

Intro

안녕하세요, 정성혜입니다.

1999. 06. 13

skyblue84411@gmail.com / 01058413253

사용자를 위한 웹 프론트엔드 개발자로 성장하고 싶습니다.

작은 변화가 사용자 경험에 큰 영향을 미치는 프론트엔드 개발의 특성을 좋아합니다. 버튼의 위치, 색상, 애니메이션 같은 요소 하나하나가 사용자 편의성을 결정한다고 생각하며, 이를 고민하고 개선하는 과정에서 보람을 느낍니다. 단순한 기능 구현을 넘어 사용자의 불편을 해소하고, 더 나은 경험을 제공하는 개발자가 되고 싶습니다.

데이터 활용을 통한 맞춤형 UI/UX 개선 경험이 있습니다.

사용자의 불편을 해결하고 더 나은 경험을 제공하는 것에 집중합니다. 단순한 기능 구현을 넘어 데이터를 활용하여 맞춤형 UI를 설계하고, 최적화된 탐색 경험을 제공하는 것을 중요하게 생각합니다. 헬로핏 프로젝트에서 데이터 전처리와 필터링을 개선한 경험이 이를 잘 보여줍니다.

작업에 몰입하며 지속적으로 성장합니다.

세부 사항까지 신경 쓰는 꼼꼼한 성향을 가지고 있지만, 프로젝트의 전체 흐름도 놓치지 않기 위해 코어 타임을 정하고 작업과 휴식을 분리하는 등 효율적으로 집중하는 방법을 고민하며 성장하고 있습니다. SQLD를 취득하며 백엔드와 원활한 협업을 준비하는 등 필요한 역량을 스스로 찾아 배우는 태도가 강점입니다.

희망 직무

프론트엔드 엔지니어

주요 기술

TypeScript / React / Next.js / Tailwind CSS

Zustand / React Query

Figma

인턴

code-it

2024. 12 - 2025. 02
Product team (FE)

사내 리소스 관리 서비스(Codeit Resources) 유지보수

URL 단축기(URL Shortener) 기능 개발

Cemware

2023. 09 - 2023. 12
R&D (FE)

수학 탐구 소프트웨어 AlgeoMath 유지보수

회사 내 Mathcore API를 활용한 웹 기반 디지털 교과서 제작

Link

<https://velog.io/@sparklhae>

<https://github.com/eqypo9>

<https://sparklhae.netlify.app/>

Internship

AWS Amplify 기반 데이터 모델링을 설계하여 유지보수성과 확장성을 고려했습니다.

Google Calendar API 유지보수로 일정 관리의 유연성 향상시켰습니다.

좌석 예약 시스템에 반복 예약 기능 추가로 사용자 편의성 증대시켰습니다.

사내 전용 URL Shortener 개발로 내부 업무 프로세스를 효율화하였습니다.

code-it

직무

Product Team (FE)

주요 기술

TypeScript / Next.js (v.14) / React (v.18)

Jotai / React Query / react-hook-form

AWS Amplify

경험

1. AWS Amplify를 활용한 데이터 모델링 및 테이블 설계

- AWS Amplify를 사용하여 데이터 모델링을 설계하고,
이를 기반으로 테이블을 구축
- 클라우드 기반으로 관리하여 유지보수성과 확장성을 고려한 설계

2. Google Calendar API 유지보수 및 기능 개선

- 기존 Google Calendar API 연동 기능을 개선하여,
회의실이 없는 일정도 불러올 수 있도록 기능 확장
- 내부 일정 관리의 유연성을 높이고, 예약 시스템의 완성도를 향상

3. 좌석 예약 시스템에 반복 예약 기능 추가

- 매일, 매주, 매월 등 커스텀 반복 예약 기능을 구현하여
사용자 편의성을 대폭 개선
- 사용자의 반복적인 좌석 예약 작업을 줄이고, 업무 효율성을 증대

4. URL Shortener 개발 및 사내 최적화

- Bitly 대신 사내에서 사용할 수 있는 자체 URL 단축 서비스 개발
- URL 그룹 지정 기능 추가
→ 사용자가 그룹을 지정하고 그에 해당하는 링크를 관리할 수 있도록 개선
- 일괄 선택 및 삭제 기능 구현
→ 필요 없는 링크를 한 번에 정리할 수 있도록 최적화
- 사내에서 직접 관리 가능한 단축 URL 서비스 제공으로 업무 생산성 향상

Internship

Git을 활용한 코드 리뷰 및 리팩토링을 수행하며 협업 프로세스를 경험했습니다.

웹 프론트엔드 성능 최적화, API 연동, UI/UX 개선 등 실무 역량을 키웠습니다.

AlgeoMath의 3D UI 성능 개선 및 유지보수를 하며 사용자에게 실질적인 가치를 부여했습니다.

Cemware

직무

R&D (FE)

주요 기술

TypeScript / Next.js / React

Unity

경험

1. AlgeoMath의 3D UI 유지보수 및 개선

- 수학 교육용 소프트웨어인 AlgeoMath의 3D 모델링 UI 조정 및 최적화
- 사용자 경험을 고려한 UI 조정 및 인터랙션 개선을 통해 직관적인 학습 환경 제공

2. TypeScript 도입 및 생산성 향상

- TypeScript를 처음 사용하면서 학습하고, 이를 실무에 적용
- TypeScript의 정적 타입 검사, 코드 자동 완성, 유지보수성 향상 등의 장점을 활용하여 코드 품질 개선
- 코드의 예측 가능성을 높이고, 런타임 오류를 줄여 개발 생산성을 증대

3. Mathcore API를 활용한 디지털 교재 개발

- 사내 Mathcore API를 활용하여 웹 기반 디지털 교재 개발
- Mathcore API의 수학 연산 기능을 활용하여 보다 직관적이고 시각적인 교육 자료 제공
- 복잡한 수학 연산을 API로 처리하여 웹 애플리케이션의 성능 및 유지보수성을 개선

1. 프로젝트 개요

기존 문제점

- 사내에서는 외부 URL 단축 서비스(Bitly)를 사용하여 링크를 관리했음
- URL이 많아질수록 관리가 어려워지고, 그룹별로 분류하는 기능이 없어 불편함 발생
- 외부 서비스 의존도가 높아 보안 및 유지보수 측면에서 비효율적

해결 방법

- 사내에서 자체적으로 사용할 수 있는 URL Shortener 서비스 개발
- 그룹별 URL 관리, 일괄 삭제 기능 추가로 효율적인 관리 지원
- AWS Amplify 및 React, TypeScript 활용

2. 주요 기능 및 구현 방식

2.1 URL 단축 기능 (Shortening URLs)

기능 설명

- 사용자가 여러 개의 URL을 입력하면, 일괄적으로 단축 URL을 생성
- 잘못된 형식의 URL이 포함되어 있을 경우, 유효성 검사 후 오류 메시지 제공
- Jotai 전역 상태 관리 라이브러리를 활용하여 선택된 URL 상태를 관리

구현 로직

- 사용자가 입력한 URL 리스트를 배열로 변환
- 유효성 검사(isValidUrl)를 통해 잘못된 URL이 있는지 확인
- transformUrlsBatches 함수를 이용해 한 번에 50개씩 배치 처리
- 변환된 URL을 상태로 저장하고, 성공 메시지를 표시

2.2 URL 그룹 관리 (Grouping URLs)

기능 설명

- URL을 B2C, 내부팀 등 그룹별로 분류하여 관리 가능
- 그룹을 생성, 수정, 삭제할 수 있으며, 중복 방지를 위한 검증 포함
- 알파벳 순으로 정렬하여 사용자가 직관적으로 선택 가능

구현 로직

- getGroupList: AWS Amplify에서 그룹 목록을 가져오고, roopFunction을 사용하여 모든 데이터를 로드
- isGroupNameExists: 특정 그룹 이름이 존재하는지 확인하는 함수
- createShortenerGroup: 그룹을 생성하고, 기존에 같은 이름이 있으면 오류 메시지 반환
- updateGroupName: 그룹 이름 변경, 기존 그룹과 중복되지 않도록 체크
- deleteShortenerGroup: 그룹을 삭제

3. 도입 효과

- 외부 서비스(Bitly) 의존도 감소 → 사내 자체 URL Shortener 서비스 제공
- URL 관리 효율성 증가 → 그룹별 관리 및 일괄 삭제 기능 추가
- 사용자 경험 향상 → 무한 스크롤, 검색 기능, 필터링 지원
- 보안 및 유지보수 개선 → 사내에서 직접 관리 가능

4. 기술 스택

- Frontend: Next.js, React, TypeScript
- Backend: AWS Amplify
- State Management: React Query, Jotai
- UI Components: Tailwind CSS
- API & Data Handling: Axios, Google Calendar API

5. 배운 점

1. 사용자 중심의 UI/UX 개선이 개발의 핵심 요소임을 경험

- 기존 UI를 단순히 수정하는 것이 아니라, 사용자가 더 쉽게 이해하고 조작할 수 있도록 최적화하는 것이 핵심임을 깨달음

2. 상태 관리와 데이터 최적화의 중요성 학습

- Jotai와 React Query를 활용하여 전역 상태를 효율적으로 관리하는 방법을 익힘
- URL Shortener의 대량 데이터 처리를 최적화하면서, 비동기 데이터 요청과 캐싱 전략을 고려하는 법을 배움
- AWS Amplify 기반의 데이터 모델링을 설계하면서, 비동기 요청 처리 및 데이터 필터링 구조에 대한 이해도를 높임

3. API 활용 및 서비스 환경에서의 데이터 처리 방식 이해

- Google Calendar API 연동 유지보수를 하면서, 외부 API와의 통신을 최적화하는 법을 학습
- URL Shortener 개발을 통해 API 요청 및 응답 설계를 경험하고, 데이터 변환 로직을 효율적으로 다루는 방법을 익힘

4. 효율적인 데이터 로딩 및 성능 최적화 경험

- 무한 스크롤(Infinite Scroll)과 페이지네이션을 적용하면서, 대량의 데이터를 효율적으로 로드하는 전략을 학습
- 검색어 및 그룹별 필터링 기능을 추가하면서, 프론트엔드에서 데이터를 효과적으로 가공하는 법을 익힘
- Batch 처리(한 번에 50개씩 변환)를 통해 네트워크 요청을 줄이고 성능을 최적화하는 방식에 대해 고민

5. 협업과 지속적인 개선의 중요성

- 코드 리뷰와 팀원 간의 피드백을 통해 더 좋은 개발 방식과 최적화 기법을 학습
- 문제를 해결하는 과정에서 단순한 기능 구현이 아니라, 팀 내에서 지속적으로 논의하고 개선하는 것이 핵심이라는 점을 경험

