Profile

스펀지 같은 개발자, 오주현입니다.

스펀지는 무엇이든 빠르게 빨아들이고, 한번 머금으면 쉽게 내뱉지 않습니다.
저도 이처럼 팀원들과 소통하면서 새로운 기술을 습득하고,
내면에 단단히 머금어 팀이 필요할 때 언제나 꺼내 쓸 수 있는 개발자입니다.

2008.05.30.
010-6637-7242
me@zuu3.kr
부산소프트웨어마이스터고등학교 재학중 (2027.01 졸업 예정)

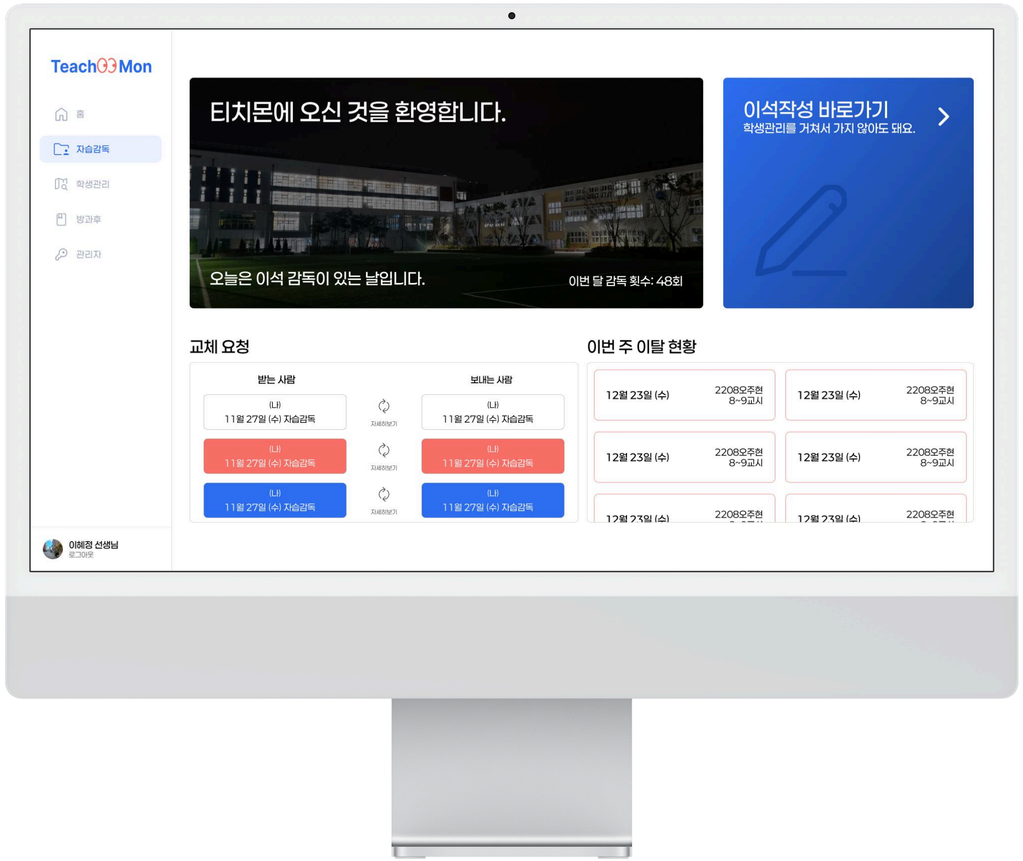
Awards	2024	교내 클론코딩 경진대회 장려상 교내 네트워크 경진대회 장려상
	2025	SJSU 역량 강화 캠프 프로젝트 Tech Excellent상 소프트웨어개발과 웹 프로그래밍 경진대회 우수상 소프트웨어마이스터고등학교 4개교 연합해커톤 우수상 U-BDIA 프로젝트 아이디어상(티치몬)
Certificate	2025	정보처리산업기사(25251030366V)
Leader	2025	SJSU 역량 강화 캠프 프로젝트 팀장
	2026	부산소프트웨어마이스터고등학교 1, 2, 3학년 반장
Activity	2025	28th Appjam 참가 교내 AI SW 공모전 참가(티치몬) U-BDIA 프로젝트 전시(티치몬) ko.react.dev 문서 기여
	2026	순복음범천교회 교회 웹사이트 외주 작업

Contents

교내 방과후 자습/이석 관리 서비스

TeachMon(티치몬)

기존 구글 시트 기반의 자습 관리를 대체하여, 선생님께서 자습 일정·이석 현황을
한눈에 파악할 수 있는 웹 서비스입니다. 감독 교체 요청, 이탈 현황 확인,
이석 신청까지 하나의 플랫폼에서 처리할 수 있습니다.



제작 기간

2024. 12. ~ 2025. 03.

