

Portfolio

Frontend Developer

01. 직무 관련 경험

- 프로필 ----- p03
- 주요 기술 스택 ----- p04

02. 프로젝트 소개

- 프로젝트 요약 ----- p05
- 행복하개(hangbokdog) ----- p06
- 동행(donghang) ----- p13
- 이집어때(ezip) ----- p19



배움에 두려움이 없는 프론트엔드 개발자 이재백입니다.

🏠 서울특별시 강남구 개포동

📞 010-2395-6787

✉️ dbswirla112@naver.com

Github 링크

Notion 링크

기술 블로그 링크

Military

• 2016.03 ~ 2017.12 육군 병장 만기 전역

제20사단 정보통신대대

🎓 Education / Experience

- 2024.07 ~ 2025.06 삼성 청년 SW·AI 아카데미(SSAFY) 12기
- 2023.01 ~ 2024.05 예비창업 지원 사업 경험
AI 예측 기술 기반 서비스 및 C2C 대여 중개 플랫폼의 초기 기획, UI/UX 설계 및 프로토타입 개발 참여
- 2021.06 ~ 2021.11 코리아IT아카데미 빅데이터 UI 전문가 과정
- 2021.04 - 2022.08 학점은행제 컴퓨터공학과 학사 취득
- 2019.05 ~ 2021.05 서울대학교병원 전산실 (사원)
Bestcare 2.0 병원 전산 시스템 및 PC 유지보수 담당
- 2015.03 ~ 2019.02 용인송담대학교 의료정보과 졸업

🏆 Award / Certificates

- 2025.05 자율 프로젝트 우수상(2등)
삼성 청년 SW·AI 아카데미(SSAFY)
- 2025.03 특화 프로젝트 우수상(1등)
삼성 청년 SW·AI 아카데미(SSAFY)
- 2024.11 관통 프로젝트 최우수상(1등)
삼성 청년 SW·AI 아카데미(SSAFY)
- 2022.06 정보처리기사
- 2022.04 SQL 개발자(SQLD)
- 2018.05 사무자동화 산업기사

01. 직무 관련 경험 주요 기술 스택



- 웹 개발에 필요한 HTML, CSS와 함께 학습 후 프로젝트 개발 경험
- async/await, Promise를 이용한 비동기 통신 및 데이터 처리 이해
- .map(), .filter(), .reduce() 고차 함수를 활용하여 효율적인 데이터 관리 경험



- 타입, 인터페이스, 제네릭 등 좀 더 명확하고 체계적인 코드 작성을 위해 학습 및 프로젝트 개발 경험
- 런타임 에러를 줄이고 코드 안정성을 높이기 위해 프로젝트에 적극 도입



- Git: 협업 코드 관리, MR 브랜치 경험
- Jira: 애자일 이슈 및 스프린트 관리 경험
- Git ↔ Jira 연동을 통한 스마트 커밋 및 개발 패널 사용 경험



- 전반적인 라이프 사이클과 리액트 흑을 이해하고 프로젝트 개발 경험
- 컴포넌트 개념을 이해하고 재사용성과 확장성을 고려한 설계 경험
- Tanstack Query, Zustand, Axios, Tailwind CSS, Shadcn UI 등 라이브러리 사용 경험



- CSR 외 SSR, ISR 등 다양한 렌더링 방식을 적용해보기 위해 학습 후 앱 라우터 기반의 클론 프로젝트 개발 경험

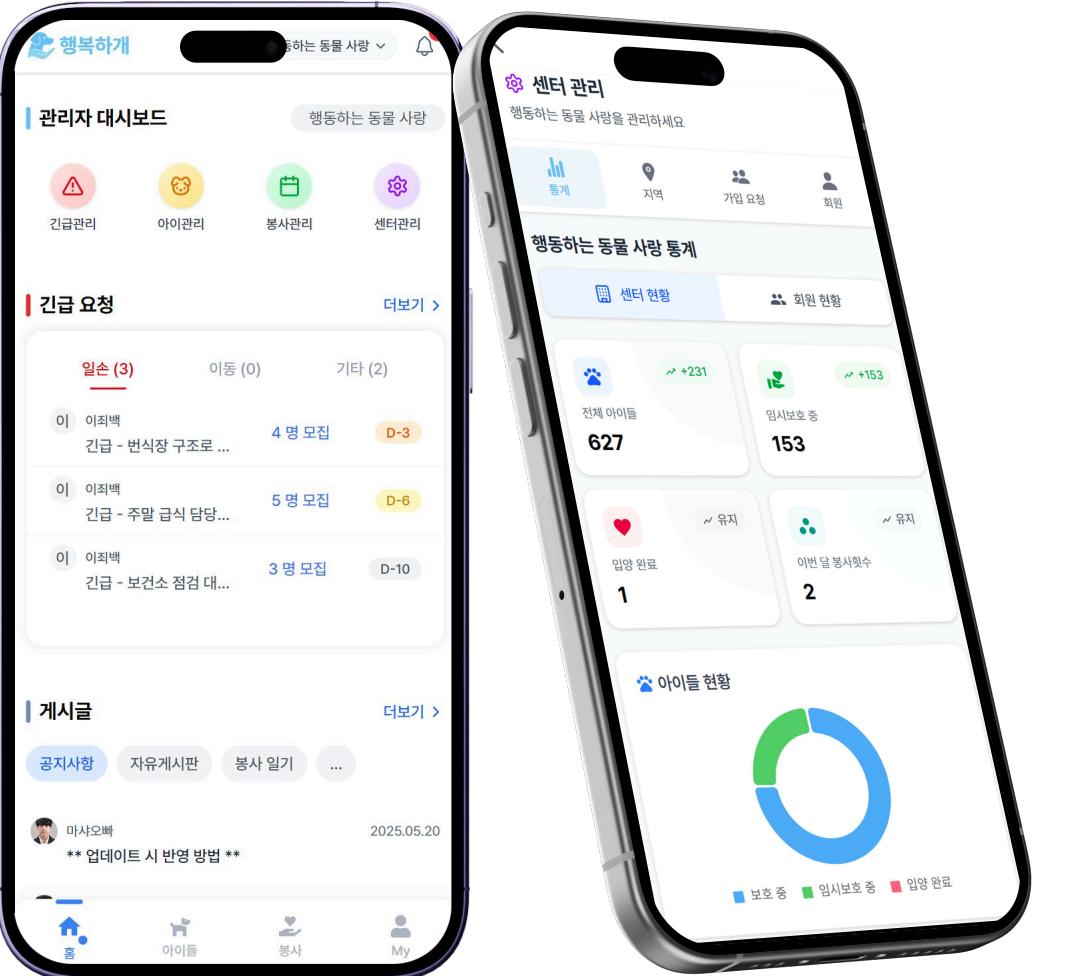


- Auto Layout을 활용한 레이어 구성
- 프로젝트 와이어 프레임, 목업 제작 경험

02. 프로젝트 소개 프로젝트 요약



유기견 보호소의 입양·임보, 봉사, 강아지 관리를 통합해 운영 효율과 소통을 향상시키는 웹앱 플랫폼



- 진행 기간
2025.04.14 ~ 2025.05.22 (6주)
- 진행 인원
4명(FE: 2명, BE: 2명)



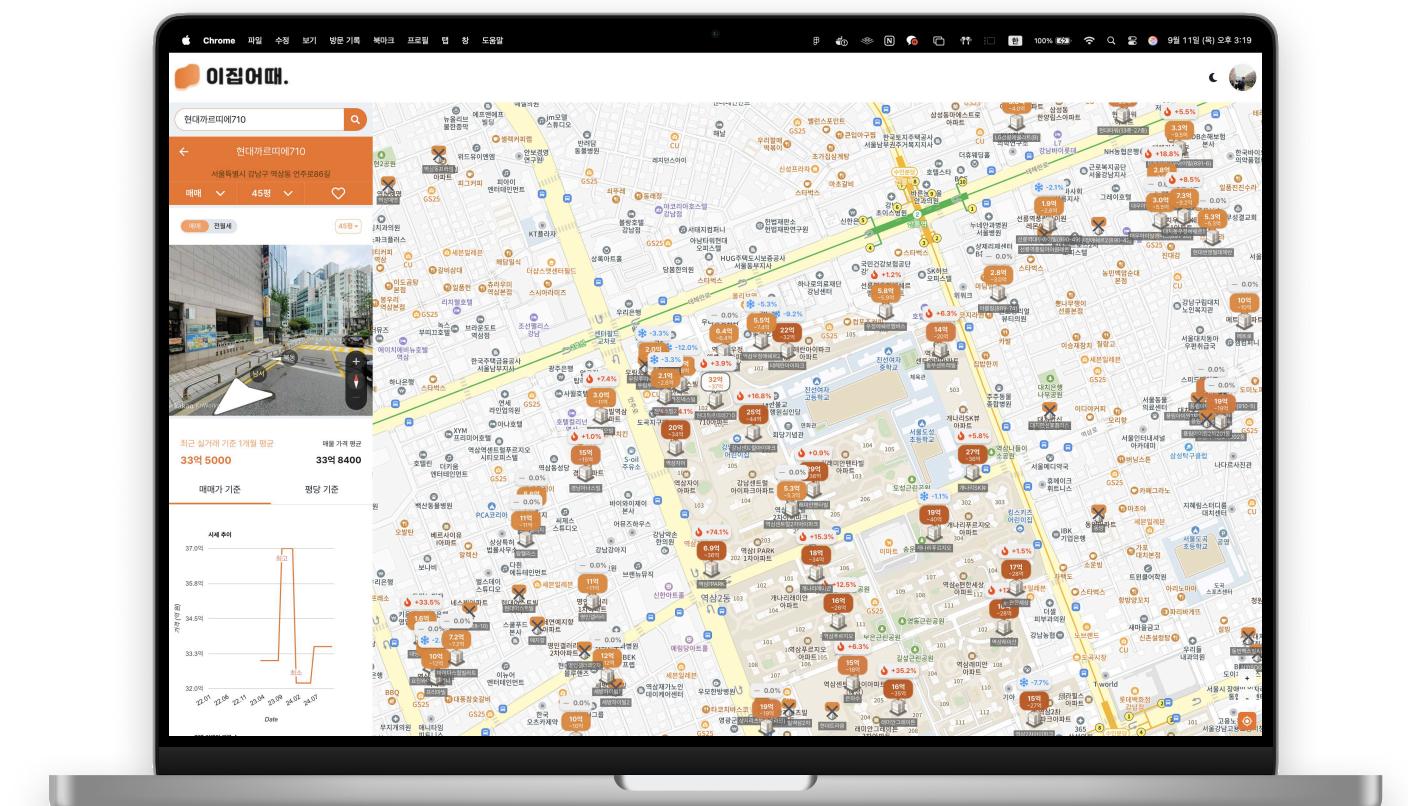
시니어를 위한 쉽고 편한 AI 기반 뱅킹 키오스크



- 진행 기간
2025.02.24 ~ 2025.04.11 (7주)
- 진행 인원
6명(FE: 2명, BE: 3명, AI: 1명)



아파트 매물 정보 및 시세 추이를 쉽게 확인하는 AI 기반 웹 플랫폼



- 진행 기간
2024.10.23 ~ 2024.11.27 (5주)
- 진행 인원
3명(FE: 1명, FULL: 1명, INFRA: 1명)

02. 프로젝트 소개 행복하개(hangbokdog)



2 SSAFY 자율 프로젝트 우수상(2등)

유기견 보호소의 입양·임보, 봉사, 강아지 관리를 통합해 운영 효율과 소통을 향상시키는 웹앱 플랫폼

- **프론트엔드 기술 스택**

TypeScript React Tailwind CSS Shadcn UI Tanstack Query Zustand Axios Firebase Cloud Messaging

- **진행 기간**

2025.04.14 ~ 2025.05.22 (6주)

- **진행 인원**

4명(FE: 2명, BE: 2명)

