김봄 | 프론트엔드

Channel

Email | Portfolio | GitHub | LinkedIn | Blog

Summary

React·Next.js·TypeScript를 기반으로 **사용자 경험과 이를 지탱하는 프론트엔드 구조를 함께 설계**하고, 운영 피드백과 **성능·행동 데이터로 개선을 반복**하는 **프론트엔드 개발자**입니다.

- 약 280명이 사용하는 교내 서비스의 관리자 웹을 구축·유지보수하며, 데이터 변경 주기에 맞춘 캐시 기준을 재설계해 최신 정보는 유지하면서, 동일 탐색 시나리오의 API 호출 횟수를 45회 → 7회, 84% 줄였습니다.
- 이미지 로딩 병목을 개선해 모바일 Lighthouse 성능을 86점 → 99점으로 높이고, LCP를 4.1초 → 1.0초로 단축했습니다.
- 현직자 피드백을 반영해 프로젝트 탐색 구조를 재설계하고, Amplitude로 877명의 고유 사용자 행동을 분석해 Contact 도달률을 39.6% → 48.3%(+8.7%p)로 개선했습니다.

Main Projects

Washer_Client v2

2025.10 - 2026.04 · 유지보수 중

Next.js, React, TypeScript, Tailwind CSS, TanStack Query, Zustand, Zod, / Architecture: FSD

배포 링크 →

GitHub →

기수 간 인수인계로 운영되는 광주소프트웨어마이스터고 세탁기 관리 시스템의 v2 클라이언트입니다.

200명 이상의 전교생을 대상으로 운영되는 서비스로, 사용자가 세탁기 상태를 확인하고 예약 및 신고 흐름을 사용할 수 있도록 구성했습니다. 실제 운영 환경에서 유지보수까지 이어가고 있습니다.

FrontEnd: 2명, Android: 2명, BackEnd: 2명, Design: 1명

프로젝트 상세 보기 →

핵심 성과

- 운영 대상이 약 210명에서 280명으로 33% 확대되는 시점에, 페**이지 이동·재진입마다 같은 데이터를 반복 요청**하는 문제 발생. 데이터 변경 주기에 맞춰 캐시 기준을 재설계해 동일 탐색 시나리오의 **API 요청 45회에서 7회로 84%** 감소
- API 호출·필터·모달·상태 관리·UI가 집중된 **1,603줄의 단일 관리자 페이지를 5개 화면·5개 도메인 구조로 분리**해 기능별 책임과 변경 범위를 명확화
- 5개 도메인의 7개 주요 API 응답·70개 DTO 필드에 런타임 검증을 적용하고, nullable 불일치 **5건 수정** 및 조회·변경 API 12개에서 발생하는 오류를 **9개 유형으로 정규화**

주요 기여

관리자 운영 정보 통합 및 조회·조치 흐름 구축

[상황]
- 기기 이상이나 사용 중 문제가 발생하면 학생이 기숙사 사관선생님에게 직접 전달하고, 관리자는 필요한 내용을 수기로 확인하며 대응
- 기기 상태·예약·사용자 패널티·고장 신고 정보가 분산되어 문제 대상을 찾은 뒤 관련 정보를 함께 확인하기 어려웠음
- 신입생 유입으로 운영 대상이 약 **210명** → **280명**으로 확대되며 기존 관리 방식의 효율 개선 필요

[수행]
- 대시보드·기기·예약·사용자·고장 신고를 관리하는 **5개의 관리자 화면** 구축
- 관리자가 단순 조회에서 끝나지 않고 **문제 대상 특정 → 관련 정보 확인 → 예약 취소·기기 상태 변경·패널티 해제·신고 처리**까지 이어갈 수 있도록 화면별 액션 연결
- 층·기기 종류·처리 상태별 검색·필터로 관리 대상을 좁히고, 기기와 고장 신고를 `machineId` 기준으로 연결해 해당 범위의 신고까지 함께 확인할 수 있도록 구성
- 현재 예약을 보던 맥락을 유지하면서 특정 기기의 이전 이용 내역을 확인할 수 있도록 History 패널로 예약 이력 연결
- Discord를 통해 운영 관리자와 상시 피드백을 주고받으며 실제 운영 과정에서 발견된 불편 사항을 지속적으로 개선

