

주도현 Dohyeon Ju

사용자의 관점에서 시작해 비즈니스로 완성하겠습니다.

기술적 선택을 서비스의 목표와 연결해,
비즈니스에 필요한 제품을 설계하고 구현합니다.

01 / PRODUCT

사용자 경험

웹 · 모바일 · 관리자 화면

02 / SYSTEM

상태와 데이터

결제 · 정산 · 집계 · 정책

03 / OPERATIONS

지속적인 운영

배포 · 관측 · 복구 · 협업

동작42

약 3억원 규모 사업의 결제·정산

한컴 · LIFEPLUS

운영 지표와 관리자 도구

ViewUp

스토어 출시와 관측·핫픽스

사용자 경험부터 데이터와 운영까지 연결합니다.

시각디자인에서 출발해 웹·앱 개발로 범위를 넓혔습니다. 제품의 맥락을 파악하고, 필요한 도구를 익혀 실제 운영에 적용해 왔습니다.

2026.04 - 현재

스마일드래곤

Frontend Engineer

앱 전환 · 제품 개편 · 배포·관측 · 개발 협업

2023.05 - 2026.04

획기획

풀스택 개발자 · 개발팀장

사용자 서비스 · 관리자 도구 · 결제·정산

2019.08 - 2022.10

주도 · 개인사업자

웹 개발 · 디자인과 운영 경험

React·Next.js·React Native

NestJS·Prisma·MySQL·PostgreSQL

Docker·AWS·CI/CD·관측과 운영

사례명과 쪽수를 누르면 해당 페이지로 이동합니다.

03	동작42 방문쿠폰 지역 상권 쿠폰·결제 서비스	↗
04	LIFEPLUS Creator Campus 대학생 크리에이터 모집·지원 서비스	↗
05	몰입도 근무·휴가·조직 관리 제품	↗
06	PoC Publisher 웹 실험 배포·운영 플랫폼	↗
07	한컴 문서운 이야기 사연 참여·맞춤 결과 캠페인	↗
08	ViewUp · 출시 이후의 개발 콘텐츠·미션 참여 리워드 서비스	↗
09	챗조도 채팅·파일·화상회의 협업 서비스	↗
10	AI와 함께 일하는 방식 맥락 전달 · 결과 검증 · 반복 작업의 자산화	↗

동작42 방문쿠폰

시민이 지역 상점에서 쿠폰을 사용하고, 상점이 지원금을 정산받는 동작구 상권 활성화 서비스입니다.

개발 리드 · 4인 개발팀

Next.js / NestJS / Prisma · MySQL / Redis · BullMQ

01 문제 예산과 결제 상태가 어긋날 여지

기존 구조는 쿠폰 발급과 예산 차감을 별도로 처리했습니다. 2025년 PG 결제를 도입하면서 동시 주문의 예산 초과, 결제 중 이탈, 취소·복구까지 같은 상태 기준으로 다뤄야 했습니다.

02 해결 예산 예약부터 실패 복구까지 설계

Serializable 트랜잭션에서 진행 중 주문도 예산에 포함했습니다. PG 알림을 검증해 결제를 확정하고, 중복 처리를 막았습니다. 내부 실패는 PG 취소로 보상하고 5분 만료 주문은 정리했습니다.

03 선택 이유 외부 결제와 내부 저장의 차이를 반영

화면 복귀만으로는 결제 성공을 보장할 수 없어 서버 알림을 기준으로 삼았습니다. PG와 DB를 한 트랜잭션으로 묶을 수 없으므로 취소 보상이 필요했고, 예산 예약에는 만료 처리를 함께 뒀습니다.

04 결과 쿠폰 사용에서 월 정산까지 운영

약 3억원 예산·150개 상점·등록 사용자 6,000명 규모의 사업에 적용했습니다. 매출 확인·본인 재인증·서명·증빙을 정산으로 연결하고, 회차·한도·기간을 재배포 없이 변경하게 했습니다.