- **담당 역할**

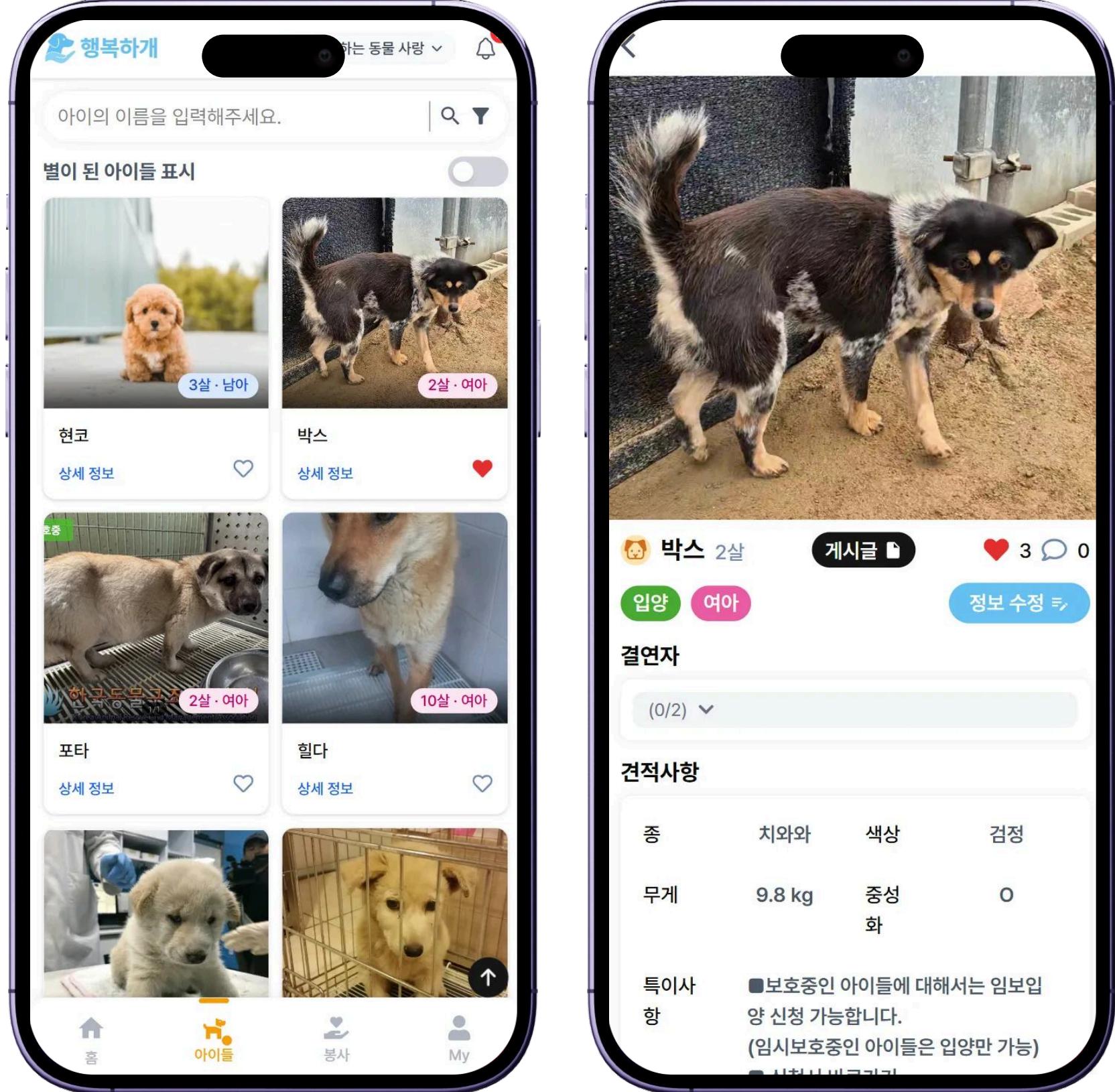
기획 및 설계, UI/UX 디자인, 프론트엔드 개발(기여도 50%)

- **기획 배경**

1. 소규모 보호소는 네이버 카페에 의존해 후원, 봉사자 등을 수작업으로 관리하여 운영 효율 저하
2. 유기견 정보가 체계적으로 관리되지 않아 데이터가 분산되어 입양 문의 및 후속 관리의 어려움

- **주요 기능**

1. 보호소(센터) 등록 및 관리 (보호소 정보 CRUD 및 권한 관리)
2. 역할 기반 회원가입 및 승인 시스템 (관리자/일반 사용자 역할 분리)
3. 봉사 및 입양/임시보호 신청 (사용자용 강아지 조회, 필터링 기능 포함)
4. 관리자 전용 통합 대시보드 (봉사, 입양, 긴급 알림 등 센터 운영 관리)
5. 보호소 통계 및 커뮤니티 기능 (회원 현황 통계 및 게시판 관리)



- **참고 링크**

Github

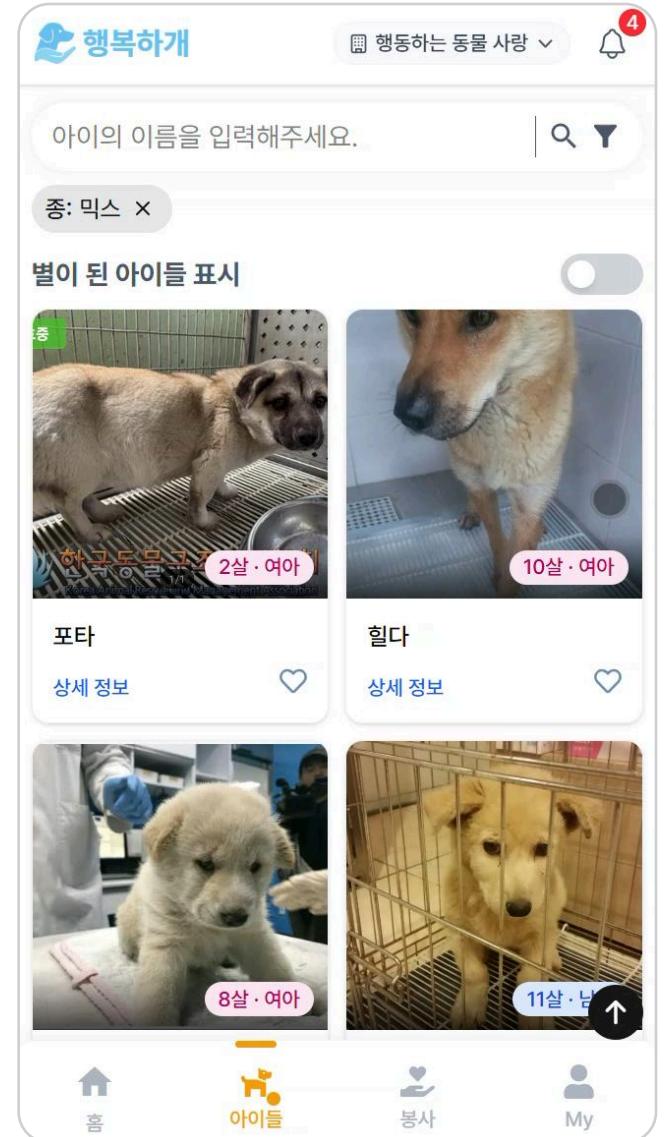
Figma

02. 프로젝트 소개 행복하개(hangbokdog)

• 구현 내용

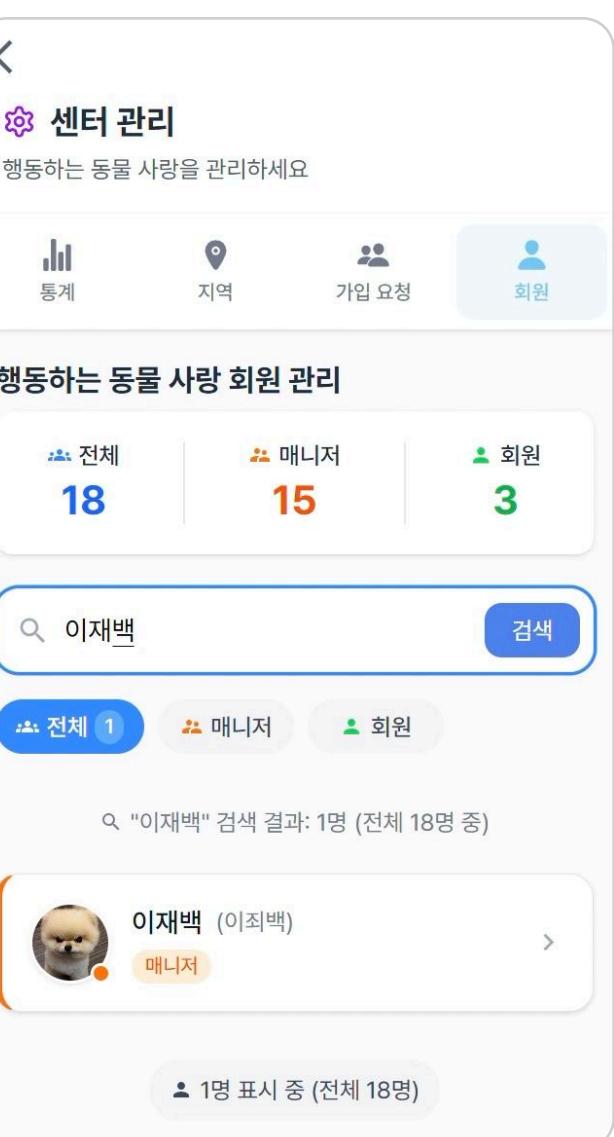
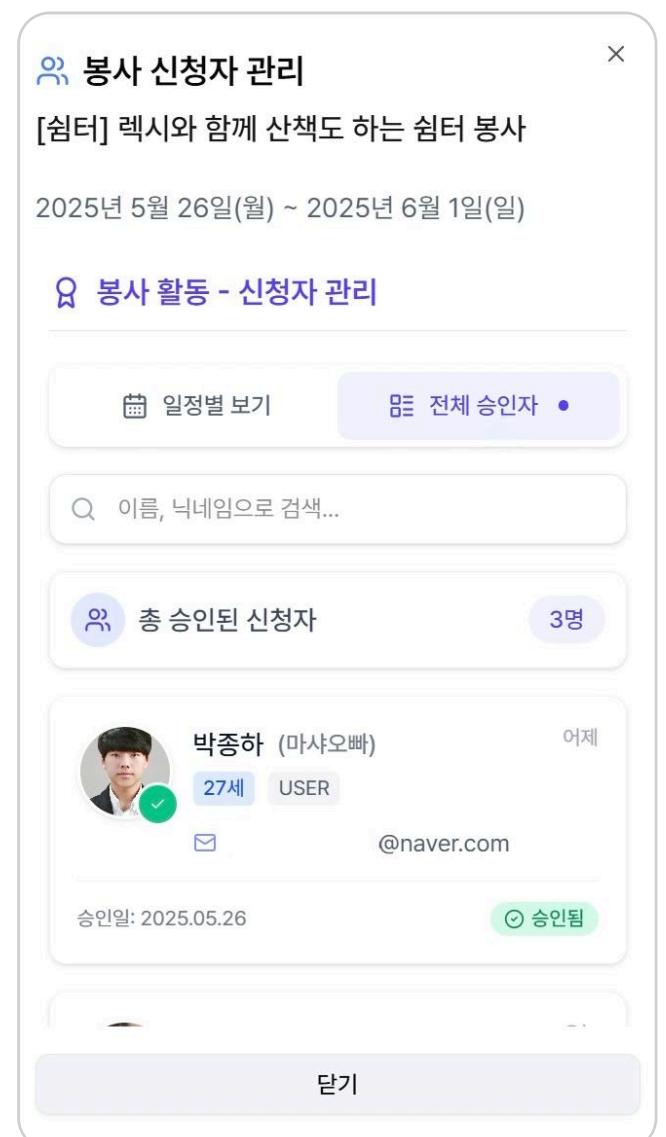
TanStack Query를 활용한 데이터 관리 최적화

- useInfiniteQuery를 사용하여 페이지 기반으로 데이터를 순차적으로 불러오는 성능 최적화된 무한 스크롤 기능 구현
- 낙관적 업데이트를 적용하여, useMutation 실행 시 서버 응답을 기다리지 않고 UI를 즉시 변경하여 사용자가 느끼는 대기 시간을 제거
- 데이터 변경이 필요한 경우, 관련된 쿼리를 무효화(Query Invalidation)하여 항상 최신 데이터를 화면에 보여주도록 보장



세션 복원 및 리렌더링 최적화를 통한 UX 완성도 향상

- sessionStorage를 활용하여 사용자가 보던 스크롤 위치나 탭(Tab) 상태를 저장하고, 페이지 재방문 시 그대로 복원하여 탐색의 연속성을 제공
- useCallback과 useMemo를 적절히 사용하여 불필요한 컴포넌트 리렌더링을 방지하고, 부드러운 UI 인터랙션을 구현
- 컴포넌트가 언마운트될 때 리소스를 정리하는 클린업(Cleanup) 함수를 철저히 구현하여 메모리 누수 방지 및 애플리케이션 안정성 확보