운영 기간

2025. 03 ~

기술스택

TS

깃허브

깃허브 바로가기

기여

홈·운영 대시보드 구현
자습/이석 일정 조회 및 감독 교체 구현
방과후 보강 처리 구현
어드민 방과후 출장/보강 구현
분기별 자습 일정 설정 구현

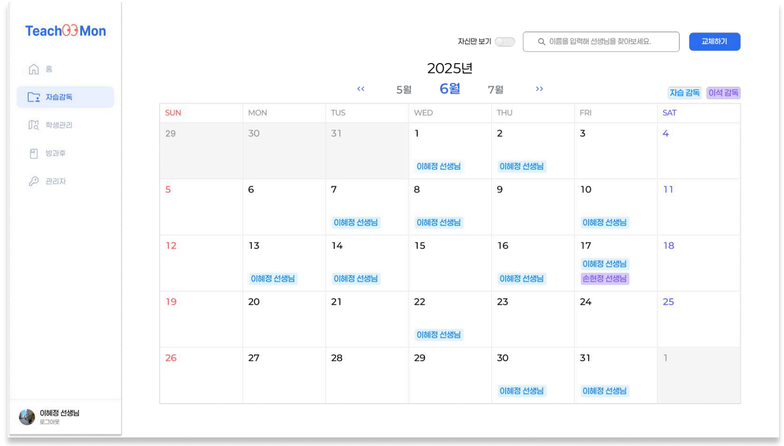
#U-BDIA 아이디어상 #실제 운영 중인 서비스

Main features & Contribution

TeachMon

자습 감독 및 일정 관리

자습/이석 일정을 캘린더로 확인하고 교체 요청을 처리하는 기능입니다. useQuery로 일정 데이터를 조회하고, 교체 요청 처리 시 useMutation과 invalidateQueries를 활용해 전체 일정에 변경 사항이 즉시 반영되도록 구현했습니다. 감독 검색에는 debounce를 적용해 입력마다 요청이 발생하지 않도록 처리했습니다.

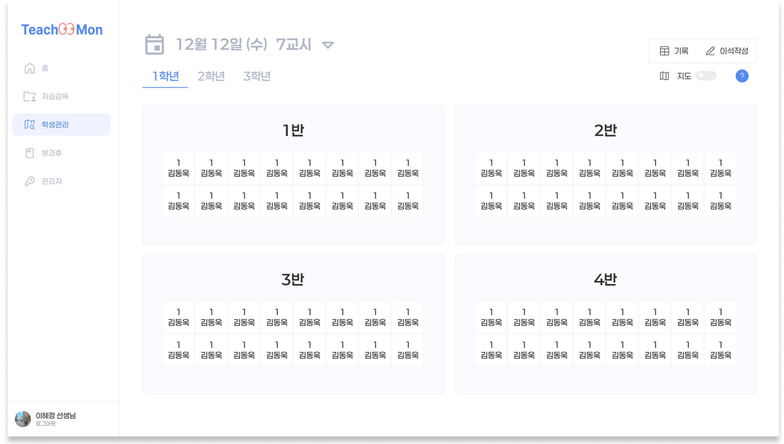


방과후 및 보강 처리

방과후 일정 관리와 출장 및 보강 처리를 지원하는 기능입니다. 복수 교시 보강을 한 번에 처리할 때 Promise.all을 활용해 여러 요청을 병렬로 처리하고, 성공 시 invalidateQueries로 목록을 갱신했습니다.

자습 일정 설정

분기별 자습 일정을 학년 단위로 설정하는 기능입니다. 학년마다 자습 일정이 달라 이 부분을 한 화면에서 미리 설정해두어 불편함을 줄일 수 있도록 구성했으며, 설정 저장 시 useMutation으로 요청을 처리했습니다.



운영 현황 대시보드

감독 횟수, 교체 요청, 이탈 현황을 한눈에 확인하는 기능입니다. 요청 처리 이후 invalidateQueries로 관련 쿼리를 갱신해 항상 최신 상태를 유지하도록 구성했습니다.

Troubleshooting - 초기 번들 구조 개선을 통한 로딩 성능 개선

TeachMon

Problem

초기 Lighthouse Performance 점수가 56점으로, 네트워크 환경이 좋지 않을 경우 첫 화면이 늦게 렌더링되는 문제가 있었습니다. 특히 초기 접속 시 모든 페이지 번들이 함께 로드되면서, 실제 화면에 필요하지 않은 코드까지 동시에 로딩되고 있었습니다.

Cause

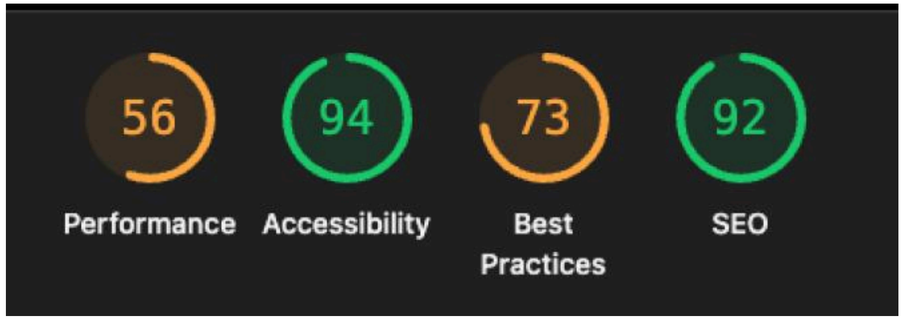
번들 구조를 분석한 결과, 라우트 단위 분리가 되어 있지 않아 초기 진입 시 전체 페이지 코드가 하나의 번들로 포함되어 있었습니다. 이로 인해 초기 렌더링에 불필요한 리소스까지 함께 다운로드되며 첫 화면 표시 시간이 지연되고 있었습니다.

