(행정학사)	2013	2017	졸업
경 력 사 항	근무처	직위/직책	담당업무	입사년월	퇴사년월	퇴직사유
	(주)알케미랩	선임연구원/팀장	프론트엔드 개발리더	2021.04	재직중	-
	(주)강인함	이사/실장	신규사업총괄	2020.02	2021.04	커리어전환
	수도기계화보병사단	중위/소대장	병력관리	2017.03	2019.07	전역

■ 자격 및 특기사항

날짜	자격증	점수/급수	발급기관
2023.07	SQL개발자(SQLD자격)	최종합격	한국데이터베이스진흥센터
2024.06	투자자산운용사	최종합격	금융투자협회
2025.08	정보처리기사	필기합격	한국산업인력공단

■ 기술 핵심 역량

분야	주요역량	상세 내용
Frontend	Next.js, React.js JavaScript, TypeScript,	Next.js App Router 기반 SPA/SSR/ISR 아키텍처 구축 가능 접근성 및 퍼포먼스 고려한 컴포넌트 아키텍처 구축
Full-Stack	Django + React	API 설계부터 UI 구현까지 단독 서비스 구축 가능
상태 관리	Zustand, React Query	데이터 특성(서버/클라이언트)에 맞는 구조적 상태관리 가능
UI/UX 구현	Tailwind CSS, 반응형 디자인	디자인 시스템 적용, 사용자 흐름 기반 인터랙션 설계 가능
실시간성	WebSocket	금융 데이터 기반 실시간 업데이트 및 렌더링 최적화
Hybrid App	Capacitor, PWA	웹 → Android/iOS 앱 전환, 앱스토어 배포 가능
배포/운영	CI/CD(AWS Amplify, GitLab)	자동화된 빌드/배포 환경 구성 및 운영
Backend	Python, DRF	REST API 설계 및 인증/인가(JWT, OAuth2.0) 구현 Full Stack Django 운영 가능
DB	SQLite, PostgreSQL (초급)	DB 모델링 기본, ORM 기반 데이터 처리 경험
Infra	Docker, AWS EC2, Lambda	컨테이너 기반 배포, 서버 운영 및 무중단 배포 경험
Design	Figma(중급)	와이어프레임, UI 컴포넌트 제작, 프로토타이핑, 디자이너 협업 기반 화면 구현

■ 경력세부사항

근무처	기간 (6년)	세부 내용
(주)알케미랩	2024.09~2025.09(1년)	<p>1. 하이브리드 웹/앱 졸보(zolbo)개발 “실시간 금융서비스 → 콘텐츠 기반 구독서비스로의 전환을 기술적으로 주도”</p> <p>[프로젝트 개요 및 목적]</p> <ul style="list-style-type: none"> - 초보 투자자의 불안 요인을 해결하고, 사용자 신뢰를 기반으로 성장하는 서비스 구조를 만들기 위한 프로젝트, (서비스의 신뢰성을 높여 초보 투자자의 진입 장벽을 낮추기 위함) - 웹과 앱(PWA·Android·iOS)을 하나의 코드 베이스로 운영해 App 유입 장벽을 낮추고 유지보수 효율성과 일관된 UX 확보 - FE 총괄 리드로, Next.js + Capacitor 기반의 하이브리드 앱(PWA·Android·iOS)을 설계 및 구현 <p>[팀 구성 및 역할]</p> <ul style="list-style-type: none"> - 프론트엔드 1명, 백엔드 1명, 풀스택 1명, 디자이너 1명, 기획자 1명 - 본인의 역할: 프로젝트 PM(과업 우선순위 설정 및 분배) 이자 FE Lead로 전반적인 구조 설계, 핵심 화면 개발 및 기능 구현, UI/UX 디자인 피드백 담당 <p>[담당업무]</p> <ul style="list-style-type: none"> - 아키텍처 설계: Next.js(App Router) 기반 SSR/CSR 통합 구조 설계 - `apps/web` 분리 구조로 PWA·AOS·iOS 빌드 통합 지원 - 상태 관리: Zustand 기반 로그인·토큰 관리 및 persist 스토리지 설계 - 데이터 관리: React Query 기반 데이터 캐싱 및 병렬 처리, WebSocket 기반 실시간 자산·주문 데이터 스트림 구현 - 빌드·배포: AWS Amplify + GitLab CI/CD 자동 빌드·배포 파이프라인 구축 - 인증 구조: Google OAuth + 휴대폰 인증 3단계 플로우통합 - 인앱 결제: Google Play 결제 연동 후 ZLB 포인트 자동 지급 로직 구현 <p>[문제 해결 과정]</p> <ul style="list-style-type: none"> - 기존에는 flutter 를 사용한 이중 개발로 App 개발이 진전이 보이지 않았음 → Next.js + Capcitor 을 활용한 단일 코드 기반으로 해결 - 초기에는 로그인/인증 과정이 복잡해 이탈률과 진입장벽이 높았음 → BottomSheet 기반 단계별 인증 UI로 개선 - 실시간 자산 데이터가 페이지 새로고침 시 끊김 →