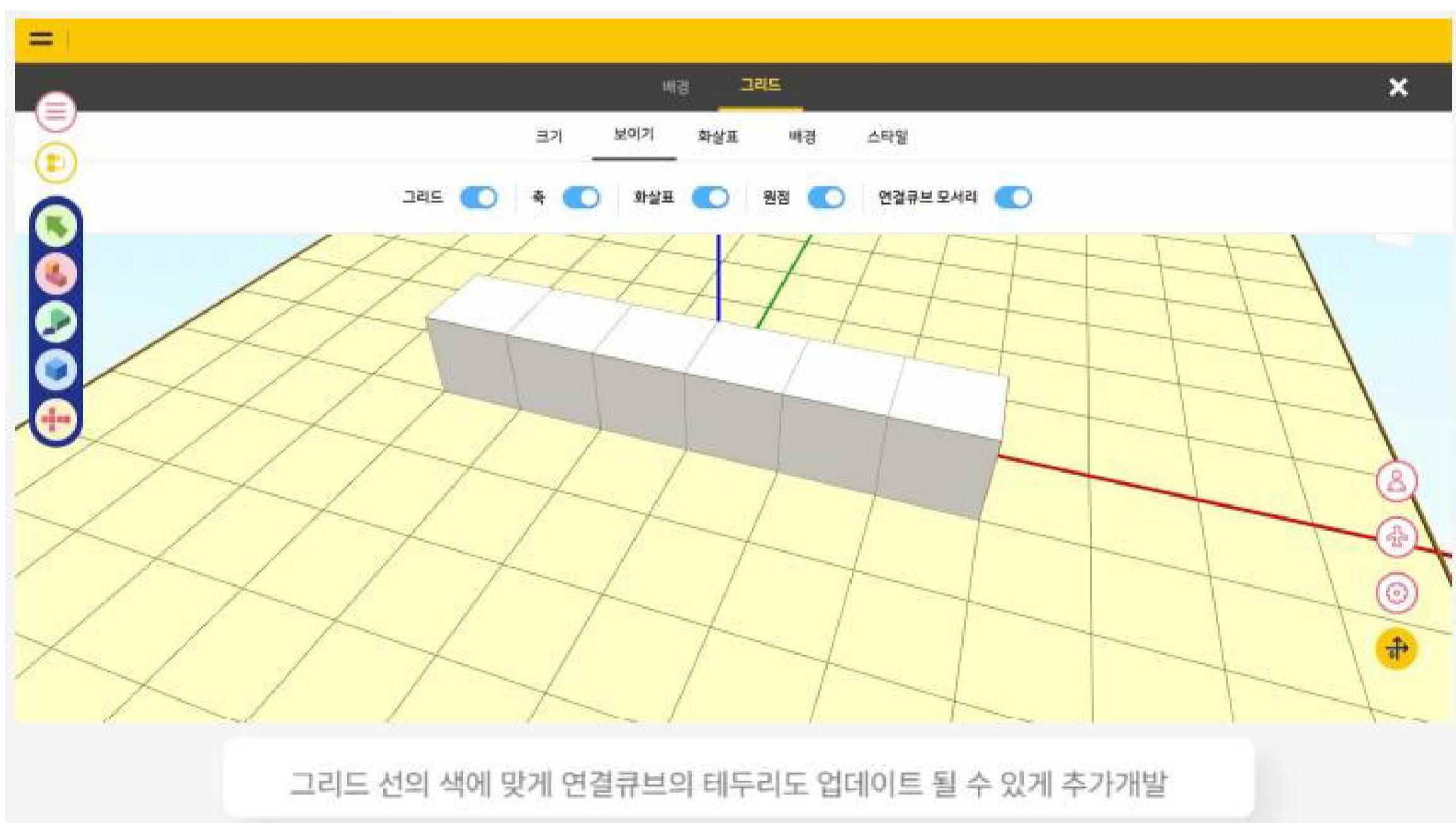
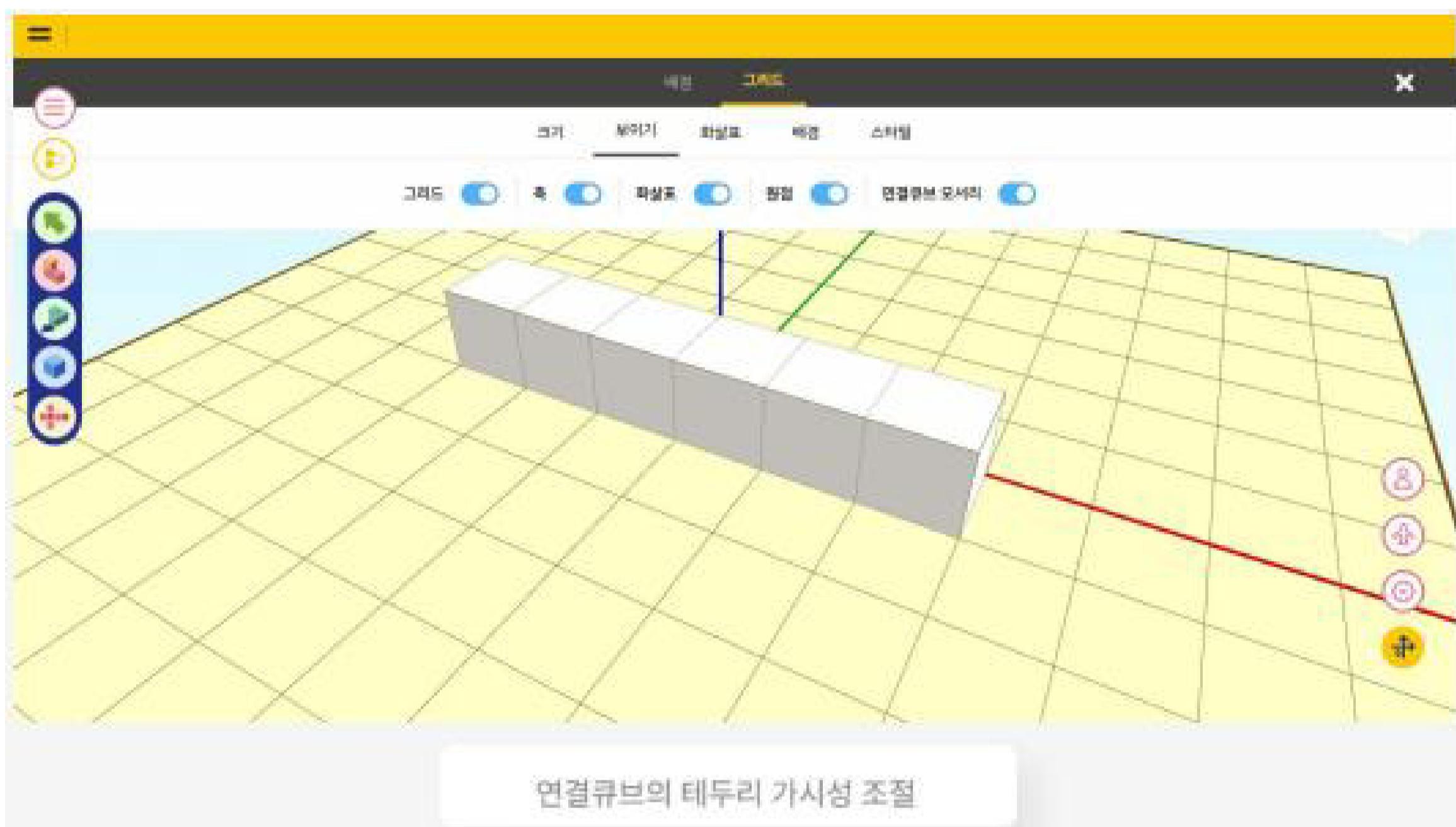
1. 수학 교육용 소프트웨어 AlgeoMath UI 개선

이용자 피드백을 반영하여 UI 개선

- 수학 교육용 소프트웨어 AlgeoMath의 UI를 사용성 중심으로 개편
- 이용자(교사 및 학생)들의 피드백을 바탕으로 가독성, 인터랙션, 배치 등을 개선
- 3D 모델링 UI를 조정하여 더 직관적이고 접근성이 높은 학습 환경 제공

배운 점

- 단순히 UI를 수정하는 것이 아니라, 사용자의 실제 학습 흐름을 고려한 UI 설계가 중요하다는 점을 배움
- 3D 모델링 UI의 조정 과정에서 입체적인 요소를 다루는 방식과 성능 최적화 기법을 익힘
- 사용자의 피드백을 수용하는 과정에서 협업과 반복적인 개선이 제품의 완성도를 높이는 핵심 요소라는 점을 경험

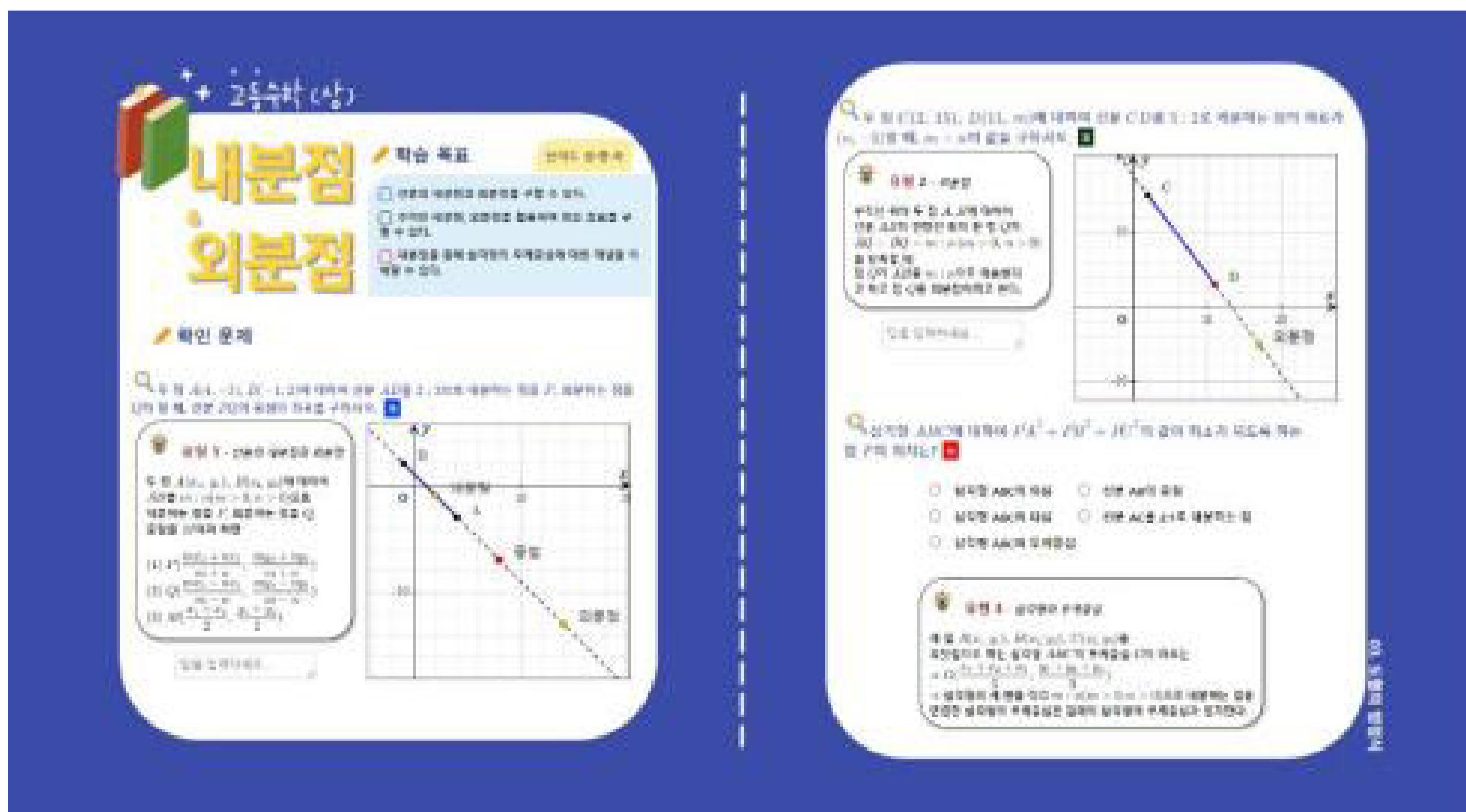
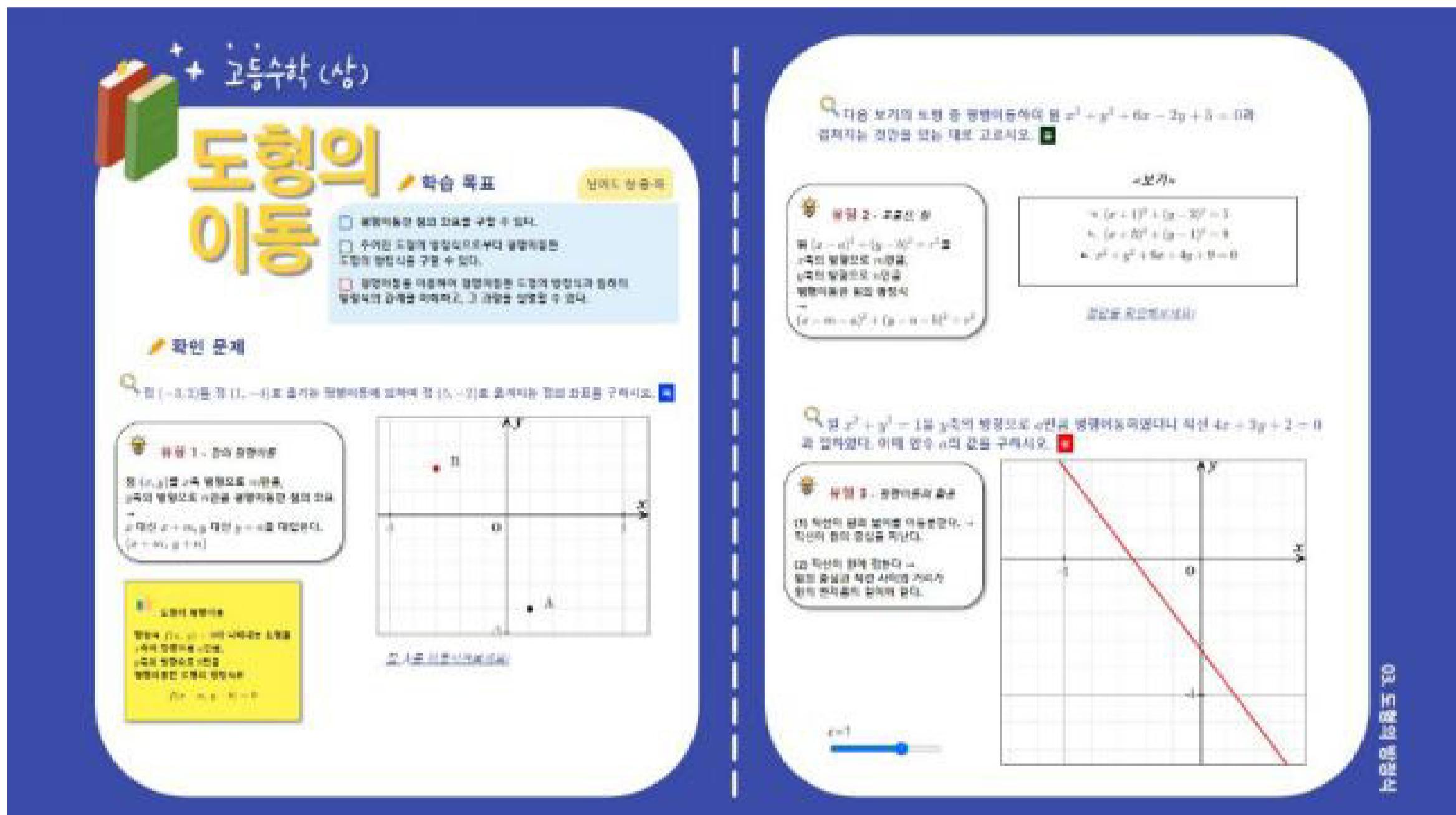