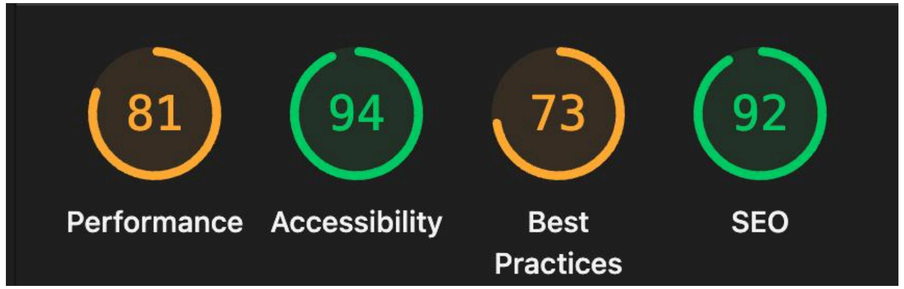
Solution

라우트 단위 코드 스플리팅을 위해 React.lazy와 @loadable/component를 비교했습니다. Loadable은 SSR 지원과 named export 처리에 강점이 있지만, 티치몬은 CSR 기반 SPA이고 SSR이 필요하지 않았습니다. React.lazy는 별도 라이브러리 의존 없이 Suspense와 조합해 로딩 fallback까지 선언적으로 처리할 수 있어 이 프로젝트에 적합하다고 판단했습니다.

Result

이를 통해 사용자가 접근하는 시점에 필요한 페이지 코드만 비동기적으로 로드되도록 구조를 개선했습니다. Performance 점수는 56 → 81로 약 44.6% 향상되었습니다.





Troubleshooting - Tanstack Query를 활용한 데이터 동기화 문제 해결

TeachMon

Problem

보강 처리나 교체 요청 이후에도 모든 화면에서 이전 데이터가 그대로 유지되어 변경된 상태가 즉시 반영되지 않는 문제가 있었습니다. 데이터 변경 이후에도 캐시된 값이 유지되면서 실제 서버 데이터와 UI 간의 불일치가 발생했습니다.

Cause

Tanstack Query를 사용해 데이터를 관리하고 있었지만, 데이터 변경 이후 관련 쿼리를 명시적으로 갱신하지 않아 기존 캐시 데이터가 그대로 유지되고 있었습니다. 이로 인해 데이터 변경이 발생하더라도 이미 조회된 데이터는 즉시 반영되지 않고, 각 화면에서 이전 상태가 남아 있는 문제가 발생했습니다.

Solution

데이터 변경 후 UI 반영 방식으로 낙관적 업데이트와 invalidateQueries를 비교했습니다. 낙관적 업데이트는 요청 전에 캐시를 먼저 수정해 UI에 즉시 반영하고, 실패 시 롤백하는 방식으로 응답 속도는 빠르지만 서버 데이터와 불일치가 생길 수 있고 롤백 로직이 추가로 필요합니다. 티치몬은 감독 교체·보강 처리 등 정확한 상태가 중요한 운영 서비스였기 때문에, 서버 기준으로 연관 쿼리를 일괄 갱신할 수 있는 invalidateQueries를 선택했습니다.

```
1  const mutation = useMutation({
2    mutationFn: updateSchedule,
3  })
```

▼

```
1  const mutation = useMutation({
2    mutationFn: updateSchedule,
3    onSuccess: () => {
4      queryClient.invalidateQueries(['schedule'])
5    }
6  })
```

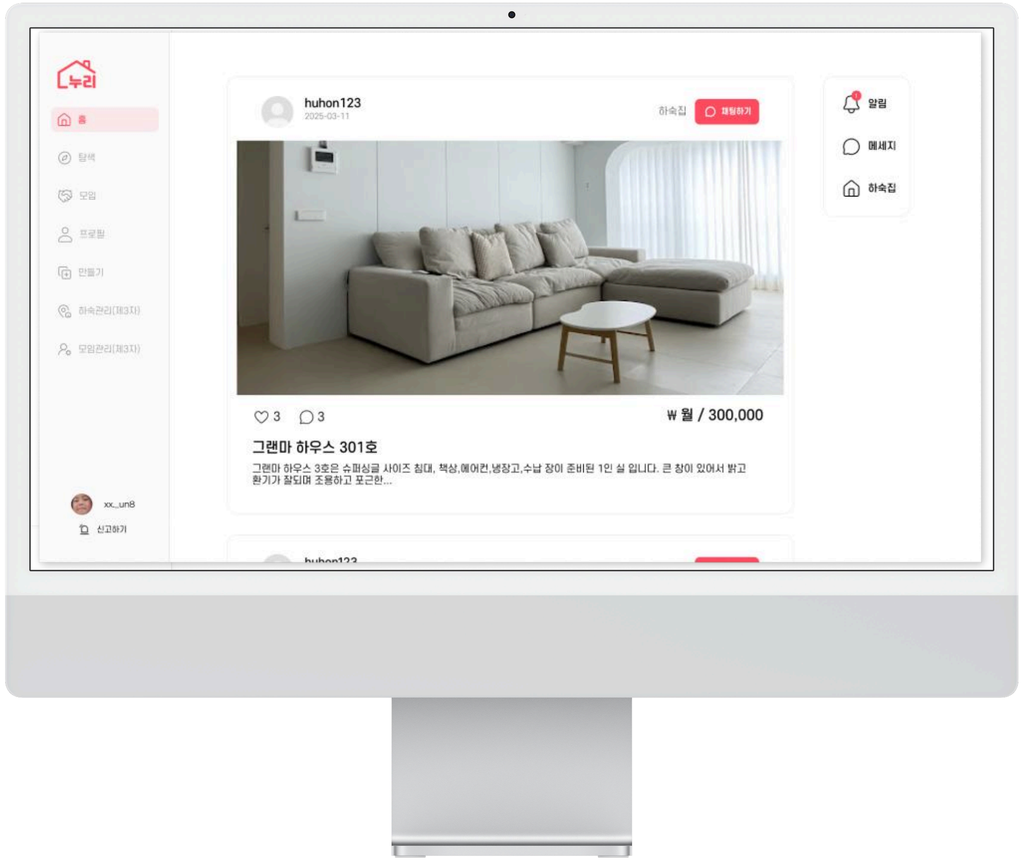
Result

데이터 변경 이후에도 이전 상태가 남아 있던 문제가 개선되어, 모든 화면에서 최신 상태를 일관되게 반영할 수 있었습니다. 사용자가 요청 처리 결과를 바로 확인할 수 있게 되었고, 운영 화면에서 데이터 신뢰도를 높일 수 있었습니다.