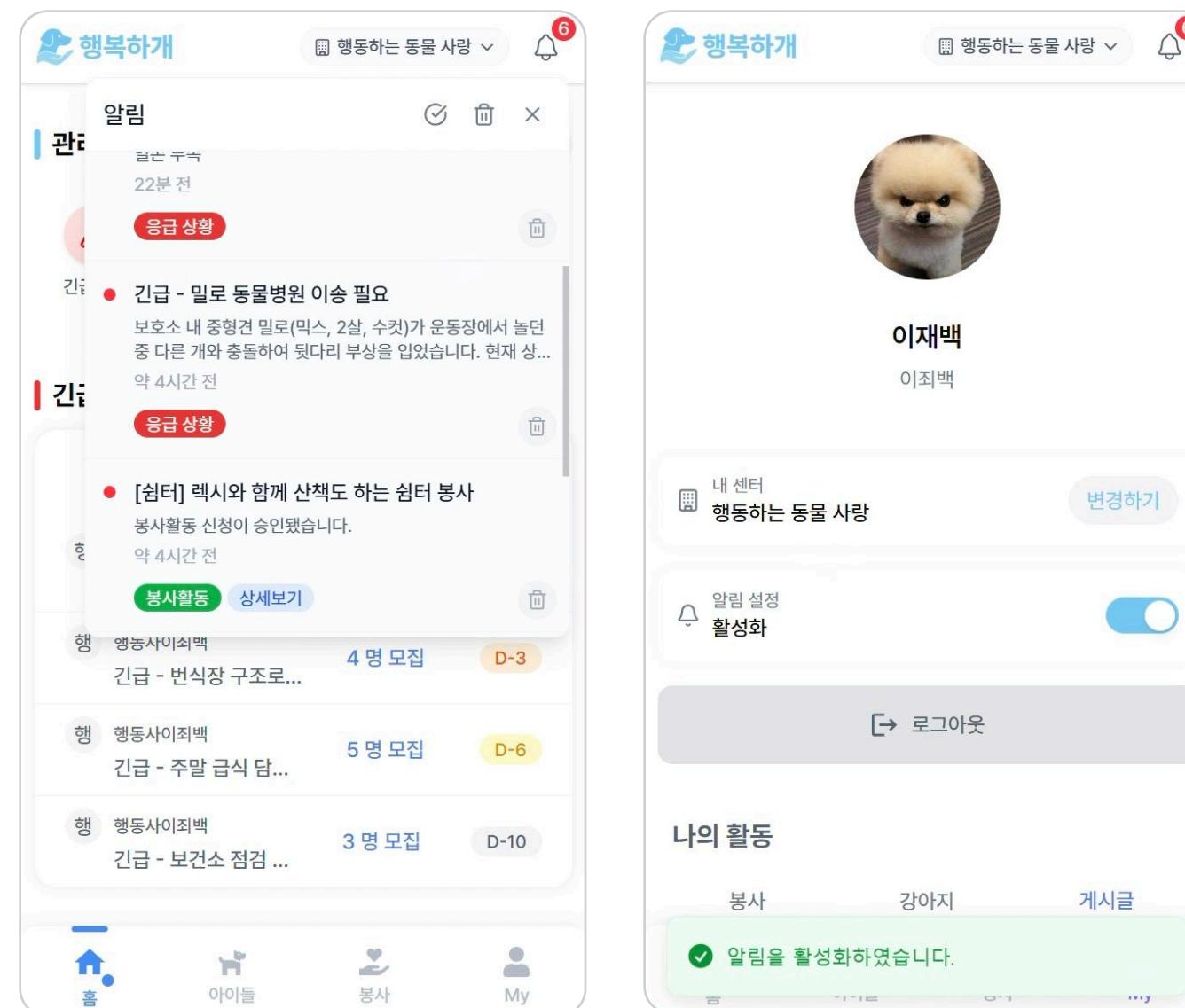
02. 프로젝트 소개

행복하개(hangbokdog)

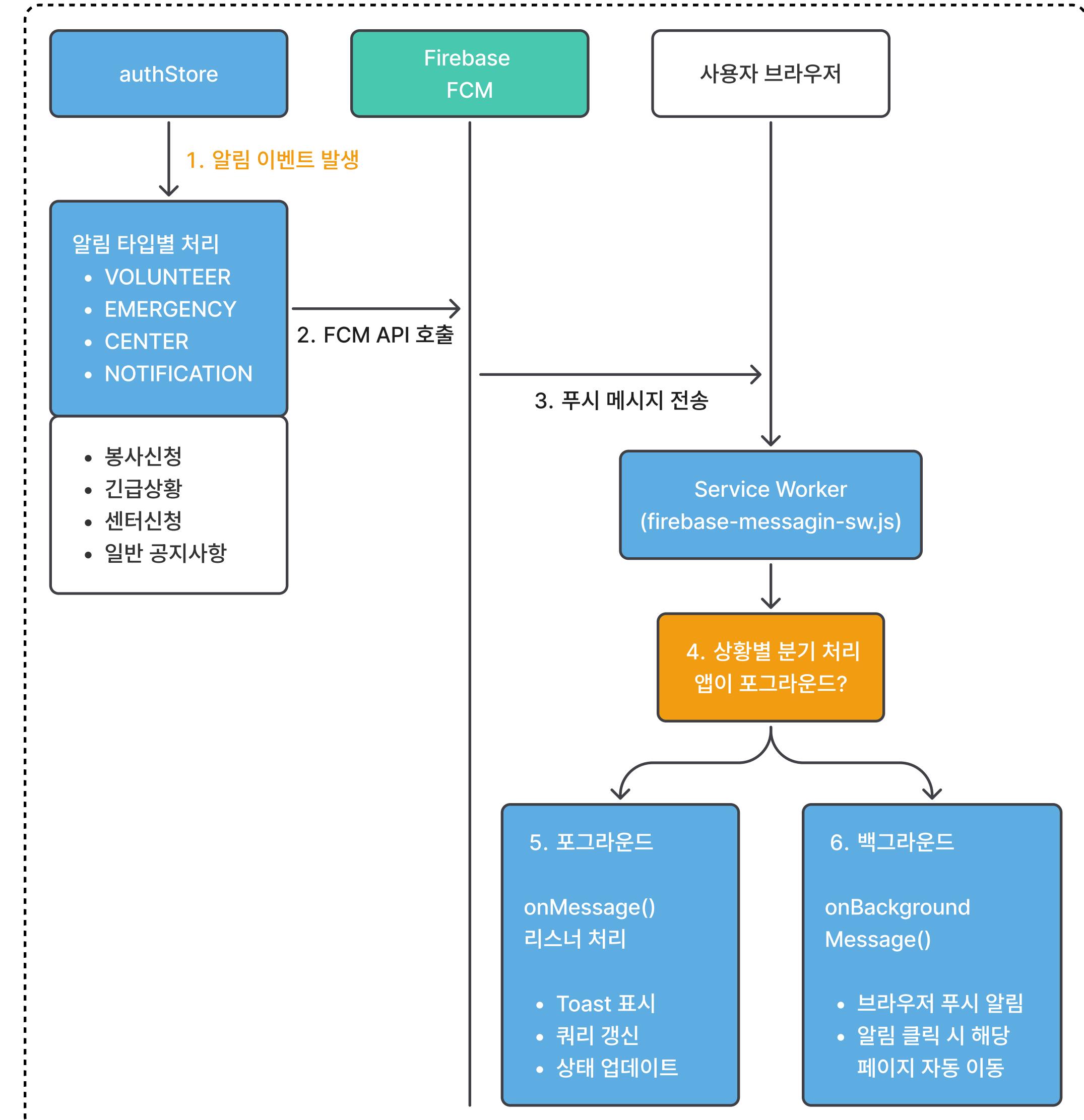
- 구현 내용

FCM 기반의 실시간 알림 시스템 구축

- Firebase Cloud Messaging(FCM)**과 **Service Worker**를 연동하여 앱이 백그라운드 상태일 때도 실시간 푸시 알림을 수신하는 기반 구축
- 수신된 알림 데이터를 타입('긴급', '봉사' 등)에 따라 파싱하고, 각기 다른 UI와 클릭 이벤트를 적용하여 유연한 알림 시스템 설계
- 앱의 활성화 상태(포그라운드/백그라운드)를 감지하여, 포그라운드 시에는 인앱 UI를, 백그라운드 시에는 브라우저 푸시 알림을 보내는 방식으로 분기 처리
- 알림 목록 UI와 수신 여부 토글 스위치를 구현하여 **사용자 중심의 알림 관리 기능** 제공



FCM 알림 시스템 아키텍처



02. 프로젝트 소개

행복하개(hangbokdog)

• 구현 내용

재사용성을 고려한 컴포넌트 아키텍처 및 UI 설계

- 공통 Layout 컴포넌트를 설계하고 **props**와 **children** 패턴을 활용하여 페이지 구조의 코드 중복을 최소화하고 개발 생산성 향상
- useNotification**, **useDayColor** 등 반복되는 비즈니스 로직을 **Custom Hook**으로 추상화하여 재사용성을 극대화하고 유지보수 비용 감소
- FullCalendar** 라이브러리를 봉사활동 일정으로 커스터마이징하였고, **Chart.js**를 활용하여 데이터를 효과적으로 시각화

The image displays the hangbokdog application interface across three main sections: General Layout, FullCalendar, and Chart.js.

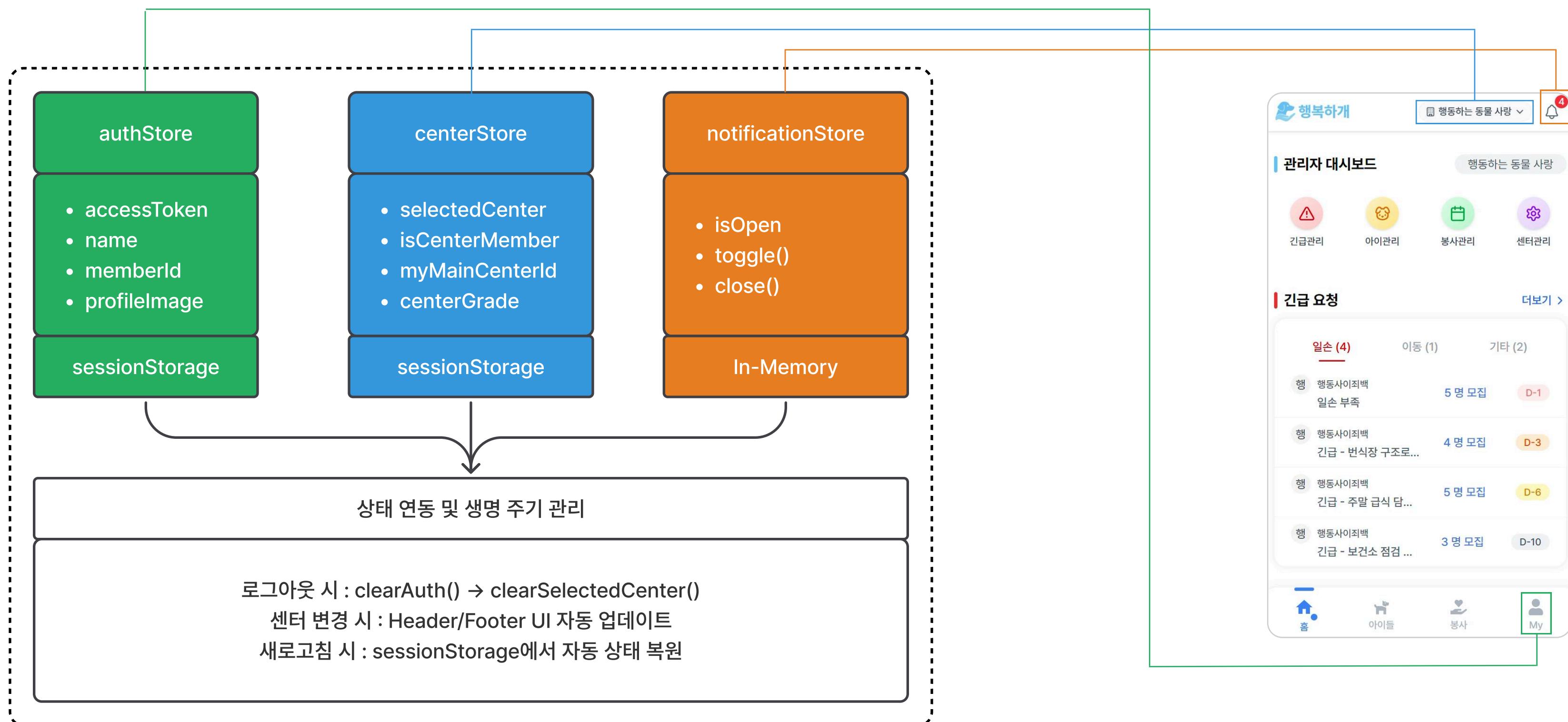
- General Layout:** Shows four mobile-like screens representing different management modules:
 - 긴급 관리**: Handles emergency situations, showing categories like 모집 중 (4), 일손 (4), 긴급 - 번식장... (0), and 긴급 - 주말... (0).
 - 아이들 관리**: Manages children, showing 아이들 전체보기 (list view) and 아이들 현황 (status view).
 - 봉사 관리**: Manages volunteer activities, showing 봉사활동을 관리할 센터를 선택해주세요.
 - 센터 관리**: Manages centers, showing 센터 선택 (list view) and 센터 현황 (status view).
- FullCalendar:** Displays the "전체 봉사활동 일정" (Full Volunteer Activity Schedule) for June 2025. The calendar shows days from 26 to 30, with specific events highlighted in yellow (e.g., [센터] 렉시와 함...). Below the calendar, there are summary statistics:
 - 6월 2025: 월, 화, 수, 목, 금, 토, 일
 - 26: [센터] 4/21(...)
 - 27: [센터] 렉시와 함...
 - 28: [센터] 렉시와 함...
 - 29: [센터] 렉시와 함...
 - 30: [센터] 렉시와 함...
 - 1: [센터] 렉시와 함...
- Chart.js:** Provides visual summaries of data:
 - 아이들 현황:** A donut chart showing 보호 중 (blue), 임시보호 중 (green), and 입양 완료 (red) counts.
 - 센터 요약 정보:** Summary statistics for the center:
 - 전체 아이들: 627마리
 - 보호 중: 474마리
 - 임시보호 중: 153마리
 - 입양 완료: 1마리