2. Mathcore API를 활용한 디지털 교과서 제작 (연구 프로젝트)

- 회사 내 Mathcore API를 활용하여 디지털 수학 교과서를 개발
- 좌표 이동, 그래프 변형, 함수 연산 등의 기능을 추가하여 학습 과정의 직관성을 높임
- 기존 종이 교재를 디지털화하여, 학생이 직접 조작하여 학습할 수 있는 환경을 구축

배운 점

- Mathcore API를 활용하여 수학적 개념을 인터랙티브하게 표현하는 방법을 익힘
- 기존의 정적인 교재를 디지털화하는 과정에서 사용자의 학습 경험을 개선하는 방향을 고민하는 것이 중요함을 깨달음
- 개발 과정에서 API의 기능을 단순히 사용하는 것뿐만 아니라, 효율적인 데이터 활용 및 성능 최적화를 고려해야 함을 배움



Projects

HELLOFIT

공공데이터를 활용하여 스포츠 이용권이 적용되는 시설을 쉽게 탐색할 수 있는 서비스입니다.

사용 기술

- TypeScript
- NextJS (v.15)
- React (v.18)
- React-Query, recoil
- SCSS
- Axios

[서비스 바로가기 \(party door\)](#)

[소스코드 보기 \(github\)](#)

Global Nomad

여행지에서 체험 상품을 쉽고 빠르게 예약할 수 있도록 돋는 서비스입니다.

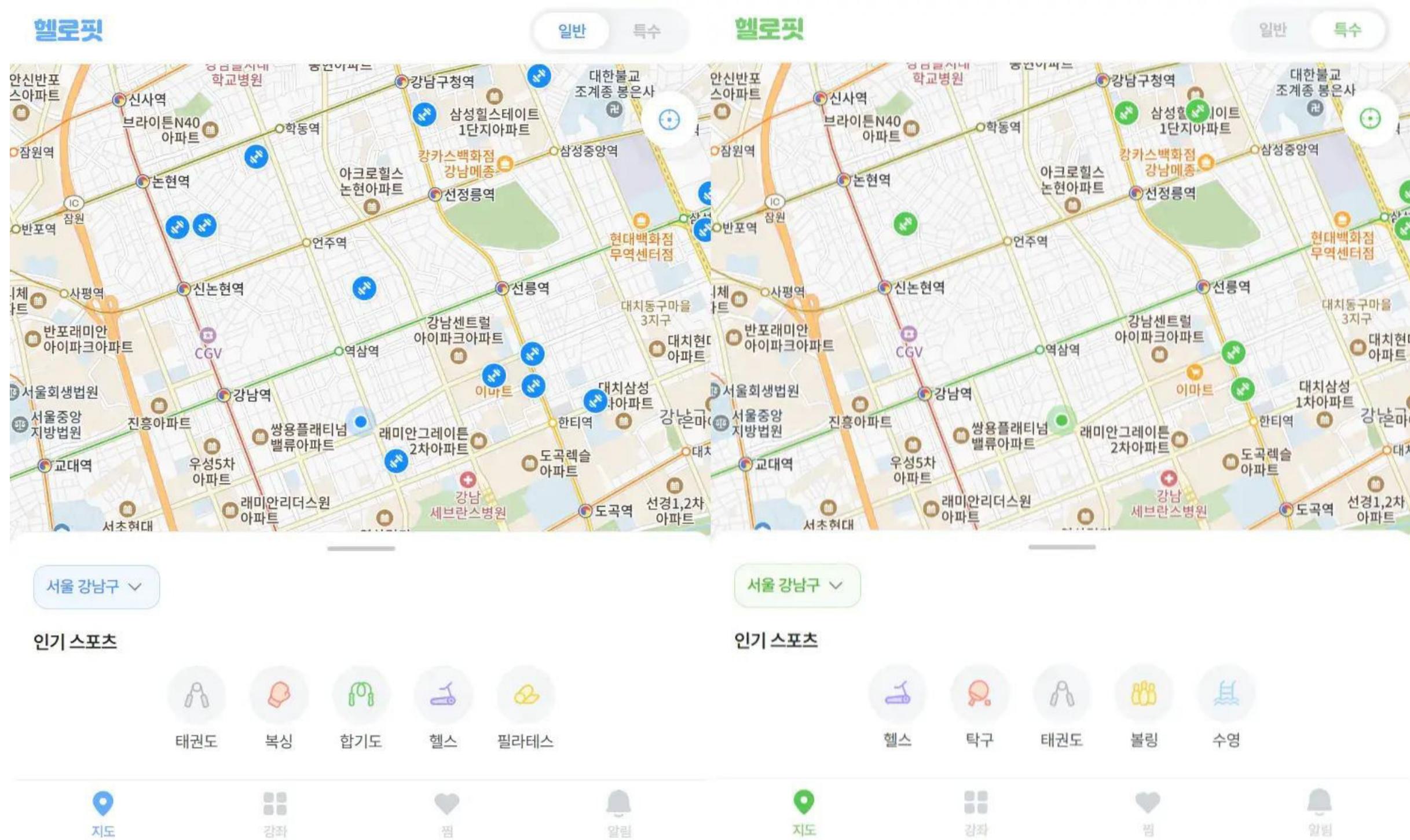
사용 기술

- TypeScript
- NextJS (v.14)
- React (v.18)
- React Query, Zustand
- TailwindCSS
- react-hook-form
- Axios

[서비스 바로가기 \(party door\)](#)

[소스코드 보기 \(github\)](#)

HELLOFIT



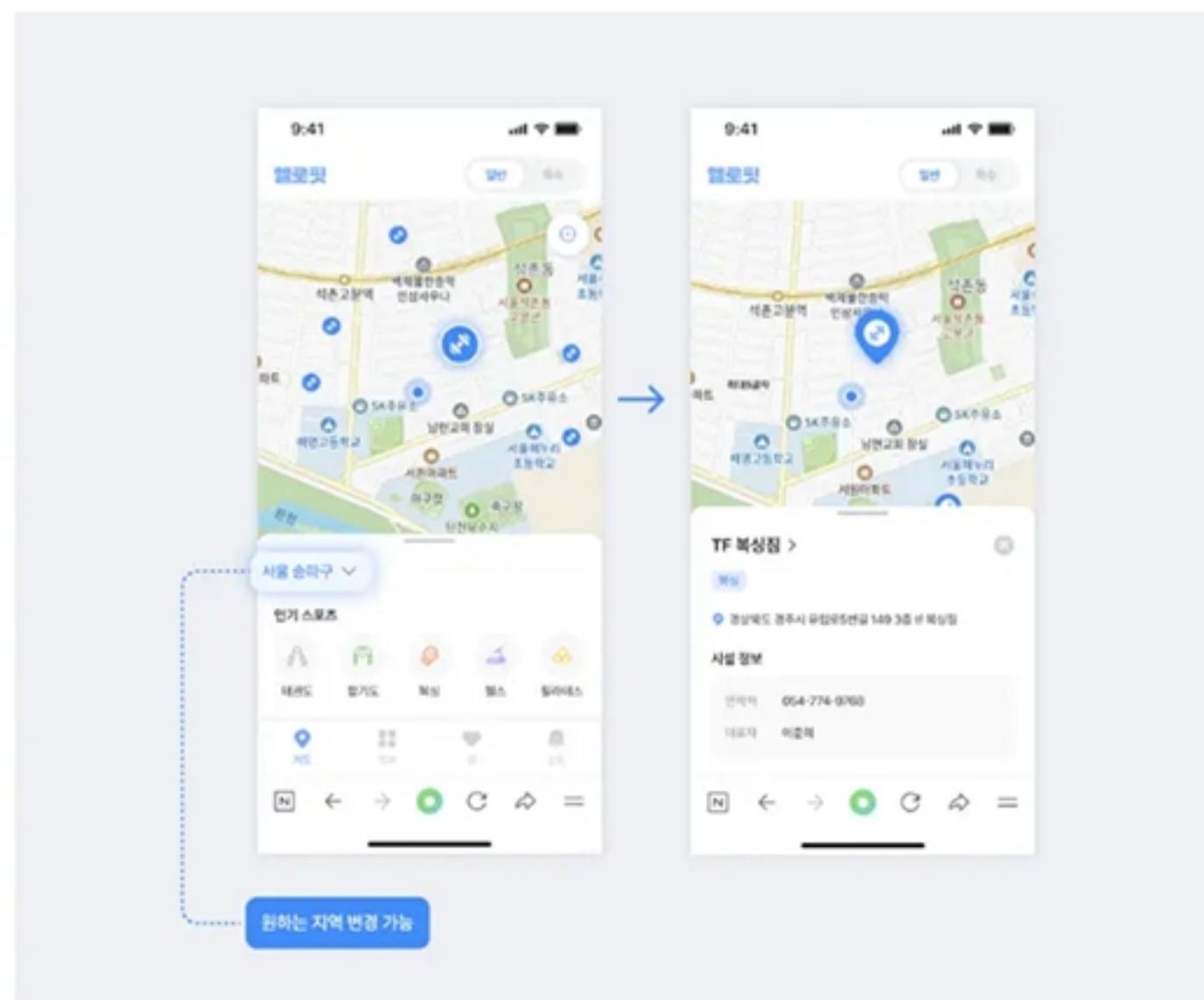
핵심 기여

- 공공데이터 API를 활용하여 스포츠 시설 데이터를 동적으로 로드하는 기능 구현
- NextAuth를 활용한 소셜 로그인 (Google, Kakao) 도입
- 사용자 경험을 고려한 UI/UX 개선: 지도 기반 탐색 최적화 및 필터링 기능 강화
- 카카오 지도 API와 Recoil을 결합하여 상태 관리를 최적화하여 성능 개선