외국인 유학생을 위한 한국 문화 적응 플랫폼

Nuri(누리)

외국인 유학생의 한국 정착을 지원하는 플랫폼입니다. 하숙집 연결부터 생활 관리, 지역 모임까지 하나의 서비스에서 제공하여 낯선 환경에서의 적응을 돕습니다.



제작 기간

2025.04. ~ 2025.11.

운영 기간

2025.10. ~ 2025.11.

기술스택

깃허브

깃허브 바로가기

기여

OAuth 로그인 및 멀티스텝 회원가입 구현
홈 상세 조회 구조 설계
하숙 업무 관리 구현
모임 일정/멤버 관리 구현

#교내 전공동아리 부스 운영

Main features & Contribution

로그인 및 회원가입

OAuth 기반 로그인 및 멀티스텝 회원가입 기능을 구현했습니다. JWT 인증 방식을 적용하고, token 만료 2분 전부터 재발급 대상으로 판단해 요청 시 선제적으로 토큰을 갱신하도록 처리했습니다.

홈 화면 상세 조회 구조

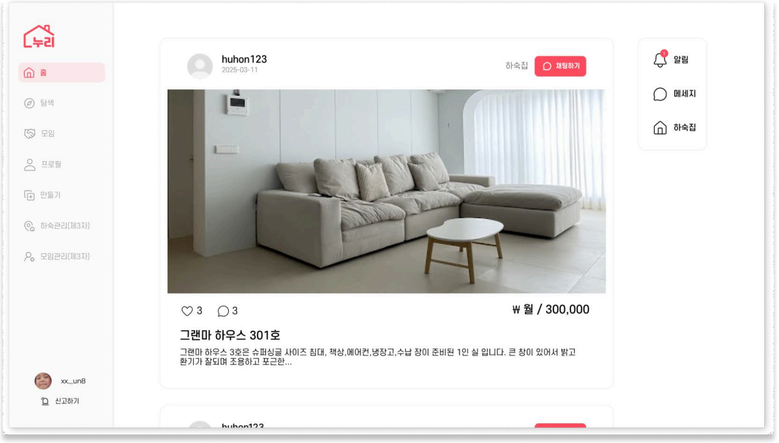
홈 화면에서 선택한 항목의 상세 정보를 조회하는 기능을 구현했습니다. Next.js의 Parallel Routes와 Intercepting Routes를 활용해 목록 페이지 위에서 상세 내용을 확인할 수 있도록 구성했으며, 직접 URL로 접근하는 경우에는 전체 페이지로 렌더링되도록 처리했습니다.

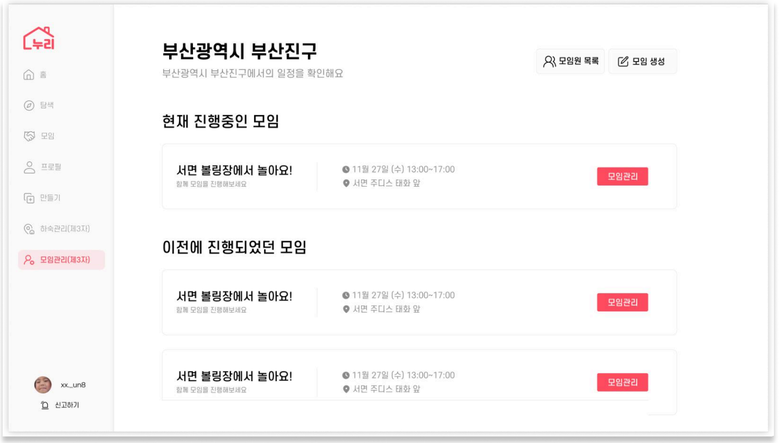
제3자 하숙 관리

캘린더에서 날짜를 선택하면 당일 업무를 조회하는 기능을 구현했습니다. Apollo Client를 활용해 데이터를 조회하고, 업무 완료 시 낙관적 업데이트를 적용해 UI에 즉시 반영되도록 처리했습니다. 요청 실패 시에는 이전 상태로 롤백해 상태 일관성을 유지했습니다.

제3자 모임 관리

모임 일정과 멤버를 관리하는 기능을 구현했습니다. Promise.all을 활용해 일정과 멤버 데이터를 병렬로 조회하고, 참가 수락/거절/추방 시 낙관적 업데이트로 UI에 즉시 반영되도록 구성했습니다. 요청 실패 시에는 전체 데이터를 재조회해 상태를 복구했습니다.





