

안녕하세요, 프론트엔드 지원자 김덕진입니다.



경기도 오산시 동부대로 332-14

ghaehfdl71@gmail.com

<https://github.com/deokjin25>

1998. 07. 14

+82-10-3379-4250

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

개발자로서의 가치관

- 비즈니스 로직을 분리하고 코드를 모듈화하는 등 리팩토링 하는 과정을 좋아합니다.
- Trouble Shooting은 항상 기록하고 더 나은 해결 방법을 모색합니다.
- 나만 이해하는 코드가 아닌, 모두가 쉽게 이해할 수 있는 코드와 설명을 지향합니다.
- 오늘의 코드는 내일의 레거시가 된다는 마음으로 항상 신중하고 깔끔하게 코드를 작성합니다.
- 기능 구현보다 “좋은 구조”를 고민하는 과정을 즐깁니다.
- 새로운 기술을 단순히 사용하는 데 그치지 않고, 왜 필요한지 이해하며 적용하려고 노력합니다.
- 문제 해결 과정에서 얻은 인사이트를 문서화하여 팀의 자산으로 남기려 합니다.

기술 스택



HTML



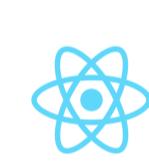
css



JavaScript



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

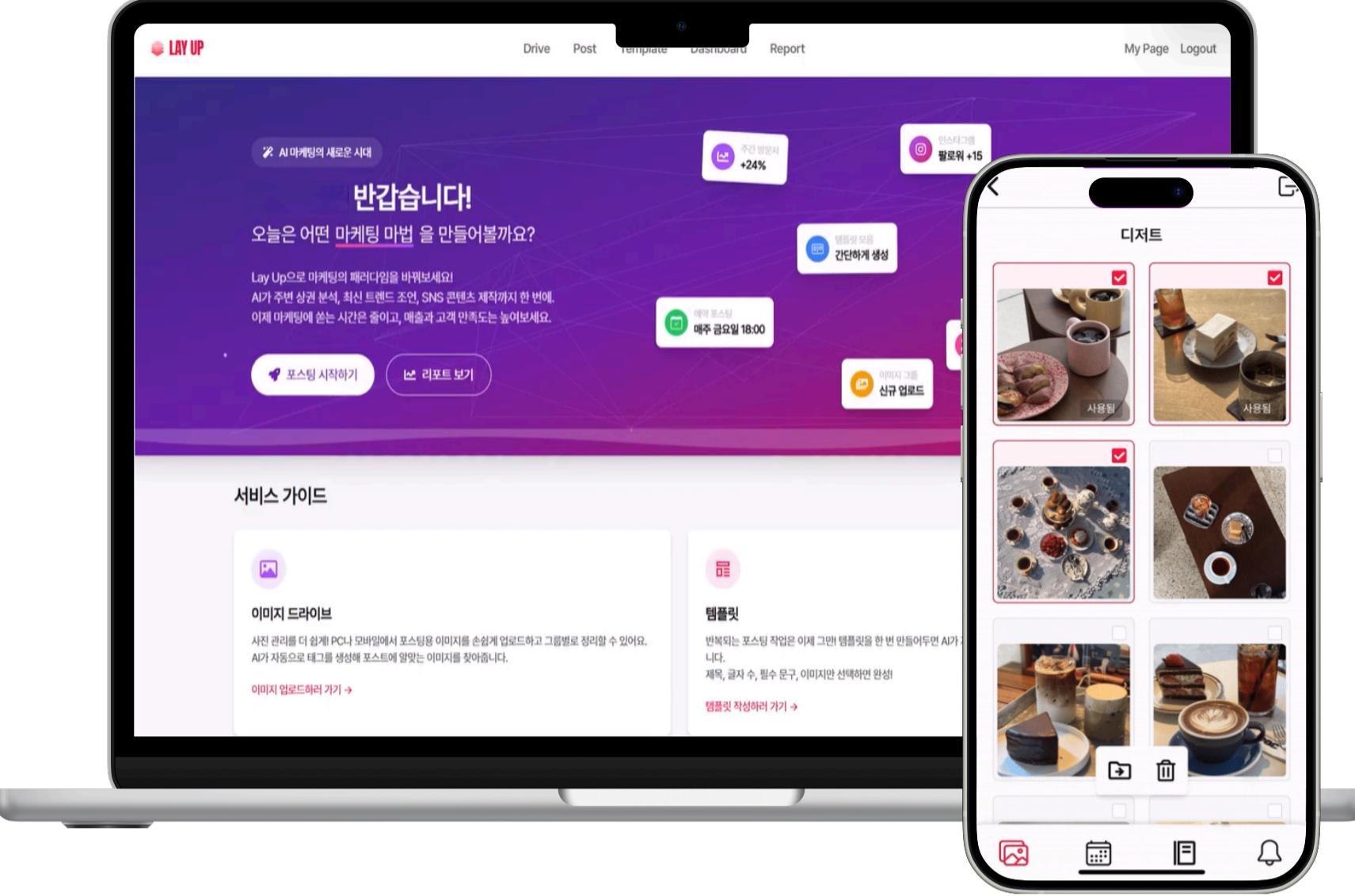
- 호텔관광경영학과 / AI 융합학사 (3.75/4.5)

