

기우경 | Frontend Developer

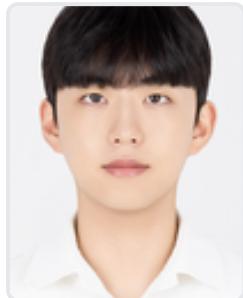
AI를 활용해 사용자 경험을 설계하는 프론트엔드 크리에이티

CONTACT

📞 010-3682-4354

✉ dnrud4354@gmail.com

🔗 github.com/kiwookyung



Q. 사용자가 계속 쓰고 싶어지는 서비스를 어떻게 만들 수 있을까?

저는 ‘사용자가 어떤 상황에서 이 화면을 쓰게 되는지’를 먼저 정의하고, 그 흐름을 끊지 않는 것을 가장 중요하게 생각합니다. FloodGuard에서는 센서 데이터·지도·차트를 한 화면에서 확인할 수 있도록 배치해 사용자가 반복적으로 모니터링하도록 만들었고, ToGather에서는 “홈 → 다이어리 작성 → 피드 타임라인” 흐름을 설계해 기록과 회고가 자연스럽게 이어지도록 했습니다. 이런 경험을 바탕으로, 화면 배치와 인터랙션을 설계할 때 항상 사용자 여정과 반복 사용 동기를 함께 고민하고 있습니다.

Q. 프론트엔드 개발에서 가장 중요하게 생각하는 점은 무엇인가요?

저는 ‘일관된 UI와 예측 가능한 상태 관리’가 좋은 사용자 경험을 만든다고 생각합니다. OrakGarak에서는 페이지마다 디자인과 상태 관리 방식이 제각각이어서 유지보수와 협업에 어려움이 있었는데, 공통 컴포넌트와 테마를 정리하고 상태를 도메인별로 분리하면서 화면 전환과 동작이 한눈에 이해되도록 개선했습니다. 이 과정에서 Zustand, Pinia, TanStack Query를 각각 로컬 상태·글로벌 상태·서버 상태에 맞게 역할을 나눠 사용해, 코드 구조와 사용자 경험의 일관성을 함께 가져가고자 했습니다.

Q. 향후 어떤 프론트엔드 개발자로 성장하고 싶나요?

웹과 모바일을 모두 이해하는 프론트엔드 아키텍트가 되는 것이 목표입니다. FloodGuard에서는 실시간 대시보드와 데이터 시각화를, ToGather에서는 모바일 네비게이션 구조와 감성적인 UX를 설계했습니다. 앞으로는 성능 최적화와 접근성, 컴포넌트 아키텍처에 더 깊이 투자하여, 다양한 서비스에서 안정적인 사용자 경험을 제공하는 개발자로 성장하고자 합니다.

2025 Project Summary

ToGather p.4-5

커플 전용 공유 다이어리 & 일정 관리 모바일 앱 (React Native + Expo)

Mobile Frontend Lead Team Project

2025.10 - 2025.11

OrakGarak p.8-9

음성 분석과 데이터 기반 음악 추천 플랫폼 (React 18 + TypeScript)

Web Frontend Team Project

2025.08 - 2025.09

FloodGuard p.6-7

AIoT 기반 실시간 침수 예방 대시보드 웹 서비스 (React + WebSocket)

Web Frontend Real-time Dashboard

2025.07 - 2025.08

기우경 | Frontend Developer

SKILLS

React

컴포넌트 재사용 구조 설계 및 Atomic 기반 화면 구성 경험.
Zustand·TanStack Query를 활용해 도메인 기반 상태 분리 및 서버 상태 관리 적용. FloodGuard에서 WebSocket 실시간 데이터 연동 및 비동기 UI(Suspense) 구성 경험. 복잡한 UI를 모듈화하여 유지보수성 향상



React Native

Expo Router 기반 탭/스택 네비게이션 전체 구조 설계. Diary, Feed, Plan 등 핵심 화면 90% 이상 직접 구현. SafeArea, Animated, FlatList 최적화 등 모바일 전용 UX 처리 경험. 이미지/텍스트 혼합 레이아웃 컴포넌트 구조화



