PORTFOLIO

LLM을 실서비스 화면에 넣고 실패까지 설계해 온 프론트엔드 엔지니어입니다.

CJ올리브영 [AI-First Track] Commerce Platform Engineer (Front-End) 지원

React · Next.js · TypeScript · LLM 파이프라인 · RAG · 멀티에이전트 개발 · 커머스

hello@anveloper.dev · github.com/anveloper · anveloper.dev

INDEX

목차

PROFILE

CONTACT

010-4175-6780

hello@anveloper.dev

LINKS

github.com/anveloper

anveloper.dev

resume.anveloper.dev

안성진

화면을 데이터로 다루고, 그 데이터를 AI가 만들게 설계하는 프론트엔드 엔지니어

#AI-Native #LLM파이프라인 #RAG #멀티에이전트 #커머스 #팀리딩

Career (총 경력 3년 5개월)

주식회사 틸스 · 기술연구소 연구2팀 팀장(과장) 2023.03 ~ 재직 중

- 멀티테넌트 커머스 플랫폼(DPS·DPS Store) PM 겸 개발 리드 · 2~7명 팀 리드
- 페이지 구성을 데이터로 다루는 렌더러 설계 - 고정형 → 자유형 → **생성형(AI)** 확장
- Claude·Codex **멀티에이전트 협업 체계**를 직접 설계해 기획·개발·검수에 적용
- 15년 된 PHP 레거시 → Next.js·React 전면 전환 주도

병역 · 육군 대위(장교) 만기전역 2015.03 ~ 2021.06

AI 활용 원칙

- **정확해야 하는 계산은 코드**, 유연한 판단만 AI
- 판단은 AI가, **조회·저장·회신 흐름은 코드**가 통제
- 규칙은 코드가 아닌 **프롬프트로 분리**해 무수정 대응
- AI가 한 일은 반드시 **관측 가능하게** 기록

Skills

- **AI · LLM** - RAG(pgvector) · SSE 스트리밍 · 구조화 출력(Zod) · 프롬프트 분리 · 토큰 관측
- **주력** - TypeScript · React · Next.js · Prisma
- **활용** - NestJS · Node.js · Python · MySQL/MariaDB · PostgreSQL · AWS · Caddy
- **Tool** - Claude · Codex · Gemini · Git · Jira · Figma

Education · Certificate

- 충남대학교 천문우주과학과 졸업 - 비전공 출발
- 삼성청년SW아카데미(SSAFY) 7기 - 팀 프로젝트 **우수상 4회**
- 항해 플러스 FE 5기 **상위 1%** · BE 9기 수료
- 정보처리기사 (2023.11) · SQLD (2022.09)

CORE COMPETENCIES

핵심역량

1 LLM을 데모가 아니라 출시되는 제품 기능으로 만들었습니다

육아 서비스에서 모든 LLM 호출을 API 한 곳으로 모으고, 임베딩 검색(RAG) 기반 챗봇을 **SSE 스트리밍 UI**로 구현해 **iOS·Android 양대 스토어 출시**까지 도달했습니다. Zod 스키마로 응답 구조를 강제하고 토큰 사용량을 기록하며, 실패는 명시적으로 남기고 재실행으로 복구하는 **관측 가능한 AI 파이프라인**을 설계했습니다.

2 화면을 데이터로 다루면, 그 데이터는 AI가 만들 수 있습니다

운영 중인 멀티테넌트 플랫폼에서 페이지 구성을 코드가 아닌 DB에 저장하고 렌더러가 해석하는 구조를 설계했습니다. 테넌트 유형을 **고정형(JSON) → 자유형(노드 빌더) → 생성형(AI)** 3단계로 확장하도록 잡았고, 정형화된 UI를 넘어선 개인화 경험이 이 경로 위에 있다고 보고 있습니다.

3 AI를 결과물이 아니라 개발 과정 자체에 넣었습니다

Claude와 Codex가 동시에 작업해도 충돌하지 않도록 진입점과 파일별 작업 범위를 문서(CLAUDE.md · AGENTS.md)로 나누는 **멀티에이전트 협업 체계**를 직접 만들었습니다. 반복 작업은 스킬로 재사용하고 작업 단위는 plan 문서로 관리해, 적은 인원으로 여러 제품을 동시에 끌고 가면서도 누락 시나리오를 사전에 잡아냅니다.

Main Topic. AI Native Engineering

AI를 제품에 넣는 방식

AI를 **어디에 쓸지보다 어디에 쓰지 않을지**를 먼저 정하는 것이 결과를 갈랐습니다.
관점 · 내재화 · 신뢰성 · 통제 · 프로세스의 다섯 갈래로 정리했습니다.