02. 프로젝트 소개 행복하개(hangbokdog)

• 구현 내용

Zustand 기반의 관심사 분리(SoC) 상태 관리 아키텍처

- authStore, centerStore, notificationStore 3개의 Store로 역할을 명확히 분리하여 코드 복잡성 감소 및 유지보수성 향상
- 페이지 새로고침 시 sessionStorage와 연동하여 로그인 상태와 선택된 센터 정보를 자동으로 복원하여 끊김 없는 사용자 경험 제공
- 로그아웃 시 관련된 Store의 상태까지 연쇄적으로 초기화되도록 설계하여 데이터 불일치 문제 원천 차단
- 컴포넌트별로 필요한 상태만 선택적으로 사용하여 불필요한 리렌더링을 방지하고 애플리케이션 성능 최적화



02. 프로젝트 소개 행복하개(hangbokdog)

• 트러블슈팅

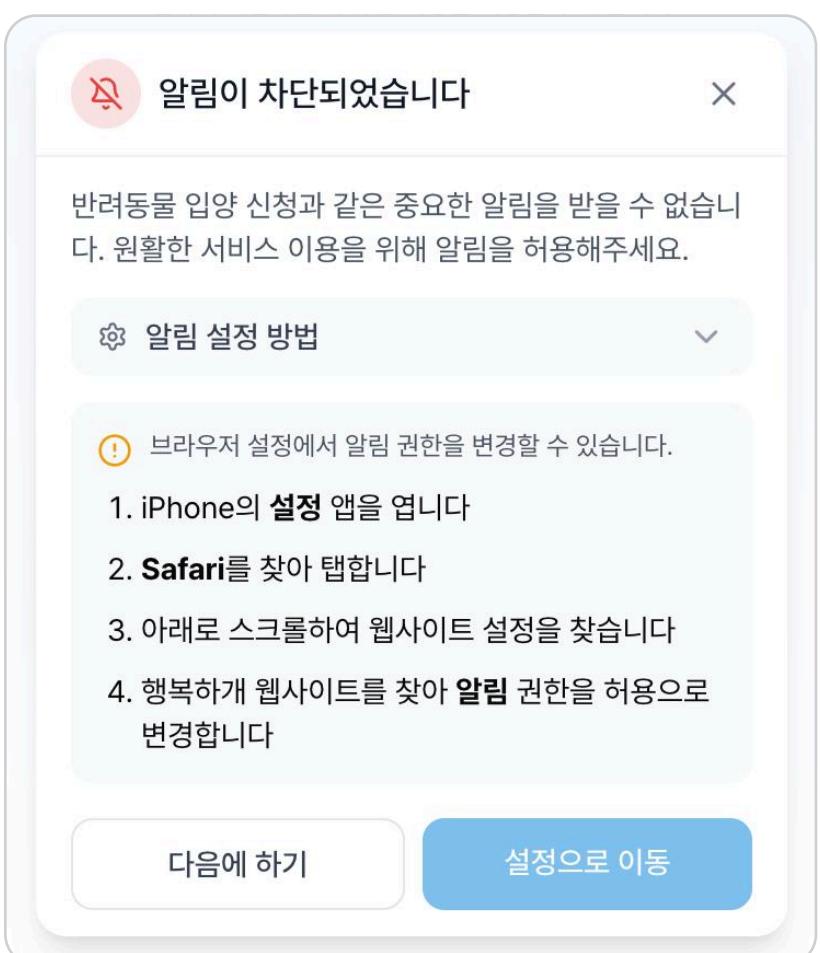
FCM 알림의 브라우저 호환성 및 권한 거부 UX 문제 해결

문제 상황

- iOS Safari 등 일부 브라우저에서 FCM 토큰 생성 오류가 발생
- 사용자가 알림 권한을 거부 시 재활성화하기 어려움 → 알림 기능 사용자 경험↓

해결 과정 및 결과

- **isSupported()** API로 브라우저의 FCM 지원 여부를 먼저 확인하고, 미지원 환경에서는 관련 기능을 비활성화하여 오류 발생을 원천 차단
- 알림 권한 상태를 **granted**, **denied**, **default**로 분기 처리하고, 권한이 거부된 사용자에게는 설정 방법을 **안내하는 UI**를 제공하여 UX를 개선
- **성과:** 브라우저 환경에 따른 잠재적 오류를 방지하고 일관된 알림 경험을 제공하여, 서비스의 전반적인 안정성을 높임



대용량 데이터 목록의 무한 스크롤 성능 최적화

문제 상황

- 대용량 목록 스크롤 시 발생하는 순간적인 끊김, 불필요한 API 호출 및 스크롤 위치 복원 오류

해결 과정 및 결과

- **Intersection Observer**의 **threshold** 옵션 조정 (**1.0** → **0.1**) → 스크롤이 끝나기 전 데이터 미리 로딩(Pre-loading)
- 컴포넌트 언마운트 시 **observer.disconnect()** 호출 → 메모리 누수 방지
- **sessionStorage**와 **setTimeout** 조합 → 깜빡임 없는 스크롤 위치 복원
- **성과:** 사용자 체감 로딩 시간 **57%** 단축, 메모리 사용량 **13%** 절감
- [자세한 성능 최적화 과정 보기 \(Tech Blog\)](#)



02. 프로젝트 소개 행복하개(hangbokdog)

- 배운 점

좋은 UX에 대한 고민

'사용자가 정말 편하게 쓸 수 있을까?'라는 질문에서 개발을 시작했습니다. 무한 스크롤 구현 시, 스크롤이 끝나기 전에 다음 데이터를 미리 로딩하도록 **Intersection Observer**의 **threshold** 값을 미세하게 조정한 것이 대표적입니다. 이 작은 변화만으로 사용자가 느끼는 경험은 '끊김'에서 '매끄러움'으로 바뀌었고, 코드 한 줄이 UX에 미치는 영향을 체감할 수 있었습니다. 검색 기능에 적용한 **디바운싱** 역시 기술 최적화를 넘어 쾌적한 사용자 경험을 만드는 과정이었습니다.

견고한 설계의 중요성

좋은 구조를 만드는 것이 결국 가장 빠른 길임을 깨달았습니다. 관리자 페이지의 공통 레이아웃을 컴포넌트화하여 코드 중복을 줄였고, **TanStack Query**와 **Zustand**로 서버와 클라이언트 상태를 명확히 분리하여 데이터 흐름의 예측 가능성과 안정성을 높였습니다. 또한, 복잡한 FCM 로직을 커스텀 훅(**useNotification**)으로 추상화하여 동료 개발자가 더 쉽게 기능을 이해하고 사용하도록 만들었습니다.

예상치 못한 문제 해결 능력

프로젝트는 언제나 계획대로만 흘러가지 않았습니다. 특히 **FCM 알림이 iOS Safari에서 동작하지 않았던 문제**를 해결하며, 다양한 사용자 환경에 유연하게 대처하는 능력을 길렀습니다. 단순히 오류를 수정하는 것을 넘어, 미지원 브라우저에서는 기능을 비활성화하고 권한을 거부한 사용자에게는 별도의 안내 UI를 제공하는 등 보다 완성도 높은 서비스를 고민하게 되었습니다.

'행복하개' 프로젝트는 사용자 경험을 최우선으로 생각하는 태도, 유지보수와 확장을 고려한 아키텍처 설계, 그리고 예상치 못한 문제에 대처하는 문제 해결 능력을 길러준 성장의 발판이었습니다.

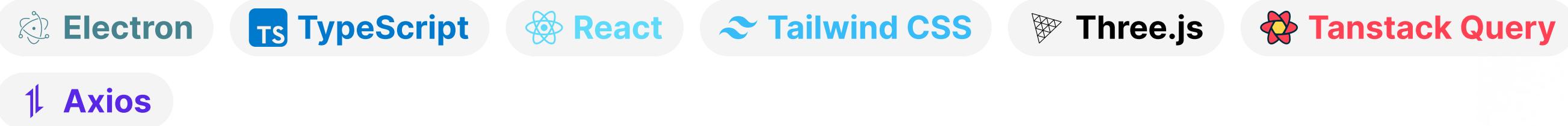
02. 프로젝트 소개 동행(donghang)



SSAFY 자율 프로젝트 우수상(1등)

시니어를 위한 쉽고 편한 AI 기반 뱅킹 키오스크

- **프론트엔드 기술 스택**



- **진행 기간**

2025.02.24 ~ 2025.04.11 (7주)

- **진행 인원**

6명(FE: 2명, BE: 3명, AI: 1명)

- **담당 역할**

기획 및 설계, UI/UX 디자인, Electron 기반 아키텍처 설계 및 메인/서브 윈도우 IPC 통신 전체 구현(기여도 50%)

- **기획 배경**

1. 디지털 금융 서비스의 빠른 확산 속에서 키오스크 사용에 어려움을 겪는 고령층의 디지털 격차 문제 발생
2. AI 기술을 통해 고령층을 대상으로 한 편의성 개선 및 보이스피싱 범죄까지 예방하는 것을 목표로 함

- **주요 기능**

1. 사용자 얼굴 인식 기반 UI 자동 전환 (일반/고령층)
2. Three.js 기반 3D AI 은행원을 통한 음성 인식 및 안내
3. AI 기반 실시간 보이스피싱 위험 탐지 및 경고 시스템
4. 사용자 맞춤형 금융 상품 추천 및 상세 안내 기능
5. Electron을 활용한 듀얼 모니터 지원 및 키패드, 시뮬레이터 구현



- **참고 링크**



[Github](#)



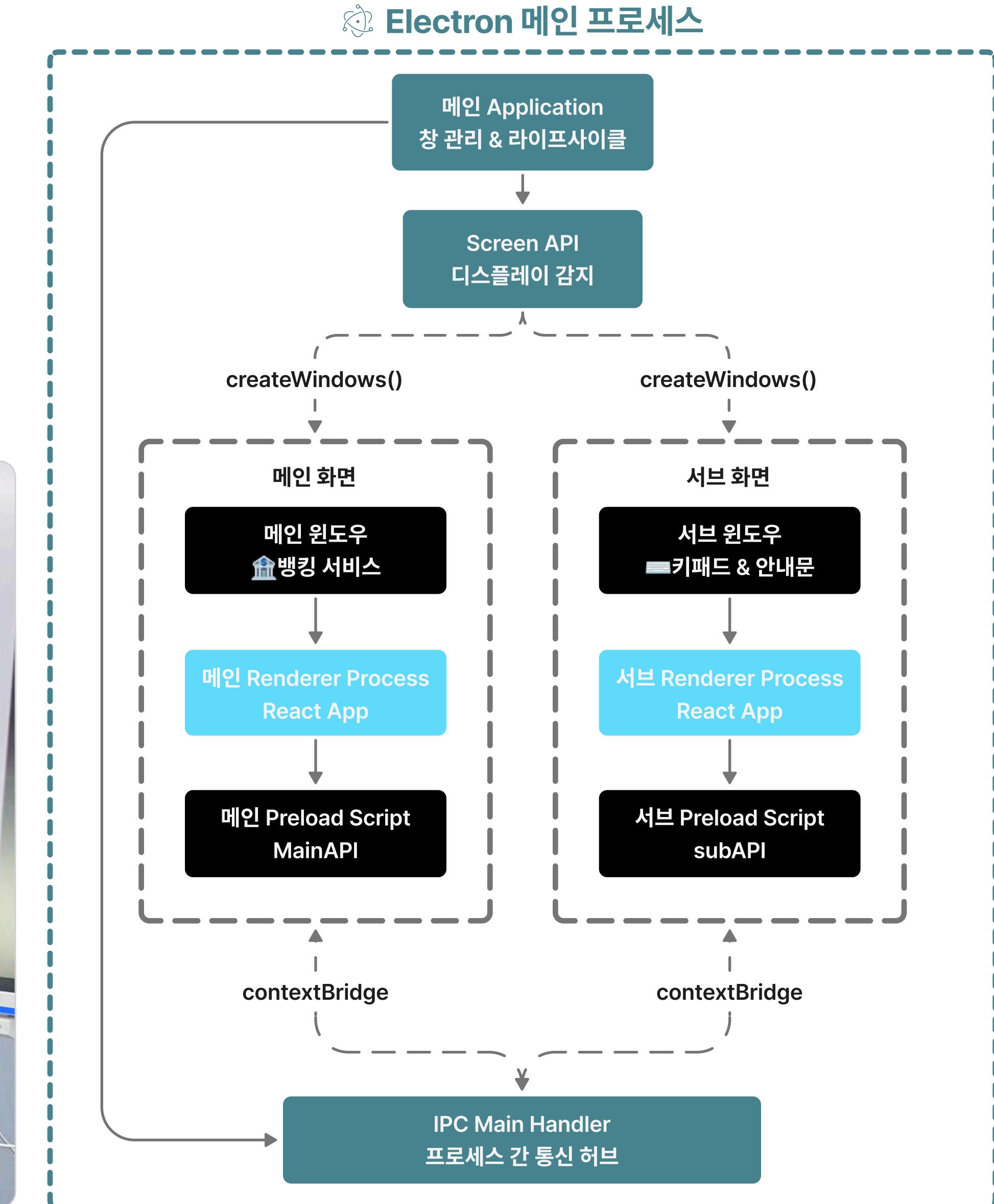
[Figma](#)