약 3억원

사업 예산

약 150곳

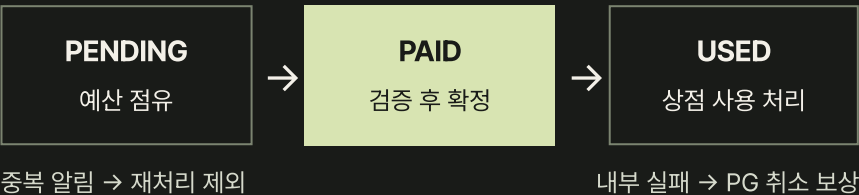
참여 상점

약 6,000명

등록 사용자

경력 기록 기준의 근사 규모

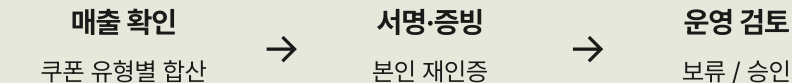
결제 확정의 기준: PG 서버 알림



Redis · BullMQ

5분간 미완료된 주문의 타임아웃 처리

상점 월 정산



코드와 경력 기록에 근거한 운영 흐름 설명도

담당 범위 전체 데이터 흐름·프론트 구조·상태 모델 설계 및 운영 대응. 실시간 리포트 데이터와 일부 관리자 UI는 동료 담당.

사례 상세 7

LIFEPLUS Creator Campus

대학생 크리에이터 프로그램의 안내·지원서 작성·포트폴리오 접수와 심사 준비를 돕는 모집 서비스입니다.

서비스 기획 · 풀스택 개발 · 팀 프로젝트

Next.js / Prisma · MySQL / AWS S3 / Docker · Jenkins

01 문제 작성 중인 지원서와 기수별 정책 변경

간 지원서는 나눠 작성해야 하고, 개인·팀 지원의 필수 조건도 달랐습니다. 기수마다 바뀌는 문항·일정을 반영하면서 접수 이후 검색과 심사용 자료 가공까지 지원해야 했습니다.

02 해결 작성·제출·심사 준비를 하나로 연결

DRAFT와 SUBMITTED의 검증 조건을 나누고, 가변 문항을 ApplicationMeta로 분리했습니다. S3 직접 업로드와 관리자 검색·상태별 조회, Excel·암호화 ZIP 내보내기를 구현했습니다.

03 선택 이유 작성 편의와 변경 범위를 함께 고려

미완성 입력은 저장하고 제출 시 필수값을 검증해 이어 쓰기를 허용했습니다. 고정 정보와 가변 문항을 나눠 기수별 변경을 수용하고, 큰 파일은 앱 서버를 거치지 않고 S3로 보내게 했습니다.

04 결과 1기 기반을 2기 모집에 재사용

1기 구조를 바탕으로 2기의 문항·일정·지원 조건을 개편하고 운영을 이어갔습니다. 지원자는 저장 후 다시 작성하고, 담당자는 접수 상태 확인부터 심사 자료 추출까지 처리할 수 있게 됐습니다.



DRAFT

작성 중에도 저장

미완성 입력 허용



SUBMITTED

제출 시 검증

필수값 + 개인·팀 조건

지원자 목록

원본 기능 재구성 / 예시 데이터

이름 검색

개인 / 팀 임시저장 / 제출완료

이름	지원유형	상태	자료 확인
지원자 A	개인	제출완료	상세 보기
지원자 B	팀	임시저장	상세 보기

선택 자료 → Excel · 암호화 ZIP / 포트폴리오 → S3 직접 업로드

관리자 코드에서 확인한 검색·필터·자료 확인 흐름. 실제 지원자 정보는 사용하지 않았습니다.

담당 범위 팀 프로젝트에서 서비스 기획과 풀스택 개발을 담당하고, 2기 개편과 운영 대응을 이어갔습니다.

사례 상세 ↗

몰입도

근무 시간 기록을 바탕으로 업무·프로젝트·휴가·계약·조직 정보를 함께 관리하는 내부 업무 제품입니다.

