

안녕하세요, 김덕진입니다.

경기도 오산시 동부대로 332-14

ghaeffd71@gmail.com

<https://github.com/deokjin25>

PPT 형식 포트폴리오

1998. 07. 14

+82-10-3379-4250

<https://velog.io/@dkqrty/posts>



끊임없이 배우고 성장하는 프론트엔드 개발자입니다.

단순한 기능 구현에서 그치지 않고 유지보수성의 개선을 위해 비즈니스 로직을 분리하고 재사용 가능한 코드를 모듈화하는 등 리팩토링 하는 과정을 좋아합니다. 작은 차이가 모여 큰 성과를 이룬다고 믿기에 미약할지라도 성능 개선을 위해 고민하고 적용해보며 확인하는 것을 중요하게 생각합니다. 더 나은 코드는 있어도 완벽한 코드는 없다고 생각합니다. 스스로의 코드를 경계하며 항상 더 나은 코드를 위해 노력하고 있습니다.

사용자 친화적인 인터페이스를 구현하며, 문제 해결과 최적화에 집중하고 있습니다.

스크롤 이후 페이지 상단으로 돌아가는 버튼, 페이지 뒤로 가기 시의 정상 응답, 페이지 이탈 후 재방문 시의 상호작용 등 무심코 지나칠 수 있는 작은 부분도 놓치지 않고 사용자의 다양한 행동 방식을 고려하여 개발하고자 합니다. 이를 위해 단순 개발자가 아닌 제가 직접 이 제품, 서비스를 이용하는 사용자라고 생각하며 개발하고 있습니다.

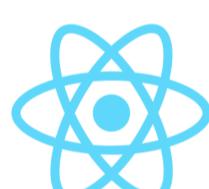
협업을 통해 함께 성장하는 경험을 중요하게 생각합니다.

학부 시절부터 다양한 팀 프로젝트를 경험하며 타인의 이야기를 경청하는 것이 얼마나 큰 의미를 가지는지 이해하고 있습니다. 팀원 간의 갈등 상황도 있었지만 의견을 조율하고 최적의 해결점을 찾아내며 소통과 협업 능력을 이끌어냈습니다. 다양한 관점을 존중하며 팀원들과 지식을 공유하고 성장하는 과정을 소중히 생각합니다.

기술 스택



TypeScript



React



NextJS



Redux



Zustand



React-Query



TailwindCSS

수상

- 삼성 청년 SW AI 아카데미 자율 프로젝트 1위 (최우수상)
- 삼성 청년 SW 아카데미 성적 우수상

자격증

- 빅데이터분석기사
- SQL 개발자(SQLD)
- ADsP(데이터분석준전문가)