경력 사항

해당 없음 (신입)



Lay Up



서비스 📊 🎯

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

작업 기간

2025.04.14 ~ 2025.05.21 (1개월)

기술 스택

TS TypeScript

N NextJS

R Redux

T TailwindCSS

PWA PWA

R React

RQ 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

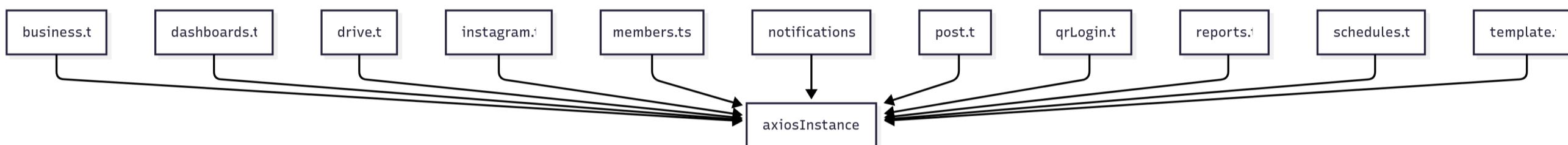
- View - Logic 관심사 분리
- 기기 검증 기반 라우팅
- 수정 사항 변경 감지

기술 선정과 도입 배경

- Next.js: SSR 학습을 위해 React 문법을 모두 지원하는 프레임워크인 Next.js를 선택하였습니다.
- Redux: 전역 상태를 체계적으로 관리하고, 복잡한 상태 변경 흐름을 예측 가능하게 만들기 위해 사용하였습니다.
- PWA: 모바일 기기에서도 서비스 일부 기능을 활용할 수 있도록 React와 PWA를 활용해 별도 모바일 프로젝트를 구성했습니다.
- React-Query: server state 와 client state를 분리하고 API 캐시를 통해 최적화하기 위해 사용하였습니다.
- TypeScript: 타입을 정확히 명시하여 프로젝트의 안정성을 높이기 위해 사용하였습니다.