풀스택 개발 · 팀 프로젝트

Next.js · React / NestJS / PostgreSQL · Prisma / BullMQ

01 문제 한 기록의 수정이 주변과 과거에 미치는 영향

근무 시간을 바꾸면 인접 기록과 겹칠 수 있어 사용자가 일일이 정리해야 했습니다. 과거 기록의 수정이 이미 확정된 휴가 산정 결과까지 자동으로 바꾸지 않도록 근거를 보존해야 했습니다.

02 해결 겹침을 정리하고 확정 시점의 근거를 저장

새 구간을 기준으로 기존 기록을 삭제·축소·분할하고 저장 전 결과를 보여줬습니다. 서버는 독립 검증을 맡고, Session의 근무 사실과 WorkReport 계산 결과를 Snapshot으로 보존했습니다.

03 선택 이유 즉시 확인하는 화면과 신뢰할 저장 경계

양쪽 규칙을 유지하는 비용을 감수해 편집 결과는 바로 보여주고 저장은 서버에서 검증하게 했습니다. 원본과 확정 결과를 분리하면 현재 기록을 고쳐도 과거의 산정 근거를 설명할 수 있습니다.

04 결과 저장 전 미리보기와 과거 결과 재현

예를 들어 09~12시 기록 중 10~11시를 바꾸면 앞·뒤 구간의 분할 결과를 즉시 확인합니다. 확정 Snapshot은 원본 수정에도 유지되며, 재산정은 별도 동작으로 처리하게 했습니다.

시간 구간 편집의 예

규칙 설명용 예시



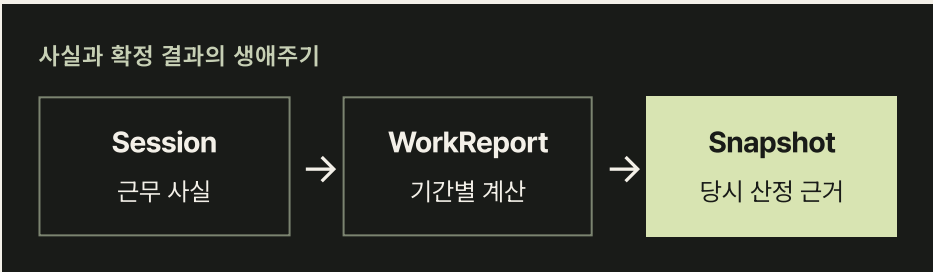
프론트 메모리 버퍼

같은 규칙으로 즉시 미리보기



서버 저장 경계

독립 검증 후 데이터 반영



현재 원본의 수정과 이미 확정된 지급 근거를 분리합니다.

담당 범위 기존 백엔드의 중첩 해소·산정 Snapshot과 개편 프론트의 편집 로직을 구현했습니다. 개편 백엔드의 조정 엔진·보상휴가 원장은 동료 담당.

사례 상세 [개](#)

PoC Publisher

작은 웹 실험을 실제 URL로 배포하고, 프로젝트별 데이터·파일·서버 함수를 관리하는 개인용 플랫폼입니다.

개인 프로젝트 · 설계 및 구현

React · Vite / Python / SQLite / Node.js / Nginx · MCP

01 문제 실험마다 반복되는 서버와 데이터 준비

작은 아이디어도 공개하려면 웹 서버·라우팅·데이터 저장·파일 업로드를 매번 준비해야 했습니다. 화면 구현 이후에도 배포 작업이 남아, 다음 실험에 재사용할 공통 기반이 필요했습니다.

02 해결 배포 단계와 실행 환경을 공통화

Draft·Preview·Publish와 Revision 롤백을 구현했습니다. Python Publisher 아래 프로젝트별 SQLite, 파일·키·백업, Node 함수 실행과 Nginx 라우팅을 묶고 관리 화면·MCP를 연결했습니다.

