# 안녕하세요, **프론트엔드 개발자 송재욱**입니다.

---

## 문제의 '왜'를 추적, 본질을 명확히 규명하는 Developer입니다 🔍

복잡한 요구사항 속에서 **진짜 문제**를 재정의하고, 가장 단순하고 비용 효율적인 솔루션을 도출합니다.

여러 공공기관으로부터 위탁받은 프로젝트들에서 쉽게 예상하기 어려운 수많은 요구사항과 피드백을 받았습니다.
이 과정에서 그대로 개발하는 대신 원청과의 기대를 조율하며, 더 나은 타협점을 제안해 낭비를 줄여 왔습니다.

디자인이 목적에 맞지 않다고 판단되면, 디자이너와의 토의를 거쳐 직접 리디자인을 주도해 사용성을 끌어올린 경험이 있습니다.
UI/UX를 심미성을 넘어선 **문제 해결**의 영역으로 바라보며, 최종적으로 모든 이해관계자에게 최선의 가치를 만드는 데 집중합니다.

## 배우는 자세로 기술의 경계를 넓혀가는 Explorer입니다 🧭

프론트엔드를 중심으로 서비스를 만들어왔지만 저는 다양한 기술을 배우는 과정 자체를 즐깁니다.

문제를 해결하다가 **MLOps**·디자이너·백엔드가 필요해지면 맡은 범위가 아니라고 넘기지 않고, 직접 학습하고 적용하며
해결 범위를 넓혀왔습니다. ✏️ MLOps 엔지니어 경험

## 책임감을 바탕으로 오너십을 실천하는 Leader입니다 🚀

서비스의 소유자가 아닌 책임자의 스탠스로, 독단보다는 소통을 통해 팀이 같은 방향을 바라보게 합니다.

**2년간 부회장**으로 활동하며 학생회와 학교 전체를 리드하고 매주 교장·교감 선생님과의 정기 회의를 통해 학교의 주요 현안을 논의했습니다. 또한 웹 개발 전공 동아리 'INCUBE'를 창단하고, **교내 최대 규모**까지 성장시켜 후배 양성을 핵심 가치로 삼아 운영해 오고 있습니다.

목적을 위해 개인적인 리소스를 아낌없이 투자하며 팀과 사용자 모두에게 유의미한 결과를 만들어내고자 합니다.

## 연락처

---


📞 010-7745-5407

s24064@gsm.hs.kr

@zaewc

@haensol

# 光탈페 플랫폼

**광주학생탈렌트페스티벌 행사 관리 공식 웹 서비스** 2025.02. - 2025.10.


## 개요

- 光탈페(광주학생탈렌트페스티벌)는 교육청과 광주고등학생의회 주도로 운영되는 오디션 프로그램으로 기존(23~24년) 예매 서비스는 교육청이 SI 업체에 위탁 운영했으나, 기능적 완성도와 UX 불편 존재
- 이에 플랫폼을 예매 기능뿐만 아니라 프로그램 소개, 슬로건 공모, 참여 신청, 실시간 투표 기능까지 아우르는 형태로 확장 개발, **1000만+ 원의 외주 입찰 비용을 절감**하는 성과
- 행사 당일 **DAU 3000+**명, Vercel 기준 누적 트래픽 **7.3만+**, GA 기준 이벤트 **5만+**건 달성

## 주요 역할

- 자체 Auth 로직 구현
- 실시간 좌석 예매 기능(SSE) 구현, 계정별 예매 여부에 따라 페이지가 달라지도록 동적 라우팅 적용
- SEO 최적화·성능 개선과 운영 중 수정 요청/피드백을 지속 반영하며 품질 향상