Vue 3

Composition API + Pinia 기반 SPA 구조 설계 및 상태 모듈화. 공통 컴포넌트·레이아웃 시스템 구성. REST API 연동 및 페이지 라우팅 최적화 경험. 직관적인 UX 흐름을 위한 UI 구조 재정립



TypeScript / JavaScript

타입 안전한 API Layer 구축, Axios 비동기 처리 타입 안전성 확보. 컴포넌트/상태/모델 타입 분리로 유지보수성 강화. ES6+ 문법 활용한 로직 단순화 및 구조화. React Native·React·Vue 전반에서 TS 기반 프로젝트 경험



Tailwind CSS & MUI

Tailwind 기반 반응형 레이아웃 및 UI 토큰 시스템 설계. MUI 컴포넌트 커스터마이징 및 라이트/다크 테마 적용. Figma 디자인 요구사항을 코드로 정확히 재현. 일관된 Design System을 위한 컴포넌트 스타일 가이드 구축



상태 관리 (Zustand / Pinia / TanStack Query)

Zustand로 로컬/전역 상태 분리 및 Store 모듈화. TanStack Query 기반 캐싱·리페치·서버 상태 관리 전략 설계. Pinia를 활용한 Vue SPA의 도메인 기반 상태 설계. 공동 작업에서도 유지보수 쉬운 구조로 확장 경험



실시간/외부 API 연동

WebSocket 기반 실시간 스트림(FloodGuard CCTV·센서 데이터) 처리. REST API 인증(JWT), 에러 핸들링, 리퀘스트 인터셉터 구성. AI 이미지/텍스트 분석 API 통합 경험 (오디오 분석, 이미지 분석 등). 비동기 데이터 흐름을 고려한 렌더링 최적화



Git / GitHub

Git Flow 기반 협업 및 체계적인 브랜치 전략 운영. PR/MR 코드 리뷰, conflict 해결 경험 다수. 릴리즈/태그 관리, 기능 단위 커밋 규칙 정립. 팀 내 코드 품질 일관성을 위한 컨벤션 구축



협업 도구 (Figma, Postman, VS Code)

Figma 시안을 기반으로 UI/UX 요소를 1:1 수준으로 구현. Postman으로 API 명세 검증·Mock 서버 활용. Prettier/Eslint 기반 코드 스타일 통일. 팀원과의 피드백 루프를 빠르게 돌리기 위한 효율적인 협업 환경 구축



기우경 | Frontend Developer

Education

조선대학교

수학과

수학적 사고와 문제 해결력을 바탕으로 데이터 분석 및 알고리즘 이해를 확장.

- 해석학·선형대수·통계학 기초 이수
- 모델링 및 논리적 문제 해결 역량 강화

2019.03 - 2025.02

SSAFY (삼성 청년 SW AI 아카데미) 13기

2025.01 - 2025.12

SW 개발자 과정

SW 교육과정 및 프로젝트를 통해 실무 경험.

- 개발 과정에서의 프론트, 백엔드, AI 등 다양한 분야에서 경험을 쌓고 있습니다.
- ReeLoom · FloodGuard · OrakGarak 등 팀 프로젝트 수행

Awards

2024-01

우수상

전남대학교 산학협력단

제품 제작 역량 강화 캠프 내 우수 아이디어/기획 발표로 수상

Certificates

2024-05

6시그마 Green Belt 실무자

커넥트 직무센터

프로세스 개선 및 데이터 기반 품질관리 실무자 과정 이수

2025-11

SSAFY 자율 프로젝트 우수상

삼성 청년 SW 아카데미

React Native 기반 커플 AI 다이어리 앱 'ToGather' 개발

- 로 SSAFY 자율 프로젝트 평가에서 우수상 수여. 전체 기획 · UI 구성 및 핵심 기능 개발을 주도하며 높은 완성도로 인정 받음.

Activities

2023.07 -

조선대학교 산학협력단

2023.08

디지털 새싹 캠프 - SW·AI 보조 강사

초·중등학생 대상 디지털 인재 양성 캠프에서 SW·AI 분야 보조 강사로 참여.