관점
화면을 데이터로, 데이터를 AI가

내재화
RAG 챗봇 · SSE 스트리밍 UI

신뢰성
구조 강제 · 관측 · 재실행 복구

통제
판단은 AI, 흐름은 코드

화면을 코드가 아니라 데이터로 다루면, 그 데이터는 사람도 AI도 만들 수 있습니다.

멀티테넌트 플랫폼에서 페이지 구성을 DB에 저장하고 렌더러가 해석하도록 설계했습니다.
이 구조 위에서 테넌트 유형을 고정형에서 자유형을 거쳐 생성형으로 확장하고 있습니다.

고정형
JSON 스키마
운영 중

→

자유형
노드 비주얼 빌더
부분 적용

→

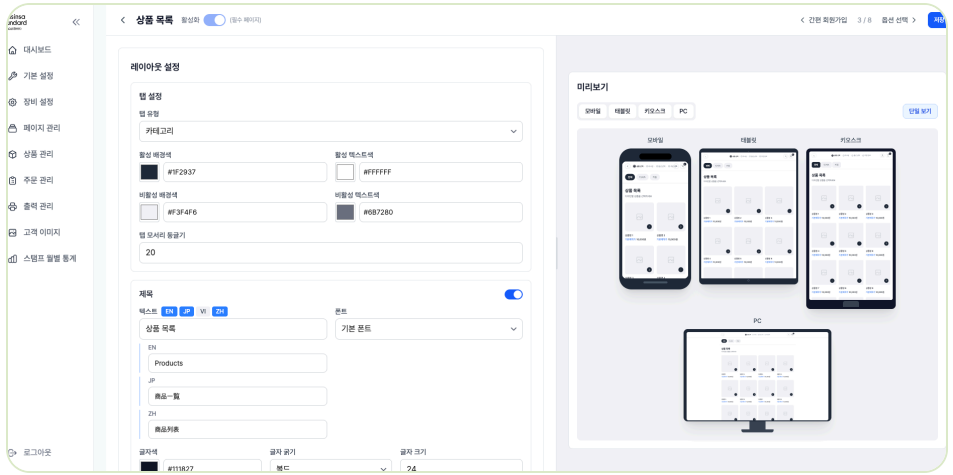
생성형
AI가 구성 생성
개발 중

→

개인화 경험
사용자마다 다른 화면

왜 이 방향인가

- 화면이 코드에 박혀 있으면 **변경마다 개발과 배포**가 따라옵니다. 개인화의 단위는 결국 사람이 손댈 수 있는 횟수로 제한됩니다.
- 구성을 데이터로 분리하면 **변경 주체를 바꿀 수 있습니다.** 개발자에서 운영자로, 운영자에서 AI로 단계적으로 옮겨갈 수 있습니다.
- 생성형 단계에서도 **데이터 정합과 주문 흐름은 코드가 통제**합니다. AI는 구성 판단만 맡습니다.

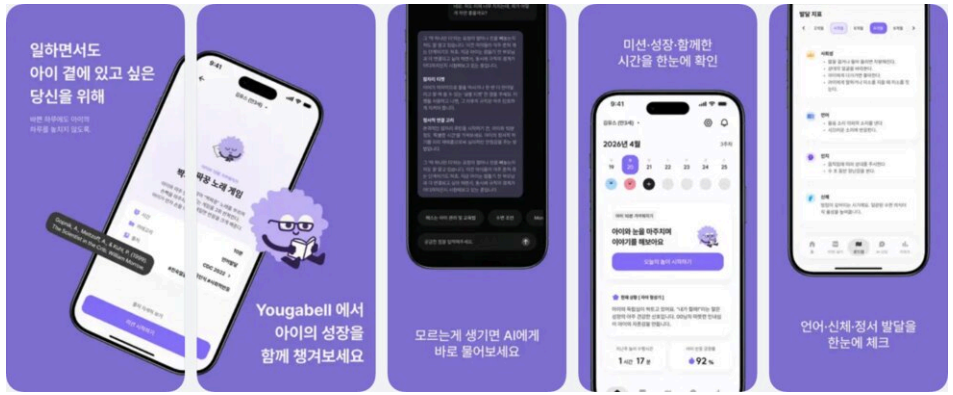


페이지 빌더 - 구성은 데이터로 저장되고, 4개 뷰포트로 즉시 렌더링

LLM을 화면 안으로 끌어들였습니다

- **호출을 한 곳으로** - 흩어지기 쉬운 LLM 호출을 NestJS API 한 지점에 모았습니다. 프롬프트·모델·토큰 정책을 한 곳에서 바꿀 수 있게 됩니다.
- **맥락 구성** - 아이 정보와 최근 기록을 묶어 맥락을 만들고, 임베딩 검색(RAG)으로 지식 조각을 붙여 답변 근거를 확보했습니다. Prisma **39개 모델**에 pgvector 임베딩을 포함해 설계했습니다.
- **스트리밍 UI** - 응답을 기다리게 하지 않고 **SSE 스트리밍**으로 토큰이 도착하는 대로 화면에 흘렸습니다. 체감 지연을 줄이는 것이 AI 기능의 사용성을 좌우했습니다.
- **화면 밖까지** - 한 주의 기록을 격려 글로 바꾸는 AI 주간 리포트를 함께 만들어, 챗봇 외의 진입점에서도 AI가 동작하게 했습니다.