API Service Layer 패턴 설계

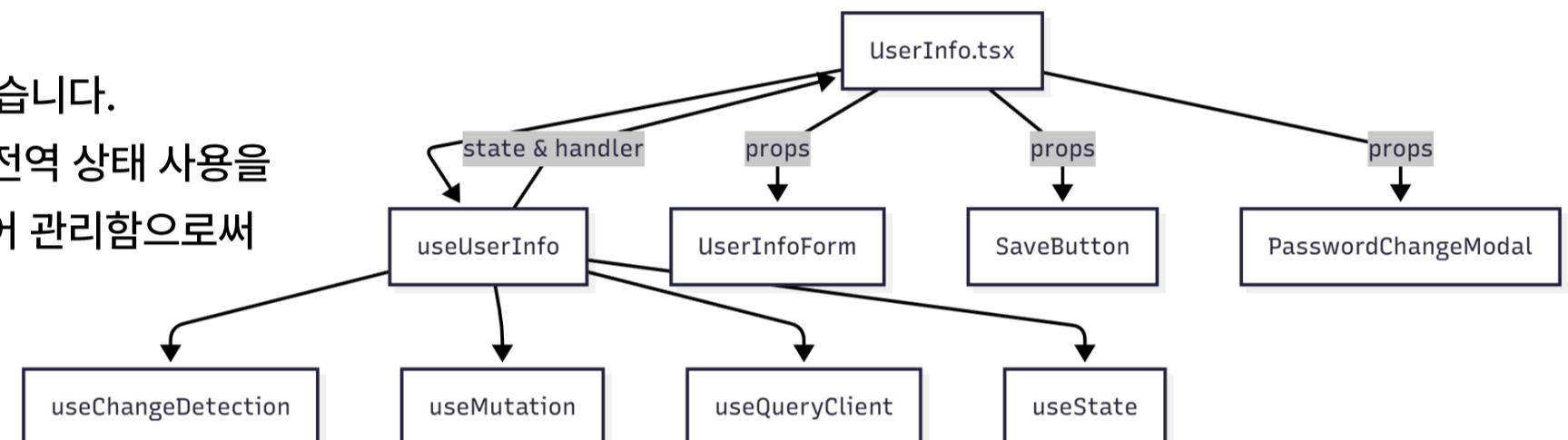
공통 API 로직을 axiosInstance.ts로 분리하여 유지보수성과 재사용성을 높였습니다. 이를 통해 (개발/배포)환경별 API 분리 설계가 용이해지고 토큰 갱신 로직의 재사용, 요청/응답 간의 timeout 설정 공유가 가능해졌습니다.



View - Logic 관심사 분리

View와 Logic을 명확히 분리하여 단일 책임 원칙을 강화했습니다.

전역 상태 관리가 불필요한 경우 props를 활용하여 과도한 전역 상태 사용을 방지하고 Logic은 상수, 유틸리티, 비즈니스 로직으로 나누어 관리함으로써 유지보수성과 가독성을 개선했습니다.



SSR 전환으로 초기 렌더링 성능 및 UX 개선

어떤 문제가 있었는가?