- Python, 인공지능(TensorFlow, Teachable Machine), 메타버스 콘텐츠 제작 등 실습형 교육 과정 기획 및 지도
- 학습자의 수준에 맞춘 콘텐츠 설명과 실습 지원, 결과물 피드백 제공
- 현장 강의자료 제작 및 기술 시연을 통해 실전 교육 커뮤니케이션 능력 향상

2024.01 -

전남대학교 창업지원단

2024.01

호남·제주권 제품 제작 역량 강화 캠프 - 창업 아이디어 기획

지역 문제 해결을 위한 창업 아이디어 발굴 및 비즈니스 모델 기획 캠프 참가.

- 팀 프로젝트로 IT 기반 창업 모델 구체화 및 비즈니스 모델 캔버스 설계
- 시장·경쟁사 분석 및 고객 타깃 정의를 기반으로 IR 발표 자료 제작
- 창업 멘토링을 통한 피드백 반영 및 실현 가능성 중심 피칭 발표 수행

ToGather

커플 전용 공유 다이어리 & 일정 관리 모바일 앱



기간: 2025.10 - 2025.11

팀 규모: 6명 (Frontend 1, Backend, AI/ML, DevOps)

역할: 프론트엔드 개발 (모바일 앱 전체 구조 설계, Expo Router 기반 탭 구조 설계, 핵심 화면 90% 이상 구현)

GitHub: <https://github.com/SSAFY-ToGather/ToGather>

역할

- Expo Router 기반 탭 구조(Home / Diary / Feed / Plan / Settings) 설계 및 네비게이션 플로우 구현
- 다이어리, 피드, 일정, 펫 등 핵심 화면 90% 이상 컴포넌트 설계 및 개발
- Zustand + TanStack Query로 도메인별 상태 관리 구조 설계 및 적용

특징

커플이 함께 쓰는 공유 다이어리 앱으로, 날짜 기반 기록·타임라인·공유 일정·펫 시스템을 하나의 흐름으로 묶은 UX 설계

같은 캘린더 컴포넌트를 다이어리와 일정 탭에서 각각 다른 맵으로 활용해, '기록 중심'과 '계획 중심' 화면을 분리해 구현

펫/아이템 시스템을 통해 기록이 누적될수록 캐릭터가 성장하는 경험을 제공하여 사용 동기를 설계

Preview



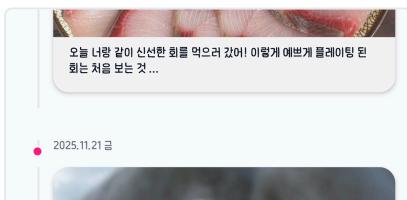
Home – 오늘의 다이어리와 일정, 펫 상태를 한 화면에서 확인



Diary – 날짜별로 사진·텍스트·위치를 섹션 단위로 기록



Plan – 공유 일정 관리를 위한 커플 캘린더



Feed – 다이어리 기록을 카드 타임라인 형태로 모아보는 피드



Pet – 기록에 따라 성장하는 커플 펫



Shop – 펫에게 장착할 아이템을 구매하는 상점 화면

개발 환경

Mobile:

React Native, Expo, TypeScript, Expo Router

State & Data:

Zustand, TanStack Query, Axios, JWT

UI & UX:

React Native Image, FlatList, SafeArea, Animated 등으로 리스트/이미지/전환 애니메이션 최적화

주요 성과

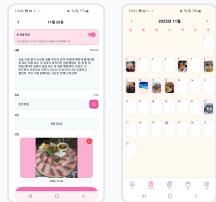
- Expo Router 기반 탭 구조(Home / Diary / Feed / Plan / Settings) 설계 및 전체 앱 네비게이션 플로우 구현
- Zustand + TanStack Query 기반 상태 · 서버 데이터 관리 체계 구축으로 도메인별 독립적인 데이터 동기화 구조 설계
- 다이어리 작성 플로우: 달력 날짜 선택 → 사진/텍스트/위치 정보를 섹션 단위로 구성하는 UI 컴포넌트 설계 및 작성/수정/삭제 기능 구현
- 피드(Feed) 타임라인: 날짜순 다이어리 피드를 카드 형태로 구성하고, 사진 개수에 따른 자동 그리드 레이아웃 생성 및 FlatList 기반 스크롤 성능 최적화
- 일정(Plan) 공유 캘린더: 다이어리와 동일한 캘린더 라이브러리를 사용하여 기록 중심과 계획 중심의 UX 맥락에 맞게 화면을 다르게 구성하여 구현
- 펫 & 아이템 시스템: 커플이 함께 키우는 펫 UI 구성, 상점에서 아이템 구매 → 펫에게 장착하는 흐름 구현 및 장착 정보를 Home 등 주요 화면과 연동