Stack - NestJS 11 · Prisma 7 · pgvector · Vercel AI SDK · Gemini 2.5 Flash · Next.js 16 · Expo

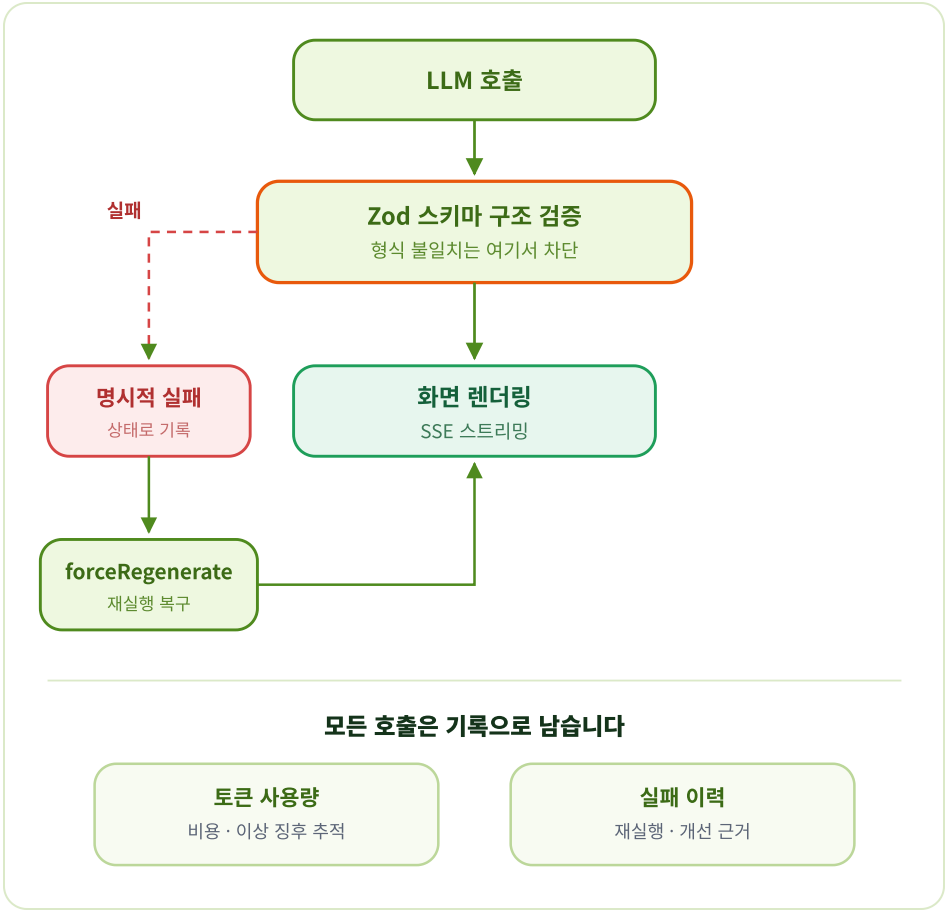


RAG 챗봇과 AI 주간 리포트가 들어간 실제 출시 화면

문제는 모델이 틀리는 것이 아니라, 틀린 줄 모르는 것이었습니다

- **구조를 강제** - Zod 스키마(Output.object)로 LLM 응답 구조를 고정했습니다. 형식이 어긋난 응답은 화면에 닿기 전에 걸러집니다.
- **사용량을 기록** - 호출별 토큰 사용량을 DB에 남겨 비용과 이상 징후를 사후에 추적할 수 있게 했습니다.
- **실패를 감춤 없이** - AI 실패를 조용히 넘기지 않고 **명시적 실패 상태**로 남깁니다. 실패가 데이터에 보이지 않으면 개선할 수도 없습니다.
- **복구 경로** - 남은 실패는 **forceRegenerate** 재실행으로 복구합니다. AI가 흔들려도 화면과 데이터가 끊기지 않습니다.