		<p>WebSocket 구독 구조를 통합해 실시간 갱신으로 구현</p> <ul style="list-style-type: none"> - 배포 시간이 길고 환경이 불안정 → Amplify + GitLab CI/CD로 환경 자동화 - 콘텐츠 소비 동기 부족으로 스스로 학습할 수 있는 "퀴즈"서비스 런칭과 리포트 기능 구축 <p>[성과]</p> <ul style="list-style-type: none"> - 하이브리드 단일 코드베이스 확립으로 App 런칭 성공 및 코드 유지보수 효율 50% 개선 - 실시간 운용 ON/OFF 시스템 도입으로 월간 유지율 25 → 40% 향상 - WebSocket 기반 실시간 자산 갱신으로 새로고침 빈도 70% 감소 - Amplify + CI/CD 자동배포로 배포시간 80% 단축 - 코드 정규화 및 Cursor.ai 도입으로 신규 기능 도입 속도 2배 향 - 실시간 로그 분석 기반 투자전환 데이터 정확도 향상
	2023.10~2024.06(8개월)	<p>2. 가상자산 거래소 WTS(트레이딩 대시보드) 개발 "FE Lead / UI·UX 기획 & 퍼블리싱까지 End-to-End 수행"</p> <p>[요약]</p> <ul style="list-style-type: none"> - OKX 기반 거래소에서 Bybit로 전환하며 실시간 대시보드 및 주문 기능을 개발 - 스택 : Django + React.js + Typescript + Zustand - TradingView 기반 실시간 트레이딩 기능 개발 - 디자이너 없이 PC/Mobile UI/UX 전면 기획 & 구현 <p>[담당업무]</p> <p>1) 핵심 기능 개발</p> <ul style="list-style-type: none"> - WebSocket 기반 실시간 데이터 스트림 및 차트 설계 - 주문·체결·포지션 업데이트 UI 및 비동기 API 연동 - Django 백엔드와 실시간 데이터 동기화 파이프라인 구성 - OKX → Bybit API 구조 차이에 따른 FE 마이그레이션 수행 - 데이터 업데이트 주기 분리로 렌더링 부하 최소화 <p>2) UI/UX 기획 + 인터랙션 디자인 + 퍼블리싱 Direct 수행</p> <ul style="list-style-type: none"> - Figma 목업 디자인 및 트레이딩 대시보드 UI/UX 구현 - TradingView 차트 커스터마이징 및 포지션별 손익 UI 구현 - 고객 핵심UX에 맞춘 직관적 UI구현과 즉각적 피드백 반영 <p>3) 팀 목표 기반 우선순위 리딩 및 FE 리드</p> <ul style="list-style-type: none"> - 2주 단위 업데이트 원칙을 세워 개발 과업 리스트업과 핵심 기능의 우선순위 도출 - 회사의 핵심 수익 서비스의 완전 이관과 안전화를 주도 - 신규 개발 기능은 과감히 후순위로 지정, 핵심 거래 기능 마이그레이션에 집중