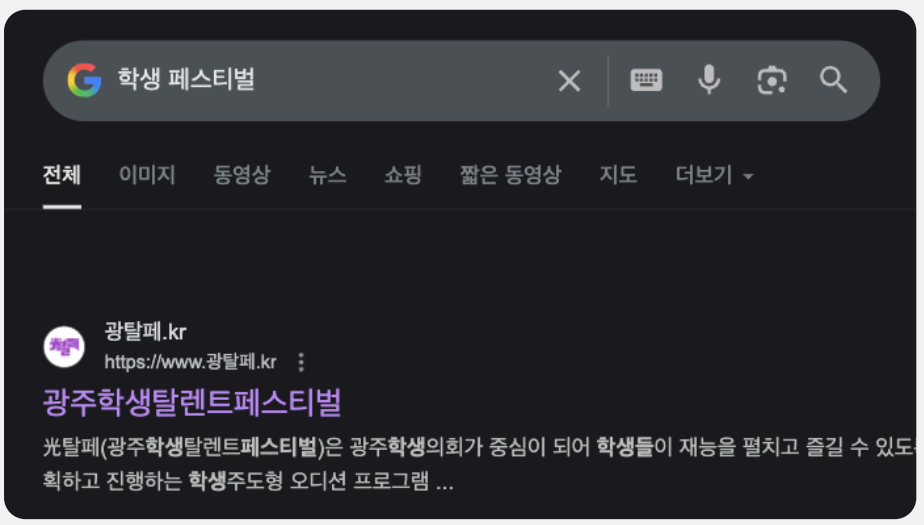
## 검색 엔진 최적화

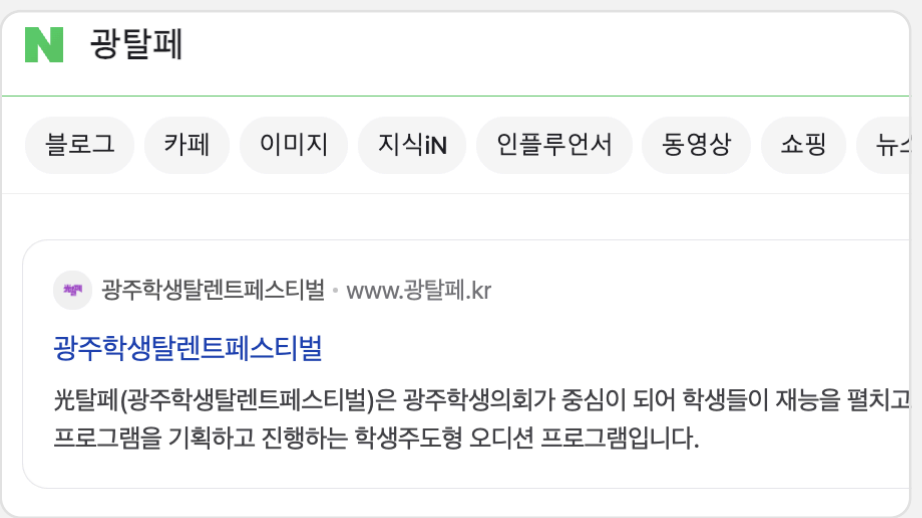
### AS IS

- 서비스 이용 고객층 중에 정보기기에 익숙하지 않은 사용자가 많으며, 검색 엔진을 통해 서비스를 탐색하는 경우가 대부분이었기 때문에 원활한 접근성을 제공할 필요가 있었습니다.

### TO BE

- next.js의 Metadata API를 이용하여 openGraph 와 keywords등을 설정했습니다.
- prebuild 시 next-sitemap이 실행되도록 하여 사이트맵 생성 과정을 자동화했습니다.
- 서비스 페이지가 대부분의 검색 엔진에서 최상단에 노출되는 결과를 달성하였습니다.