AI 기능의 신뢰는 잘 동작할 때가 아니라 실패할 때 결정됩니다

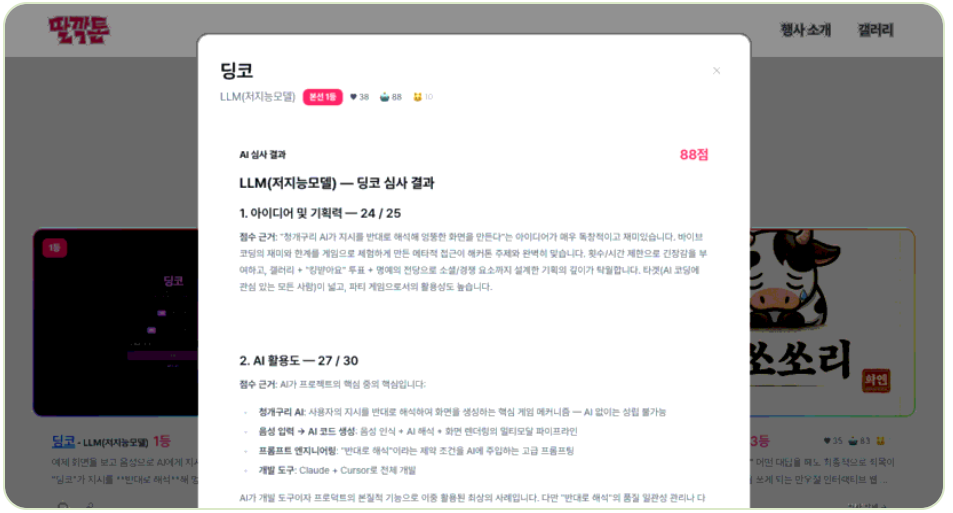


구조 강제 → 명시적 실패 → 재실행 복구 → 기록으로 축적

판단은 AI가, 흐름은 코드가 통제합니다

딸깍톤 - 심사위원 수라는 상한을 구조로 넘다

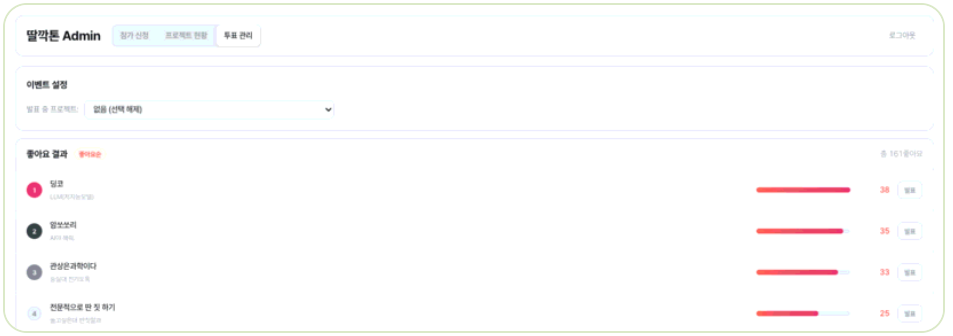
- 해커톤 심사는 **심사위원 수가 곧 처리량의 상한**입니다. 참가팀이 늘면 시간과 비용이 그대로 따라 늘어납니다.
- 디스코드 명령 한 줄로 호출되면, 서버가 팀의 프로젝트 정보를 불러오고 GitHub·데모·영상 링크를 직접 확인한 뒤 점수와 근거를 생성하고 파싱해 DB에 저장한 다음 회신합니다.
- **심사 규칙을 코드가 아닌 프롬프트로 분리**했습니다. 기준이 바뀌어도 서버 코드를 건드리지 않고, 한 팀이 여러 프로젝트를 낸 경우까지 확장했습니다.
- 판단은 AI가 하되 조회·저장·회신 흐름은 코드가 통제해, **행사 진행 중에도 결과가 흐트러지지 않고 축적**됐습니다.



AI 자동 심사 결과 - 항목별 점수와 근거 서술

날별 - 계산과 해석을 분리하다

- 사주 계산은 절기와 율리우스적일을 쓰는 **결정론적 엔진**이 맡고, AI는 그 결과를 풀어 쓰기만 합니다.
- 덕분에 **모델을 바꿔도 계산 값이 달라지지 않습니다.** 정확해야 하는 영역과 유연해야 하는 영역을 나눈 사례입니다.



운영 어드민 - 투표 집계와 발표 제어는 코드가 통제

AI를 결과물이 아니라 개발 과정 자체에 넣었습니다

- **작업 범위를 문서로 분리** - Claude와 Codex가 동시에 작업해도 충돌하지 않도록 진입점과 파일별 담당 범위를 **CLAUDE.md · AGENTS.md**로 나누는 체계를 직접 만들었습니다.
- **반복은 스킬로** - 매번 되풀이되는 작업은 스킬로 만들어 재사용합니다. 같은 지시를 다시 쓰지 않게 하는 것이 실제 속도 차이를 만들었습니다.
- **작업 단위는 plan 문서로** - 디렉터리에 plan 문서를 두고 작업 단위를 관리해, 진행 상태와 결정 근거가 코드 밖에 남습니다.
- **코드 생성에만 쓰지 않음** - 기획 검토와 코드 리뷰에도 함께 넣어, 적은 인원으로 여러 제품을 동시에 끌고 가면서도 **놓친 시나리오와 예외를 사전에 포착**합니다.