트러블슈팅

SafeArea / KeyboardAvoidingView 관련 UI 깨짐 문제

문제: iOS/Android 기종마다 달력/피드/작성 화면이 노치·홈 버튼·키보드 때문에 밀려 UI가 잘리는 문제가 발생했습니다.

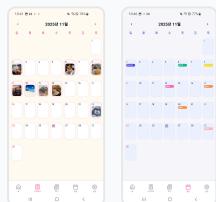
해결: React Native의 SafeAreaView와 useSafeAreaInsets 툴을 활용하여 노치와 홈 버튼 영역을 정확히 계산하고, KeyboardAvoidingView의 behavior 속성을 플랫폼별로 분기 처리하여 iOS는 padding, Android는 height로 설정했습니다. 또한 ScrollView와 함께 사용하여 키보드가 올라올 때 컨텐츠가 자연스럽게 스크롤되도록 구현하고, 각 화면의 헤더와 탭바 위치를 safe area와 키보드 높이를 고려하여 재계산하여 모든 기종에서 안정적인 UI를 제공했습니다.



캘린더 데이터 동기화 이슈

문제: 일정과 다이어리가 같은 캘린더 컴포넌트를 사용하면서 데이터 충돌이 발생했습니다.

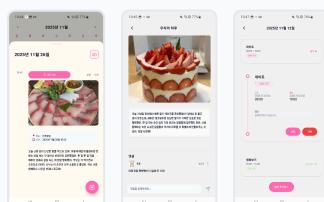
해결: Query key를 도메인별로 분리하고 의존 관계를 명확히 분리하여 다이어리와 일정이 서로 영향을 주지 않는 안정적인 구조로 개선했습니다.



RN 네비게이션 & 상태 사라짐 문제

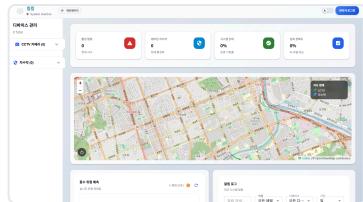
문제: 상세 화면에서 뒤로가기 시 화면 상태가 초기화되거나 누락되는 문제가 발생했습니다.

해결: useFocusEffect로 화면 재진입 시 상태 관리를 분리하고, 전역·로컬 상태의 역할을 정의하여 화면 전환이 자연스럽고 사용자 경험이 안정화되도록 구현했습니다.



FloodGuard

AIoT 기반 실시간 침수 감지·예측·자동 제어 대시보드
웹 서비스



기간: 2025.07 - 2025.08

팀 규모: 6명 (Frontend 1, Backend 1, AI/IoT 4)

역할: 프론트엔드 개발 (React 대시보드 구현, 실시간 데이터 시각화, UI/UX 설계)

GitHub: <https://github.com/kiwookyung/floodguard>

역할

- React와 MUI를 활용한 실시간 모니터링 대시보드 구현 및 6명 팀에서 프론트엔드 단독 담당
- WebSocket을 통한 실시간 데이터 통신 및 자동 재연결 로직 구현
- Leaflet 지도를 활용한 지리적 데이터 시각화 및 인터랙티브 UI 구현