Troubleshooting - Intercepting Routes 직접 접근 렌더링 문제 해결

Problem

홈 화면에서 상세 페이지를 Intercepting Routes로 구현했을 때, 목록에서 접근하는 경우와 직접 URL로 접근하는 경우의 렌더링 결과가 달라 일관되지 않은 UI가 표시되는 문제가 발생했습니다. 특히 직접 접근했을 때도 모달 형태로 렌더링되어 흐름이 어색한 문제가 있었습니다.

Cause

Intercepting Routes는 기존 페이지 위에 내용을 덮어쓰는 방식으로 동작하기 때문에, 접근 방식에 따라 동일한 페이지라도 다른 렌더링 결과가 발생했습니다. 이로 인해 “모달로 보여야 하는 경우”와 “독립된 페이지로 보여야 하는 경우”를 구분하지 못하는 구조였습니다.

Solution

일반 모달(useState) 방식은 클라이언트 접근 시에는 모달을 띄울 수 있지만, 직접 URL로 접근했을 때 동일한 UI를 제공할 수 없었습니다. Parallel Routes와 Intercepting Routes를 함께 활용하면 접근 방식에 따라 렌더링을 분리할 수 있어 이 조합을 선택했습니다. 목록에서 접근한 경우에는 모달 형태로, 직접 URL로 접근한 경우에는 전체 페이지로 렌더링되도록 구성했습니다.

Result

접근 방식에 관계없이 일관된 사용자 경험을 제공할 수 있게 되었으며, 페이지 구조를 상황에 맞게 유연하게 제어할 수 있도록 개선했습니다.

```tsx
export default function DetailPage({ params }) {
  return (
    <Modal>
      <DetailContent id={params.id} />
    </Modal>
  );
}
```

```tsx
// 목록 접근 → 모달
// app/(home)/@modal/(.)detail/[id]/page.tsx
export default function DetailModal({ params }) {
  return <Modal><DetailContent id={params.id} /></Modal>;
}
```

```tsx
// 직접 접근 → 페이지
// app/detail/[id]/page.tsx
export default function DetailPage({ params }) {
  return <DetailContent id={params.id} />;
}
```

Troubleshooting - JWT 재발급 과정에서 동시 요청 제어 문제 해결

Problem

여러 요청이 동시에 발생하는 상황에서 access token이 만료되면, 하나의 요청에서 401 응답이 발생한 이후 다른 요청들도 연달아 401을 반환하는 문제가 있었습니다. 이로 인해 일부 요청은 실패 상태로 남거나, 토큰 재발급 이전 상태에서 요청이 계속 진행되는 비효율적인 흐름이 발생했습니다.

Cause

토큰 만료 상황에서 요청 간 상태를 공유하지 못하고, 각 요청이 독립적으로 서버 응답을 처리하는 구조였습니다. 이로 인해 하나의 토큰 만료가 전체 요청 흐름에 반영되지 못하고, 이미 만료된 토큰으로 요청이 계속 진행되는 문제가 발생했습니다.

Solution

토큰이 만료된 상황에서 여러 요청이 동시에 발생하면 모든 요청이 각각 401을 받고 재발급을 시도해 불필요한 서버 부하가 발생했습니다. refreshLock을 구현해 최초 401 응답 시 재발급을 한 번만 수행하고, 나머지 요청은 재발급이 완료될 때까지 대기했다가 갱신된 토큰으로 한 번에 처리하도록 구성했습니다.

Result

토큰 만료 상황에서도 요청 흐름이 일관되게 유지되도록 개선했으며,
불필요한 401 응답과 요청 실패를 줄일 수 있었습니다.

```ts
if (status === 401) {
  await refreshAccessToken();
  return request();
}
```

```ts
let refreshLock: Promise<string | null> | null = null;

export async function withRefreshLock(refreshFn: () =>
Promise<string | null>) {
  if (refreshLock) return refreshLock;

  refreshLock = (async () => {
    try {
      return await refreshFn();
    } finally {
      refreshLock = null;
    }
  })();

  return refreshLock;
}
```

교내 유휴 자원 활용 관리형 배포 클라우드 플랫폼

M-ADP(마듭)

AWS 등 클라우드 서비스를 처음 접하는 학생·선생님이 비용 걱정 없이 배포를 경험할 수 있도록, 교내 유휴 서버 자원을 활용한 관리형 클라우드 플랫폼입니다. 잘 모르고 사용하다가 요금이 발생하는 일 없이, 안전한 환경에서 배포를 연습할 수 있습니다.