02. 프로젝트 소개 동행(donghang)

- 구현 내용

Electron 듀얼 모니터 아키텍처 구현

- Main Process 중심의 아키텍처 설계:** 창 관리, 디스플레이 감지, IPC 통신 허브 등 핵심 로직을 Main Process에 집중시켜, 각 Renderer Process(메인/서브 화면)가 UI 렌더링에만 집중할 수 있는 안정적인 구조를 구축
- screen.getAllDisplays() API**를 활용하여 연결된 디스플레이를 실시간으로 감지하고, 단일/듀얼 모니터 환경에 따라 **자동으로 윈도우를 배치하는 시스템** 구축
- Main/Renderer/Preload 프로세스를 분리하고 **Context Bridge**를 적용하여 **보안 아키텍처**를 설계
- 보안이 강화된 **Context Bridge** 기반의 **IPC 통신 구조**를 설계하여, 메인(뱅킹 서비스)과 서브(키패드) 윈도우 간의 안정적인 양방향 데이터 동기화를 구현
- 서브 모니터의 숫자 입력 및 확인/취소 액션을 메인 화면에 즉시 반영하고, 메인 윈도우의 상태에 따라 서브 윈도우의 모드(키패드 탑재, 시니어 모드)를 동적으로 제어



02. 프로젝트 소개 동행(donghang)

- 구현 내용

AI 연령 인식에 따른 일반/시니어 모드 UI 동적 전환 구현

- React **Context API**를 활용하여 AI가 분석한 연령대 결과를 전역 상태('시니어 모드')로 관리
- Context 상태에 따라 조건부 렌더링으로 일반/시니어용 컴포넌트를 동적으로 교체, UI/UX 최적화
- UserContext, AIContext** 등 도메인별로 Context를 분리하여 상태 관리 구조의 직관성 및 유지보수성 향상
- WebSocket**으로 수신한 AI 결과를 Context에 반영하여, 페이지 새로고침 없는 실시간 UI 전환 구현
- 모드별 상호작용 방식 차별화**

- 시니어 모드:** AI 아바타와의 음성 대화를 통해 진행하며, 음성 명령에 따라 자동으로 페이지가 전환되는 방식 구현
- 일반 모드:** 사용자가 직접 버튼을 터치하여 단계를 진행하는 직관적인 UI 제공

물리적 키오스크 환경을 위한 가상 시뮬레이터 UI/UX 구현

- 실제 키오스크의 **카드, 현금, 신분증 투입구**를 3D CSS(**perspective**) 효과를 적용하여 사실적인 애니메이션으로 시뮬레이션
- 현재 진행 중인 은행 업무 단계에 따라 해당 투입구의 표시등이 자동으로 활성화되도록 구현하여, 사용자에게 다음에 할 일을 직관적으로 안내
- 각 시뮬레이터의 상태(카드 삽입 여부, 현금 투입 등)를 **Context API**로 관리하여 앱 전반에서 일관된 상태를 유지하도록 설계



- 카드 투입 애니메이션 보러가기

02. 프로젝트 소개 동행(donghang)

- 구현 내용

시니어를 위한 접근성(WCAG) 최적화 설계

- WCAG 2.1 AA 레벨을 목표로, 모든 UI 요소의 명암비를 4.5:1 이상으로 유지하여 저시력 사용자도 명확하게 콘텐츠를 인지할 수 있도록 설계
- 시니어 모드에서는 일반 모드 대비 폰트 크기를 25% 확대하고, 가독성이 높은 전용 서체를 사용하여 정보 전달력을 높임
- 모든 버튼과 상호작용 요소의 터치 영역을 최소 44x44px 이상으로 확보하고, 시니어 모드에서는 추가로 30% 확대하여 오터치를 방지



실시간 보이스피싱 탐지 및 경고 시스템 구현

- AI의 실시간 음성 분석 결과를 WebSocket으로 수신하여 보이스피싱 의심 패턴을 즉시 감지하는 시스템을 구축
- 보이스피싱 위험 감지 시, 전체 화면을 덮는 시각적 경고 오버레이(framer-motion 활용)와 안내 문구를 통해 사용자의 주의를 즉각적으로 환기시키도록 구현
- 메인/서브 모니터에 걸쳐 다계층 경고 시스템을 적용하여, 사용자가 위험 상황을 명확히 인지하고 거래를 중단할 수 있도록 유도



02. 프로젝트 소개 동행(donghang)

• 트러블슈팅

Electron 듀얼 모니터 실시간 렌더링 동기화 문제 해결

문제 상황

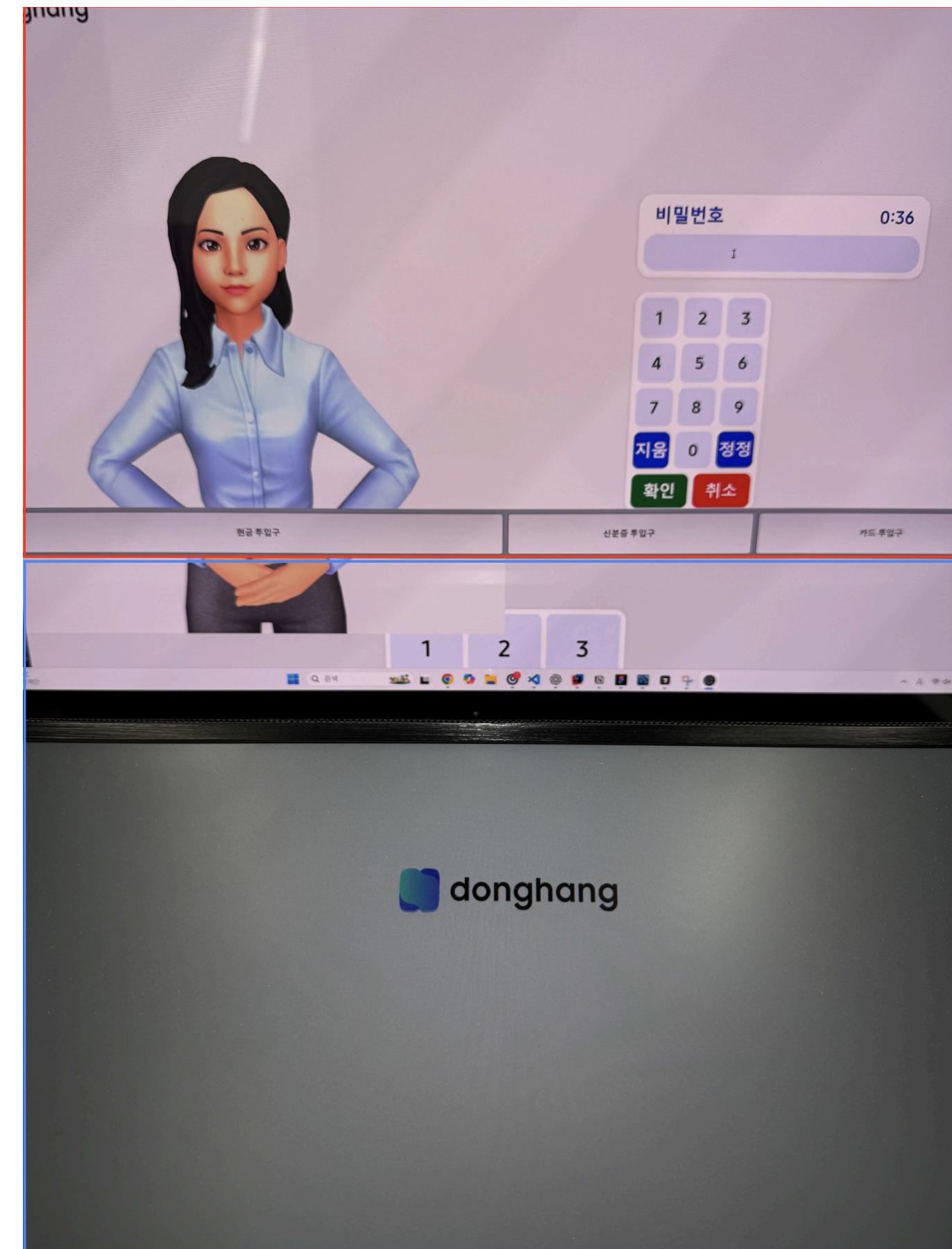
- 초기 아키텍처(단일 BrowserWindow를 두 모니터에 확장) 사용 시, 메인 모니터의 상태 변경이 서브 모니터에 실시간으로 렌더링되지 않는 문제가 발생
- 창 포커스를 수동으로 변경(Alt+Tab)해야만 서브 모니터의 UI가 업데이트되는 등, 키오스크 사용에 치명적인 사용자 경험(UX) 저하를 초래

해결 과정 및 결과

- 각 모니터에 독립적인 BrowserWindow 인스턴스를 생성하는 구조로 아키텍처를 전면 재설계하여 각 화면이 독립적으로 렌더링
- 두 윈도우 간의 통신을 위해 IPC 통신 시스템을 구축하고, Main Process가 데이터 중계 허브 역할을 하도록 구현
- contextBridge를 통해 Renderer Process에 안전하게 IPC API를 노출시켜 보안을 강화했으며, useSubMonitorListeners 커스텀 푸드으로 복잡한 통신 로직을 추상화

성과

- 두 모니터 간 < 50ms의 자연 시간으로 실시간 양방향 상태 동기화를 구현하여, 별도 조작 없이도 즉각적인 UI 반응을 경험할 수 있는 끊김 없는 듀얼 모니터 키오스크 환경을 완성
- [🔗 자세한 성능 최적화 과정 보기 \(Tech Blog\)](#)



02. 프로젝트 소개 동행(donghang)

- 배운 점

복합 시스템 설계의 깊이

Electron 듀얼 모니터 아키텍처를 직접 설계하며, 독립된 윈도우 간의 상태를 IPC 통신으로 실시간 동기화하는 복잡성을 해결했습니다. 메인 화면의 변화에 서브 화면이 즉각 반응하는 디테일이 '불편함'을 '자연스러움'으로 바꾸는 핵심임을 깨달았습니다.

실시간 AI 처리와 성능 최적화

WebSocket으로 AI의 실시간 영상/음성 분석 결과를 처리하며 성능과 정확도의 균형점을 찾는 과정을 배웠습니다. 연령 감지 시 노이즈 필터링, VAD 시스템의 임계값 미세 조정 등을 통해 최적화된 알고리즘 설계의 중요성을 체감했습니다.

접근성을 고려한 포용적 설계

'진정한 접근성'을 고민하며, Context API를 활용해 단순히 UI만 바꾸는 것을 넘어 상호작용 방식 자체를 다르게 설계했습니다. 시니어 모드에서는 3D 아바타와의 음성 대화로, 일반 모드에서는 직관적인 터치로 업무를 진행하도록 하여 기술로 디지털 격차를 해소하는 경험을 했습니다.

'동행' 프로젝트는 단순한 웹 개발을 넘어, 복잡한 시스템 아키텍처 설계와 기술을 통한 사회적 가치 창출에 대해 깊이 고민하게 해준 기술적 성장의 전환점이었습니다.