교육 & 활동 내역

삼성 청년 SW · AI 아카데미(SSAFY) 12기 수료

2025.01 ~ 2025.06

- 2번의 FrontEnd 프로젝트와 BackEnd 프로젝트 1회 수행

삼성 청년 SW · AI 아카데미(SSAFY) 12기 이수

2024.07 ~ 2024.12

- Java, SpringBoot, VueJS, MySQL, 알고리즘 등 웹 풀스택 과정 이수

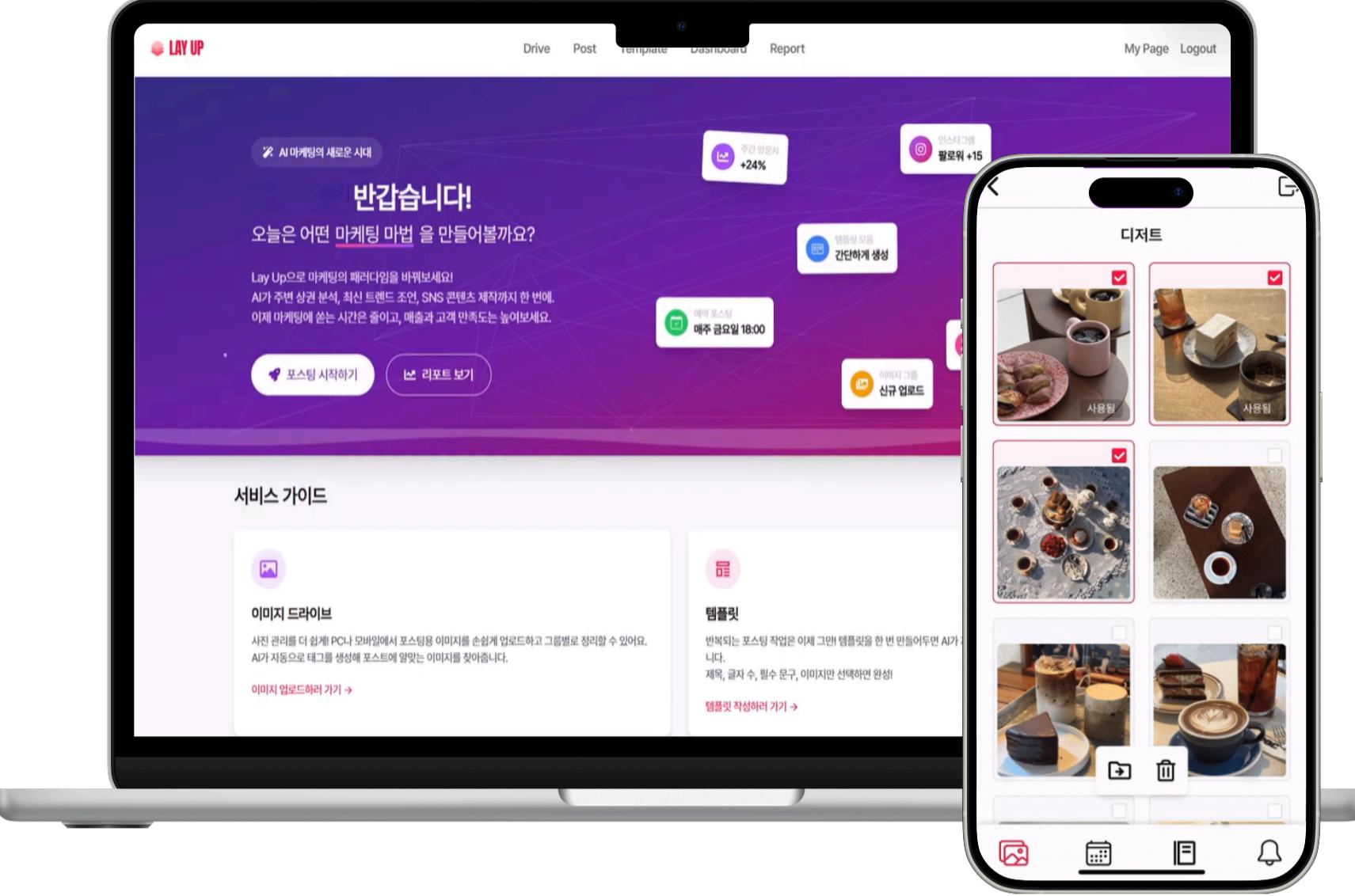
세종대학교 졸업

2018.03 ~ 2024.02

- 호텔관광경영학과 / AI 융합학사 (3.75/4.5)



Lay Up



서비스

[소상공인의 SNS 홍보 마케팅을 돋기 위한 AI Agent 마케팅 서비스]

작업 기간

2025.04.14 ~ 2025.05.21 (1개월)

프론트엔드 기술 스택

TypeScript

NextJS

Redux

TailwindCSS

PWA

React

React-Query

Figma

구성원

총 6인 (FE 3인 / BE 3인)

성과

삼성 청년 SW · AI 아카데미 자율 프로젝트 최우수상 수상



주요 업무 및 상세 역할

[PC]

- 메인페이지 구현
- 대시보드 페이지 구현
- 포스트 목록 페이지 구현
- 보고서 목록 및 상세 페이지 구현
- 마이페이지 구현
- QR로그인 구현

[모바일]

- 캘린더 탭 구현
- 알림 기능 구현
- QR 로그인 구현

[설계]

- Barrel Pattern 설계
- API Service Layer 패턴 설계
- Client / Server Component 분리 설계
- React-Query의 prefetchQuery & dehydrate 구조 설계

- View - Logic 관심사 분리
- 기기 검증 기반 라우팅
- 수정 사항 변경 감지

기술 선정과 도입 배경

왜 PWA를 사용하였는가?