특징

AIoT 기반 침수 예방 시스템의 웹 대시보드로, 실시간 센서 데이터 모니터링과 자동 제어 기능 제공

WebSocket을 통한 실시간 데이터 스트리밍 및 Leaflet 지도를 활용한 센서 위치 시각화

Web Worker와 requestAnimationFrame을 활용한 렌더링 최적화로 초당 30-50개의 센서 데이터도 부드럽게 처리

Preview



물이 차오를 때 센서가 침수를 감지하는 과정



센서 신호를 받아 차수막이 자동으로 닫히는 과정



차수막이 완전히 닫혀 침수를 방지한 최종 상태

개발 환경

Frontend:

React, Vite, MUI, Tailwind CSS

State & Data:

Zustand, Recharts, Leaflet, WebSocket, JWT

Performance:

Web Worker, requestAnimationFrame, React.memo를 통한 렌더링 최적화

주요 성과

- React와 MUI를 활용한 직관적인 실시간 모니터링 대시보드 구현 및 6명 팀에서 프론트엔드 단독 담당
- WebSocket을 통한 실시간 데이터 통신 및 자동 재연결 로직 구현으로 안정적인 실시간 스트리밍 환경 구축
- Leaflet 지도를 활용한 지리적 데이터 시각화 및 인터랙티브 UI 구현으로 센서 위치를 직관적으로 표시
- Web Worker와 requestAnimationFrame을 활용한 렌더링 최적화로 지도와 차트 동시 업데이트 시에도 60fps 유지
- Zustand 상태 관리 최적화, 데이터 버퍼링 및 디바운싱, React.memo를 통한 리렌더링 방지로 초당 30-50개의 센서 데이터도 부드럽게 처리

트러블슈팅

WebSocket 연결 안정성 문제

문제: 실시간 센서 데이터 수신 중 네트워크 불안정 시 연결이 끊어지고 재연결이 되지 않는 문제가 발생했습니다.

해결: WebSocket 연결 상태 모니터링 및 자동 재연결 로직을 구현하고, 연결 실패 시 exponential backoff 전략을 적용하여 안정적인 실시간 데이터 스트리밍 환경을 구축했습니다.

대용량 실시간 데이터 렌더링 성능

문제: 초당 30-50개의 센서 데이터가 들어오면서 React 컴포넌트가 과도하게 리렌더링되어 UI가 멈추는 현상이 발생했습니다.

해결: Zustand를 활용한 상태 관리 최적화, 데이터 버퍼링 및 디바운싱 적용(500ms), React.memo를 통한 불필요한 리렌더링 방지로 부드러운 실시간 대시보드를 구현했습니다.

JWT 토큰 관리 및 인증 구조

문제: 토큰 만료 시 사용자가 갑자기 로그아웃되고, 만료 전 갱신 로직이 없어 사용자 경험이 저하되는 문제가 발생했습니다.

해결: Axios 인터셉터를 활용한 자동 토큰 갱신 로직을 구현하고, refresh token을 통한 seamless 인증 흐름을 설계하여 사용자 경험을 개선했습니다.

Leaflet 지도와 Recharts 차트 성능 병목

문제: 지도와 차트가 동시에 업데이트되면서 메인 스레드가 블로킹되어 애니메이션이 끊기는 문제가 발생했습니다.

해결: Web Worker를 활용한 데이터 처리 분리, requestAnimationFrame을 통한 렌더링 최적화, 지도/차트 업데이트를 독립적으로 분리하여 각각 60fps를 유지했습니다.

OrakGarak

빅데이터 기반 음성 맞춤형 음악 추천 플랫폼



기간: 2025.08 - 2025.09

팀 규모: 6명 (Frontend 2, Backend 3, Data/AI 1, DevOps 1)

역할: 프론트엔드 개발 (React/TypeScript 기반 UI 구현, 오디오 처리, 인터랙티브 UX 설계)

GitHub: <https://github.com/kiwookyung/orakgarak>

역할

- React 18과 TypeScript를 활용한 타입 안전한 컴포넌트 아키텍처 설계
- Web Audio API를 통한 실시간 음성 녹음 및 시각화 기능 구현
- Framer Motion을 활용한 3D 캐러셀 및 인터랙티브 애니메이션 구현

특징

음성 분석 기반 음악 추천 플랫폼으로, Web Audio API를 통한 실시간 오디오 처리와 Framer Motion을 활용한 인터랙티브 UI 제공