주요기능

지도 기반 스포츠 시설 탐색

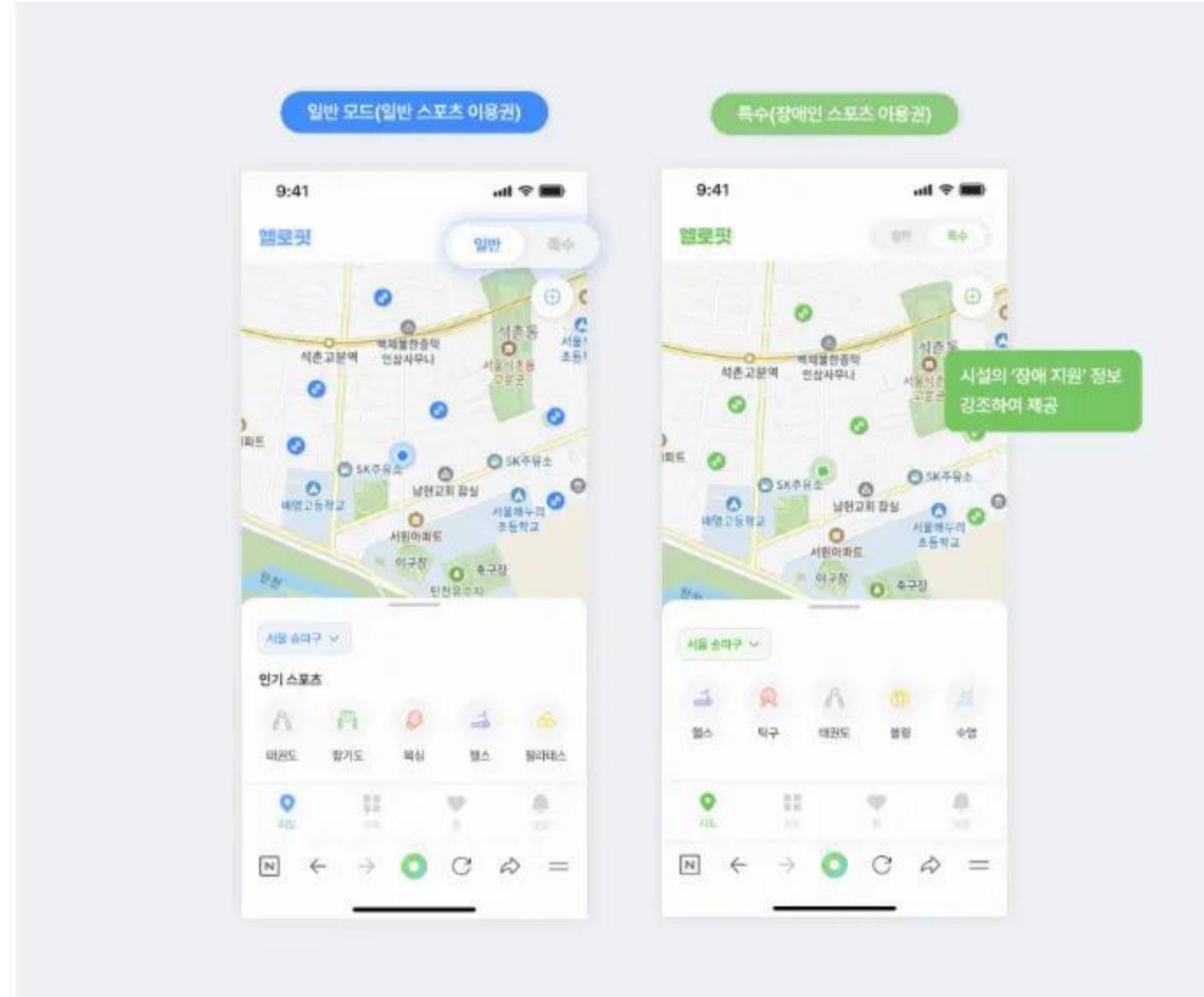
- 사용자의 현재 위치를 기반으로 가까운 스포츠 시설을 한눈에 확인할 수 있음
- 필터를 활용하여 원하는 시설을 더욱 빠르게 찾을 수 있음



HELLOFIT

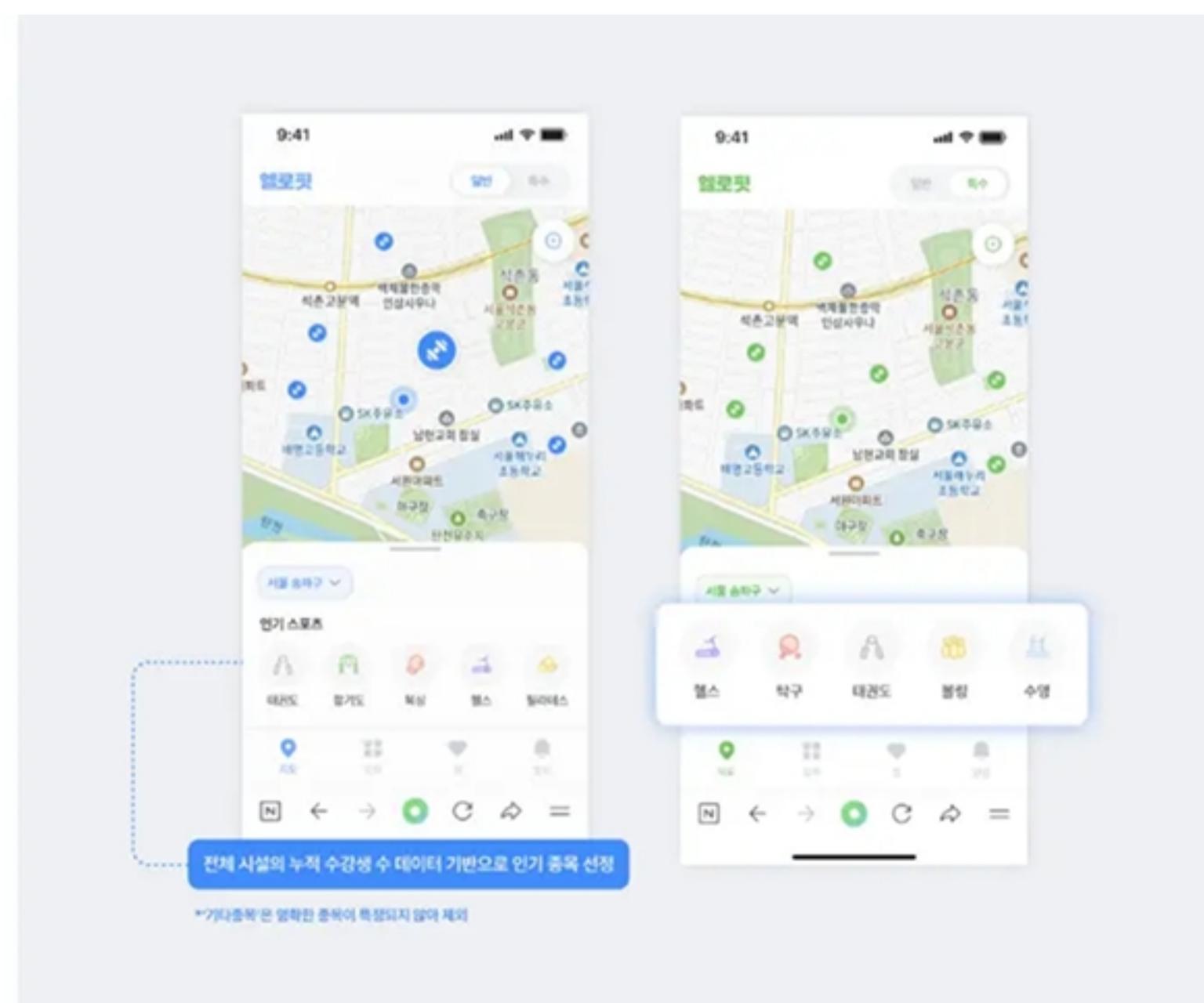
이용권 유형별 맞춤 정보 제공

- 일반 이용권과 장애인 이용권을 구분하여 맞춤형 정보를 제공
- 사용자는 자신의 이용권에 해당하는 강좌만 확인할 수 있어 검색 효율이 높아짐



누적 수강생 수 기반 인기 스포츠 종목 추천

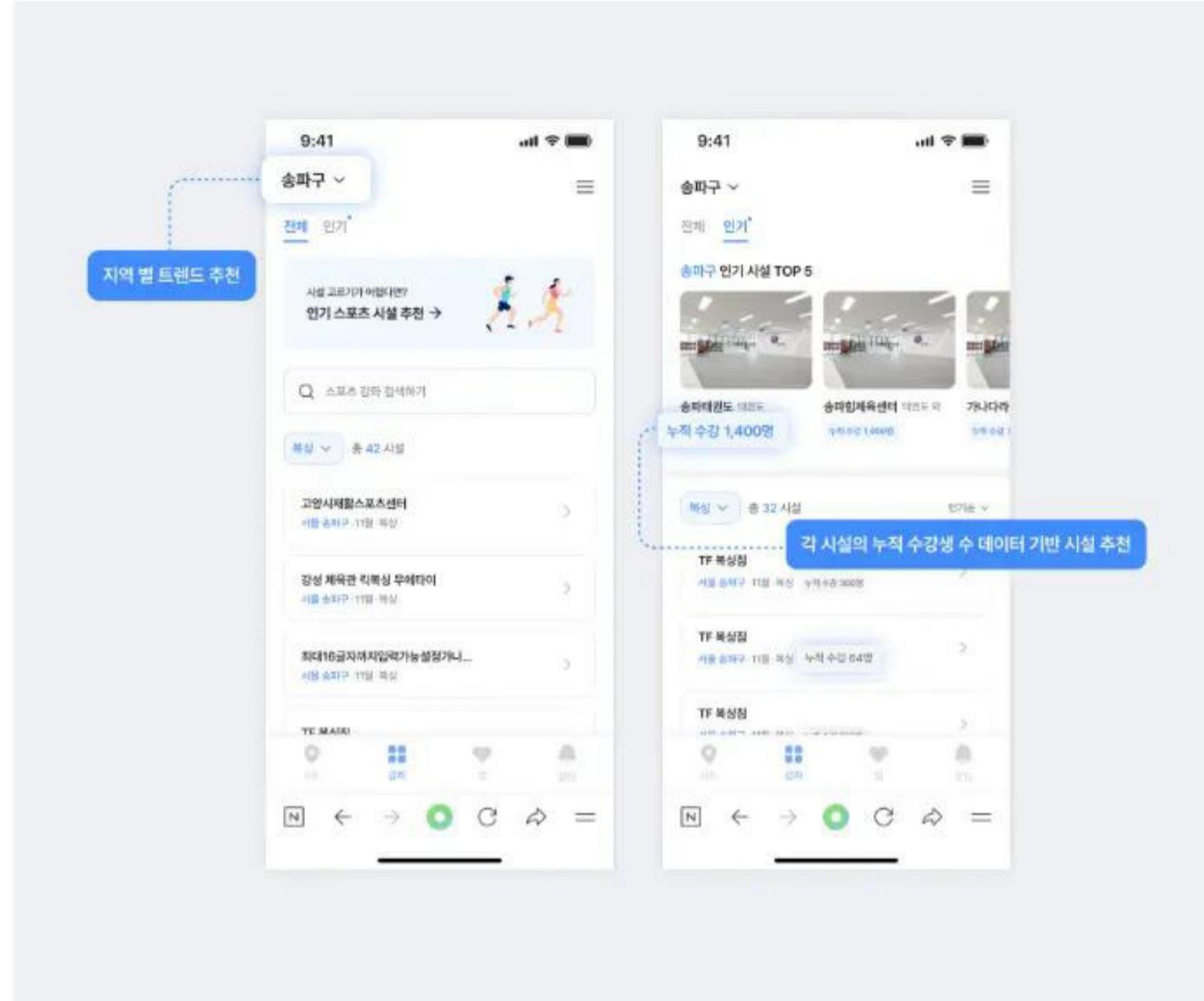
- 이용자 데이터를 분석하여 가장 인기 있는 스포츠 종목을 추천
- 이를 통해 사용자들은 인기 있는 강좌를 쉽게 찾을 수 있음