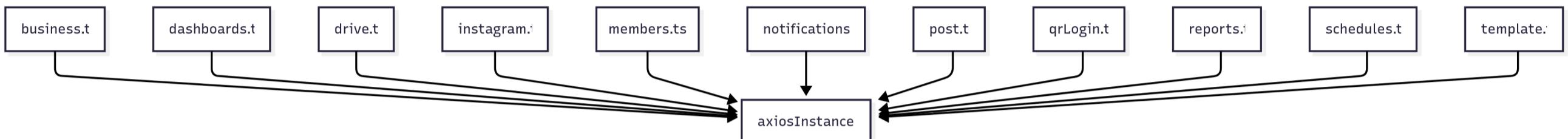
- 서비스 특성상 사용자가 촬영한 이미지를 업로드해야 했습니다. 일반적으로 사진은 모바일 기기로 촬영 및 보관하기에 이미지 업로드 경험을 개선하고 모바일 기기에서도 서비스 일부 기능을 활용할 수 있도록 React와 PWA를 활용해 별도 모바일 프로젝트를 구성했습니다.

왜 Redux를 사용하였는가?

- 이전에 Zustand를 사용하며 Flux 패턴을 경험한 바 있습니다. 이번 프로젝트에서는 Redux를 학습 목적으로 도입하여, 보일러플레이트 구조와 대규모 상태 관리 효율을 직접 비교하고 이해하며 상태 관리 경험을 확장하고자 했습니다.

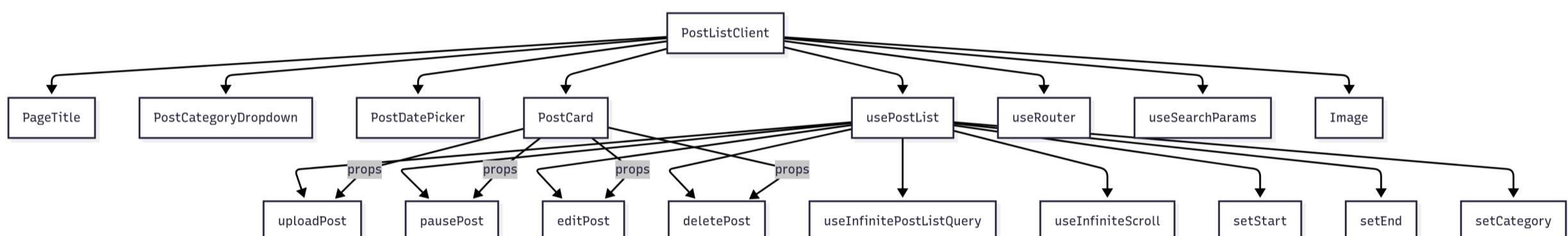
API Service Layer 패턴 설계

- 공통 API 로직을 axiosInstance.ts로 분리하여 유지보수성과 재사용성을 높였습니다. 이를 통해 (개발/배포)환경별 API 분리 설계가 용이해지고 토큰 갱신 로직의 재사용, 요청/응답 간의 timeout 설정 공유가 가능해졌습니다.



View - Logic 관심사 분리

- View와 Logic을 명확히 분리하여 단일 책임 원칙을 강화했습니다. 상태 관리가 필요 없는 경우 props를 활용하여 불필요한 전역 상태 사용을 방지하고 Logic은 상수, 유ти리티, 비즈니스 로직으로 나누어 관리함으로써 유지보수성과 가독성을 개선했습니다.