		[성과] <ul style="list-style-type: none"> - OKX/Bybit API 연동 구조 확립, 실시간 거래 안정화 - 마이그레이션 소요시간 2개월 → 3주로 단축 - 실시간 차트 렌더링 지연률 40% 감소 - 주문 응답 시간 600ms→280ms로 단축, 거래 안정성 확보 - 포지션/성과 탭 도입으로 이탈률 50% 감소
	2023.02~2023.07(6개월)	3. 가상자산 펀드마켓 솔루션 대시보드 개발 <p>[요약]</p> <ul style="list-style-type: none"> - 누구나 거래소 포트폴리오 기반 펀드 운용 및 투자자 모집이 가능한 구조 설계 - 퍼스널 펀드마켓 화이트라벨 솔루션 '화라폴' 총괄 개발 <p>[담당업무]</p> <ul style="list-style-type: none"> - Django(SSR, DB) 기반 멀티테넌트 구조 설계 (운용사별 계정/보수/계약 관리) - 기본보수·성과보수·회원관리 자동화 - 브랜드 테마 커스터마이징(로고·컬러·모드) 기능 구현 - 퍼블렛·지브릭 등 파트너사 API 연동 및 표준화 - 법적 정산 및 보수 프로세스 자동화 설계 - 운용사 온보딩 대시보드 및 관리자 시스템 구축 <p>[성과]</p> <ul style="list-style-type: none"> - MC9·Inus·Lia Holdings 등 파트너사 계약 체결 및 상용화 - 점조직형 펀드 모델 정산 자동화로 운영 리소스 30% 절감 - 브랜드 커스터마이징 기능으로 온보딩 속도 2배 향상 - 화이트라벨의 본질은 기술보다 프로세스와 신뢰 설계에 있다는 인사이트 확보
	2022.11~2022.11(1개월)	4. 자사 홈페이지 리뉴얼 및 브랜드 아이덴티티 기획 <p>[요약]</p> <ul style="list-style-type: none"> - 브랜드 리뉴얼 총괄: 미션·비전·전략·가치 재정의 및 웹 반영 - AWS EC2 + Docker 기반 배포 환경 구성 <p>[담당업무]</p> <ul style="list-style-type: none"> - 25개 기업 벤치마킹을 통한 알케미랩 핵심가치 구조화 - 내부 워크숍·인터뷰 통해 '핀테크 자본주의를 만드는 사람' 키워드 도출 - 브랜드 언어·UI 가이드·스토리 섹션 통합 설계 - 디자인팀·개발팀 간 협업으로 메시지 일관성 확보 <p>[성과]</p> <ul style="list-style-type: none"> - 구성원 공감도 90% 이상, 페이지 조회수 2.3배 상승 - 외부 투자 피칭용 홈페이지 활용

		- "누구나 편안하게 투자할 수 있는 금융생태계"라는 조직 정체성 명문화
	2022.05~2022.10(6개월)	5. 가상자산 자문형 서비스 화면 개발 [요약] <ul style="list-style-type: none"> - Binance API 기반 실시간 시세·차트 연동 - TradingView 도입으로 신뢰성 높은 실시간 데이터 시각화 [담당업무] <ul style="list-style-type: none"> - TradingView 차트 커스터마이징 및 위젯 구조 설계 - 실시간 자문 포트폴리오 데이터 시각화 - 차트 반응형 업데이트 및 사용자 테마 구성 기능 구현 - 환경 분리 및 GitLab CI/CD 자동배포 파이프라인 구축 [성과] <ul style="list-style-type: none"> - 실시간 시각화로 사용자 리텐션 30% 증가 - 체류시간 2배(4→9분), 차트 렌더링 지연 40% 감소 - "데이터는 보여지는 순간 신뢰가 된다"는 FE 문화 정립
	2022.02~2022.04(3개월)	6. 가상자산 옵션 트레이딩 MTS 개발 [요약] <ul style="list-style-type: none"> - Deribit API 기반 실시간 포지션·주문 관리 시스템 구축 - EC2 + Docker 기반 서버 배포 자동화 [담당업무] <ul style="list-style-type: none"> - WebSocket 실시간 체결/포지션 업데이트 구현 - 주문/거래내역 데이터 시각화 및 반응형 UI 구성 - 환경 변수 관리 및 보안체계 구축 [성과] <ul style="list-style-type: none"> - 실시간 거래 데이터 처리속도 45% 향상, 주문응답시간 700ms → 300ms 단축 - 사용자 체류시간 1.8배 증가, 운영비 20% 절감 - 안정적인 실시간 거래 구조로 이후 프로젝트(WebSocket 표준화) 기반 마련
	2021.12~2021.01(2개월)	7. 자사 홈페이지 개발 <ul style="list-style-type: none"> • Django + Javascript + CSS 기반 웹사이트 개발 • AWS EC2 + Nginx 기반 풀스택 개발
	2021.08~2022.03(7개월)	8. Binance 일임형 서비스 "zolbo.ai" WEB 기획 및 개발 [요약] <ul style="list-style-type: none"> - 금융 지식 없이 3주 만에 1일결산 로직을 직접 설계 후 Django + JS로 구현 - Binance API 연동, 자산 결산, 포인트 차감 로직 개발 [담당업무]