HELLOFIT

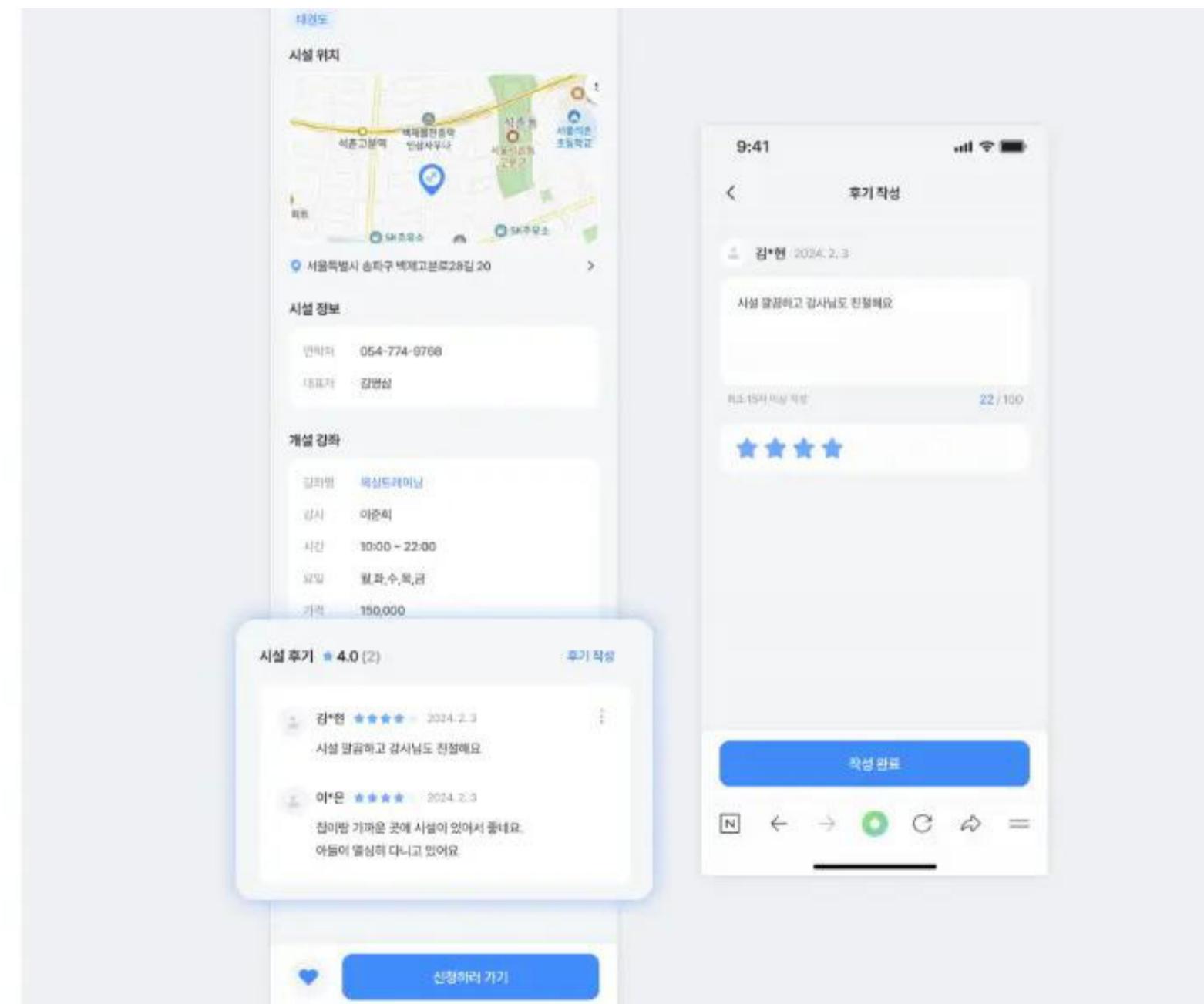
지역별 인기 시설 추천

- 누적 수강생 수 데이터를 기반으로 지역별 인기 시설을 추천하여 맞춤형 탐색 기능을 제공



시설 후기 및 별점 시스템

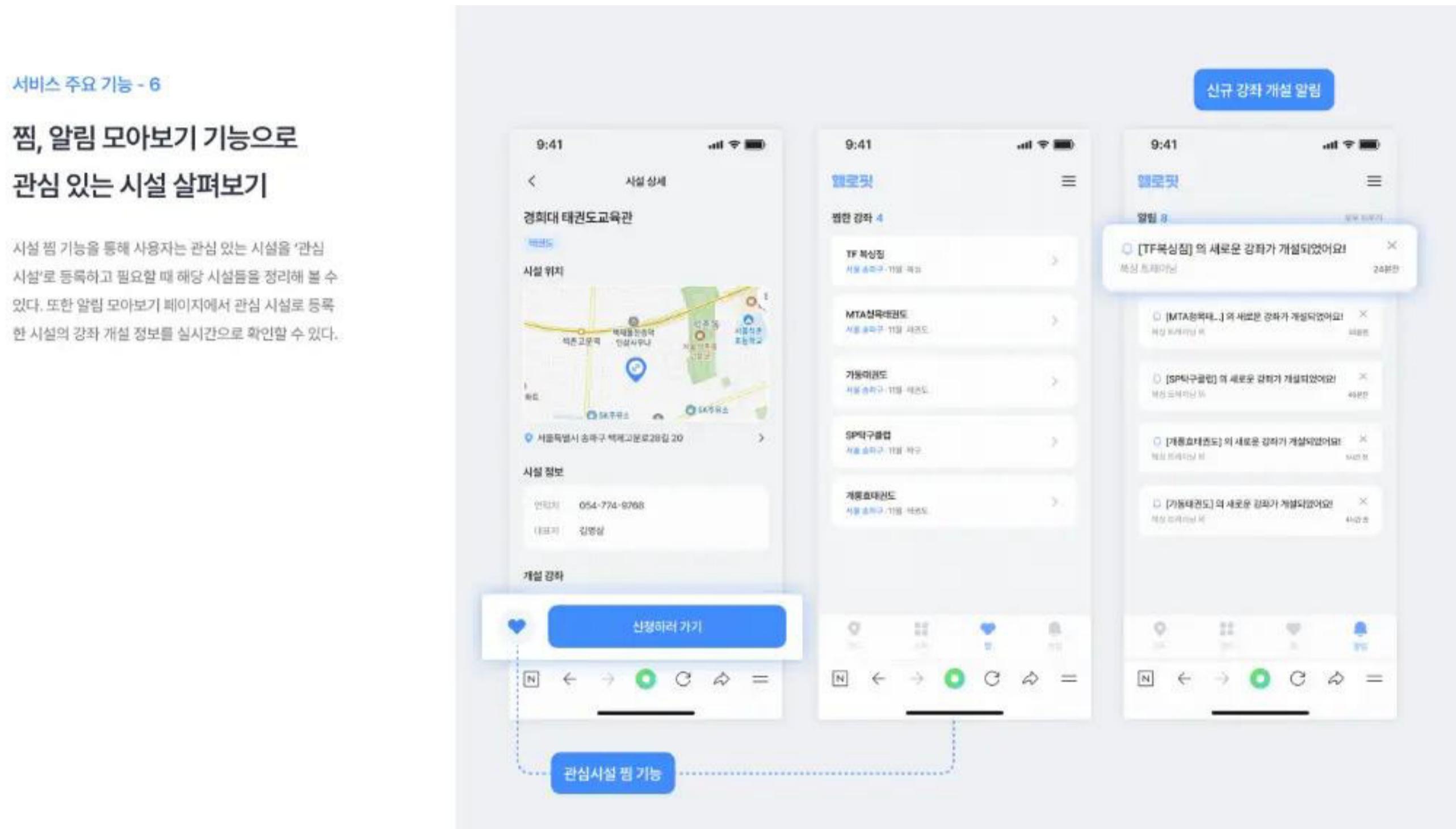
- 사용자들이 직접 남긴 후기를 확인하고 별점 평가를 참고하여 최적의 시설을 선택할 수 있음



HELLOFIT

관심 시설 저장 및 알림 기능

- 자주 이용하는 시설을 저장하고, 신규 강좌가 개설될 경우 알림을 받을 수 있는 기능을 제공



개발 과정

카카오 지도 API 성능 최적화

- 카카오 지도 API의 불필요한 중복 실행을 방지하기 위해 useKakaoMapLoader 훅을 도입
- 지도 초기화 과정에서 useEffect를 활용하여 API 스크립트 로드와 상태 관리를 최적화
- 역지오코딩을 적용하여 사용자의 현재 위치를 기반으로 지역명을 설정

마커 렌더링 최적화

- 기존 방식에서는 시설 목록이 변경될 때마다 모든 마커를 삭제하고 다시 생성하여 성능 저하 발생
- useFacilityMarkers 훅을 활용하여 기존 마커를 유지하면서 필요한 경우만 업데이트
- 좌표 변환을 병렬 처리하여 로딩 속도를 개선하고, UI 깜빡임 문제를 해결

```
const coordinates = await Promise.all(facilities.map(facility =>
  fetchCoordinates(facility)));
```

HELLOFIT

Recoil을 활용한 지도 필터 상태 관리

- Recoil을 활용하여 사용자가 선택한 스포츠 종목 필터를 상태로 관리
- 필터 상태가 변경될 때마다 마커가 갱신되도록 구현

```
import { atom, selector } from 'recoil';

export const selectedSportState = atom({
  key: 'selectedSport',
  default: '전체',
});

export const filteredFacilitiesState = selector({
  key: 'filteredFacilities',
  get: ({ get }) => {
    const selectedSport = get(selectedSportState);
    return facilities.filter(facility => facility.sport === selectedSport);
  }
});
```

검색 및 API 호출 최적화

- AbortController를 활용하여 검색 중 새로운 요청이 들어오면 기존 요청을 취소하여 불필요한 API 호출 방지
- useMemo와 debounce를 적용하여 연속적인 검색에도 성능 저하가 발생하지 않도록 개선
- 검색 결과가 없을 경우 사용자에게 명확한 피드백을 제공하여 UX 개선

```
if (abortControllerRef.current) { abortControllerRef.current.abort(); }
```

UI/UX 개선

- 이용권 유형별 필터 제공 (일반/장애인)으로 사용자 맞춤 탐색 기능 강화
- 반응형 디자인을 적용하여 모바일 환경에서도 원활한 탐색 가능
- UX 테스트를 통해 검색 및 필터링 로직을 개선하여 사용자의 피드백을 반영

```
const filteredFacilities = facilities.filter(facility => facility.type === selectedType);
```

HELLOFIT

트러블슈팅

마커 플리커 현상 & 과도한 API 호출 문제

- 시설 목록이 변경될 때마다 기존 마커를 삭제하고 새로 생성하여 UI가 깜빡이며, 과도한 API 요청이 발생하여 성능이 저하됨

해결 방법

- `seFacilityMarkers` 훅을 도입하여 기존 마커를 유지하며 필요한 경우만 업데이트
- `AbortController`를 활용하여 불필요한 API 요청을 방지
- 좌표 변환(`fetchCoordinates`)을 별별 처리하여 렌더링 속도 개선