Tools - Claude · Codex · Gemini

에이전트를 팀 구성원처럼 배치하고 책임 범위를 설계

AI를 붙인 자리마다 사람의 손이 상한이던 지점이 사라졌습니다

영역	AS-IS	TO-BE
해커톤 심사	심사위원 수가 처리량 상한, 팀 수에 비례한 시간·비용	**명령 한 줄로 조회·평가·저장·회신 자동화, 규칙은 프롬프트로 무수정 대응**
AI 응답 품질	형식이 깨져도 화면에 그대로 노출, 실패를 사후 추적 불가	**Zod 구조 강제로 사전 차단, 명시적 실패 기록 + 재실행 복구**
AI 운영 비용	사용량 가시성 없음	**호출별 토큰 사용량 DB 기록으로 비용·이상 징후 추적**
페이지 구성	요구마다 코드 수정과 배포	**데이터로 분리해 운영자가 직접 구성, 생성형(AI) 단계로 확장 중**
개발 프로세스	1인분 속도, 검토 누락은 사후 발견	**멀티에이전트 병렬 작업, 기획·리뷰 단계에서 예외 사전 포착**

육아뱰 iOS · Android 양대 스토어 출시 · 딸깍톤 실제 행사 운영 · 개발 커밋 약 85% 담당

배운 점

- AI 기능의 신뢰는 잘 동작할 때가 아니라 **실패할 때** 결정됩니다. 실패를 감추지 않고 데이터로 남기는 설계가 제품 품질을 좌우했습니다.
- AI를 빠른 코드 생성기로만 쓰면 개인의 속도가 조금 빨라지지만, **작업 범위와 책임을 설계**하면 팀의 구조가 바뀝니다.

01

Company. 주식회사 틸스

DPS - B2B 주문형 굿즈·인쇄 제조 커머스 플랫폼

2024.08 ~ 진행 중 (운영·기능 개선 지속)

프로젝트 배경

- 24년 정부지원사업 프로토타입에서 출발해 실서비스·매출 단계까지 확장한 커머스 플랫폼
- 관리자·파트너·스토어·크리에이터 4개 역할군 멀티테넌트 구조 (인증은 NextAuth 3역할 분리)

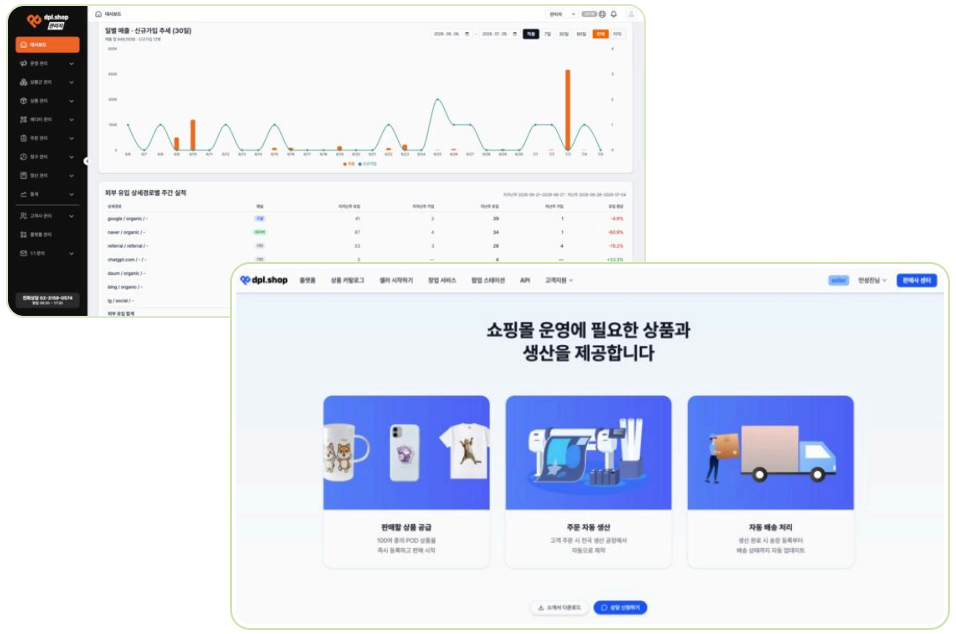
주요 역할 - 총 개발 리더 (팀 2~7명 · 개발 기여 약 60%)