03 선택 이유 개인용 규모에 맞는 단순한 경계

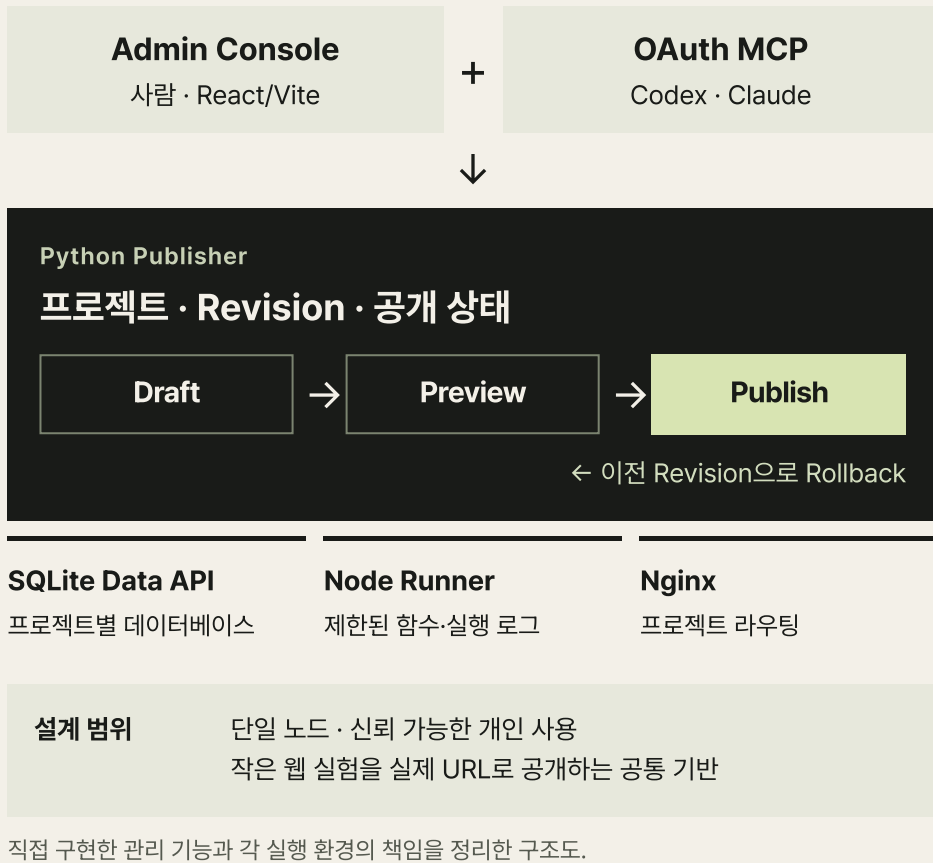
신뢰 가능한 개인 사용과 단일 노드를 전제로, 프로젝트별 SQLite로 데이터·백업 경계를 명확히 했습니다. 검토와 공개를 분리하고 사람·AI가 같은 프로젝트 모델을 써 배포 경로를 공유하게 했습니다.

04 결과 화면을 넘어 동작하는 서비스로 공개

데이터 저장·파일·함수가 필요한 실험도 실제 URL에서 동작하게 했습니다. Codex·Claude가 프로젝트를 만들고 갱신할 수 있으며, 다음 실험은 기존 기반에서 시작하고 이전 Revision으로 복구할 수 있습니다.

하나의 프로젝트, 두 개의 진입점

구조 설명도



담당 범위 기획·아키텍처·관리 화면·API·배포 환경을 직접 구현했습니다. 단일 노드에서 신뢰 가능한 개인 사용을 전제로 운영합니다.

사례 상세 7

한컴 문서운 이야기

문서 작업에서 겪은 사연을 입력하면 유형별 부적과 한컴어시스턴트 기능을 소개하는 참여형 캠페인입니다.

풀스택 개발

Next.js / Prisma · MySQL / AWS Bedrock / Chart.js

01 문제 제각각인 사연을 분류하고 후속 행동 파악

짧은 사연도 표현이 다양해 단어 일치만으로 의미를 나누기 어려웠습니다. 욕설 차단과 의미 분류를 구분하고, 응모 이후 공유·다운로드·제품 방문까지 확인할 운영 지표가 필요했습니다.