02. 프로젝트 소개 이집어때(ezip)



SSAFY 자율 프로젝트 우수상(1등)

아파트 매물 정보 및 시세 추이를 쉽게 확인하는 AI 기반 웹 플랫폼

- **프론트엔드 기술 스택**

TypeScript React Chakra UI Redux Tanstack Query Framer Motion

Axios

- **진행 기간**

2024.10.23 ~ 2024.11.27 (5주)

- **진행 인원**

3명(FE: 1명, FULL: 1명, INFRA: 1명)

- **담당 역할**

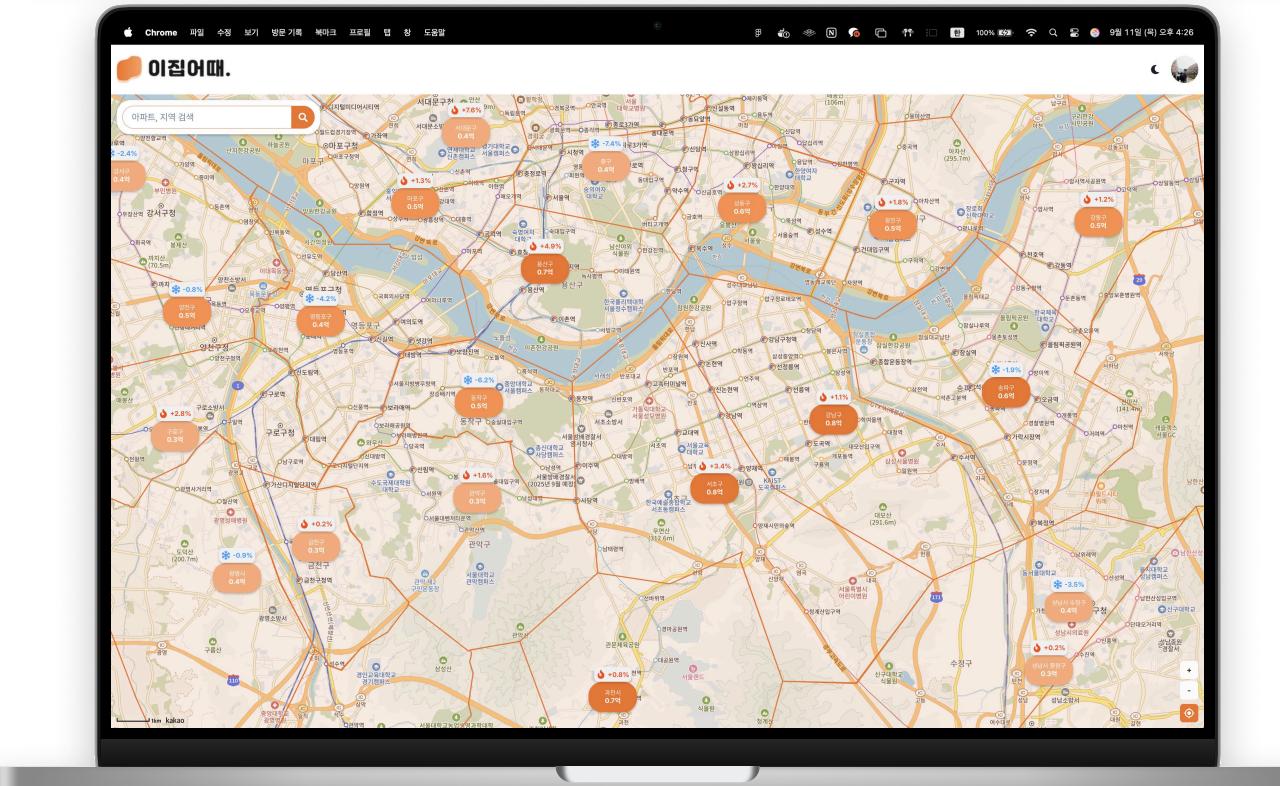
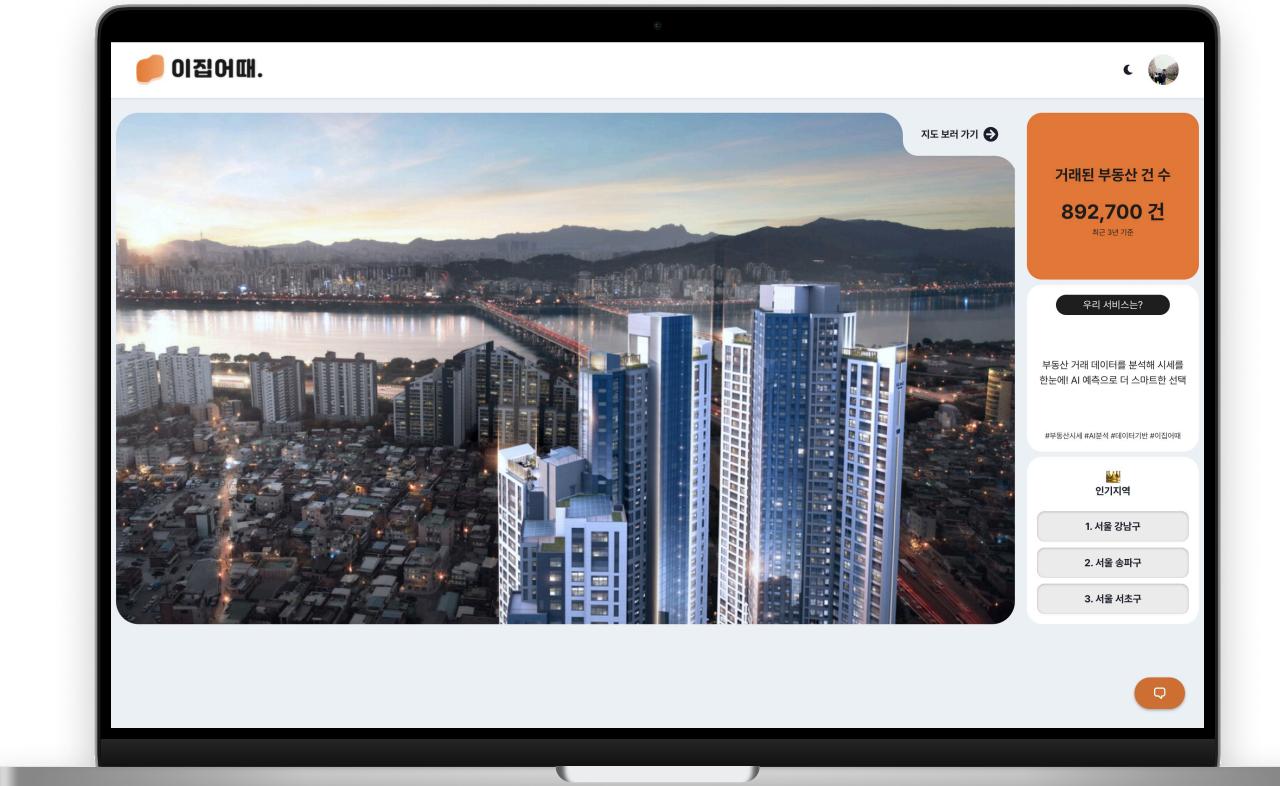
기획 및 설계, UI/UX 디자인, 프론트엔드 아키텍처 설계 및 핵심 기능(지도, AI 예측) 전체 구현 담당

- **기획 배경**

1. 부동산 투자 시 정확한 시세 정보와 미래 전망의 부재로 인한 투자 리스크 증가
2. 복잡하고 분산된 부동산 정보로 인한 정보 접근성 문제

- **주요 기능**

1. 줌 레벨별 카카오 맵 연동 (아파트 → 동 → 구 → 시 단위 시세 표시)
2. ChatGPT API 연동을 통한 AI 기반 부동산 시세 예측 및 챗봇 상담
3. 네이버 뉴스 API 크롤링으로 실시간 부동산 뉴스 제공
4. Google Charts 기반 거래 내역 및 시세 추이 시각화
5. 핫한 매물 시스템 - 전달 대비 증가율 기준 인기 부동산 추천



- **참고 링크**

Github

Figma

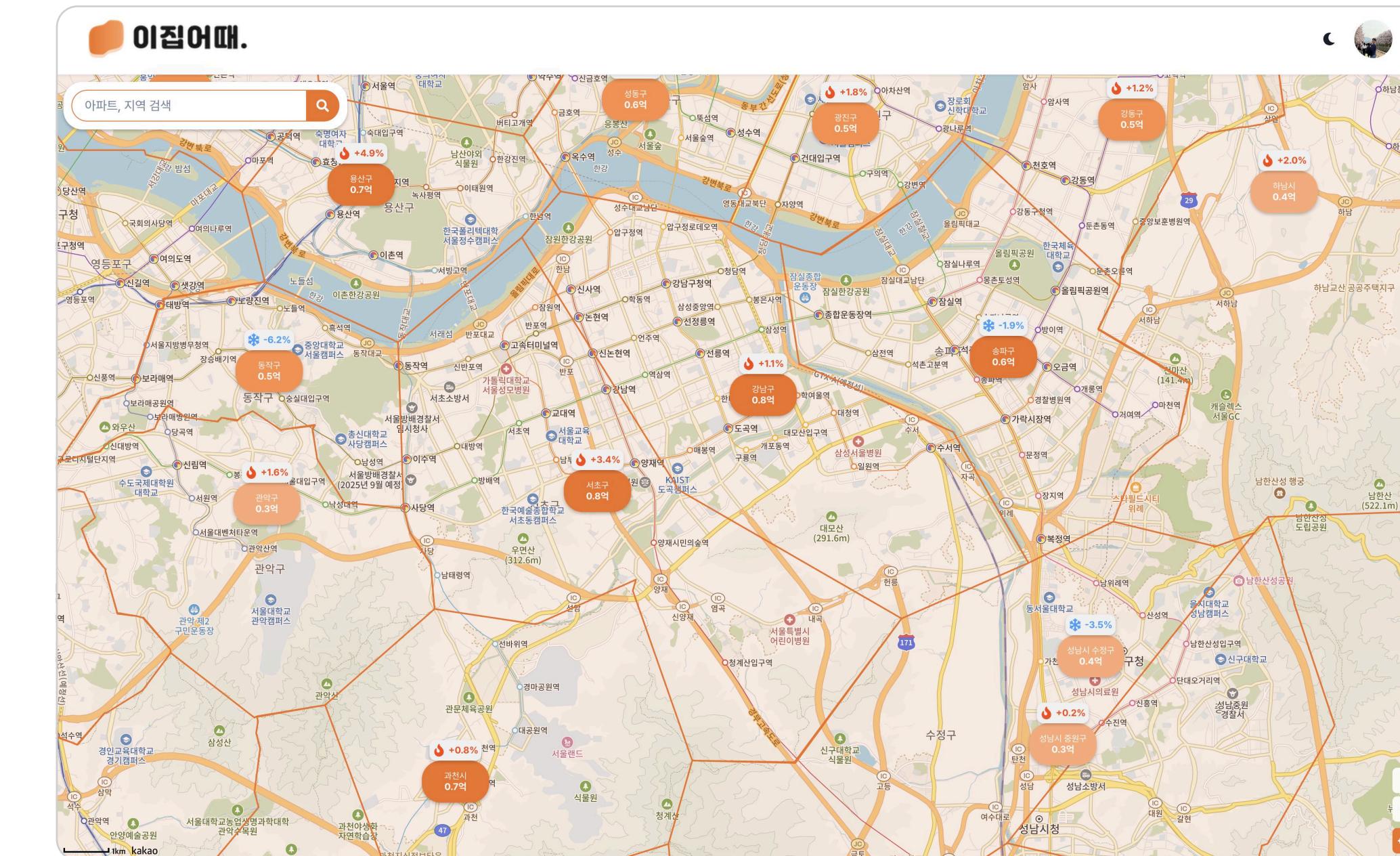
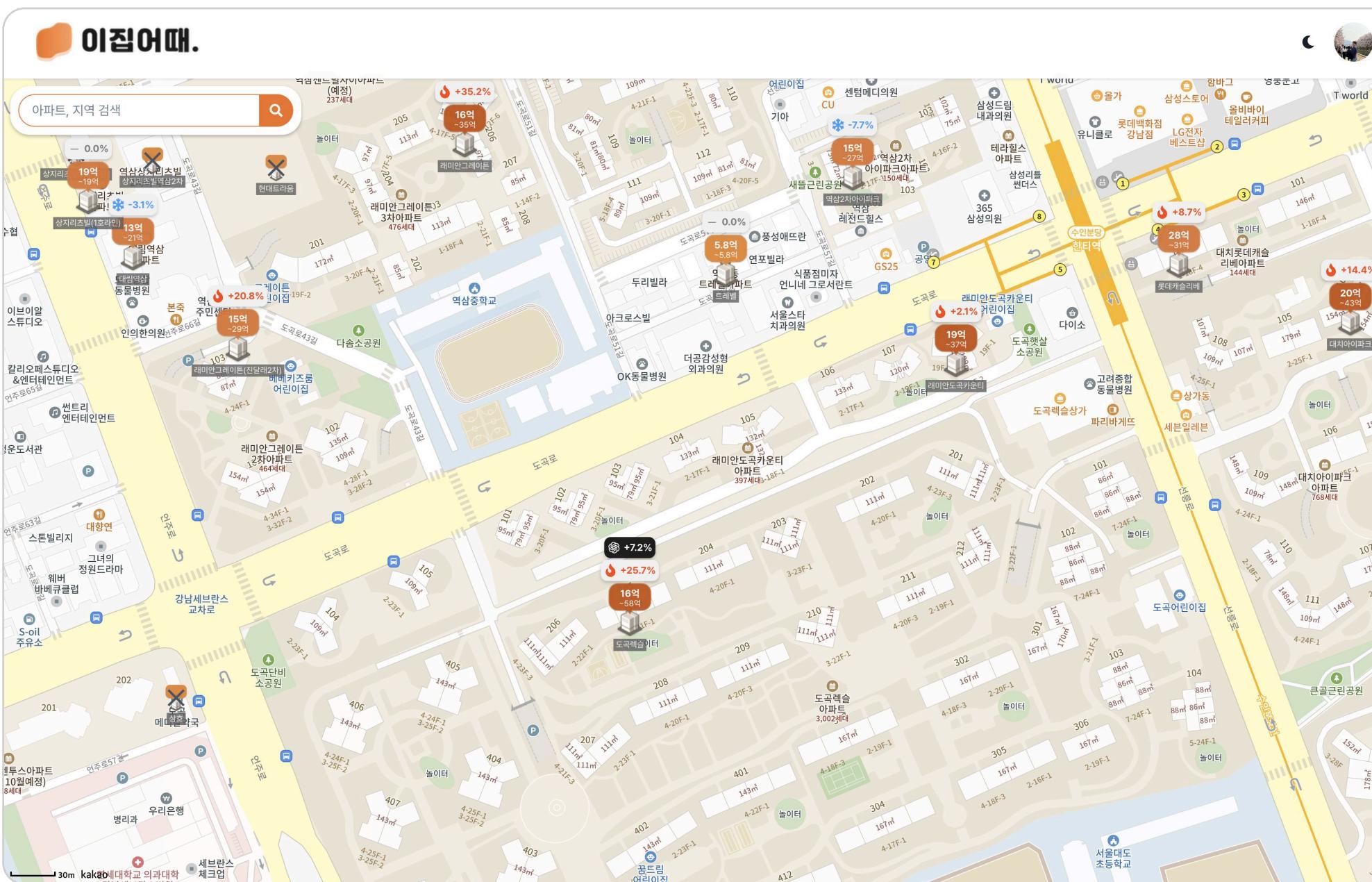
02. 프로젝트 소개