제작 기간

2025.12. ~ 2026.04

운영 기간

2026. 04 ~ 현재

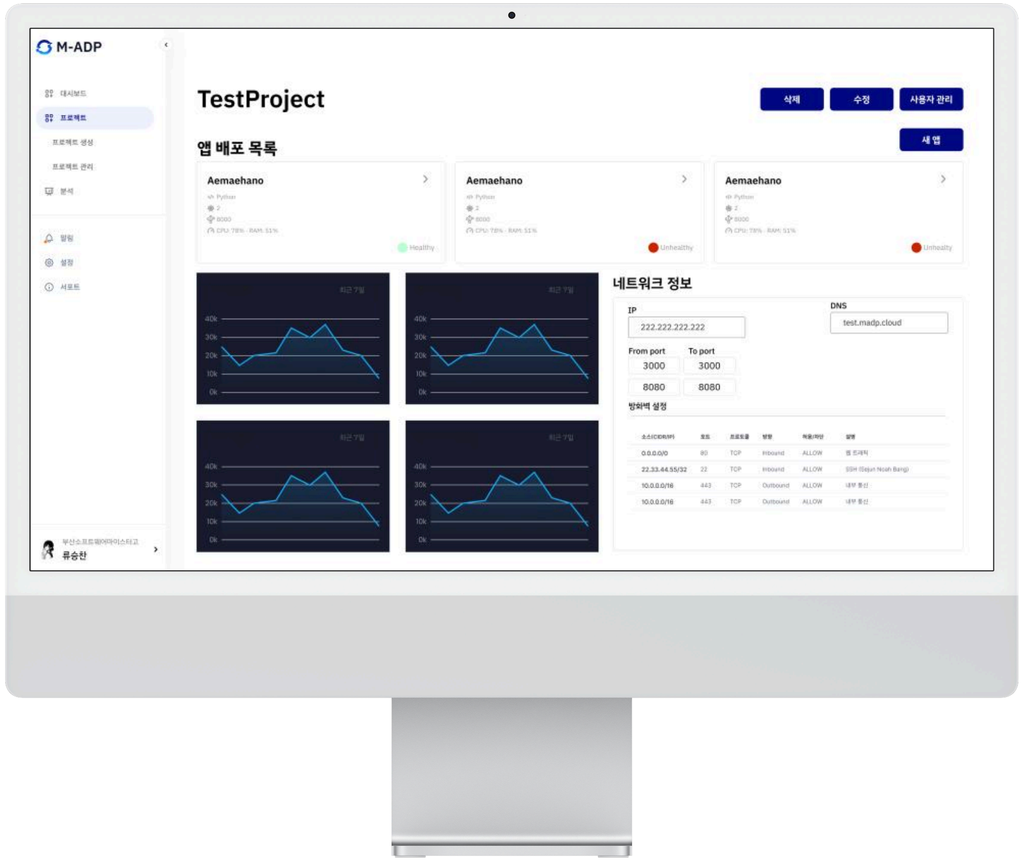
기술스택

깃허브

깃허브 바로가기

기여

Google/Github OAuth 연동 로그인 구현
프로젝트 생성 구현
프로젝트 대시보드 구현
애플리케이션 생성 구현
ChatOps 구현



#교내 AI 경진대회 출품작 #AI EXPO 전시

Main features & Contribution

프로젝트 상세 조회

애플리케이션 목록, CPU·Memory·Disk 자원 사용량, 멤버 정보를 관리하는 대시보드입니다. OWNER/VIEWER 역할에 따라 수정 권한을 분기 처리했습니다.

애플리케이션 생성

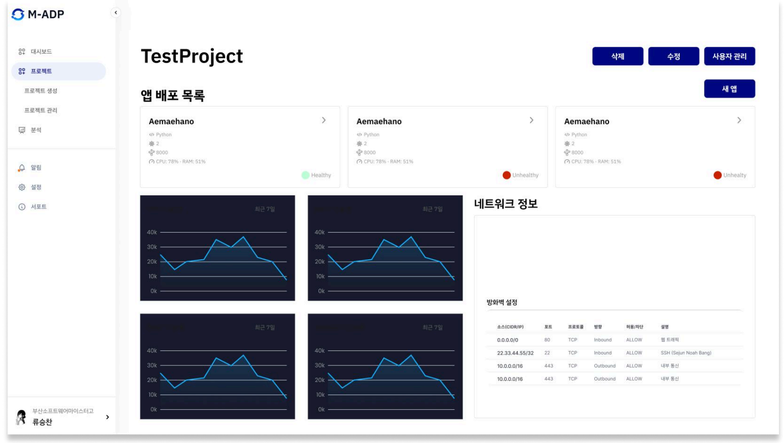
프로젝트 내 애플리케이션을 생성하는 화면입니다. 앱 이름 입력 시 한글·대문자를 실시간으로 제거해 백엔드 네이밍 규칙을 입력 단계에서 강제했습니다.

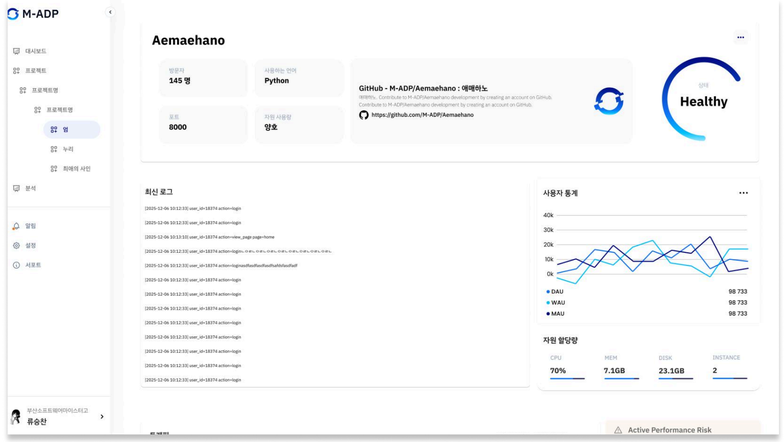
ChatOps

채팅으로 배포 환경을 제어하는 LLM 기반 인터페이스입니다. native EventSource는 Authorization 헤더를 지원하지 않아 fetch + ReadableStream으로 직접 SSE를 구현했습니다. 연결 실패 시 exponential backoff로 재연결하고, Last-Event-ID로 누락 이벤트를 복구했습니다.

로그인

Google OAuth와 GitHub OAuth를 순차적으로 연동하는 로그인 흐름을 구현했습니다. OAuth 인증 후 발급받은 code를 백엔드에 전달하고, 응답으로 받은 토큰을 쿠키에 저장했습니다. 응답의 is_authenticated 값으로 GitHub 연동 여부를 판단해, 이미 Google 로그인이 완료된 사용자는 GitHub 단계를 건너뛰도록 분기 처리했습니다.