02 해결 의미 분류와 행동 집계를 각각 구현

Titan 임베딩을 342개 키워드 벡터와 비교해 7개 유형을 선택하고, Aho-Corasick으로 금치어를 검사했습니다. 참여 이벤트를 사용자·응모 데이터와 묶어 일자별·누적 통계와 Excel로 제공했습니다.

03 선택 이유 의미의 유사성과 문자열 일치의 차이

같은 뜻을 다른 말로 쓴 사연에는 벡터 유사도를, 정해진 금치어에는 문자열 탐지를 적용했습니다. 두 판단을 분리하고, 응모 건수와 후속 행동도 별도로 집계해 참여 흐름을 살필 수 있게 했습니다.

04 결과 참여 경험과 캠페인 운영을 연결

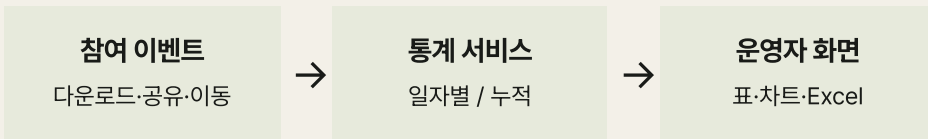
사연 입력에서 유형별 결과·공유·제품 안내까지 이어지는 흐름을 구현했습니다. 담당자는 날짜별 추이·인기작·목표 달성률을 확인하고 데이터를 추출할 수 있게 됐습니다.

이벤트 통계 · 일자별

표시 항목 재구성 / 예시 데이터

참여와 후속 행동					Excel 내보내기
예시 일자	누적 참여자	누적 응모작	부적 다운로드	공유	제품 유입
1일차	12	18	9	5	3
2일차	20	31	16	8	6
추가 제공: 인기작 좋아요 · 목표 달성률 · 기간별 차트					

원본 통계 코드의 집계 항목을 재구성했습니다. 숫자는 실제 성과가 아닙니다.



의미 분류

사연 → 임베딩 → 7개 유형

Titan · 코사인 유사도
342개 사전 키워드 벡터와 비교

금치어 검사

입력 → 문자열 규칙

Aho-Corasick
정해진 금치어 목록을 탐지

담당 범위 응모·분류·관리자·지표 기능을 구현했습니다. 참여 데이터 수집부터 운영 화면의 집계·조회까지 연결했습니다.

사례 상세 ↗

ViewUp · 출시 이후의 개발

콘텐츠 조회와 미션 참여로 포인트를 적립하는 리워드 서비스로, 웹을 네이티브 앱으로 전환해 운영했습니다.

Frontend Engineer · 스마일드래곤

React Native · Expo / Next.js / EAS / SigNoz / Storybook · MSW

01 문제 출시 이후 오류 추적과 잦은 변경 대응

웹을 앱으로 옮긴 뒤에도 디자인·API 변경과 운영 오류에 대응해야 했습니다. 사용자 오류와 서버 요청의 맥락이 끊겨 있었고, 앱 바이너리와 OTA 업데이트의 호환 기준도 필요했습니다.

02 해결 관측·배포·화면 검토를 연결

Next.js·Expo 에러 리포팅과 traceparent로 백엔드 trace를 연결했습니다. EAS 환경 ·runtimeVersion·OTA 기준을 정리하고, Storybook UI Picker·MSW로 디자인·API 변경을 화면에서 검토했습니다.

03 선택 이유 실패 원인과 변경 영향을 확인할 기준

화면 오류만으로 원인을 알 수 없어 요청·응답과 서버 trace를 함께 보게 했습니다. OTA는 호환되는 런타임에 배포하고, 디자이너·백엔드 개발자가 실제 화면에서 변경 영향을 확인하도록 했습니다.

04 결과 스토어 출시와 크래시 3건 핫픽스

기존 웹 기능을 React Native·Expo 앱으로 이관해 스토어에 출시했습니다. SigNoz로 프로덕션 크래시 3건을 포착해 핫픽스로 대응했고, 화면 수정 위치와 API 변경을 함께 검토하는 흐름을 마련했습니다.