이집어때(ezip)

• 구현 내용

Kakao Map API 기반 동적 데이터 시각화 구현

- 줌 레벨(Zoom Level)에 따른 차등적 데이터 렌더링 시스템을 구축하여, 지도 축척에 따라 시/구/동 단위의 평균 시세와 개별 아파트 정보를 효율적으로 시각화
- Tanstack Query를 활용하여 지도의 현재 좌표(bounds)와 줌 레벨을 queryKey로 사용하여 지도 이동 및 축소/확대에 따른 데이터 요청을 자동화 및 캐싱 처리
- 지도 위에 표시되는 아파트 정보에는 전월 대비 실제 증감률과 ChatGPT가 예측한 시세 증감률을 함께 표시하여, 사용자에게 다각적인 데이터 기반 인사이트를 제공
- GeoJSON 데이터를 활용하여 행정구역 경계를 폴리곤(Polygon)으로 시각화하고, react-kakao-maps-sdk의 커스텀 오버레이 기능을 적극 활용하여 UI 구현



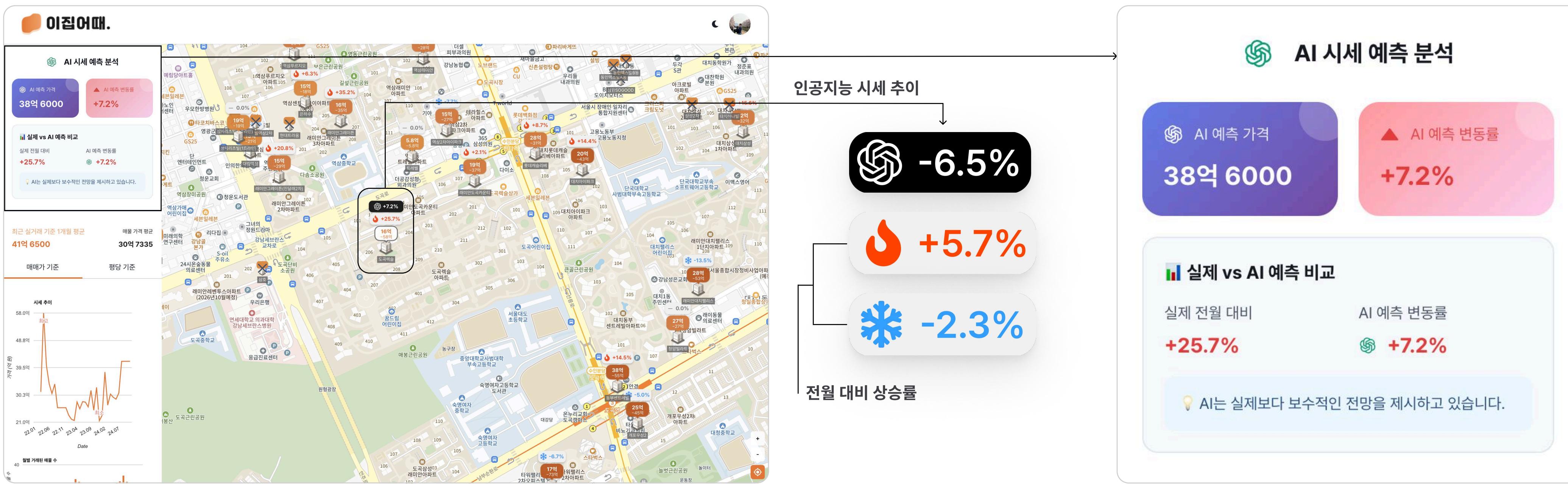
02. 프로젝트 소개

이집어때(ezip)

• 구현 내용

ChatGPT API 연동 부동산 시세 예측 기능

- 사용자가 특정 아파트를 조회할 때, 해당 아파트의 데이터를 기반으로 **ChatGPT API에 시세 예측을 요청하고 그 결과를 실시간으로 받아와 사용자에게 제공**
- 복잡한 AI 응답 데이터를 사용자가 이해하기 쉬운 핵심 지표(예측 가격, 예상 증감률)로 가공하여 **React Google Charts**와 연동, 시각적인 설득력을 높임
- 사용자의 질문 이력을 **sessionStorage**에 저장하여, 대화의 연속성을 유지하고 채팅 형태의 UI를 통해 자연스러운 AI 인터랙션 경험을 제공



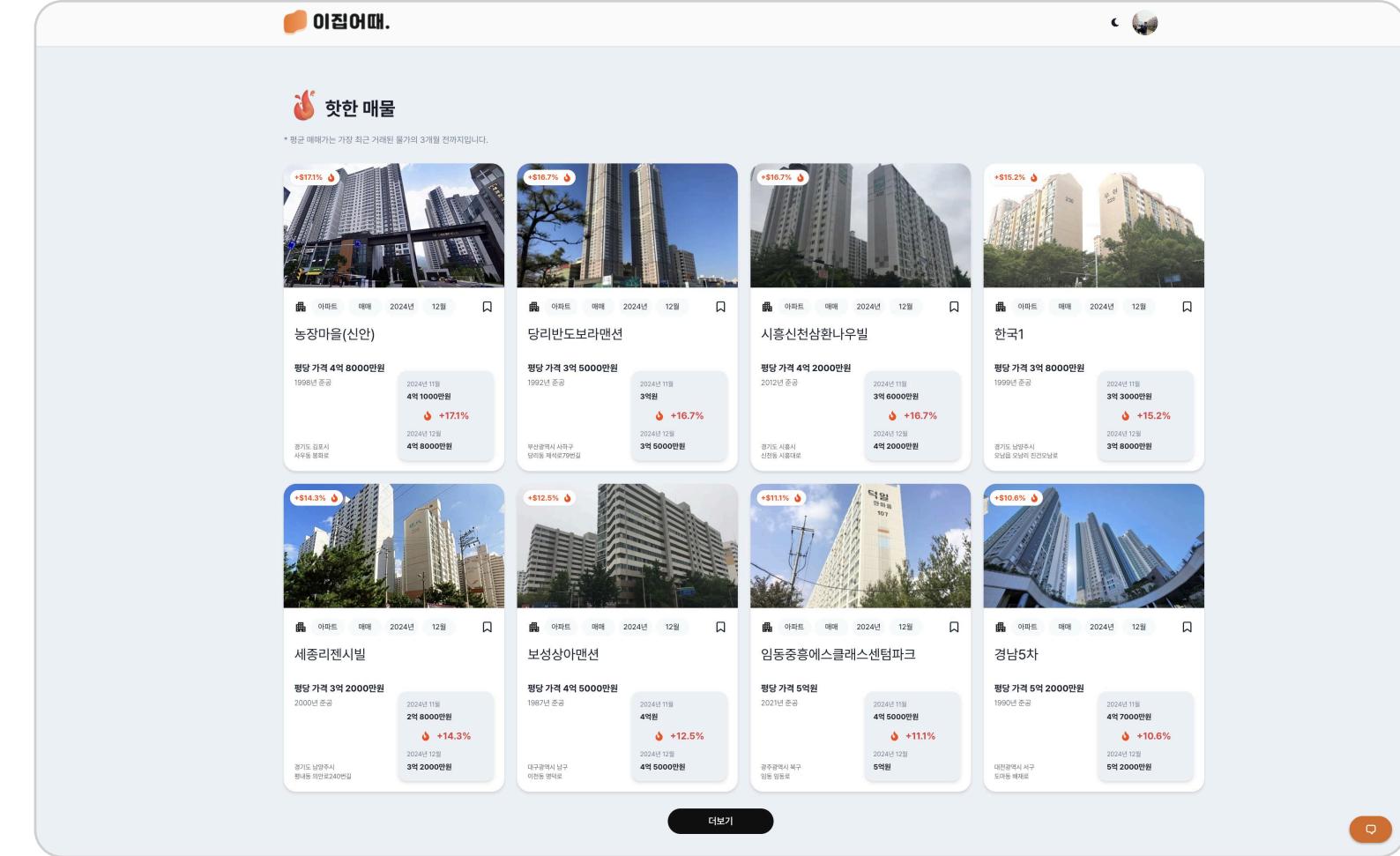
02. 프로젝트 소개

이집어때(ezip)