Troubleshooting - 새로고침 시 인증 상태 소실 문제 해결

Problem

Zustand 메모리에만 토큰을 저장하고 있어, 새로고침 시 인증 상태가 초기화되며 로그인 페이지로 리다이렉트되는 문제가 있었습니다.

Cause

메모리에만 저장된 토큰은 새로고침 시 유지되지 않아, 인증이 필요한 페이지에서 토큰이 null로 판단되어 로그인 페이지로 강제 이동되는 구조였습니다.

Solution

localStorage는 JavaScript로 직접 접근 가능해 XSS에 취약하기 때문에 쿠키를 선택했습니다. refreshToken은 백엔드에서 httpOnly 쿠키로 설정해 JavaScript 접근을 차단하고, accessToken은 클라이언트 쿠키에 저장했습니다. 또한 인증 체크 로직을 Next.js 미들웨어로 옮겨 클라이언트 렌더링 이전 요청 단계에서 바로 검증하도록 구성했습니다.

Result

새로고침 이후에도 인증 상태가 유지되도록 개선했고, 불필요한 로그인 페이지 리다이렉트를 방지했습니다.

```ts
// authStore.ts
partialize: (state) => ({ step: state.step }), // ⚠️ token 미포함

// MainLayout.tsx
const token = useAuthStore((state) => state.token);
useEffect(() => {
  if (!isAuthPage && !token) {   // 새로고침 → token === null
    router.replace('/login');
  }
}, [isAuthPage, token, router]);
```

```ts
// middleware.ts
export function middleware(request: NextRequest) {
  const token = request.cookies.get('token')?.value;
  if (!token && !isAuthPage) {
    return NextResponse.redirect(new URL('/login', request.url));
  }
  return NextResponse.next();
}
```

Troubleshooting - Authorization 헤더 미지원으로 인한 SSE 직접 구현

Problem

LLM 기반 ChatOps 기능에서 SSE로 응답을 스트리밍하려 했으나, native EventSource가 커스텀 헤더를 지원하지 않아 인증이 필요한 엔드포인트에 연결할 수 없었습니다.

Cause

EventSource는 브라우저 스펙상 GET 요청만 지원하고 헤더를 직접 설정할 수 없습니다. Authorization 헤더 없이는 인증된 스트리밍 연결 자체가 불가능한 구조였습니다.

Solution

fetch + ReadableStream으로 SSE를 직접 구현했습니다. Last-Event-ID를 헤더와 쿼리 파라미터 양쪽으로 전송해 프록시 환경에서도 누락 이벤트를 복구하고, exponential backoff(최대 5회)로 재연결하도록 구성했습니다. 4xx는 재시도하지 않아 잘못된 요청 반복을 방지했습니다.

Result

백엔드 롤링 배포 중에도 메시지 유실 없이 응답을 끝까지 수신할 수 있게 되었습니다. 4xx 분기 처리와 AbortController 정리로 불필요한 재시도와 좀비 커넥션을 방지했습니다.

```
const evtSource = new EventSource('/api/stream');
// Authorization 헤더 설정 불가
```

```
const res = await fetch('/api/stream', {
  headers: {
    Authorization: `Bearer ${token}`,
    'Last-Event-ID': lastSequenceRef.current,
  },
  signal: abortController.signal,
});
const reader = res.body!.getReader();
```

매주 바뀌는 주보와 월간 일정을 관리하는 교회 웹

순복음범천교회 웹사이트

교회 소개, 예배 안내, 주보·월간 일정을 제공하는 웹사이트입니다. 담당자가 개발자 없이도 매주 콘텐츠를 직접 갱신할 수 있도록, 기획부터 백엔드 설계, 배포까지 진행했습니다.

제작 기간

2026.01. ~ 2026.05.