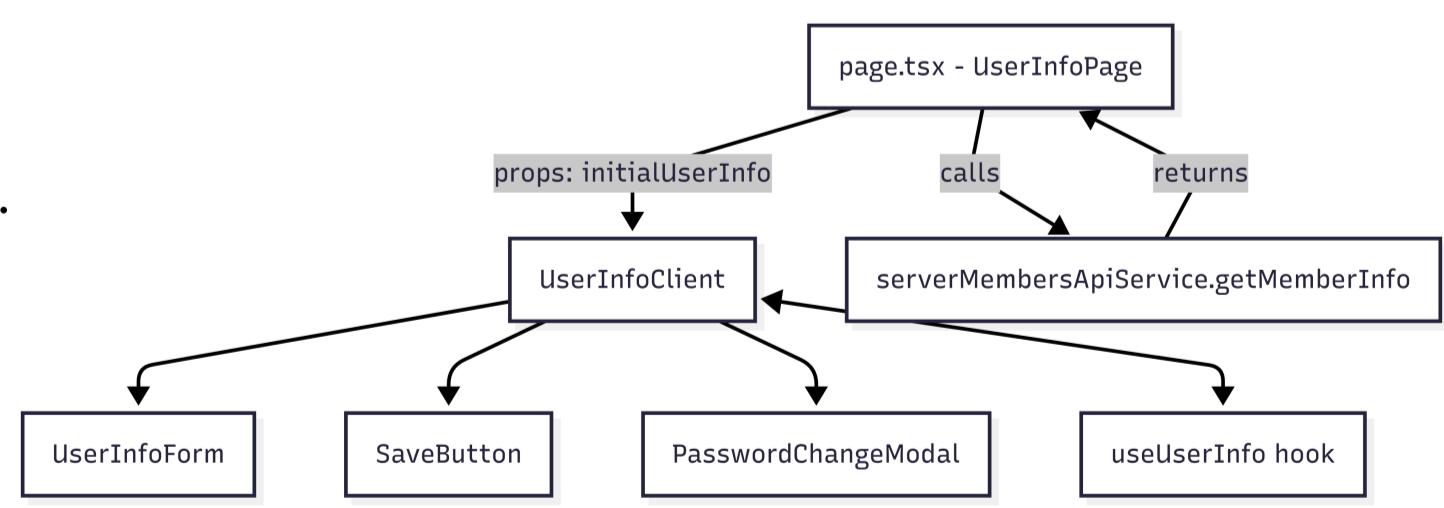
- CSR(Client Side Rendering) 방식에서는 상태 변수의 초기값을 빈 문자열("")로 설정하고 useEffect에서 API 응답으로 상태를 업데이트하는 구조였기 때문에 초기 렌더링 시점에서의 상태 변경으로 인한 컴포넌트 깜박임 현상이 발생하였습니다.

어떻게 해결했는가?

- SSR(Server Side Rendering) 방식을 채택했습니다. 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.

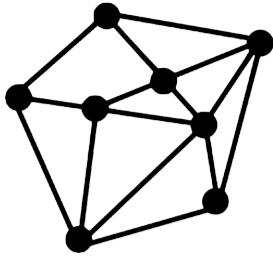
성과

- 컴포넌트 깜박임 현상이 해결되었습니다.
- 초기 렌더링 시점에서 상태값이 정확히 반영되어 UX가 개선되었습니다.
- Lighthouse SEO 점수 100점을 달성할 수 있었습니다.



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.



Altteul



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

WebSocket

React

Zustand

TailwindCSS

Figma

주요 업무 및 상세 역할

- 개인전 매칭 페이지 구현
 - 동적 리더 할당: 방장 퇴장 시 자동 권한 이양
 - 실시간 대기열 업데이트: ENTER/LEAVE 소켓 이벤트 처리
 - 타이머 동기화: 서버 기준 카운트다운으로 공정성 보장
- 팀전 매칭 페이지 구현
 - 팀 구성 → 매칭 대기 → 게임 준비 단계별 채널 관리
 - 동적 팀 할당: 서버 응답에 따른 아군/적군 자동 분류
 - 매칭 중 취소: 대기 중 팀 구성으로 롤백 기능
- 랭킹 페이지 구현
 - react-intersection-observer 라이브러리의 useInView 훅을 사용하여 무한 스크롤 구현
 - 닉네임 검색 및 유저 등급, 사용 언어 필터링 기능
- 채팅 및 게임 초대 기능 구현
 - 공통 레이아웃과 로직을 추출해 재사용 가능한 모달 컴포넌트로 추상화
 - 모달 상태를 전역 상태로 관리하여 일관된 제어 구조 마련
 - 중앙 집중화를 위해 중간 관리 컴포넌트(ModalManager.tsx)를 도입하여 다양한 모달을 상황별로 렌더링

기술 선정과 도입 배경

- React: 컴포넌트 기반 구조를 통해 재사용성과 유지보수성을 높이고 동적인 사용자 인터페이스를 효율적으로 구현하기 위해 사용하였습니다.
- Zustand: 전역 상태를 보다 가볍고 직관적으로 관리하기 위해 사용하였습니다.
- TypeScript: 타입을 정확히 명시하여 프로젝트의 안정성을 높이기 위해 사용하였습니다.
- WebSocket: 게임 매칭과 팀원 간 코드 동시 편집 등 실시간 협업 기능이 필요했기에 양방향 통신을 지원하는 WebSocket을 도입했습니다.
- Tailwind CSS: 직관적인 유ти리티 클래스 기반 스타일링을 통해 빠르게 UI를 구성하고 일관된 디자인 시스템을 유지하고자 사용하였습니다.