SSR 전환으로 초기 렌더링 성능 및 UX 개선

어떤 문제가 있었는가?

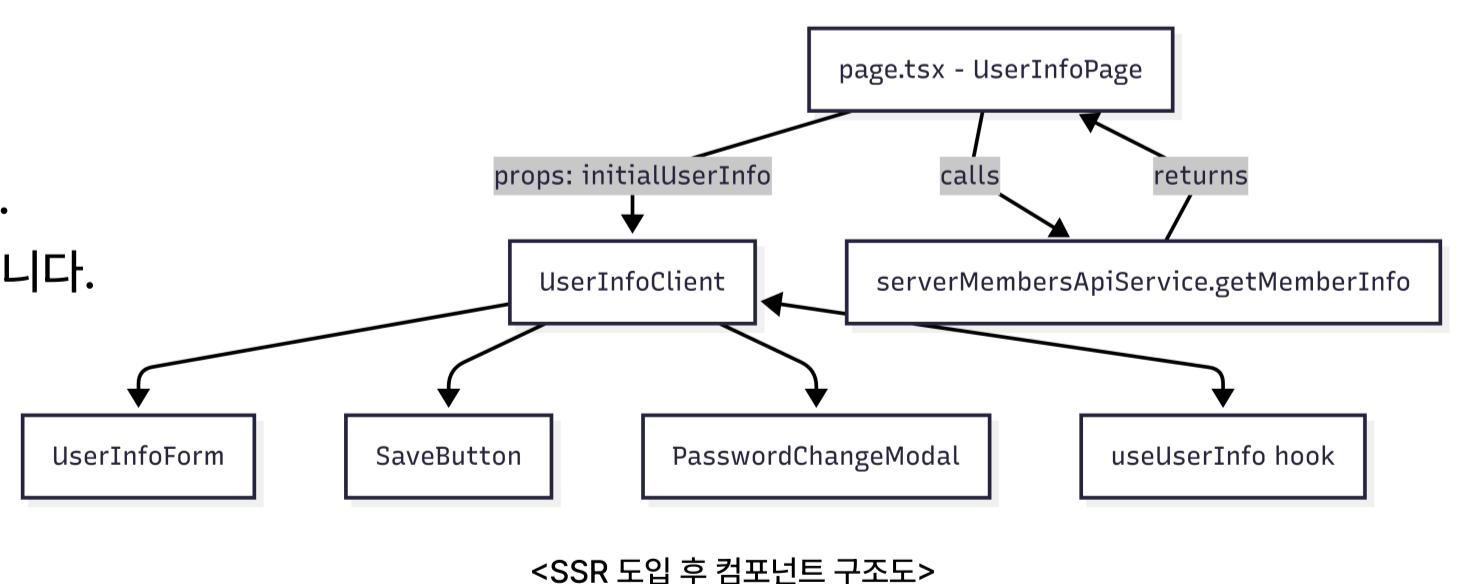
- マイ페이지의 저장하기 버튼은 사용자가 특정 항목을 수정했을 때만 활성화되도록 구현하려 했습니다. 그러나 CSR(Client Side Rendering) 방식에서는 상태 변수의 초기값을 빈 문자열("")로 설정하고, useEffect에서 API 응답으로 상태를 업데이트하는 구조 때문에, 초기 렌더링 시점에서 상태 변경이 수정 사항으로 감지되어 버튼이 항상 활성화되는 문제가 발생했습니다. 초기값을 API 응답으로 설정하는 과정과 변경 감지를 위한 커스텀 Hook의 구조가 맞지 않아 복잡성이 증가했습니다.

어떻게 해결했는가?