운영 기간

2026. 05 ~ 현재

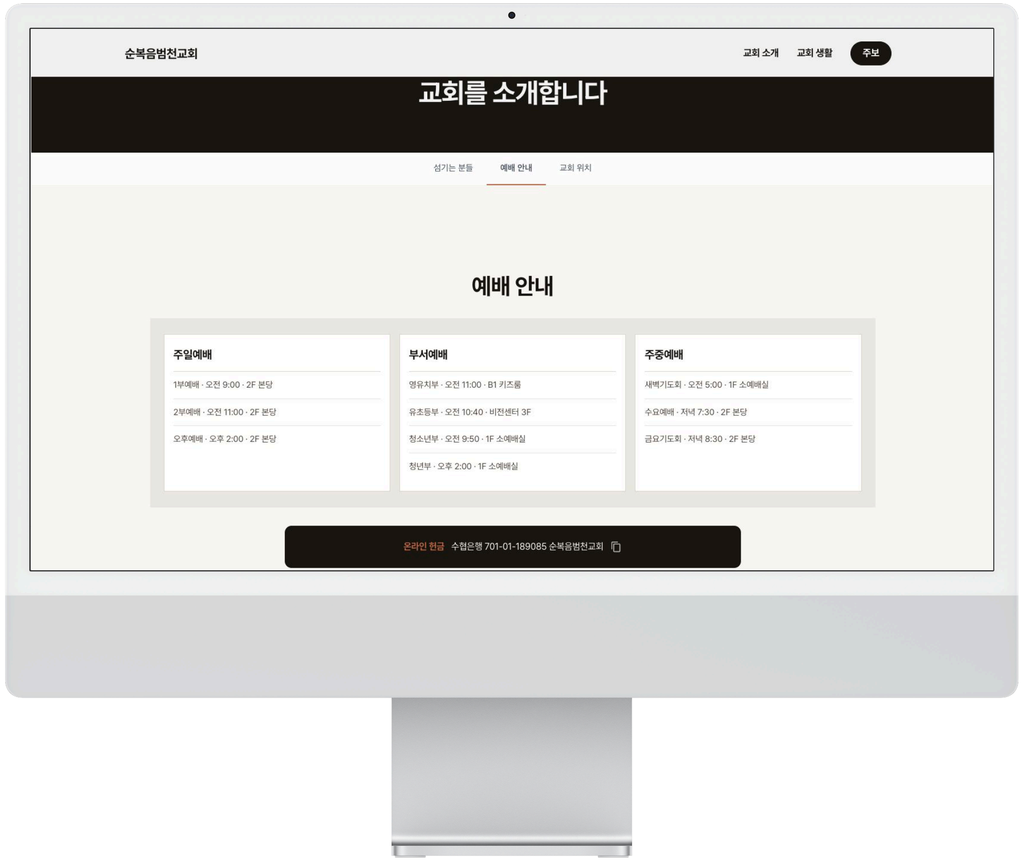
기술스택

깃허브

깃허브 바로가기

기여

교회 소개·예배 안내 등 전체 페이지 구현
주보·월간 일정 어드민 페이지 구현
FastAPI 기반 백엔드 API 설계
MySQL 데이터베이스 스키마 설계
Docker 컨테이너화 및 카페24 서버 배포



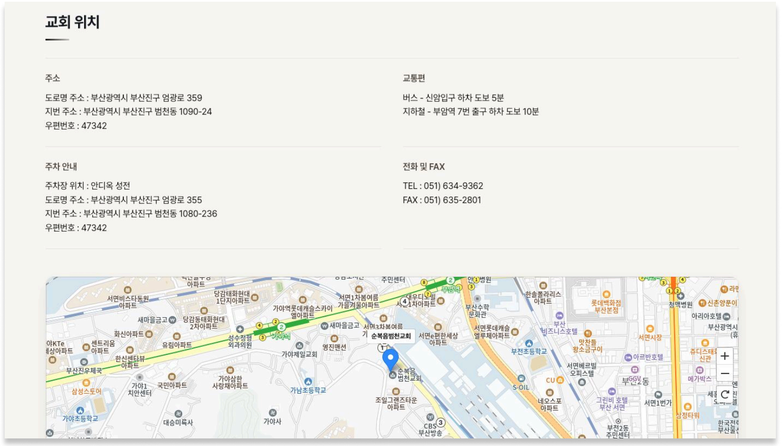
#교회 실사용 서비스 #서비스 바로가기

Main features & Contribution

기독교대한하나님의성회
순복음범천교회

교회 소개·안내 페이지

교회 소개, 예배 안내, 오시는 길 등 방문자가 접하는 전체 페이지를 구현했습니다. Next.js 기반으로 페이지 구조를 설계하고 라우팅을 구성했습니다.

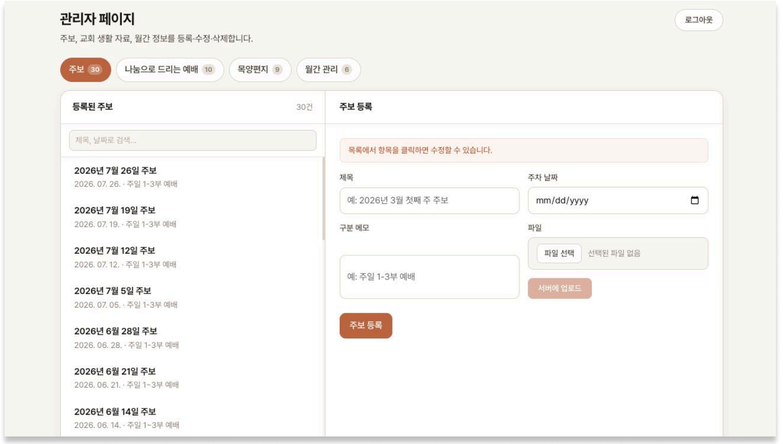


주보·월간 일정 어드민 페이지

담당자가 개발자 없이도 매주 주보와 월간 일정을 직접 등록·수정할 수 있는 어드민 페이지를 구현했습니다. FastAPI 기반 API로 CRUD 요청을 처리하고, 등록 즉시 실제 서비스 화면에 반영되도록 했습니다.

백엔드 API 및 데이터베이스 설계

FastAPI로 백엔드를 처음부터 설계하고, MySQL 스키마를 직접 구성했습니다. 콘텐츠 구조(주보, 일정, 안내 정보)를 나눠 테이블을 설계하고, 프론트엔드에서 필요한 데이터만 효율적으로 조회할 수 있도록 API를 구성했습니다.



배포 및 운영

Docker로 백엔드를 컨테이너화해 카페24 리눅스 서버에 배포했고, 프론트엔드는 Vercel로 분리 배포했습니다. 운영 시작 이후에도 수정 요청이 들어올 때마다 직접 반영하며 서비스를 관리하고 있습니다.

오주현 Frontend Engineer 2026

Thank you for watching

+82 10-6637-7242
me@zuu3.kr