배운 점

- API 호출을 최적화하는 것이 성능 개선에 중요한 영향을 미친다는 점을 실감
- 특히, 불필요한 리렌더링을 줄이는 것이 사용자 경험(UX)에 미치는 영향을 직접 확인할 수 있었음

```
const coordinates = await Promise.all(facilities.map(facility =>
  fetchCoordinates(facility)));
```

현재 위치 탐색 시 불필요한 API 호출 문제

- 사용자가 현재 위치 버튼을 여러 번 클릭하면 같은 API 요청이 반복 호출되면서 성능이 저하됨

해결 방법

- `usePositionButton` 훅을 활용하여 사용자의 위치를 캐싱하고, 한 번 가져온 위치를 재사용하도록 개선
- 위치 허용 여부를 확인하여 허용하지 않을 경우 기본 위치를 설정

배운 점

- 사용자의 행동 패턴을 고려하여 API 요청을 최적화하는 것이 중요
- 불필요한 중복 요청을 줄이는 방식(캐싱, 상태 관리 등)을 적용하면 성능과 UX를 동시에 개선할 수 있음

```
const [cachedLocation, setCachedLocation] = useState(null);
const moveToUserLocation = () => {
  if (cachedLocation) { map.setCenter(cachedLocation); return; }
  navigator.geolocation.getCurrentPosition(position => {
    const userLatLng = new kakao.maps.LatLng(position.coords.latitude, position.coords.longitude);
    setCachedLocation(userLatLng);
    map.setCenter(userLatLng);
  });
};
```

HELLOFIT

지도 이동 시 마커 초기화 문제

- 지도를 이동할 때 기존 마커가 유지되지 않고 다시 그려지는 문제 발생

해결 방법

- useFacilityMarkers 훅에서 기존 마커를 재사용할 수 있도록 개선
- setCenter 호출 시 기존 마커가 유지되도록 수정
- useEffect 의존성을 최소화하여 불필요한 리렌더링 방지

배운 점

- 불필요한 상태 변경을 방지하고, 최소한의 리렌더링으로 데이터를 유지하는 것이 성능 최적화에 큰 영향을 미침
- useEffect의 의존성을 관리하는 것이 중요한 이유를 다시 한번 실감

지역 검색 시 정확하지 않은 검색 결과

- 검색된 지역명이 너무 길거나 복잡하여 API가 정확한 좌표를 반환하지 않는 경우 발생

해결 방법

- simplifyRegionName 유틸 함수를 사용하여 시/군/구 단위로 변환
- kakao.maps.services.Geocoder() 호출 시 검색 결과를 검증하여 첫 번째 결과만 사용

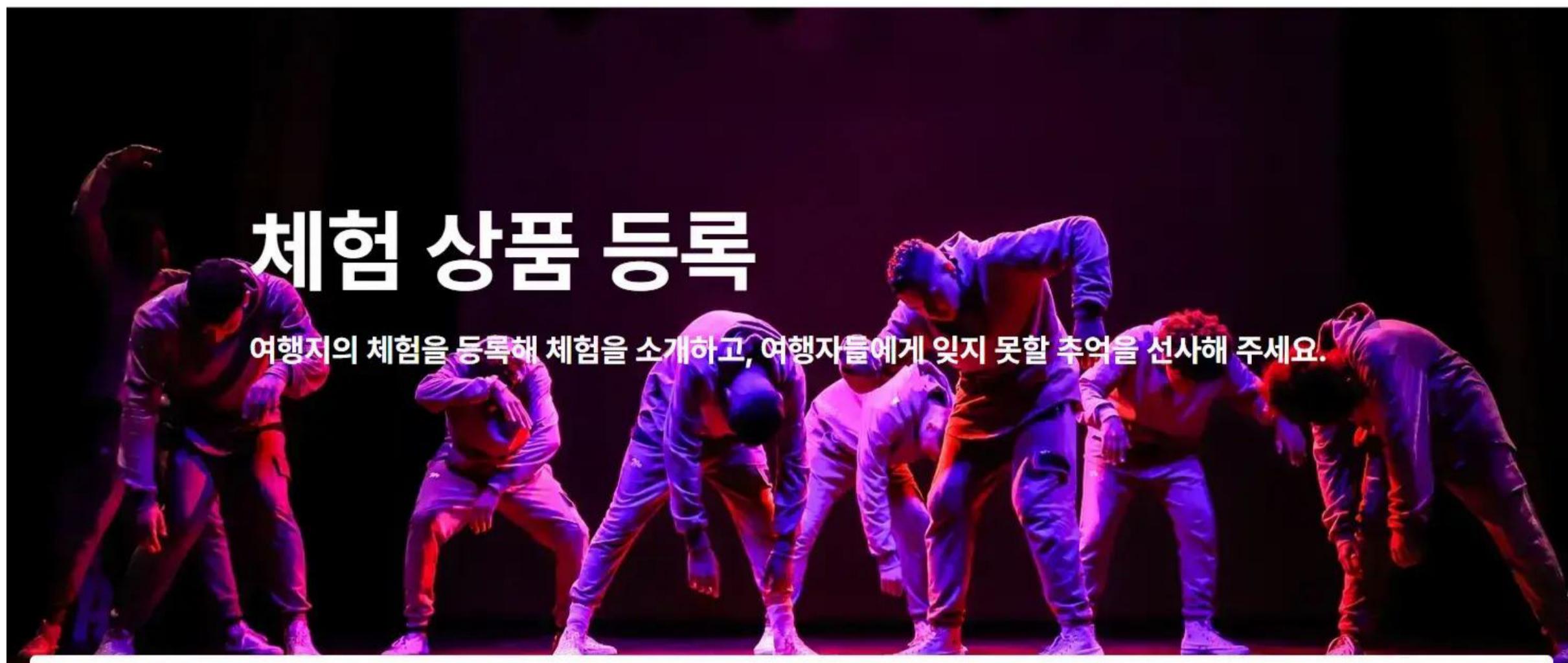
배운 점

- 외부 API를 사용할 때는 데이터가 항상 정확하다고 가정하면 안 된다는 점
- 데이터를 정제하고 가공하는 로직이 사용자 경험에 중요한 영향을 미침

Global Nomad

GlobalNomad®

GlobalNomad® walwal



체험 상품 등록

여행자의 체험을 등록해 체험을 소개하고, 여행자들에게 잊지 못할 추억을 선사해 주세요.

무엇을 체험하고 싶으신가요?

검색하기

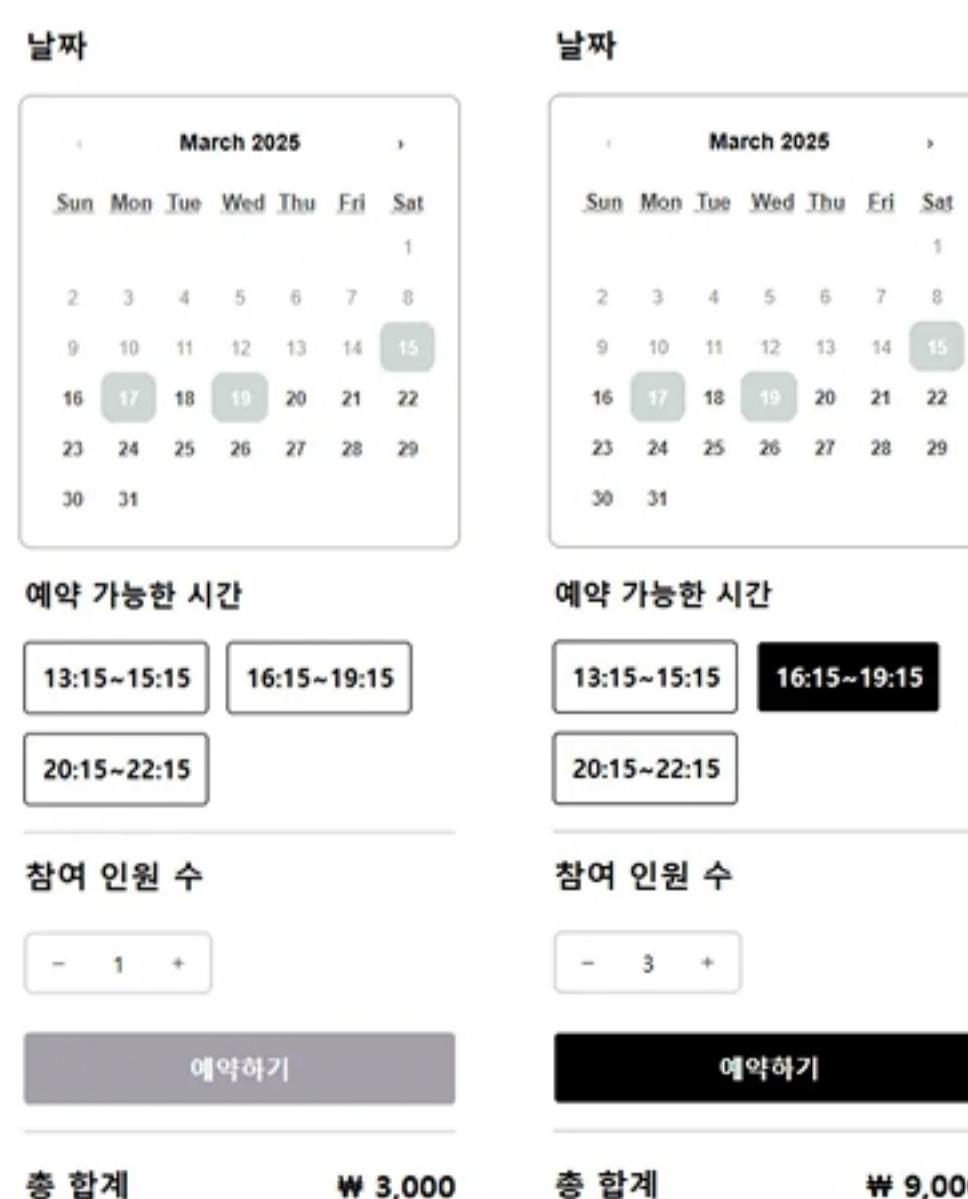
핵심 기여

- React Query를 활용하여 예약 데이터의 비동기 상태 관리 최적화
- Zustand를 사용하여 전역 상태 관리 간소화 및 성능 개선
- SSR을 적용하여 SEO 최적화 및 초기 로딩 속도 개선
- Tailwind CSS로 반응형 UI를 구축하여 다양한 디바이스에서 최적의 UX 제공
- Airbnb ESLint를 도입하여 코드 일관성 유지 및 협업 환경 개선

주요기능

체험 상품 예약 시스템

- 여행 중 체험 상품을 간편하게 예약할 수 있는 기능 제공



날짜

날짜

예약 가능한 시간

예약 가능한 시간

참여 인원 수

참여 인원 수

예약하기

예약하기

총 합계 ₩ 3,000

총 합계 ₩ 9,000

Global Nomad

지도 기반 위치 검색

- 사용자가 원하는 지역의 체험 상품을 지도에서 쉽게 탐색할 수 있도록 지원



잠실한강공원 자연형 물놀이장에서 한강의 낭만적인 노을과 야경 그리고 수영장을 비추는 멋진 조명과 함께 물 위에 둉둥 떠서 즐기는 시네마 풍당!



사용자 리뷰 및 별점 시스템

- 체험 상품에 대한 사용자 리뷰와 별점 평가 기능 제공

후기

4.8 매우 만족
★ 5개 후기



test1234 | 2024.08.08

직원이 친절해요



test1234 | 2024.08.08

친절하게 잘 알려주셔서 마음에 들게 만들었어요!