[결과]
- 현장 방문 없이 관리자 웹에서 **기기 상태와 예약·고장 신고 정보를 원격으로 확인하고 후속 조치까지 수행할 수 있는 운영 흐름 구축**
- 분산된 운영 정보를 기기와 층 기준으로 연결해 전체 목록을 반복 탐색하지 않고 문제 범위를 빠르게 좁힐 수 있도록 개선
- 약 **280명 규모의 운영 대상에 대한 기기·예약·신고·패널티 관리**를 하나의 관리자 서비스에서 처리할 수 있는 환경 구축
- Discord 기반 운영자 피드백을 유지보수 과정에 반영하는 반복 개선 구조 마련

관리자 화면 구조 및 컴포넌트 경계 재설계

[상황]
- V1 관리자 페이지가 **1,603줄까지 증가**하며 API 호출·데이터 변환·모달 상태·사용자 액션·UI 렌더링이 한 파일에 집중
- 유사한 화면을 일괄 공통화할 경우 도메인별 API와 업무 흐름까지 범용 컴포넌트에 결합될 위험이 존재

[수행]
- 세탁기·건조기처럼 **데이터만 다르고 상태 확인·변경 흐름이 같은 기능은 동일 컴포넌트로 재사용**
- 예약 이력·기기 관리처럼 외형은 비슷하지만 API와 사용자 행동이 다른 기능은 별도 도메인 컴포넌트로 유지
- 도메인과 관계없이 반복되는 패널 외형은 `StatusPanelShell`, 액션 배치는 `StatusRowActions`로 분리해 **공통 UI와 도메인 기능의 책임을 구분**

[결과]
- 1,603줄의 단일 관리자 페이지를 5개 화면·5개 도메인 구조로 분리
- 화면의 유사성이 아니라 **변경 이유와 사용자 행동을 기준으로 컴포넌트 경계 설정**
- 반복 UI의 수정 지점을 공통 컴포넌트로 모으고, **도메인별 기능 변경이 다른 관리 영역으로 전파되는 범위 축소**

서버 상태 캐시 및 요청 구조 최적화

[상황]
- 대시보드·기기·예약·신고·사용자 화면을 이동·재진입할 때 동일 데이터를 반복 요청했고, 고정된 관리자 탐색 시나리오에서 **API 요청 45회 발생**
- 기기 상태처럼 자주 변경되는 데이터와 관리자 정보처럼 거의 변경되지 않는 데이터에 **별도의 신선도 기준 없이 동일한 캐시 정책을 적용**

[수행]
- 페이지의 직접 API 호출을 제거하고 HTTP Client → 도메인 API 함수 → TanStack Query Hook으로 데이터 흐름 분리
- 도메인별 Query Key를 중앙화하고 기기·예약·사용자 등 데이터 변경 빈도에 따라 `staleTime`을 구분
- 예약 삭제·기기 상태 변경·패널티 해제 이후 **관련 Query만 무효화해 변경된 데이터만 다시 조회하도록 갱신 범위 설정**

[결과]
- 동일 관리자 탐색 시나리오의 API 요청 **45회 → 7회, 약 84% 감소**
- 기기·예약처럼 변경 빈도가 높은 데이터의 최신성을 유지하면서, 변경이 적은 데이터의 불필요한 반복 요청 최소화
- 데이터의 변경 주기와 실제 갱신 시점을 기준으로 서버 상태 관리 정책 정립

API 응답 검증 및 데이터 경계 재설계

[상황]
- TypeScript 제네릭으로 서버 응답 타입을 지정했지만, 실제 API 연동 과정에서 nullable 필드와 선언 타입이 일치하지 않는 사례 발견
- 런타임 검증 없이 응답이 이후 로직으로 전달되어 필드 누락·타입 불일치·정의되지 않은 상태값을 API 진입 단계에서 감지하기 어려웠음