		<ul style="list-style-type: none"> - Chart.js로 운용성과 시각화 및 백엔드 데이터 연동 - 일일 운용성과 기준의 포인트 결산 프로세스 구축 <p>[성과]</p> <ul style="list-style-type: none"> - 데이터 시각화의 가치 체득, 금융 로직 직접 설계 경험 - '도메인 이해'의 중요성을 깨달으며 서비스 사고력 확장
(주)강인함	2020.02~2021.04(1년 3개월)	<p>9. 신사업 홈페이지 개발 및 CRM 프로세스 구축</p> <p>[프로젝트 개요 및 목적]</p> <ul style="list-style-type: none"> - 교육입시컨설팅 브랜드 "강인함"의 디지털(온라인) 전환 프로젝트로, 기존 업계의 오프라인 중심의 상담·결제·등록 프로세스를 온라인 자동화 시스템으로 전환하는 것을 목표로 함. <p>[주요 업무]</p> <ul style="list-style-type: none"> - WordPress 기반 홈페이지 구축 및 SEO 구조화 - Google Spreadsheet 기반 CRM 자동화 (상담 → 결제 → 등록 데이터 파이프라인 설계) - GA4 + GTM 기반 상담 전환 추적 및 리포트 자동화 - 광고 캠페인(네이버/카카오/구글) 성과 분석 및 UX 개선 - 랜딩페이지, 배너, 포스터 등 온라인 브랜드 에셋 제작 <p>[문제 해결 과정]</p> <ul style="list-style-type: none"> - CTA와 광고 타겟팅이 최적화되어 있지 않아 이탈률이 업종평균대비 2배 높음 → 홈페이지 랜딩 구성 최적화 및 키워드 셋팅 최적화(1차, 2차고객) 진행 - CRM 데이터가 산발적으로 관리되어 상담 이력 추적이 어려움 → Google Spreadsheet로 통합 관리 - 광고 효율 저하 → 전환율 기반 GA4 트래킹 도입 및 캠페인 리디자인 <p>[성과]</p> <ul style="list-style-type: none"> - 초기 이탈률 70% 감소 - 광고 전환율 2.1% → 4.3%(+104%) - 월 트래픽 3.2배 증가, 고객 응답속도 60% 단축 - CRM 자동화로 운영 리소스 50% 절감

[소개]

5년차 프론트엔드 개발자로서, 실시간 데이터 기반 핀테크 서비스에서 기술적 한계와 사용자 경험 간의 간극을 줄이는 데 집중해 왔습니다. 일례로 수수료 기반의 로보어드바이저 서비스인 사내 B2C 프로젝트 '졸보(zolbo)'를 학습 퀴즈, 리포트 등 **콘텐츠 중심의 구독형(투자안심패스) 모델**로 확장하는 과정을 주도하며 사용자가 지속적으로 서비스를 찾을 수 있는 콘텐츠 소비 경험(학습형 퀴즈, 리포트, 인사이트)을 직접 설계하고 구현했습니다.

초기 종전 사내의 모든 프로젝트들은 Django 기반 SSR/WebView 구조였기 때문에 성능과 유지보수의 제약이 컸습니다. 이 불편을 사용자 '신뢰' 문제로 바라보고 실시간성과 UX가 중요한 로보어드바이저 서비스를 React 기반 컴포넌트 아키텍처로의 전환하는 것을 주도했습니다. 또한 웹 중심 운영의 한계를 해결하기 위해 Next.js와 Capacitor 기반의 **하이브리드 앱 출시**까지 이끌며 단일 코드베이스로 웹.앱을 운영할 수 있는 체계를 확립했습니다. 사내에 관련 경험이 없던 환경에서도 Android/iOS 빌드 및 앱스토어 배포 전 과정을 스스로 구축한 경험으로 새로운 기술을 빠르게 흡수하고 실서비스에 적용하는 실행력을 길러 왔습니다. 저는 개발로 서비스의 비즈니스 모델과 사용자 경험이 함께 성장할 수 있도록 문제를 정의하고, 구조를 개선하며, **팀의 속도를 높이는 개발자**라고 생각합니다.