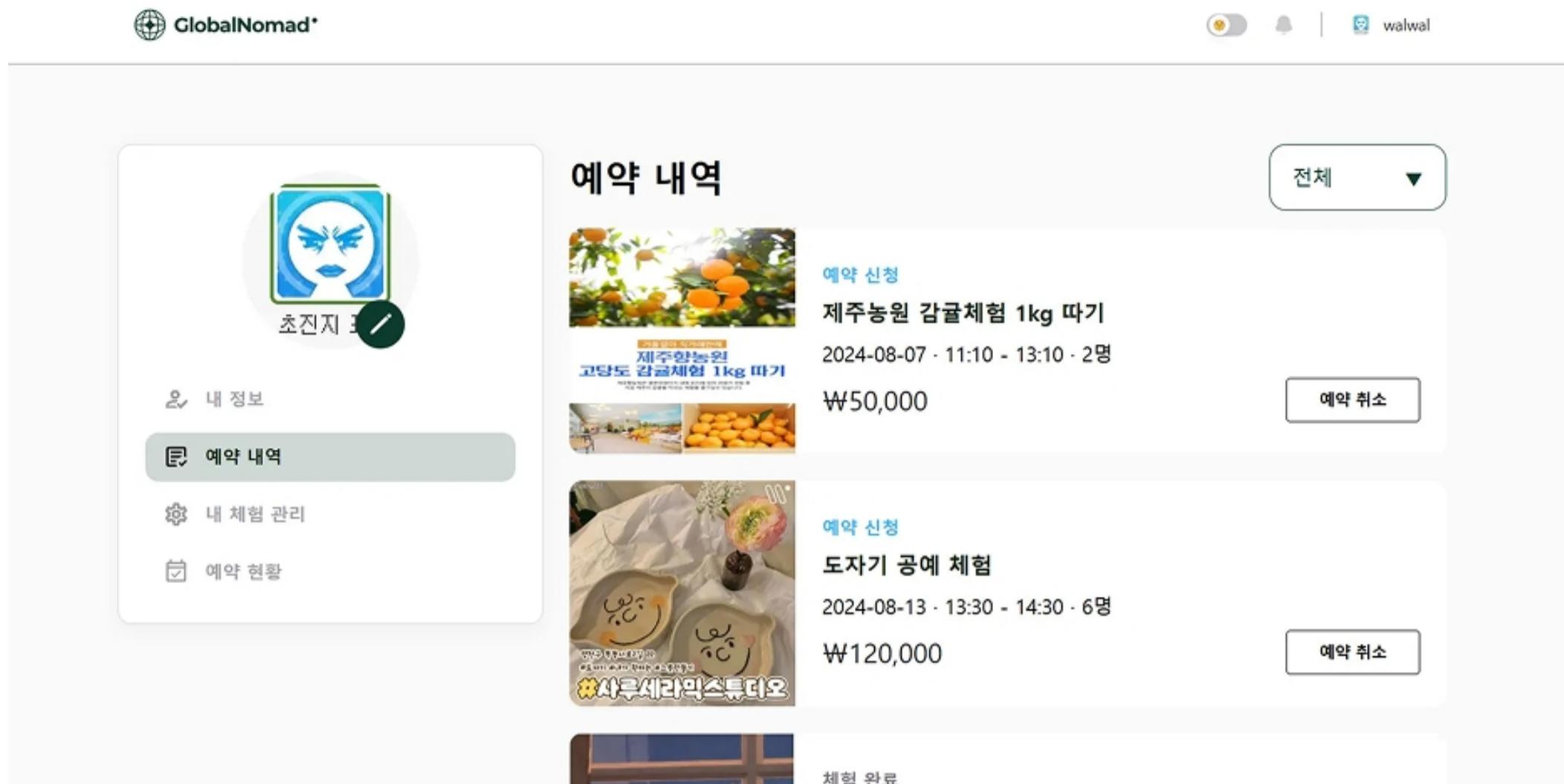
test1234 | 2024.08.08

만들어서 선물로 줬는데 너무 좋아해서 만족합니다~

Global Nomad

예약 내역 관리

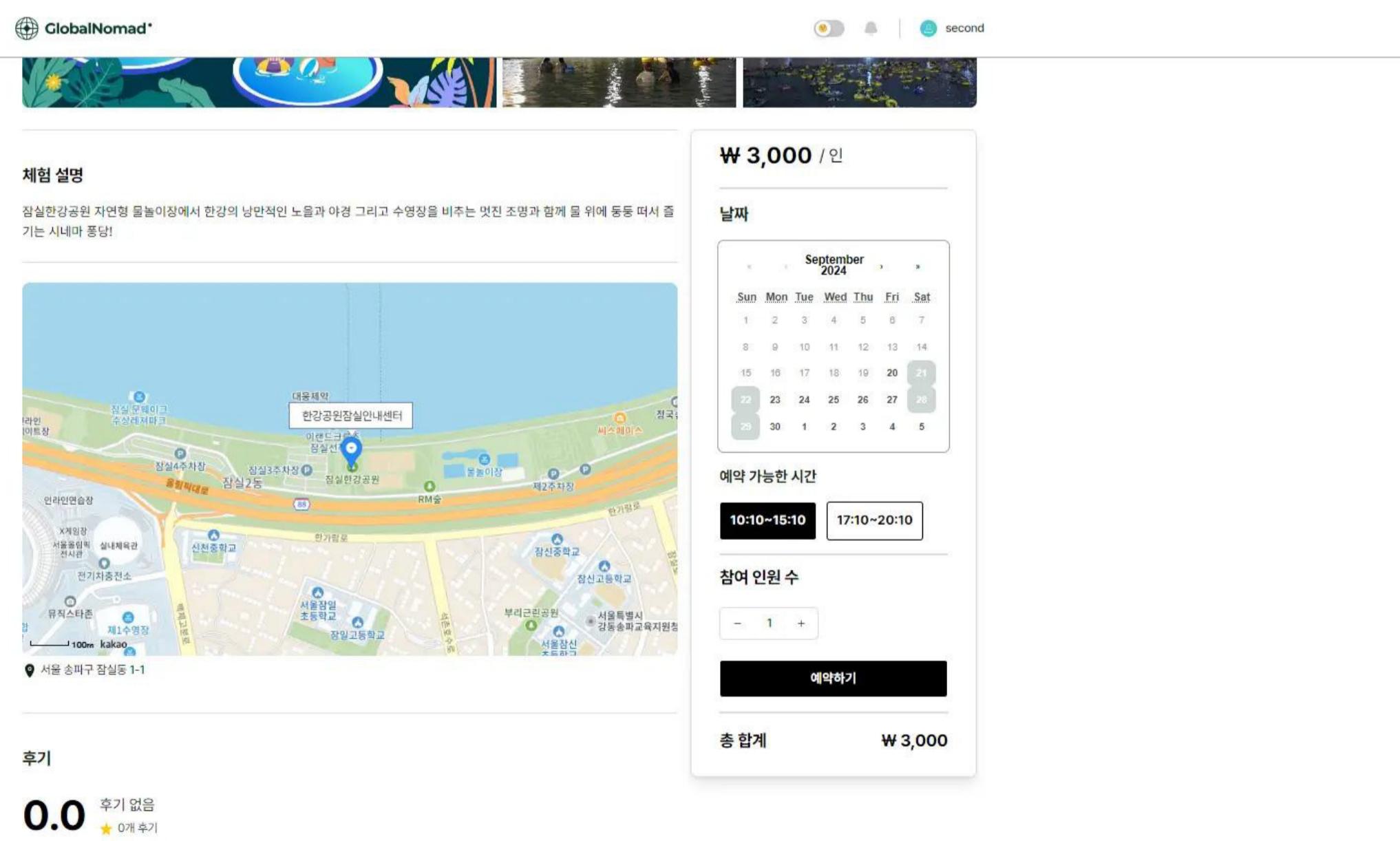
- 사용자가 예약한 체험 상품을 관리하고 변경할 수 있는 기능 제공



반응형 예약 UI

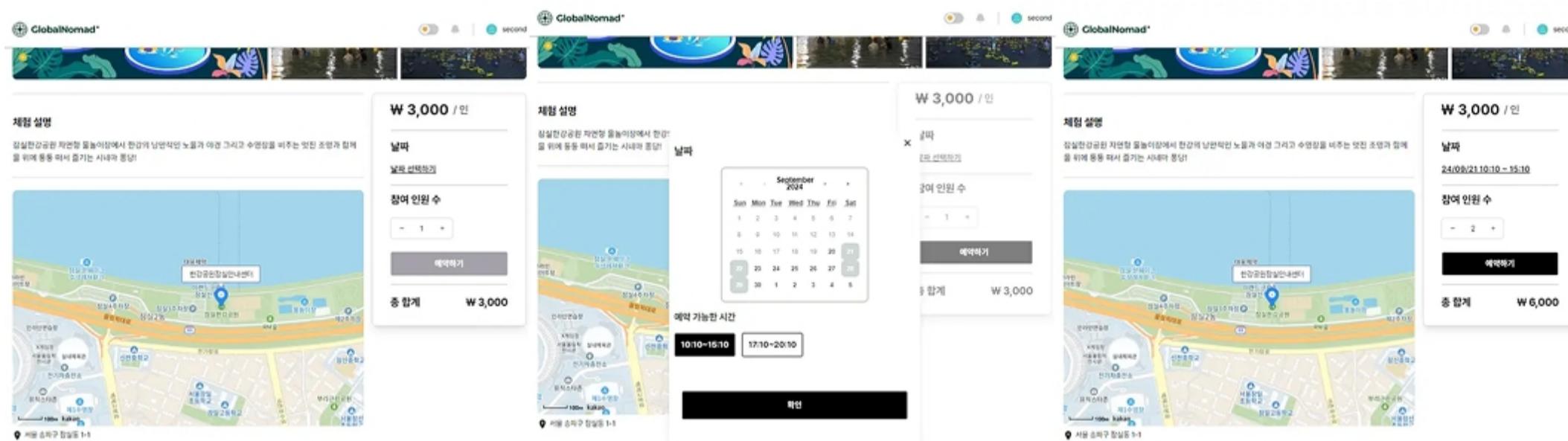
- PC, 태블릿, 모바일 환경에 맞게 최적화된 예약 UI 제공

PC 버전

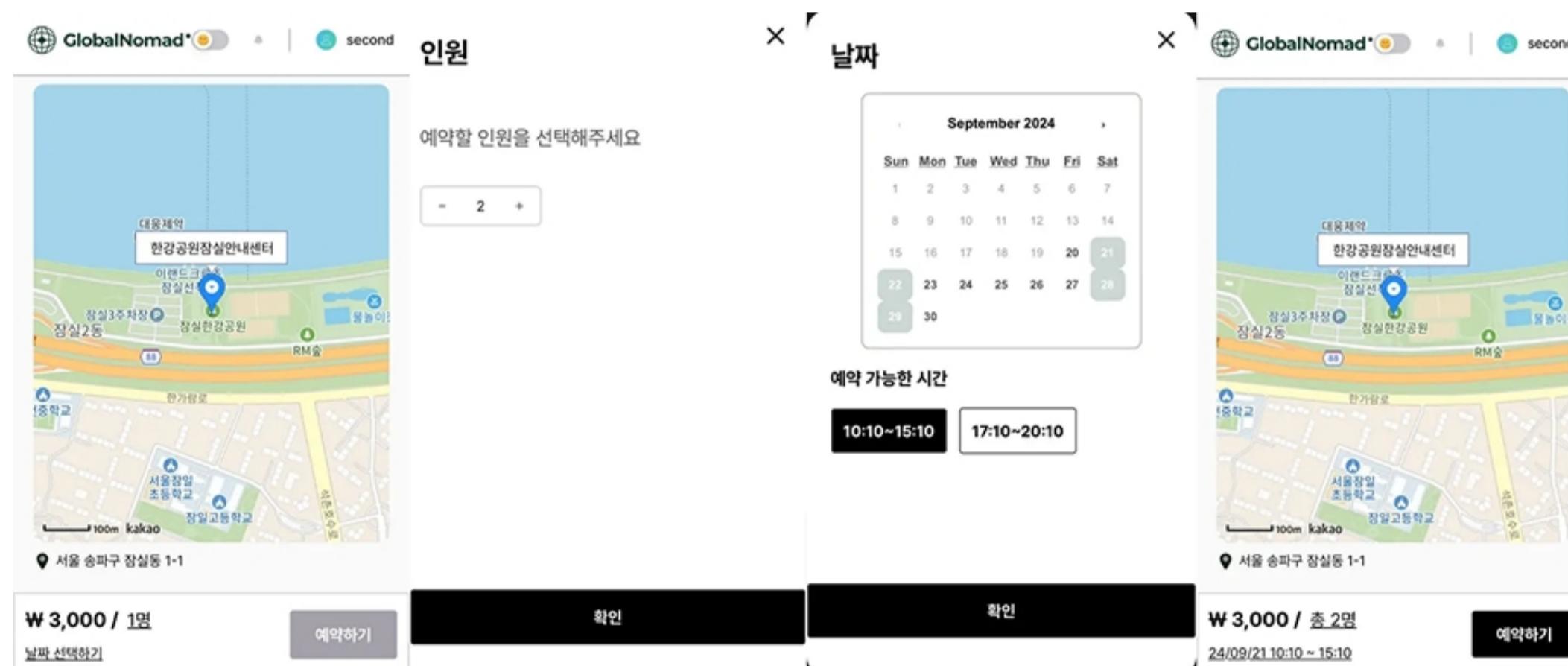


Global Nomad

Tablet 버전

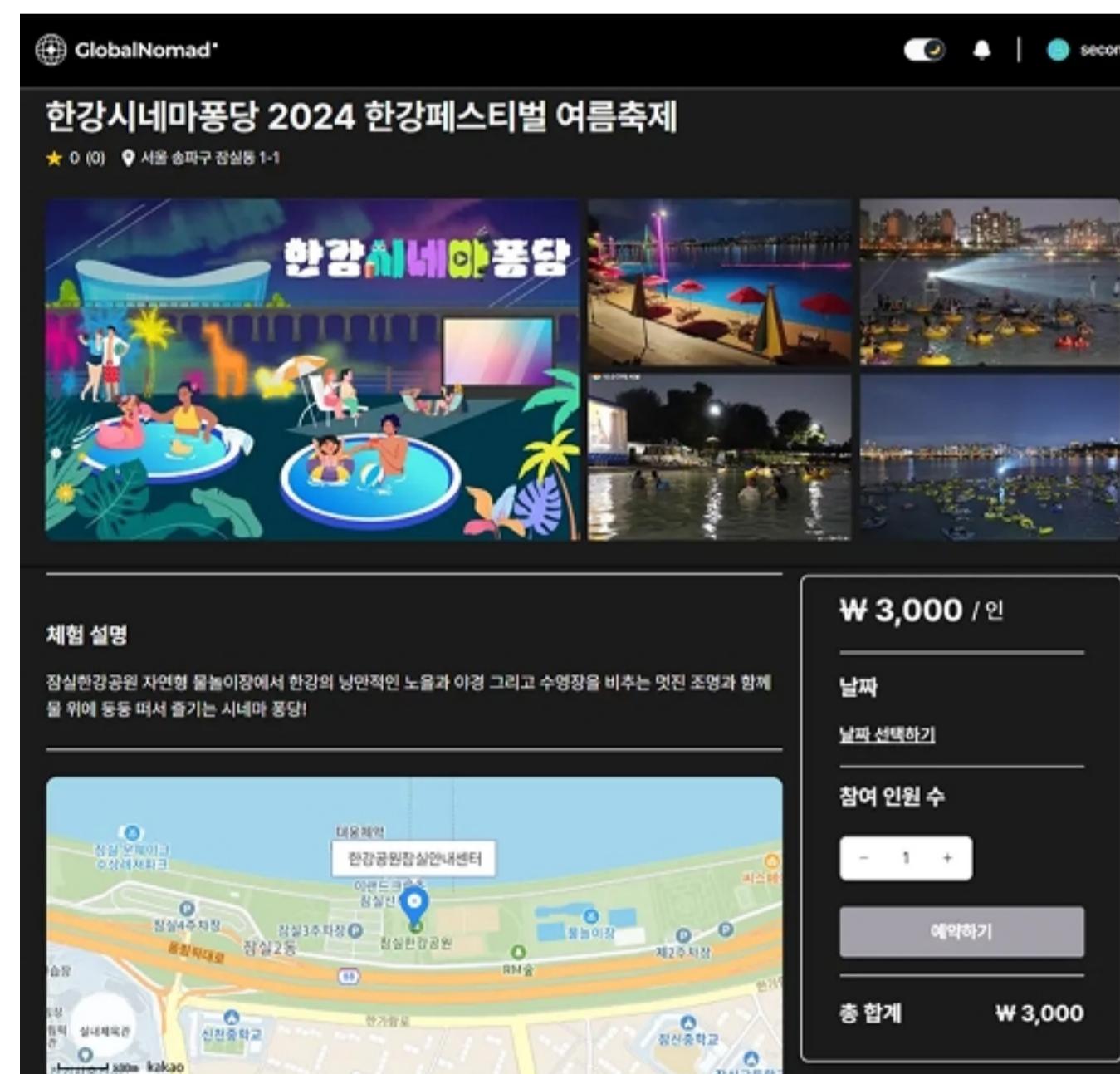


Mobile 버전



다크 모드 지원

- 사용자의 시각적 편의를 고려한 다크 모드 기능 추가



Global Nomad

개발 과정

화면 크기에 따른 예약 UI 렌더링

- PC, 태블릿, 모바일 환경에 맞춰 최적화된 UI를 제공하기 위해 `useMediaQuery` 커스텀 툴을 도입
- 각 디바이스에 맞춰 `FloatingCard`, `TabletCard`, `MobileCard` 컴포넌트를 분리하여 렌더링
- 화면 크기 변경을 감지하고 적절한 UI를 적용하는 방식을 구현

```
const isTablet = useMediaQuery(768, 1024);
const isMobile = useMediaQuery(0, 768);

const renderBookingUI = () => {
  if (isMobile) return <MobileCard schedules={schedules} price={price} />;
  if (isTablet) return <TabletCard schedules={schedules} price={price} />;
  return <FloatingCard schedules={schedules} price={price} />;
};
```

예약 시스템 최적화

- React Query의 `refetchInterval`을 활용하여 실시간 예약 상태 유지
- 비동기 요청을 `useMutation`으로 관리하여 예약 프로세스 최적화
- 페이지네이션을 적용하여 리뷰 데이터를 효율적으로 로드하도록 개선

```
const { data, refetch } = useQuery(['reservation', userId], fetchReservation, {
  refetchInterval: 30000
});
```

SSR을 활용한 SEO 최적화

- 검색 엔진 최적화를 위해 Next.js의 `getServerSideProps`를 활용하여 서버 사이드 렌더링 적용
- SEO를 고려한 `meta` 태그와 `Open Graph` 설정 추가
- 동적 페이지에서도 SNS 미리보기가 정상적으로 표시되도록 최적화

```
export const getServerSideProps: GetServerSideProps<Props> = async (context) => {
  const { id } = context.query;
  const activityId = typeof id === 'string' ? parseInt(id, 10) : undefined;

  if (!activityId) return { notFound: true };

  return { props: { id: activityId } };
};
```

Global Nomad

Zustand를 활용한 전역 상태 관리

- Redux 대비 가벼운 Zustand를 활용하여 전역 상태 관리를 간소화
- 예약 정보 및 사용자 데이터를 효율적으로 관리하도록 개선
- 상태 변경 시 불필요한 리렌더링을 방지하여 성능 최적화

```
const useStore = create((set) => ({ reservation: null, setReservation: (data) => set
```

다크 모드 상태 관리

- Zustand를 활용하여 다크 모드 상태를 전역적으로 관리
- Tailwind CSS를 사용해 UI 스타일을 동적으로 변경
- 사용자가 다크 모드를 선택하면 상태가 저장되어 페이지 새로고침 후에도 유지되도록 구현

```
const useDarkMode = create((set) => ({ isDarkMode: false, toggleDarkMode: () => set((state) => ({ isDarkMode: !state.isDarkMode }))), );
```