- SSR(Server Side Rendering) 방식을 선택했습니다. 서버 컴포넌트에서 react-query의 prefetchQuery를 활용해 API를 호출한 후 응답 데이터를 dehydrate하여 클라이언트 컴포넌트에서 캐시된 데이터를 활용하도록 구조를 변경했습니다. 이를 통해 초기 렌더링 시점에 사용자별 데이터를 상태값으로 설정할 수 있었고, 이후 상태 변화 감지도 정확하게 동작하여 저장하기 버튼의 활성화 조건을 정상적으로 구현할 수 있었습니다.

성과

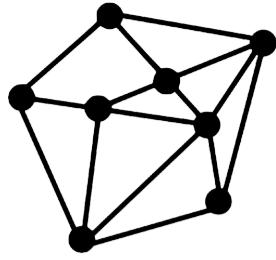
- 컴포넌트 깜박임 현상이 해결되었습니다.
- 초기 렌더링 시점에서 상태값이 정확히 반영되어 UX가 개선되었습니다.
- Lighthouse를 통해 확인한 결과 SEO 점수 100점을 달성할 수 있었습니다.
- 초기 렌더링 속도를 192ms → 86ms로 약 55% 단축하였습니다.



<SSR 도입 후 컴포넌트 구조도>

기기 검증 기반 라우팅

- 접속 기기에 따라 사용자 경험을 최적화하기 위해 PC와 모바일 페이지를 분리 설계했습니다.
 - 모바일에서 PC 전용 URL 접근 시 → 안내 페이지 제공
 - PC에서 모바일 전용 URL 접근 시 → 앱 설치 유도 페이지 제공
- 이를 통해 기기별 접근성을 높이고, 사용자가 올바른 환경에서 서비스를 이용하도록 유도했습니다.



서비스

[알고리즘 코딩 배틀 플랫폼]

구성원

총 6인 (FE 3인 / BE 3인)

작업 기간

2025.01.06 ~ 2025.02.21 (6주)

프론트엔드 기술 스택

TypeScript

Zustand

WebSocket

TailwindCSS

React

Figma

주요 업무 및 상세 역할

- 개인전 매칭 페이지 구현
- 팀전 매칭 페이지 구현
- 랭킹 페이지 구현
- 채팅 및 게임 초대 기능 구현



프로젝트 회고

협업과 조율을 통한 프로젝트 성공 경험

- 팀장으로서 기한 내 업무 단계를 달성하기 위해 프로젝트를 빠르게 추진하는 과정에서 팀원들과 갈등이 발생한 경험이 있습니다. 업무 진행 속도와 접근 방식이 팀원마다 다르다는 점을 인지하고 이후에는 일정 관리와 업무 우선순위 조정, 참고 자료 제공 등 간접적인 지원과 조율을 통해 팀원들이 각자의 강점을 발휘할 수 있도록 돋는 방식으로 전환했습니다. 그 결과 프로젝트를 성공적으로 마무리할 수 있었고, 단순히 함께 일하는 것을 넘어 팀원 간의 화합을 바탕으로 최상의 결과를 만들어 내는 것이 진정한 협업임을 깨달을 수 있었습니다.

팀원과 함께 문제를 해결하며 프로젝트 품질을 지킨 경험

- 한 팀원이 프로젝트에 익숙하지 않은 기술로 코드를 작성하는 과정에서 AI가 생성한 코드를 그대로 활용하다 보니 프로젝트 구조와 융화되지 않는 문제가 발생했습니다. 마감 기한이 임박한 상황에서 프로젝트 전반을 이해하고 있던 팀장으로서 책임감을 가지고 해당 코드를 리팩토링하여 기능을 정상화하고 성능을 개선했습니다. 이를 통해 팀원의 노력과 프로젝트 목표를 모두 살리면서 팀 전체의 코드 품질을 높일 수 있었습니다.

기술 선정과 도입 배경

왜 Zustand를 사용하였는가?

- 처음 진행한 React 프로젝트였기 때문에, 러닝 커브가 낮고 보일러플레이트가 적은 Zustand를 상태 관리 도구로 선택했습니다.