사용자 오류에서 서버 원인까지

관측 흐름 설명도

- 01 **웹·앱 사용자 오류**
Next.js · Expo 에러 리포팅 SDK
- 02 **프론트 요청과 응답**
traceparent 전파
- 03 **백엔드 trace**
관련 API 실패 원인 탐색

3건 프로덕션 크래시 포착·대응
SigNoz 관측 → 원인 확인 → 핫픽스

Storybook에서 검토 범위를 구체화

디자이너 UI Picker로 위치 지정 → 같은 화면에서 수정 확인

백엔드 개발자 MSW 응답 수정 → 계약 변경의 화면 영향 확인

크래시 건수는 경력 기록 기준. 화면·계약 검토와 전체 앱 검증을 함께 운영했습니다.

담당 범위 앱·웹 프론트엔드, 에러 리포팅 SDK 연결, 관측·릴리즈 개선을 담당하며 디자이너·백엔드 개발자와 협업했습니다.

사례 상세 ↗

챗조도

주제별 메시지·댓글·파일 공유와 화상회의를 한 공간에서 사용하는 실시간 팀 협업 서비스입니다.

풀스택 · 메인 개발자

Next.js / NestJS / Socket.IO / React Query / WebRTC

01 문제 하나의 이벤트가 여러 화면을 바꾸는 구조

새 메시지는 대화 목록·상세·읽지 않음 상태에 함께 반영되어야 합니다. 파일 업로드와 화상회의도 비동기로 상태가 바뀌어, 통신 처리가 화면에 섞이면 이벤트마다 여러 컴포넌트를 손봐야 했습니다.

02 해결 통신·모델 변환·화면 반영의 책임 분리

Socket 응답을 화면 모델로 변환해 MessageBus에 발행하고, Receiver가 React Query 캐시를 갱신하게 했습니다. API·파일·Socket·WebRTC 동작은 일반 TypeScript 객체로 구성했습니다.

03 선택 이유 화면이 달라져도 핵심 동작을 재사용

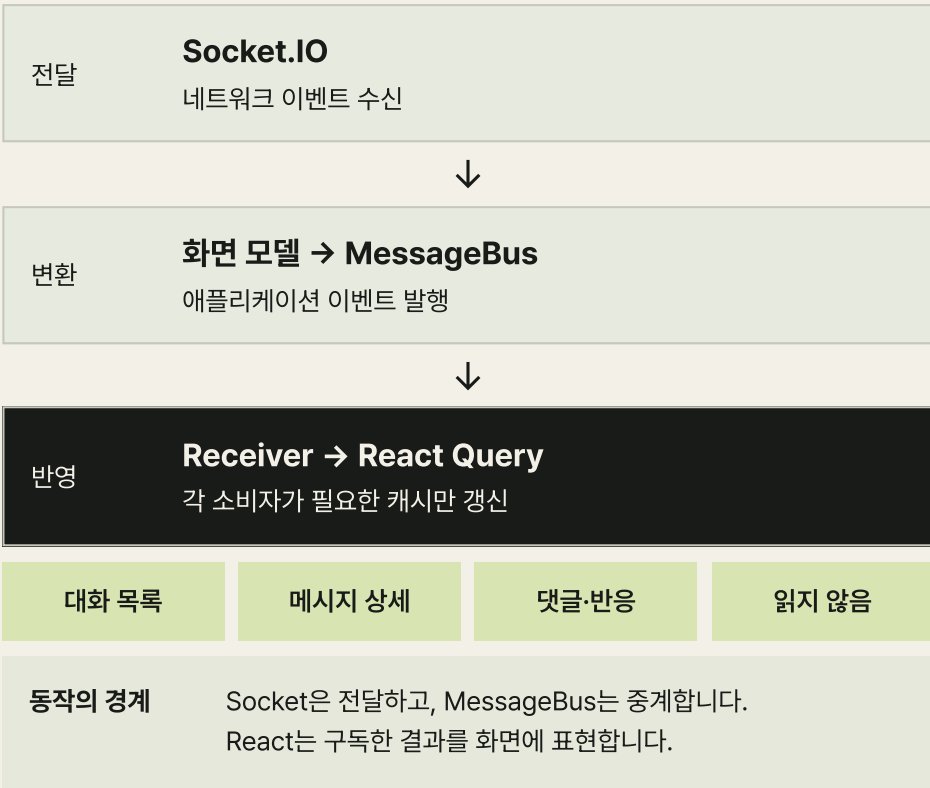
통신 방식과 React 생명주기의 결합을 줄이기 위해 동작을 화면 밖에 뒀습니다. 각 Receiver가 필요한 상태만 갱신하면 전달 코드를 반복하지 않고, 여러 화면에서 같은 서비스 동작을 조합할 수 있습니다.