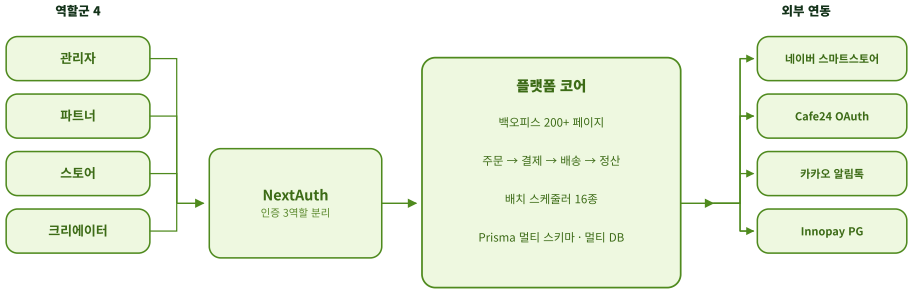
- 역할군별 백오피스 설계·구축 - **200+ 페이지 · 400+ 컴포넌트**
- 상품상세·옵션·장바구니·주문서 등 고객 플로우 프론트엔드 구현
- 네이버 스마트스토어·Cafe24 OAuth, 카카오 알림톡, Innopay PG 연동
- Prisma 멀티 스키마·멀티 DB 데이터 모델 설계

핵심 성과 · 트러블슈팅

- 수기로 관리되던 주문 승인 → 결제 → 배송 추적 → 정산 흐름을 **16개 배치 스케줄러**로 자동화해 상태 누락 제거
- 프로토타입을 실서비스로 확장, 인쇄 전시회 출품 등 사업 확장 지원

Stack - Next.js · React · TypeScript · Prisma · MySQL/MariaDB · Zustand · TanStack Query · NextAuth · AWS S3





관리자·파트너·스토어·크리에이터 4개 역할군 - NextAuth 3역할 인증으로 하나의 코어 사용

02

Company. 주식회사 틸스

PHP 레거시 마이그레이션 & 외부 커머스 연동

2023.03 ~ 2024.01 (Cafe24 연동 유지보수 지속)

프로젝트 배경

- 파일 하나가 **5천~1만 줄**에 달하는 15년 된 PHP 레거시 - 신규 입사자가 적응하지 못해 연쇄 퇴사하던 상황
- 신규 스택 전환을 상급자에게 건의하고, 단독 담당 프로젝트로 직접 증명하는 전략 수립

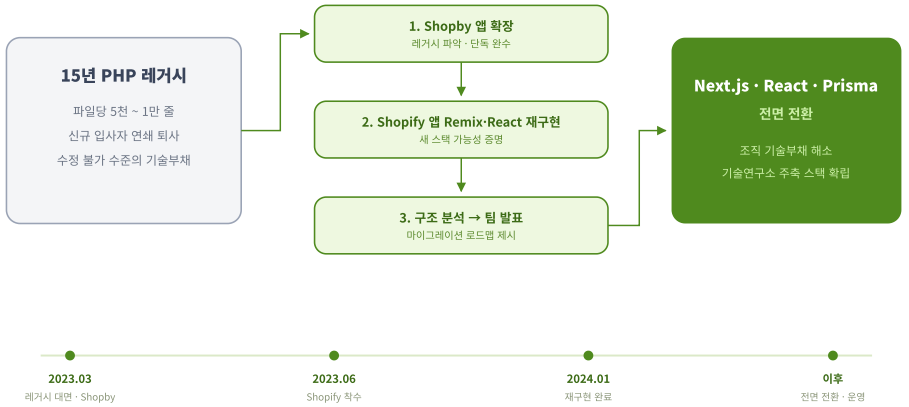
주요 역할 - 단독 담당

- Shopby(NHN) 앱스토어 확장 - 구형 에디터 솔루션 플랫폼 확장 (2023.03~05)
- Shopify 앱스토어 확장 - 구형 솔루션을 **Remix·React·TypeScript로 재구현**, Liquid 테마 연동 (2023.06~2024.01)
- Cafe24 앱스토어 에디터 솔루션 연동 개발·유지보수 (2023.03~)
- 스파게티 코드의 디렉터리 구조·핵심 로직을 분석해 팀에 공유, 마이그레이션 로드맵 제시

핵심 성과

- 기술연구소 주축으로 **회사 솔루션의 Next.js·React·Prisma 전면 전환**을 이끌어 기술부채 해소
- 주문·재고·정산 데이터가 외부 플랫폼 사이에서 어긋나지 않도록 정합 유지

Stack - Remix.js · React · TypeScript · PHP · VanillaJS · Liquid



단독 담당 연동 앱으로 새 스택을 증명 → 조직 전체의 전환으로

03

Side Project. 4인 팀 (기획 1 · 디자인 1 · 개발 2)

딸깍톤 - AI 해커톤 운영·자동 심사 플랫폼

2026.02 ~ 2026.04 (약 6주) · 실제 행사 운영

프로젝트 배경

- 해커톤 심사를 사람이 일일이 수행하는 비용을 줄이기 위해, 디스코드 명령 한 줄로 호출하는 AI 자동 심사 플랫폼을 기획

주요 역할 - 개발 2인 중 백엔드·AI 파이프라인 담당 (개발 커밋 약 85%)