[수행]
- API 응답을 unknown으로 수신한 뒤 Zod Schema를 적용해 **5개 도메인의 7개 주요 API 응답·70개 DTO 필드를 검증**
- Schema에서 TypeScript 타입을 추론해 **타입 선언과 런타임 검증 규칙을 단일화**
- 검증된 DTO만 Mapper와 UI에 전달하고, 응답 구조 오류와 관리자가 처리해야 하는 정상 상태값을 구분
- DTO와 UI Model의 분리 기준을 프론트엔드 팀원과 논의해 별도 변환이 필요한 경우에만 Mapper를 적용하고, 그렇지 않은 경우 검증된 DTO를 그대로 사용하도록 기준 설정

[결과]
- 필드 누락·타입 불일치·정의되지 않은 상태값을 **API 진입 단계에서 감지**하고, 실제 응답과 선언이 달랐던 **nullable 불일치 5건 수정**
- 검증되지 않은 외부 데이터가 Mapper와 UI로 전달되는 흐름 차단
- 타입·검증 기준은 Schema, 화면 표현을 위한 변환은 Mapper로 분리해 서버 응답과 UI 변경 시 확인 지점을 명확화

인증 오류 및 세션 복구 흐름 개선

[상황]
- 내 정보 조회 실패 원인을 구분하지 않고 모든 오류를 로그인 만료로 처리
- 서버·네트워크·응답 검증 오류에서도 토큰과 Query Cache가 삭제되어 **인증과 무관한 오류로 강제 로그아웃될 가능성**이 있었음

[수행]
- 요청 실패 여부만으로 세션을 종료하지 않고, 오류 원인과 복구 가능성을 기준으로 인증·권한·서버·네트워크·응답 검증 등의 **공통 오류 모델로 분류**
- 최초 `401` 응답은 토큰 재발급으로 복구하고, 재발급까지 실패한 경우에만 세션 정리
- 서버·네트워크·응답 검증처럼 **인증과 무관한 오류에서는 토큰과 Query Cache를 유지하고 오류 유형별 안내·재시도 흐름 제공**
- 세션 정리 책임을 `AdminLayout`에서 공통 인증 로직으로 분리

[결과]
- 12개 API 흐름에서 발생하는 오류를 **인증·권한·서버·네트워크·응답 검증 등 9개 유형으로 정규화**
- 인증과 무관한 오류로 인한 불필요한 강제 로그아웃 방지
- 실제 인증 복구가 실패한 경우에만 토큰과 캐시를 정리하도록 **세션 복구 기준 명확화**
- API 계약 오류와 인증 오류를 분리해 원인 추적과 사용자 안내 개선

kimbom.dev

2025.08 - 2026.06 · 유지보수 중

Next.js · React · TypeScript · GSAP · ScrollTrigger · Lenis · SCSS · Amplitude

배포 링크 →

GitHub →

제가 어떤 문제에 관심을 가지고 어떤 방식으로 만드는 사람인지 **하나의 웹 경험으로 전달**하기 위해 **직접 기획·디자인·개발**한 개인 웹사이트입니다.

정보 구조와 인터랙션부터 프론트엔드 구현까지 직접 연결하고, 배포 이후에는 현업자들의 피드백과 **사용자 행동·성능 데이터를 기반**으로 경험과 구조를 지속적으로 개선하고 있습니다.

Planning · Design · Frontend Development | 1명

프로젝트 상세 보기 →

핵심 성과

- 되감기·상태 유지·Pin 전환 등 서로 다른 스크롤 동작을 **공통 lifecycle**로 재설계해 **Animation의 실행·정리 흐름을 표준화**
- 초기 이미지 로딩 우선순위 경쟁을 추적해 리소스 요청 구조를 개선하고, 모바일 **Lighthouse 86 → 99 · LCP 4.1초 → 1.0초**
- 현업자들의 피드백을 반영해 프로젝트 탐색 정보 구조를 재설계하고, Amplitude로 877명의 고유 사용자 행동을 분석해 **Contact 도달률 39.6% → 48.3%(+8.7%p)** 개선