• 구현 내용

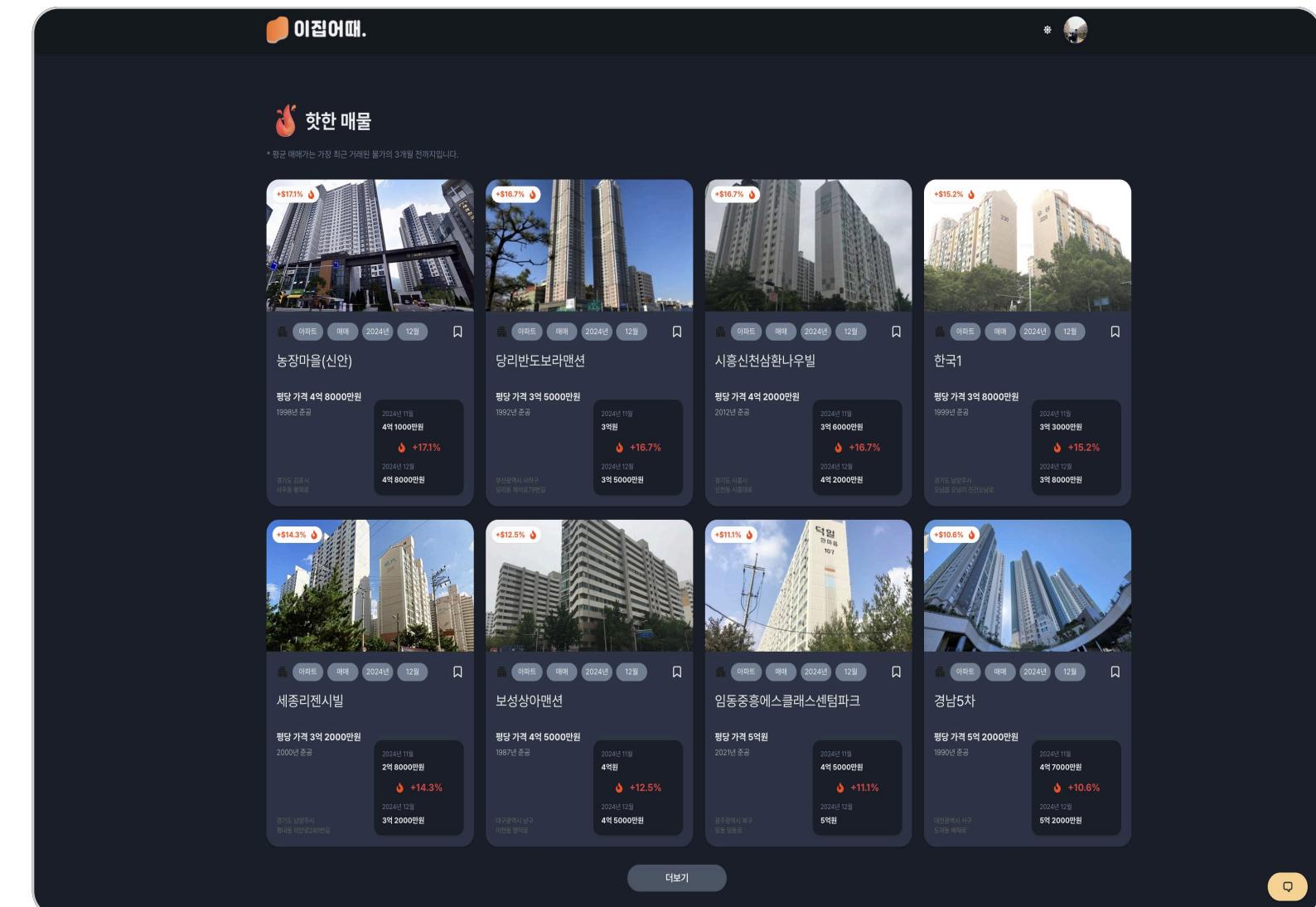
데이터 기반 '핫 매물' 시스템 구현 및 시각화

- 가격 상승률 기반 '핫 매물' 선정 로직 구현하고, **Chakra UI**의 카드 컴포넌트를 활용하여 사용자 주목도를 높임
- 증가율, 평당 가격 등 복잡한 수치 데이터를 **상승(🔥)**, **하락(❄️)** 등 직관적인 아이콘과 함께 시각화하여 사용자가 정보를 빠르게 인지할 수 있도록 설계
- 아파트 상세 페이지에서는 **React Google Charts**를 활용하여 **월별 시세 추이(라인 차트)**와 **거래량(바 차트)**을 시각화하여 사용자의 깊이 있는 데이터 분석을 지원



다크모드 및 반응형 웹 디자인 구현

- Chakra UI**의 **useColorModeValue** 툥과 커스텀 테마를 활용하여 동적 다크모드 시스템 구축
- 브레이크포인트 시스템을 기반으로 화면 크기에 따라 1~4단으로 변하는 반응형 그리드 레이아웃 설계
- 모바일 환경에서는 로고를 축소하고, 특정 페이지(지도)에서는 레이아웃 제약을 동적으로 해제하여 UX 최적화



02. 프로젝트 소개 이집어때(ezip)

• 트러블슈팅

Grid-based Caching을 통한 지도 API 성능 최적화

문제 상황

- 초기 구현 시, Tanstack Query의 queryKey에 지도의 실시간 좌표(bounds)를 직접 사용
 - 하지만 지도를 1px만 움직여도 bounds 값이 변경되어 queryKey가 달라짐
 - 이로 인해 동일한 지역을 탐색함에도 매번 새로운 API 요청이 발생하는 문제가 발생

해결 과정 및 결과

- (**Grid-based Caching** 전략 도입) 연속적인 좌표 값을 일정한 격자(Grid) 단위로 변환하여 queryKey를 생성 → **Math.floor(좌표 / 격자크기)** * 격자크기 공식을 통해, 특정 격자 내에서는 항상 동일한 queryKey를 갖도록 하여 캐시 재사용성을 높임
 - 또한 줌 레벨에 따라 격자 크기를 동적으로 조절하는 적응형 격자 시스템을 적용하여 상세 뷰(줌 레벨 낮음)에서는 촘촘한 격자로, 광역 뷰(줌 레벨 높음)에서는 넓은 격자로 캐싱

성과

- Grid-based Caching 전략으로 동일 지역(강남역, 역삼역 부근) 테스트
→ 불필요한 API 호출을 80% 감소
 - 그 결과 로딩 지연 없는 매끄러운 지도 탐색 경험을 제공하고 서버 부하를 크게 줄임
 -  자세한 성능 최적화 과정 보기 (Tech Blog)

Name	Status	Type	Initiator	Size	Ti...
bound?bottomLat=37.4975879249...	200	xhr	apt.ts:107	5.7 kB	30...
bound?bottomLat=37.4975878059...	200	xhr	apt.ts:107	5.6 kB	26...
bound?bottomLat=37.4975877064...	200	xhr	apt.ts:107	5.8 kB	27...
bound?bottomLat=37.4975875431...	200	xhr	apt.ts:107	5.8 kB	29...
bound?bottomLat=37.4975874517...	200	xhr	apt.ts:107	6.2 kB	28...
bound?bottomLat=37.4975873311...	200	xhr	apt.ts:107	6.2 kB	34...
bound?bottomLat=37.4976321982...	200	xhr	apt.ts:107	6.4 kB	32...
bound?bottomLat=37.4976320709...	200	xhr	apt.ts:107	6.7 kB	30...
bound?bottomLat=37.4976319865...	200	xhr	apt.ts:107	6.8 kB	31...
bound?bottomLat=37.4976318420...	200	xhr	apt.ts:107	7.0 kB	32...



Name	Status	Type	Initiator	Size	Ti...
⌚ bound?bottomLat=37.4976318420...	200	xhr	apt.ts:107	7.0 kB	45...
⌚ bound?bottomLat=37.497624482...	200	xhr	apt.ts:107	5.3 kB	26...



02. 프로젝트 소개 이집어때(ezip)

• 트러블슈팅

페이지 새로고침 시 인증 상태가 초기화되는 문제 해결

문제 상황

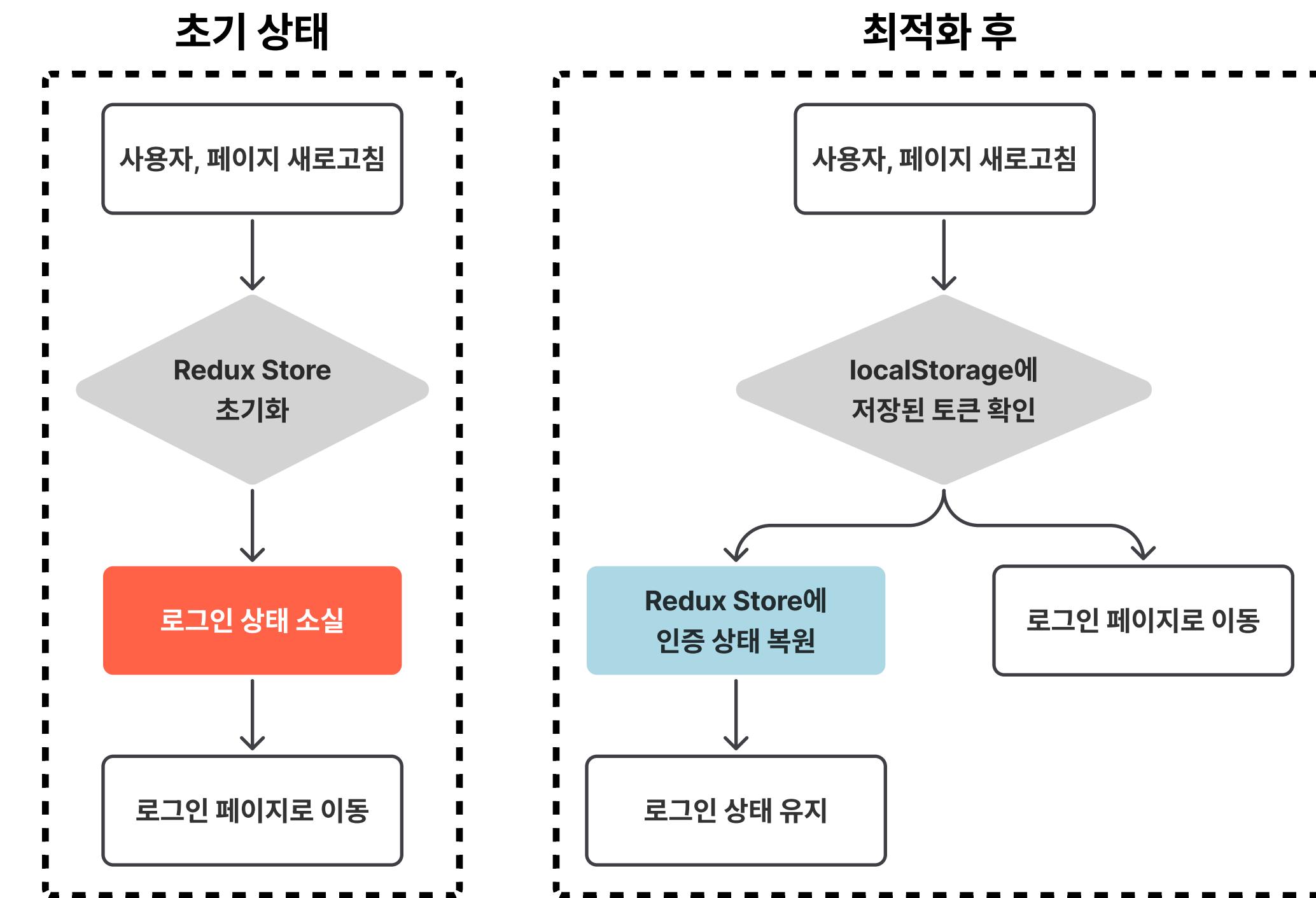
- 최초 로그인 시 Redux 스토어에만 인증 상태를 저장하여, 사용자가 페이지를 새로고침하면 로그인 상태가 초기화되는 문제가 발생
- 이로 인해 인증이 필요한 페이지에 접근할 수 없는 등 **UX의 일관성이 저하됨**

해결 과정 및 결과

- 인증 상태 영속성 확보를 위해 Redux-persist와 같은 라이브러리 대신, **보안과 관리 포인트를 고려하여** 이중 관리 체계 직접 구축
 - 로그인 성공 시:** JWT는 localStorage에, 사용자 정보는 Redux Store에 저장
 - 애플리케이션 로드 시:** localStorage의 토큰 유무를 먼저 확인하여 Redux 인증 상태를 복원하는 로직 추가

성과

- 새로고침 및 재방문에도 인증 상태가 안정적으로 유지되어 **끊김 없는 UX 제공**
- (학습 및 회고) 나아가 localStorage의 XSS 취약점을 학습, **HttpOnly 쿠키와 Refresh Token의 필요성을 인지하며 더 안전한 인증 시스템에 대한 기술적 시야 확보**



02. 프로젝트 소개 이집어때(ezip)

- 배운 점

본질을 파고드는 성능 최적화

지도 기능의 잦은 API 호출 문제를 해결하기 위해, 단순 캐싱을 넘어 **Grid-based Caching** 전략을 직접 설계하고 도입했습니다. 연속적인 좌표를 격자 단위로 변환하여 **queryKey**를 생성함으로써 불필요한 API 호출을 80% 이상 줄였고, 이를 통해 서비스 특성에 맞는 캐싱 전략을 최적화하는 능력을 길렀습니다.

기술을 통한 사용자 가치 창출

ChatGPT API를 연동하여 시세 예측 기능을 구현하며, 최신 기술을 사용자 가치로 연결하는 경험을 했습니다. 복잡한 AI 응답을 핵심 지표로 가공하고 **React Google Charts**로 시각화하는 과정에서, **기술을 사용자 친화적으로 풀어내는 데이터 처리 및 시각화 역량**을 키웠습니다.

견고한 설계와 일관된 UI/UX

Tanstack Query와 **Redux Toolkit**으로 서버와 클라이언트 상태를 명확히 분리하여 데이터 흐름의 안정성을 높였습니다. 또한, Chakra UI 기반의 **다크모드**와 **반응형 웹**을 구현하며, 다양한 환경의 사용자에게 일관된 UI/UX를 제공하는 것의 중요성을 배웠습니다.

'이집어때'는 복잡한 데이터를 다루는 기술적 자신감과 사용자 중심의 개발 철학을 길러준 프로젝트입니다. 특히 성능 병목의 본질을 파고들어 고도화된 전략으로 해결했던 경험은 가장 큰 자산이 되었습니다.