04 결과 메시지·파일·화상회의를 같은 흐름으로

채팅, 파일 업로드 진행률, WebRTC track 변화를 공통 이벤트 흐름으로 연결했습니다. 화면은 필요한 결과를 구독하고 해제 시 정리하는 역할을 맡아, 협업 기능을 한 제품 안에 통합했습니다.

새 메시지 하나가 UI에 반영되기까지

구조 설명도



파일 업로드 진행률과 WebRTC track도 같은 이벤트 흐름으로 전달했습니다.

담당 범위 팀의 메인 개발자로 프론트·백엔드 초기 구조를 잡고, 메시지·파일 업로드·화상회의의 주요 기능을 구현했습니다.

사례 상세 ↗

AI에 실행을 맡기고, 실제 결과로 확인합니다.

Claude Code·Codex를 구현에 활용하면서, 맥락 전달과 검증 기준을 함께 정리합니다.
사람의 판단이 필요한 제품 흐름과 운영 위험은 확인 가능한 지점으로 남깁니다.

01 / INPUT

같은 원본에서 시작

Figma 링크·수정 위치·개정 내용을 직접 입력으로 연결합니다. 사람이 화면을 다시 설명하는 단계를 줄이고 최신 디자인을 기준으로 작업합니다.

02 / REVIEW

작은 화면부터 전체 흐름까지

Storybook에서 화면·상태를 확인하고 개발환경 앱에서 이동·데이터·실제 사용 흐름을 검증합니다. 테스트와 CI로 회귀를 확인합니다.

03 / REUSE

다음 작업에 남는 기준

반복한 구현 규칙과 검토 방법을 Skill·테스트·배포 작업으로 축적합니다. 개인의 AI 작업 환경과 팀에서 사용하는 협업 도구는 구분해 운용합니다.

개인 프로젝트 / AI REELS GENERATOR

블로그 글을 짧은 영상으로 바꾸는 제작 도구입니다.

문제	완성 영상에서야 구성을 확인하면 수정할 때 음성과 영상을 다시 만들어야 합니다.
해결	글에서 장면 명세와 스토리보드를 만들고, 검토 후 음성·영상 렌더로 이어지게 했습니다.
선택 이유	수정할 지점을 렌더 전에 확인하도록 스토리보드 검토를 음성 생성보다 앞에 배치했습니다.
결과	레이아웃 검증과 제한된 재시도 루프를 적용하고, 단계별로 검토할 수 있는 제작 흐름을 구현했습니다.

[제작 기록 >](#)

풀스택 개발자로서 만들어가고 싶은 것

사용자의 문제를 풀고, 오래 쓰이는 제품을 만들겠습니다.

사용자와 운영자가 겪는 문제를 이해하고,
화면·API·데이터를 하나의 흐름으로 설계하겠습니다.

배포 이후의 오류와 피드백까지 책임지며,
AI를 활용한 실험을 팀과 함께 안정적으로 운영할 수 있는 제품으로 발전시키고 싶습니다.

주도현 Dohyeon Ju

syate@naver.com ↗

포트폴리오

portfolio.dohyeon.kr ↗

GitHub

github.com/dohyeon-kr ↗

개발 기록

blog.dohyeon.kr ↗