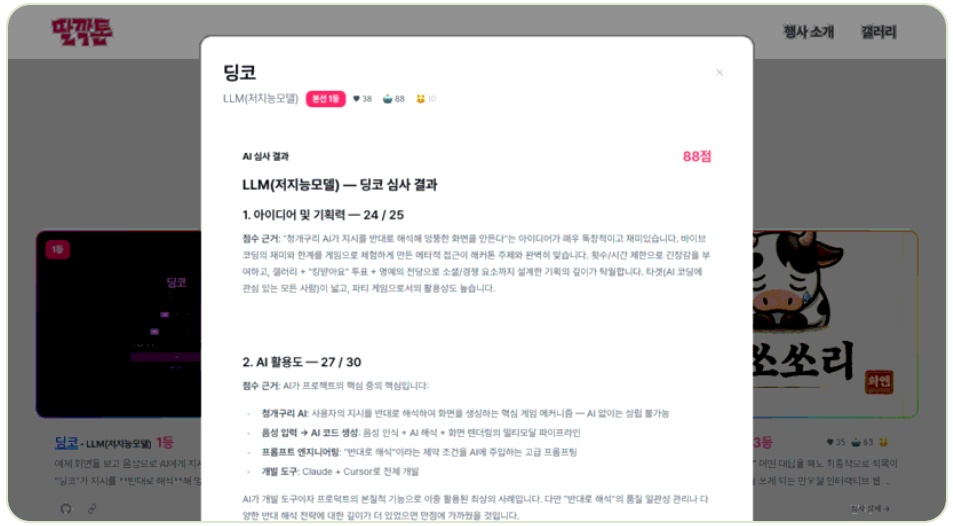
- 프로젝트 조회 → GitHub 코드·제출 정보 검토 → 평가 → 파싱 → DB 저장 → 디스코드 회신의 심사 파이프라인 설계
- 심사 규칙을 코드가 아닌 **프롬프트로 분리** - 기준 변경 시 코드 무수정 대응
- 심사 결과를 **마크다운 타이핑 효과 + TTS**로 현장 스크린에 연출 - 행사장의 몰입 요소 구현

핵심 성과

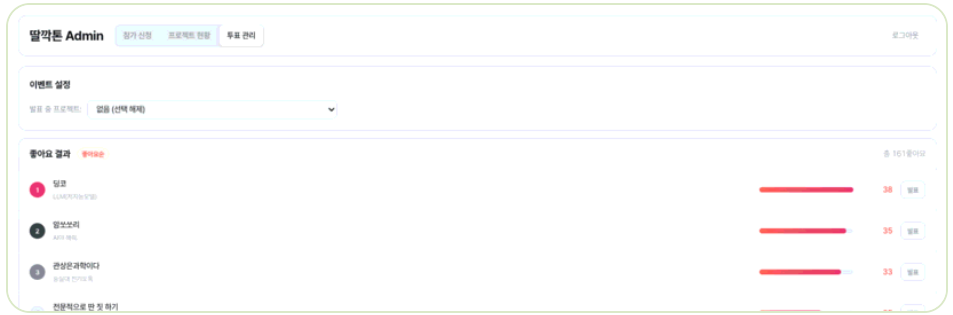
- 실제 해커톤 행사에서 운영 - 판단은 AI가, 조회·저장·회신 흐름은 코드가 통제하는 구조로 **행사 중에도 안정적으로 동작**
- 심사 결과를 **마크다운 타이핑 효과·TTS**로 현장 스크린에 연출 - AI 출력을 실시간 경험으로 만드는 감각

Stack - Next.js 16 · TypeScript · Prisma 7 · Neon Postgres · Vercel

인터랙티브 랜딩 (motion · canvas-confetti)



AI 자동 심사 결과 - 항목별 점수와 근거 서술 (본선 1등 팀 "딩코" · 88점)



운영 어드민 - 참가자 투표 집계(총 161표) · 프로젝트별 발표 제어

04

Side Project. 4인 팀 (기획 1 · 디자인 1 · 개발 2)

육아밸 - RAG 챗봇 · AI 주간 리포트 육아 서비스

2026.06 · iOS·Android 양대 스토어 출시

프로젝트 배경

- 육아 기록을 기반으로 질문에 답하는 RAG 챗봇과 AI 주간 리포트를 제공하는 모바일 서비스
- Expo 네이티브 셸 + Next.js 웹뷰의 **하이브리드 앱** 구조 - 웹 기술로 앱 경험을 구현