주요 기여

프로젝트 탐색 구조 재설계 및 사용자 행동 검증

[상황]
- 긴 스크롤형 포트폴리오 특성상 방문자가 어떤 콘텐츠까지 확인하고 어느 구간에서 이탈하는지 파악하기 어려웠음
- 정성적인 피드백과 단순 방문자 수만으로는 콘텐츠 구조 변경의 필요성과 개선 효과를 판단하기 어려웠음
- 현업자 피드백을 통해 **여러 프로젝트가 하나의 서사 흐름 안에 묶여 있어, 원하는 프로젝트를 빠르게 탐색하기 어렵다는 문제**를 확인

[수행]
- `Build`와 `Projects`를 분리해 **프로젝트를 독립적으로 탐색**할 수 있도록 **IA(정보 구조)** 재설계
- Amplitude Browser SDK를 적용하고, 섹션 도달·외부 링크 클릭 등 **탐색 흐름을 확인**하기 위한 **커스텀 이벤트 설계**
- 섹션이 **화면 중앙 20% 영역에 진입한 경우를 도달 기준으로 정의**하고, 섹션명·순서·화면 너비를 함께 수집
- 중복 제거 기준 **877명의 고유 사용자 행동을 분석**하고, 개편 전후 Contact 도달률을 동일한 기준으로 비교해 **IA 변경 효과 검증**

[결과]
- 정보 구조 개편 이후 **Contact 도달률 39.6% → 48.3%, +8.7%p 개선**
- 정성적 피드백 중심의 개선 방식을 **피드백 → 구조 변경 → 사용자 행동 검증으로 이어지는 반복 개선 방식**으로 전환

서로 다른 스크롤 동작을 예측 가능한 실행 구조로 재설계

[상황]
- 각 Stage에는 스크롤 방향에 따라 되감기는 장면, 한 번 노출된 상태를 유지하는 장면, 특정 구간을 화면에 고정해 콘텐츠를 전환하는 장면 등 **서로 다른 방식의 인터랙션**이 존재
- 여러 Stage와 Scene의 DOM 탐색 · Animation 생성 · ScrollTrigger 연결 · cleanup이 각 Hook에 혼재해, **Animation의 실행 순서**와 **수정·정리 위치**를 파악하기 어려웠음

[수행]
- Animation의 동작 자체를 하나로 합치지 않고, 초기화 → 실행 → 정리의 **공통 lifecycle**을 기준으로 실행 구조 재설계
- Stage는 각 장면의 큰 DOM 구간만 수집하고, Scene에서는 내부 Animation에 필요한 세부 요소만 탐색하도록 **DOM 탐색 범위**와 **책임을 분리**
- Stage Hook이 실행 환경에 맞는 Scroll Config를 선택하고, `DOM 수집 → Controller 생성 → ScrollTrigger 연결 → cleanup` 순서로 **Animation 실행 흐름을 조율**하도록 구성
- 되감기·상태 유지·Pin 전환 등 **장면별 실행 규칙은 유지**하되, 서로 다른 동작을 공통 로직에 억지로 통합하지 않도록 분리

[결과]
- 여러 파일이 연결된 구조에서도 DOM 수집부터 Timeline 실행까지 **Animation 실행 흐름을 한 방향으로 추적**할 수 있도록 개선
- 새로운 Animation을 추가할 때 DOM 수집 → Controller 생성 → Trigger 연결 → 정리의 확장 절차를 표준화
- Trigger와 Timeline을 각각 생성한 계층에서 직접 정리하도록 구성해 자원 생성과 cleanup의 책임을 명확화

과도한 이미지 로딩 우선순위로 발생한 LCP 병목 개선

[상황]
- 모바일 Lighthouse 측정에서 **LCP가 4.1초**로 나타나 초기 화면의 주요 콘텐츠 노출이 지연됨
- Chrome Performance 패널로 확인한 결과, 초기 화면의 LifeMotion 이미지가 LCP 요소로 측정됨
- 첫 8개 이미지에 모두 높은 로딩 우선순위를 적용해, 실**제로 필요한 이미지와 나중에 필요한 이미지가 동시에 요청**되고 있었음