왜 WebSocket을 사용하였는가?

- 실시간 게임 매칭과 팀원 간 코드 동시 편집 등 실시간 협업 기능이 필요했기에, 안정적 양방향 통신을 지원하는 WebSocket을 도입했습니다.

상태 접근 일원화로 실시간 데이터 안정성 확보

어떤 문제가 있었는가?

- WebSocket을 통해 유저 입장, 퇴장, 게임 시작 알림을 구독하면서 매칭 대기 인원 수를 실시간으로 갱신해야 했습니다.
하지만 일부 세션에서는 인원이 정상 갱신되지 않거나, 방장 전환이 원활하지 않은 문제가 발생했습니다.

원인은 무엇인가?

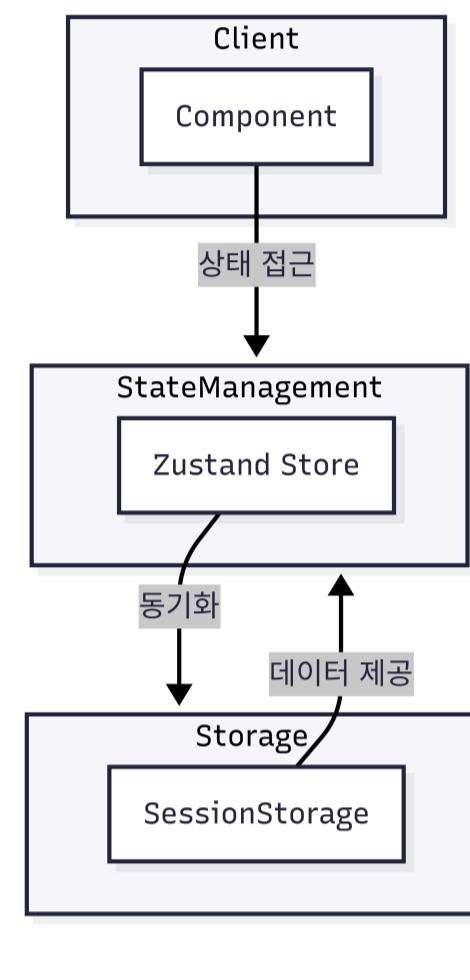
- 주요 원인은 다음과 같았습니다.
 - useEffect 실행 시점에 따른 비동기 흐름 문제
 - Zustand와 컴포넌트가 각각 sessionStorage에 접근하며 데이터 일관성 붕괴

어떻게 해결했는가?

- 모든 컴포넌트가 Zustand를 통해서만 상태에 접근하도록 구조를 재정리했습니다.
더불어 Zustand 내부에서만 sessionStorage와 동기화되도록 하여 데이터 흐름을 단일화했습니다.

성과

- 데이터 흐름 단일화로 실시간 데이터 처리 안정성을 확보하였습니다.
- 방장 전환 및 매칭 인원 동기화 오류가 재현되지 않았습니다.
- 코드 수정 시 문제 발생 지점 명확해져 유지보수의 효율이 향상되었습니다.



< 변경 후 구조 >

WebSocket 연결 단일화·모듈화로 효율성과 안정성 확보

어떤 문제가 있었는가?

- 기존에는 각 페이지 진입 시마다 WebSocket 핸드셰이크를 시도하고 언마운트 시 연결을 해제하는 방식이어서 불필요한 재연결과 리소스 낭비가 발생했습니다.

어떻게 해결했는가?

- 로그인 시점에 한 번만 WebSocket을 연결하고 로그아웃 시에만 해제하는 전역 연결 구조로 변경했습니다.
- 양방향 연결과 구독 등록/해제 로직을 모듈화하여 유지보수성을 높이고, 각 페이지 진입 시 필요한 데이터 흐름에 맞춰 구독 경로를 선택적으로 등록/해제하도록 설계했습니다.

성과

- WebSocket 연결의 효율성을 극대화 할 수 있게 되었습니다.
- 실시간 데이터 처리 안정성을 확보할 수 있었습니다.