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?

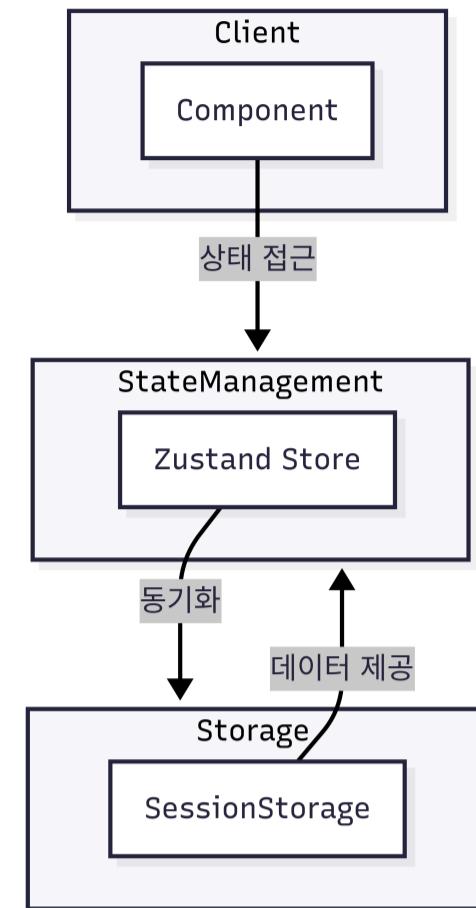
- 주요 원인은 다음과 같았습니다.
 - useEffect 실행 시점에 따른 비동기 흐름 문제
 - Zustand와 컴포넌트가 각각 sessionStorage에 접근하며 데이터 일관성 붕괴

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 지점이 명확해져 유지보수의 효율이 향상되었습니다.



< 변경 후 구조 >

WebSocket 연결 단일화·모듈화로 효율성과 안정성 확보

어떤 문제가 있었는가?

- 각 페이지 진입 시마다 WebSocket 핸드쉐이크를 시도하고 언마운트 시 연결을 해제하는 방식으로 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 연결 안정성 보장**
 - 자동 재연결: 네트워크 불안정 상황에서도 안정적인 연결 유지
 - 최대 재시도 횟수 제한: 무한 재연결 방지로 리소스 최적화
- **구독 관리의 고도화**
 - 구독 상태 영속화: sessionStorage를 활용하여 새로고침 시에도 구독 복구
 - 중복 구독 방지: Map 자료구조로 효율적인 구독 관리
 - 단일 핸들러: 채팅 관련 모든 소켓 메시지를 중앙에서 통합 처리
- **경로 기반 메시지 필터링**
 - 컨텍스트 인식: 특정 페이지에서만 필요한 소켓 처리
 - 불필요한 연산 방지: 관련 없는 페이지에서 리소스 절약
- **메모리 누수 방지 설계**
 - 자동 정리: 컴포넌트 언마운트 시 구독 해제
 - 메모리 최적화: 불필요한 구독 누적 방지

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 모달의 UI가 달라져야 했는데 점차 조건 분기와 UI 중복이 쌓이면서 모달 컴포넌트의 복잡도가 크게 증가하고 신규 모달 생성 시 코드 작성에 많은 시간이 소요되었습니다.

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

- 디버깅에 소요되는 시간을 1시간에서 10분으로 약 85% 단축
- 신규 모달 추가에 드는 작업 시간을 2시간에서 30분으로 약 75% 단축



나 혼자 간다



서비스

[혼행인을 위한 여행 편의 제공 서비스]

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.