[수행]
- Chrome Performance 패널의 LCP 마커와 실제 DOM 요소를 확인해 **이미지 로딩 병목 추적**
- 반복 갤러리 이미지에 적용된 `priority`를 제거하고 **Next.js Image의 기본 지연 로딩 적용**
- 이미지 품질을 75에서 60으로 조정하고, 화면 크기에 맞는 반응형 `sizes` 설정 유지

[결과]
- 모바일 Lighthouse 성능 점수 **86점 → 99점**
- LCP **4.1초 → 1.0초**, 약 **76%** 단축

Other Projects

효잇

2025.09 – 현재

React Native · Expo · TypeScript · TanStack Query · Zustand · pnpm Workspace

GitHub →

프로젝트 상세 보기 →

부모·자녀 역할을 연결한 가족 안부 확인 모바일 서비스 (React Native)
- 부모·자녀 역할을 분리하고 **역할 선택 → 가족 연결 → 역할별 홈**으로 이어지는 사용자 흐름 설계
- Expo Router와 Zustand를 연계해 **인증·역할·온보딩 상태에 따른 라우팅 및 접근 제어** 구현
- TanStack Query로 사용자·알림·가족 연결의 **서버 상태 조회 및 갱신 흐름** 관리
- pnpm Workspace 기반 모노레포로 모바일 앱과 공통 패키지를 분리하고 **역할별 기능과 공통 코드의 책임 경계** 구성
- **팀 리드**로 제품 방향과 기능 우선순위를 정하고 디자인·개발 일정 및 작업 범위 조율

NOVA

2024.03 - 2024.12

React Native · Expo · JavaScript · React Navigation · Axios · OpenAI API

GitHub →

프로젝트 상세 보기 →

전공 탐색부터 학습 실행까지 연결한 개발자 성장 앱
- 학생 사용자 조사를 바탕으로 전공 선택 문제를 탐색 → 선택 → 실행 → 성장 확인의 경험으로 재정의하고 전체 흐름 설계

- **Frontend · Backend · iOS · Android · NOVA 캐릭터 5종을 직접 제작**해 전공을 선택 가능한 시각적 정체성으로 표현
- 설문 응답 기반 추천을 **캐릭터 선택 → 메인 화면까지 상태로 연결**하고, 미션 완료·개발자 단계·활동 보드로 성장 피드백 프로토타이핑
- OpenAI Chat Completions API를 연결해 **사용자 메시지·응답 대기·실패 상태를 반영한 모바일 상담 인터랙션** 구현
- **팀 리드**로 기획·디자인·개발 전반을 조율하고 **광주SW마이스터고 아이디어 페스티벌 최우수상(교내 1위)** 수상

Awards

과학기술정보통신부 장관상 · 전국 SW마이스터고 연합 해커톤 1위

2025.11 | 과학기술정보통신부·정보통신기획평가원(IITP) 주관

- 다문화 가정 학부모의 학교 공문서 이해를 돕는 AI 플랫폼 **ClipCat** 개발
- AI 서비스 UX 설계 및 Next.js 기반 문서 업로드·변환 결과·챗봇 인터페이스 구현

  ClipCat 프로젝트 자세히 보기 →

AppJam 27th 우수상

2024.08 | K-디지털 플랫폼 주관 · 생활정보 부문

- 목표 달성 과정을 퀘스트와 성장 경험으로 구성한 목표 관리 서비스 **Cando** 개발
- 목표 진행률 시각화와 AI 추천 콘텐츠 흐름 설계 및 클라이언트 구현

  Cando 프로젝트 자세히 보기 →

광주SW마이스터고 아이디어 페스티벌 최우수상

2025.01 | 교내 아이디어 페스티벌 1위

- 전공별 미션과 캐릭터 성장으로 관심 분야를 탐색하는 진로 탐색 앱 **NOVA** 개발
- 팀 리드로 서비스 기획·React Native 클라이언트 개발·UI 시스템 및 캐릭터 디자인 담당

  NOVA 프로젝트 자세히 보기 →