더욱 속도를 높이기 위해 AI 도구를 활용해 **문서 자동화, 로그 분석, 개발 품질 관리**의 효율을 높여온 경험도 있습니다. AI가 대세가 되기 전부터 개발 방식의 근본적 문제(반복되는 코드, 노가다성 개발)를 해결하기 위해 Copilot, GPT, Gemini, Cursor 등을 먼저 사용하며 사내에 도입을 주도하고 개발팀에 생산성의 확산을 만들었습니다. 단순히 개인 생산성을 높이는 수준이 아니라 코드 품질 표준화, 지식 격차 해소, 리팩터링 자동화, 그리고 빠른 실험과 검증이 가능한 프로토타이핑 모드를 팀 문화에 정착시키는 데 집중했습니다. AI를 실서비스에 성공적으로 적용하며 프로젝트를 더 빠르고, 더 정확하게, 더 멀리 나아가게 하는 역할을 해왔다고 생각합니다.

[일의 원칙]

저는 아래와 같은 3단계의 방법으로 일을 추진합니다.

1. 킥오프

모든 프로젝트는 "이 일을 왜 하는가, 누구를 위해 하는가"라는 질문에서 시작합니다. 개발을 하기 전에 목적을 먼저 정의하고, 그 목적이 팀과 고객의 방향성에 맞는지부터 확인합니다. 이후 변하지 않는 것과 변하는 것, 할 수 있는 것과 할 수 없는 것에 대해 정의합니다.

2. 추진 방식

프로젝트 내부에 흔들리지 않는 유스케이스와 가설을 세우고, 관련 서비스와 주변 레퍼런스를 빠르게 조사합니다. 핵심 **UI/UX를 프로토타입**으로 만들어 실제로 '손에 잡히는 형태'로 검증하고, 이 과정을 바탕으로 구체적인 기획과 개발 전략을 수립합니다. 이후 디자이너, 기획자, 개발팀과 함께 방향을 다듬으며 완성도를 높여갑니다.

3. 개발

현재 가장 자신이 있는 Next.js, TypeScript, Zustand, Tailwind, Capacitor 기반의 하이브리드 앱(PWA/Android/iOS) 개발 경험을 바탕으로, 사용자 중심의 인터랙션과 안정적인 배포 환경을 구현해왔습니다. 서비스 기획과 기술 구현을 모두 경험했기에, 기술적 의사결정과 사용자 경험 개선을 균형 있게 이끌 수 있습니다.