Airbnb ESLint 적용 경험

- Airbnb ESLint를 도입하여 코드 컨벤션을 통일하고 가독성을 향상
- 기본적인 Best Practice(예: 화살표 함수, 객체 구조 분해, 불필요한 변수 제거 등)를 적용하는 습관을 형성
- Prettier와 함께 사용하여 자동 코드 포맷팅을 적용하고, 팀원 간 코드 일관성을 유지

```
const fetchData = async () => {
  console.info('Fetching data ... ');
  const response = await fetch('/api/data');
  return response;
};
```

Global Nomad

트러블슈팅

useMediaQuery 초기 렌더링 오류

- Next.js의 서버 사이드 렌더링(SSR)에서 window 객체를 참조하여 useMediaQuery가 초기 렌더링 시 오류 발생

해결 방법

- useEffect를 사용하여 클라이언트에서만 window 객체를 참조하도록 변경

배운 점

- Next.js 환경에서는 SSR 시 window 객체를 직접 사용할 수 없으므로, 클라이언트 사이드에서만 실행되도록 분리해야 함

```
const isTablet = useMediaQuery(768, 1024);
const isMobile = useMediaQuery(0, 768);
useEffect(() => { setIsMobile(window.innerWidth <= 768); }, []);
```

다크 모드 상태 유지 문제

- Zustand를 활용하여 다크 모드 상태를 관리했지만, 페이지 새로고침 시 설정이 초기화됨

해결 방법

- Zustand의 persist 미들웨어를 사용하여 localStorage에 저장되도록 설정

배운 점

- 사용자의 설정을 유지하려면 전역 상태 관리뿐만 아니라 브라우저 저장소(localStorage)를 활용하는 것이 중요함

```
const useDarkMode = createPersist((set) => ({ isDarkMode: false,
  toggleDarkMode: () => set((state) => ({ isDarkMode: !state.isDarkMode })) }), { name: 'dark-mode' }));
```

Global Nomad

Airbnb ESLint 적용 후 코드 수정 이슈

- ESLint Airbnb 스타일 가이드 적용 후 기존 코드에서 여러 규칙 위반이 발생하여 빌드 실패

해결 방법

- ESLint 규칙을 학습하고, `eslint --fix`를 실행하여 자동으로 코드 스타일을 맞춤
- 불필요한 `console.log` 제거, 화살표 함수 사용 등 Airbnb 스타일 가이드에 맞게 코드 리팩토링

배운 점

- 린트 도구를 적용하면 코드 일관성이 유지되지만, 기존 코드와 충돌할 수 있어 점진적인 적용이 필요함

React Query 캐싱 문제로 예약 데이터가 즉시 반영되지 않음

- 예약 후 데이터를 새로고침하지 않으면 UI에서 즉시 반영되지 않음

해결 방법

- `useQueryClient`의 `invalidateQueries`를 활용하여 특정 쿼리를 캐시를 무효화하여 최신 데이터가 반영되도록 수정

배운 점

- API 데이터를 실시간으로 반영하려면 적절한 캐시 무효화 전략을 적용해야 함

```
const queryClient = useQueryClient();
const handleReservation = async () => {
  await makeReservation();
  queryClient.invalidateQueries(['reservation']);
};
```

SEO 최적화 시 메타 태그 적용 문제

- `next/head`를 사용한 메타 태그 설정이 일부 동적 페이지에서 적용되지 않음

해결 방법

- `getServerSideProps`에서 메타 정보를 동적으로 생성하여 SSR에서 올바르게 렌더링되도록 수정

배운 점

- SEO 최적화를 위해 SSR과 SSG의 차이를 이해하고, 동적 컨텐츠에는 SSR을 활용해야 함

```
export const getServerSideProps: GetServerSideProps = async (context) => {
  const { id } = context.query;
  const data = await fetchActivityData(id);
  return { props: { metaTitle: data.title, metaDescription: data.description } };
};
```