복잡한 모달 로직을 추상화·중앙 관리하여 유지보수성과 확장성 개선

어떤 문제가 있었는가?

- 게임 결과(승/패)와 사용자의 선택(재도전, AI 코칭, 상대방 코드 보기 등)에 따라 모달이 달라져야 했는데, 점차 조건 분기와 UI 중복이 쌓이면서 모달 컴포넌트의 복잡도가 크게 증가했습니다.

어떻게 해결했는가?

- 공통 레이아웃과 로직을 추출해 재사용 가능한 모달 컴포넌트로 추상화하였습니다.
- 모달 상태를 Zustand 전역 상태로 관리하여 일관된 제어 구조 마련하였습니다.
- 다양한 모달을 상황별로 렌더링하기 위해 ModalManager.tsx라는 중간 관리 컴포넌트를 도입하여 컴포넌트 간 책임을 분리하였습니다.

성과

- 모달 관리 구조가 단순화되어 신규 모달을 추가할 때에도 유지보수 비용이 크게 줄었고
UI 중복을 최소화하여 코드 가독성과 확장성을 개선할 수 있었습니다.



나혼자간다



서비스

[나혼자간다]

구성원

총 2인 (FE 1인 / BE 1인)

작업 기간

2024.11.20 ~ 2024.12.03 (2주)

프론트엔드 기술 스택

JS JavaScript

Vue.js

pinia

Bootstrap

주요 업무 및 상세 역할

- 로그인/회원가입
- 여행 큐레이션 페이지 및 상세 모달 구현
- 블로그 여행 소식 모음 페이지 구현
- 동행 구인 게시판 구현
- 댓글 및 대댓글 구현
- AI 여행 플래너 페이지 구현
- 마이페이지 구현



프로젝트 회고

사용자 관점에서 바라본 UI/UX 설계 및 개선

1. Navbar hover UX 개선

Navbar의 블랙/화이트 대비 문제로 마우스 위치 인지가 어려웠기 때문에 특정 영역 hover 시 밑줄 표시 효과를 추가하여 사용자가 현재 선택한 메뉴를 직관적으로 확인할 수 있도록 개선했습니다.

2. 카드형 컴포넌트 사진 탐색

사용자가 상세 모달을 열지 않고도 여행지 사진을 확인할 수 있도록 외부 카드형 컴포넌트에서 이미지 슬라이드 기능을 구현하여 직관적인 콘텐츠 탐색을 지원했습니다.

3. 서비스 내 기능 연계 버튼

여행지 상세 모달에서 사용자가 블로그 후기 조회, 동행 모집 등 연관 기능으로 자연스럽게 이동할 수 있도록 페이지 링크 버튼을 배치하여 서비스 흐름을 매끄럽게 연결했습니다.

4. 무한 스크롤 + 스크롤 최상단 버튼

무한 스크롤이 적용된 여행 소식 페이지에서 사용자가 길게 스크롤한 후 빠르게 최상단으로 이동할 수 있도록 우측 하단에 '위로 가기' 버튼을 개발하여 탐색 편의성을 높였습니다.

5. AI 여행 플랜 마이페이지 연동

AI로 생성된 여행 플랜이 단순 조회에서 끝나지 않도록 마이페이지와 연동해 바로 저장·수정할 수 있는 기능을 구현하여 사용자가 즉시 계획을 관리할 수 있도록 UX를 개선했습니다.

6. 첫 방문자 안내를 위한 스크롤 기반 UX 설계

메인 페이지에 SnapScroll 기능을 구현하고 페이지 하단에 플로팅 화살표 버튼을 배치하여 사용자가 자연스럽게 스크롤하도록 유도했습니다. 이를 통해 서비스의 핵심 기능을 직관적으로 안내할 수 있었고 첫 방문 사용자가 별도의 안내 없이도 주요 기능과 구조를 이해할 수 있었습니다.