실 패 극 복 사 례	<p>[실패는 배움의 시작이었다]</p> <p>초기 zolbo.ai(Binance 일임투자 서비스) 개발 당시, 펀드 결산 로직의 오류로 인해 결산 결과가 일치하지 않는 문제가 반복되었습니다. 결산일 기준가와 수익률 계산이 누락·역산 오류를 일으키며 고객의 수익률이 실제 펀드 성과와 달라졌습니다. 당시 저는 단순히 코드를 수정하는 대신, 문제의 본질이 '도메인을 모르고 개발했다는 것'에 있음을 깨달았습니다. 그래서 “무엇을 모르는지부터 정의하자”는 원칙 아래, 3주간 하루 다섯 가지 이상의 질문을 스스로 설정했습니다. 펀드 회계, 기준가, 좌수, 선입선출, 대차대조표 등 20여 개의 금융 개념을 논문과 도서를 통해 학습했고, 이를 토대로 입출금 시 마진 밸런스를 역산하는 스냅샷 혼합형 결산 로직 구현을 직접 개발했습니다. 결과적으로 결산 오류율을 99%에서 0%로 낮출 수 있었고, 당연하게도 성과 불일치 문제를 완전히 해결할 수 있었습니다.</p> <p>핵심 성과 요약</p> <ul style="list-style-type: none">- 결산 오류율 100% → 0%- 3주간 100회 이상 질문 & 2회 내부 공유 세션 진행- 질문 중심 학습 문화' 정착 → 온보딩 레퍼런스 문서화 <p>이 경험은 단순한 기술 습득이 아니라, 학습 구조를 스스로 설계한 과정이었습니다. 매일 대표와 사수에게 질문하고, 팀원과 학습 내용을 공유하면서 '질문은 성장의 시작'이라는 문화가 자연스럽게 팀에 자리 잡았습니다. GPT나 관련 내부 자료가 전혀 없던 시기였기에, 논문·기사·YouTube 등 다양한 채널을 통해 정보를 수집하고, 정리된 내용을 후임 개발자가 쉽게 이해할 수 있도록 내부 레퍼런스 문서로 남겼습니다. 이 프로젝트를 통해 저는 “모르는 것을 빠르게 정의하고, 학습 가능한 체계로 전환하는 능력”이 개발자의 가장 중요한 역량임을 깨달았습니다. 문제를 해결한다는 것은 단순히 오류나 버그를 잡는 일이 아니라, 이해할 수 있는 세계관을 넓히는 것임을 깨달았습니다.</p> <p>“성장은 문제를 해결할 때가 아니라, 모르는 것을 인정할 때부터 시작된다.”</p>
팀 생 산 성 개 선	<p>[팀의 집중력을 높이는 환경을 만들고]</p> <p>개발은 개인의 역량만으로 완성되지 않습니다. 저는 팀이 더 집중할 수 있는 환경을 설계하는 일에 가장 큰 보람을 느낍니다. 업무 효율을 높이는 것은 개인의 속도를 높이는 일이 아니라, 불필요한 소음을 제거해 몰입을 돕는 일이라고 믿고 있습니다.</p> <p>먼저 회의 구조를 재설계했습니다. 매일 진행하던 구두 데일리리 문서형 리포트로 전환해 불필요한 회의 시간을 60% 줄였고, 주간 회의는 전사 공지와 담당자별 스크럼 체계로 개편하여 실행률을 35% 향상시켰습니다. 회의가 시간 소비가 아니라 집중 시간 확보의 수단으로 바뀌었습니다.</p> <p>다음으로 협업 툴의 한계를 해결했습니다. Slack의 휘발성 메시지로 인해 결정 이력이 사라지는 문제가 있었기에, Swit 도입을 제안하고 팀을 설득하여 Slack → Swit 마이그레이션 전체를 주도했습니다. 그 결과 커뮤니케이션 효율이 40% 향상되었고, 업무 이력 추적률이 0%에서 95%로 개선되었습니다. 채팅 중심의 협업 문화를 기록 중심의 협업 구조로 바꾸어, 누구나 의사결정의 맥락을 쉽게 공유할 수 있게 했습니다. (Swit은 쉽게 말해 Slack + Notion)</p> <p>마지막으로 배포 프로세스를 자동화했습니다. AWS Amplify와 GitLab CI/CD를 조합하여 무중단 자동배포 파이프라인을 구축하고, 브랜치별 자동 빌드·QA 검증 체계를 만들어 배포 시간을 80% 단축, QA 처리 속도를 두 배 이상 높였습니다. 이후 팀원들은 반복적인 수동 작업에서 벗어나 문제 해결과 개선에 더 집중할 수 있는 환경을 갖추게 되었습니다.</p>