사용자 친화적인 음성 녹음 인터페이스와 3D 캐러셀을 통한 추천 결과 시각화

Zustand 도메인별 분리 및 TanStack Query를 통한 서버 상태 독립 관리로 복잡한 상태 관리 구조 개선

Preview



음성 녹음 – Web Audio API를 활용한 실시간 음성 녹음 및 시각화



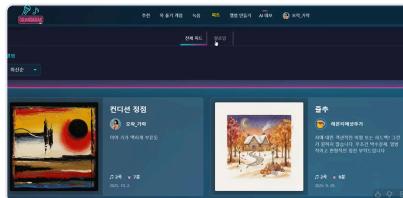
추천 결과 – 3D 캐러셀을 통한 음악 추천 결과 시각화



추천 상세 – Framer Motion을 활용한 인터랙티브 애니메이션



앨범 – 사용자별 음악 앨범 관리



피드 – 커뮤니티 피드 및 음악 공유



트랙리스트 – 플레이리스트 및 재생 목록 관리

개발 환경

Frontend:

React 18, TypeScript, Vite, TailwindCSS, MUI

State & Data:

Zustand, TanStack Query, Axios, JWT

UI & UX:

Framer Motion, Web Audio API를 통한 실시간 오디오 처리 및 인터랙티브 애니메이션

주요 성과

- React 18과 TypeScript를 활용한 타입 안전한 컴포넌트 아키텍처 설계 및 팀 프로젝트에서 프론트엔드 2명 중 1명으로 참여
- Web Audio API를 통한 실시간 음성 녹음 및 시각화 기능 구현 및 크로스 브라우저 호환성 확보 (Chrome, Safari, iOS Safari)
- Framer Motion을 활용한 3D 캐러셀 및 인터랙티브 애니메이션 구현 및 will-change와 transform 속성을 활용한 GPU 가속으로 60fps 유지
- Zustand 도메인별 분리 및 TanStack Query를 통한 서버 상태 독립 관리로 복잡한 상태 관리 구조 개선
- 요청 큐 시스템을 통한 JWT 토큰 갱신 중 동시 요청 처리로 안정적인 인증 흐름 구현

트러블슈팅

Web Audio API 브라우저 호환성 문제

문제: Chrome에서는 정상 작동하지만 Safari에서 음성 녹음이 되지 않고, iOS Safari에서는 권한 요청이 제대로 처리되지 않는 문제가 발생했습니다.

해결: 브라우저별 API 차이를 감지하고 폴리필을 적용하며, getUserMedia API의 브라우저별 구현 차이를 처리하는 래퍼 함수를 작성하여 크로스 브라우저 호환성을 확보했습니다.



JWT 토큰 갱신 중 동시 요청 처리

문제: 토큰 만료 시 여러 API 요청이 동시에 발생하면 refresh token API가 중복 호출되어 401 에러가 연쇄적으로 발생했습니다.

해결: 요청 큐 시스템을 구현하여 토큰 갱신 중인 요청들을 대기시키고, 갱신 완료 후 일괄 처리하는 방식으로 중복 호출을 방지하고 안정적인 인증 흐름을 구현했습니다.



3D 캐러셀 성능 최적화

문제: Framer Motion으로 구현한 3D 캐러셀에서 많은 음악 카드가 렌더링되면서 스크롤 시 프레임이 떨어지는 문제가 발생했습니다.

해결: will-change CSS 속성 적용, GPU 가속을 위한 transform 속성 활용, 가상화를 통한 보이지 않는 카드 렌더링 제거로 60fps를 유지하고 부드러운 애니메이션을 구현했습니다.



복잡한 상태 관리 구조

문제: 음성 녹음, 추천 결과, 플레이리스트 등 여러 도메인의 상태가 얹혀 있어 데이터 동기화가 어려운 문제가 발생했습니다.

해결: Zustand를 도메인별로 분리하고, TanStack Query를 통해 서버 상태를 독립적으로 관리하며, 각 store 간의 의존성을 최소화하는 구조로 리팩토링하여 유지보수성을 향상시켰습니다.