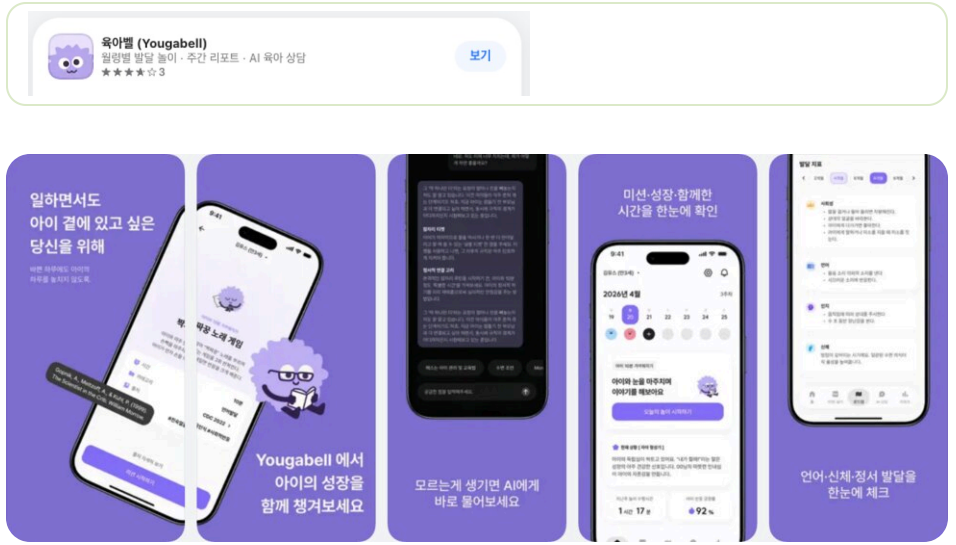
주요 역할 - API·백엔드 약 80% 기여 (커밋 약 80%)

- 모든 LLM 호출을 NestJS API 한 곳으로 모으고, 임베딩 검색(RAG) 기반 챗봇을 **SSE 스트리밍 UI**로 구현
- Zod 스키마(Output.object)로 **LLM 응답 구조를 강제**하고 토큰 사용량까지 DB에 기록 - AI 실패는 명시적 실패로 처리하고 **forceRegenerate 재실행**으로 복구하는 관측 가능한 AI 파이프라인
- **OpenAPI(Swagger) 명세**를 단일 계약으로 삼아 멀티레포(API·웹·어드민·모바일) 타입 동기화 - 4갈래 작업이 어긋나지 않는 협업 구조 구축
- Prisma **39개 모델**(pgvector 임베딩 포함) 데이터 설계

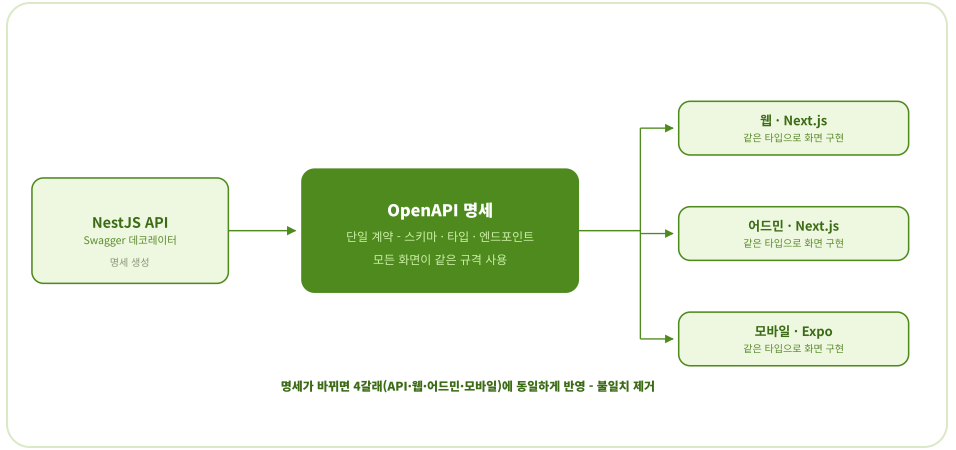
핵심 성과

- 기획·디자인·개발이 나뉜 팀에서 타입 계약 기반 협업으로 **짧은 기간에 iOS·Android 양대 스토어 출시까지 도달**
- 스트리밍 응답·구조 검증 등 AI 기능을 실사용 품질로 다듬는 프론트-백 통합 감각 확보

Stack - NestJS 11 · Prisma 7 · Next.js 16 · Expo · Vercel AI SDK · Gemini 2.5 Flash · Supabase



iOS·Android 양대 스토어 출시 - App Store 등록 카드와 공식 스크린샷



기획·디자인·개발이 나뉜 4인 팀 - 명세를 단일 계약으로 삼은 협업 구조

올리브영에서
정형화된 화면을 넘어선
커머스 경험을 만들겠습니다.

감사합니다.

AI-First Track · Front-End 지원자 안성진