	<p>[AI 활용으로 팀 속도를 높이다]</p> <p>AI는 단순 자동화 도구가 아니라, 사람이 집중해야 할 문제에 더 많은 시간을 주는 도구라고 생각합니다. 그래서 저는 일찍부터 AI Pair Development 문화를 팀에 도입하기 위해 노력했습니다. 먼저, 반복적이고 비창의적인 작업을 AI에게 위임했습니다. GitHub Copilot과 GPT를 활용해 보일러플레이트, 문서화, 테스트 코드, 타입 정리 등을 자동화했고, PR(MR) 템플릿을 표준화하여 리뷰 기준을 일관되게 유지하도록 했습니다. 최근에는 Cursor를 도입해 레거시 코드 분석과 리팩터링 속도를 높였고, 문서 기반 자동 QA 체크와 로그 분석 자동화도 함께 구현했습니다.</p> <p>또한, 초기에 명확한 기획이 없는 상황에서 AI를 활용한 프로토타이핑 및 실서버 테스트를 진행해 핵심 UX를 의사결정자가 빠르게 체감할 수 있도록 했습니다. 이 과정은 기획-개발-검증 사이클을 획기적으로 단축시켰습니다. 그 결과 팀이 더 중요한 문제에 집중할 수 있는 환경이 마련되었습니다.</p> <ul style="list-style-type: none">- 문서화 및 관리 작업 40% 감소- 신규 기능 검증 사이클 25% 이상 단축- 불확실한 가설에 대한 의사결정 속도 개선- 레거시 코드 파악 및 품질 표준화 구축 <p>저는 앞으로도 사람이 해야 하는 일(본질 파악, 문제 정의, 구조 설계, 품질 판단)과 AI가 더 잘할 수 있는 일(자동화·참조·리팩터링)을 적극적으로 분리하여 팀 전체가 더 크게 성장할 수 있는 환경을 설계하는 개발자가 되고자 합니다.</p>
코드 분석 및 적용 경험	<p>[하나부터 열까지 스스로 성장한 여정]</p> <p>처음부터 끝까지 스스로 부딪히며 성장한 경험이 있습니다. TradingView 기반의 실시간 차트 시스템을 구현할 당시, 문서가 불충분하고 내부 버전은 이미 지원이 종료된 상태였습니다. 저는 “누구도 모른다면, 직접 해석한다”는 마음으로 라이브러리의 핵심 소스코드를 한 줄씩 분석하고, 각 함수의 호출 구조와 반환값을 `console.log`로 추적하며 동작 원리를 복원했습니다. 그 결과, WebSocket 렌더링 지연 시간을 3.2초에서 0.7초로 단축하고, 실시간 데이터 반영 속도를 개선해 사용자 체류 시간을 2배 이상 증가시켰습니다. 문서보다 코드를 먼저 믿고, 코드보다 사용자의 반응을 먼저 본 경험이었습니</p> <p>이후 OKX → Bybit 마이그레이션 프로젝트에서는 거래소 간 API 구조가 완전히 달라 단순 포팅이 불가능한 상황이었습니다. 먼저 변하는 것과 변하지 않는 것을 표로 정리하고, 할 수 있는 것과 할 수 없는 것을 구분했습니다. 이 과정에서 OKX의 세분화된 계약 단위(Contract·Coin·USDT)가 Bybit에서는 통합되어 있음을 파악할 수 있었고 이에 따라 수량 계산과 주문 체결 로직 전체를 새로 설계해야 했습니다. 저는 공식·비공식 문서를 모두 수집해 API 응답을 1:1 매핑하고, 코인/USDT 기준으로 환산하는 표준화 파서(Parser)를 직접 구현했습니다. 또한 Bybit가 포지션 정보를 WebSocket으로 제공하지 않는 한계를 극복하기 위해, REST 기반 자산 데이터와 Ticker WebSocket을 결합한 혼합형 스트림 구조를 새로 설계했습니다. 이를 통해 데이터 부하를 30% 이상 줄이고, 거래 오류율을 0.3% 이하로 안정화시켰으며, 예상 기간의 절반인 3주 만에 전체 마이그레이션을 완수했습니다.</p> <p>이 두 프로젝트는 공통적으로 “문서를 읽는 개발자”에서 “시스템을 해석하는 개발자”로 성장한 순간이었습니다. 누가 알려주지 않아도, 필요한 답을 스스로 찾고 구조를 새로 세우는 과정 속에서 저는 진짜 성장의 의미를 배웠습니</p>

<p>주 도 적 문 제 해 결</p>	<p>[문제 해결은 시스템의 한계를 이해하는 것]</p> <p>2023년 OKX 기반의 거래소 서비스 초기에, Django 기반 PWA에서 네트워크가 불안정할 때 차트·주문 모듈이 반복적으로 오류를 일으키는 문제가 있었습니다. 단순한 성능 이슈로 보이지만, 실제로는 Django의 SSR 특성과 비동기 요청 간 캐시 불일치로 인해 매 요청마다 비캐시 리소스를 새로 로드하는 구조적 한계가 원인이었습니다.</p> <p>저는 이를 “속도 문제”가 아니라 “사용자 신뢰 하락 거래 중단 매출 저하”라는 비즈니스 병목으로 정의하고, Service Worker Cache + Browser Cache + Django Cache를 병행 적용했습니다. 정적 리소스 재렌더링을 최소화하고, 네트워크가 끊겨도 JS·CSS·차트 데이터가 로컬 캐시에서 즉시 복구되도록 구조를 설계했습니다.</p> <p>그 결과</p> <ul style="list-style-type: none"> - 페이지 로딩 속도 6.8초 -> 2.1초 (69% 개선) - 차트 오류율 80% 감소 - 사용자 거래 유지율 15% 상승(이탈률 유의미적 감소) <p>이 경험을 통해 초기 데이터가 많아질 경우에 사용자 경험 측면에서의 Django SSR의 한계를 체감하고, React 기반 CSR 구조 전환의 필요성을 직접 제안했습니다. 이는 이후 1년뒤 Next.js 기반 하이브리드 앱으로 발전하는 기술적 전환점이 되었고, “문제 해결은 시스템의 한계를 이해하는 것에서 출발한다”는 제 첫번째 개발 철학으로 자리 잡았습니다.</p>
<p>기 역 에 남 는 개 발 경 험</p>	<p>[사용자의 불안을 해결해야한다]</p> <p>2025년 9월, 코엑스에서 열린 동아 재테크쇼에서 약 500명의 투자자(잠재 고객)를 대상으로 직접 리서치를 진행했습니다. 그 결과, 대부분의 사용자가 기술적 문제나 보다 ‘전문성 부족으로 인한 심리적 불안’으로 인해 가상자산 투자를 망설이고 있다는 사실을 확인했습니다.</p> <p>이 인사이트를 기반으로 비즈니스 모델을 ‘AI 자동투자’ → ‘안심패스(심리 안정 구독형+학습형)’ 으로 전환했습니다. 기존의 “기술 편의 중심 UX”를 버리고, “사용자의 불안을 해소하며 신뢰를 쌓는 경험 구조”로 리디자인했습니다. 단순히 성능이 좋은 서비스를 만드는 것이 아니라, “기술이 사람의 감정을 설계할 수 있는가”라는 질문에 처음으로 진지하게 고민해볼 수 있었습니다.</p> <p>현장에서 얻은 데이터는 서비스의 철학을 바꾸는 근거가 되었습니다. 설문에 응한 사람 중 70%가. “비트코인 지금이야 사!”라고 답하면서도 실제 행동으로 옮기지 못했고, 가장 큰 불안 요인은 가격 출렁임(35%), 전문성 부족(25.4%), 거래소 불신(18.3%) 임을 확인할 수 있었습니다. 이를 통해 zolbo가 해결해야 할 핵심은 ‘수익률’이 아니라 ‘심리적 신뢰’라는 점을 명확히 인식했습니다.</p> <p>이후 TF팀을 구성해 개발·기획·디자인이 함께 “사용자가 배우며 안심할 수 있는 서비스 구조”를 만들기 위해 협업했습니다. 내부 익명 설문 결과, 구성원 90% 이상이 새로운 방향성에 공감했고, 서비스는 기술 중심에서 사용자 중심으로 재탄생할 수 있었습니다.</p> <p>비록 회사의 경영 여건으로 서비스는 11월에 종료되지만, 이 프로젝트는 팀 전체가 “우리가 왜 이 서비스를 만드는가”라는 질문에 다시 집중하는 계기가 되었습니다. 성과는 사라질 수 있지만, 그 과정에서 얻은 사용자 중심 철학과 조직의 공감대는 남았습니다.</p> <p>“개발은 기능을 위한 도구가 아니라, 사람의 불안을 해결하는 언어다.”</p>

정 체 성	<p>[정체성]</p> <p>저는 단순히 화면을 개발하는 사람이고 싶지 않습니다. 서비스의 본질적 가치를 기술로 재해석하고 확장시키는 개발자로 기억되고 싶습니다. 기존 하이브리드 로보어드바이저 웹/앱 서비스에서 수수료 중심 BM의 한계를 해결하기 위해, 콘텐츠 중심의 구독 모델(안심패스) 전환을 추진했습니다. 그 과정에서 학습 퀴즈, 리포트, 인사이트 등 사용자가 계속해서 서비스를 찾을 수 있는 콘텐츠 소비 경험을 설계하고, UI 구조와 사용자 흐름 전반을 개선한 경험이 있습니다.</p> <p>또한, Javascript → React → Next.js → Capacitor 기반 앱 출시까지 기술 스택 확장과 아키텍처 전환을 스스로 주도하며 접근성과 성능, 운영 효율을 동시에 확보했습니다. 저에게 프론트엔드라는 영역은 단순히 화면이나 인터페이스가 아니라, 서비스의 가치가 사용자에게 가장 먼저 닿는 공간입니다. 사용자의 실제 문제를 정확히 정의하길 좋아하고, 개발로 신뢰를 높이는 서비스를 만드는 것에 즐거움을 느낍니다. 앞으로도 서비스의 본질을 탐구하고 사용자의 신뢰와 경험을 설계하며 회사의 매출에 기여하는 사람이 되고 싶습니다.</p>
-------------	--