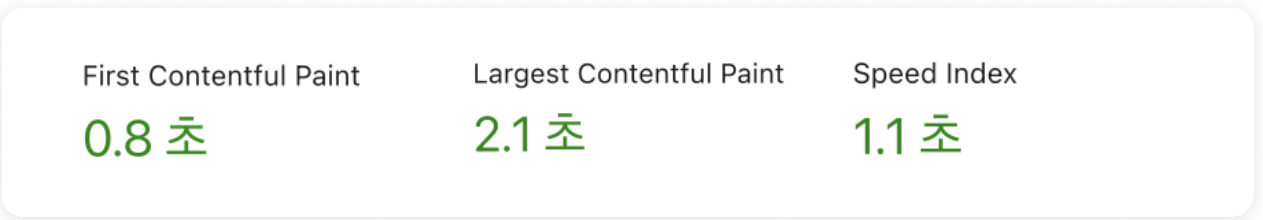



## /home 페이지 초기 로딩 성능 개선

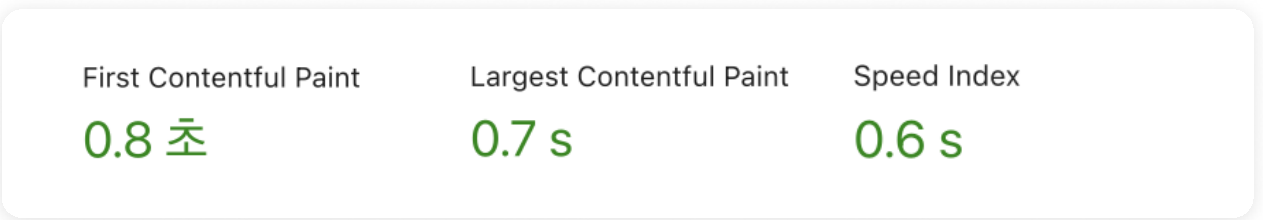
### AS IS

- 정성적 리서치 결과 /home 페이지의 로딩 속도가 느리다는 피드백이 있었습니다.
- 대용량 비디오와 이미지를 포함한 모든 섹션이 한 번에 로드되어, 초기 로딩 속도가 저하되는 문제였습니다.




### TO BE

- SSG와 섹션별 Lazy Loading을 도입해 초기 부하를 저하시켰습니다
- 히어로 섹션 즉시 로드 + Above-the-fold 이미지를 priority preload로 처리하여 일관된 UX를 유지했습니다.
- Lighthouse 측정 결과 FCP, LCP, SI를 각각 약 60% 개선하는 성과를 보였습니다.



## Tanstack Query 캐시가 업데이트 돼도 UI가 갱신되지 않는 문제 해결

### 문제 상황

- SSE를 통해 좌석 상태 변경 이벤트는 정상적으로 수신되고 Tanstack Query 캐시도 올바르게 업데이트되었지만, UI에 표시되는 섹션 별 좌석 예약 된 좌석 수 정보는 페이지를 새로고침하기 전까지 갱신되지 않았습니다.

### 원인 파악

- 기존 코드에서는 useMemo의 의존성 배열에 queryClient 객체를 포함했으나, 이는 캐시 내용이 변경되어도 참조값이 동일하게 유지되는 불변 객체라는 것을 알았습니다.
- queryClient.getQueryData()는 호출 시점의 캐시 값만 반환하는 일회성 조회 방식으로, Tanstack Query의 reactive 시스템과 연결되지 않았던 것입니다.

```
const seatInfoMap = useMemo(() => {
  const map: Record<Section, string> = {} as Record<Section, string>;

  SECTIONS.forEach(section => {
    const cachedSeats = queryClient.getQueryData<Seat[]>(seatQueryKeys.seatState(section));

    if (cachedSeats) {
      const occupied = cachedSeats.filter(seat => seat.status === SEAT_STATUS.OCCUPIED).length;
      map[section] = `${occupied}/${total}`;
    }
  });

  return map;
}, [isPrefetching, queryClient]);
```

### 해결

- useQueries를 사용하여 모든 섹션의 좌석 상태 쿼리를 동시에 구독 → 섹션의 캐시가 변경될 때마다 해당 쿼리 객체가 자동으로 업데이트되도록 했습니다.
- useMemo의 의존성 배열을 sectionQueries 배열로 변경함으로써 캐시 내용 변경 시 좌석 점유율을 재계산합니다.

# GOGO

**전국 중·고등학교 대상 스포츠 행사 플랫폼** 2025.03. - 2025.05.


프로젝트 기획서 레포지토리

## 개요

- 기존 체육행사 시스템에서는 신청서를 수기로 작성하여 제출해야 했기에 신청서 작성을 위해 반장들이 직접 발로 뛰어야 했고 각 종목에 출전할 인원 모집을 위해 따로 디스코드 서버를 파서 배정된 인원을 확인해야하는 불편함이 있었습니다.
- GOGO는 이를 개선하기 위해 팀 신청부터 동적 대진표 및 일정 관리 기능을 제공하여 행사 관리자, 체육 선생님과 학생들을 도와주고  승부예측과 같은 시스템을 도입하여 학생들의 참여도와 관심도를 높히기 위한 서비스입니다.
- 본교 체육대회에서 **DAU 300+**명, GA 기준 **누적 이벤트 17만+**건을 기록하며 서비스 중이고, 전국 단위 확장 이라는 OKR 목표 아래 발전하고 있습니다.

## 주요 역할

- 대회 운영 MVP 플로우(스테이지/대진표/팀/매치) 설계·구현
- DnD 기반 동적 대진표(Bracket) 기능 구현 및 팀 수/턴 등 edge case 대응
- 커뮤니티 이미지 첨부/조회수 표시 구현, 댓글 입력/정렬/반응형 등 UX 개선

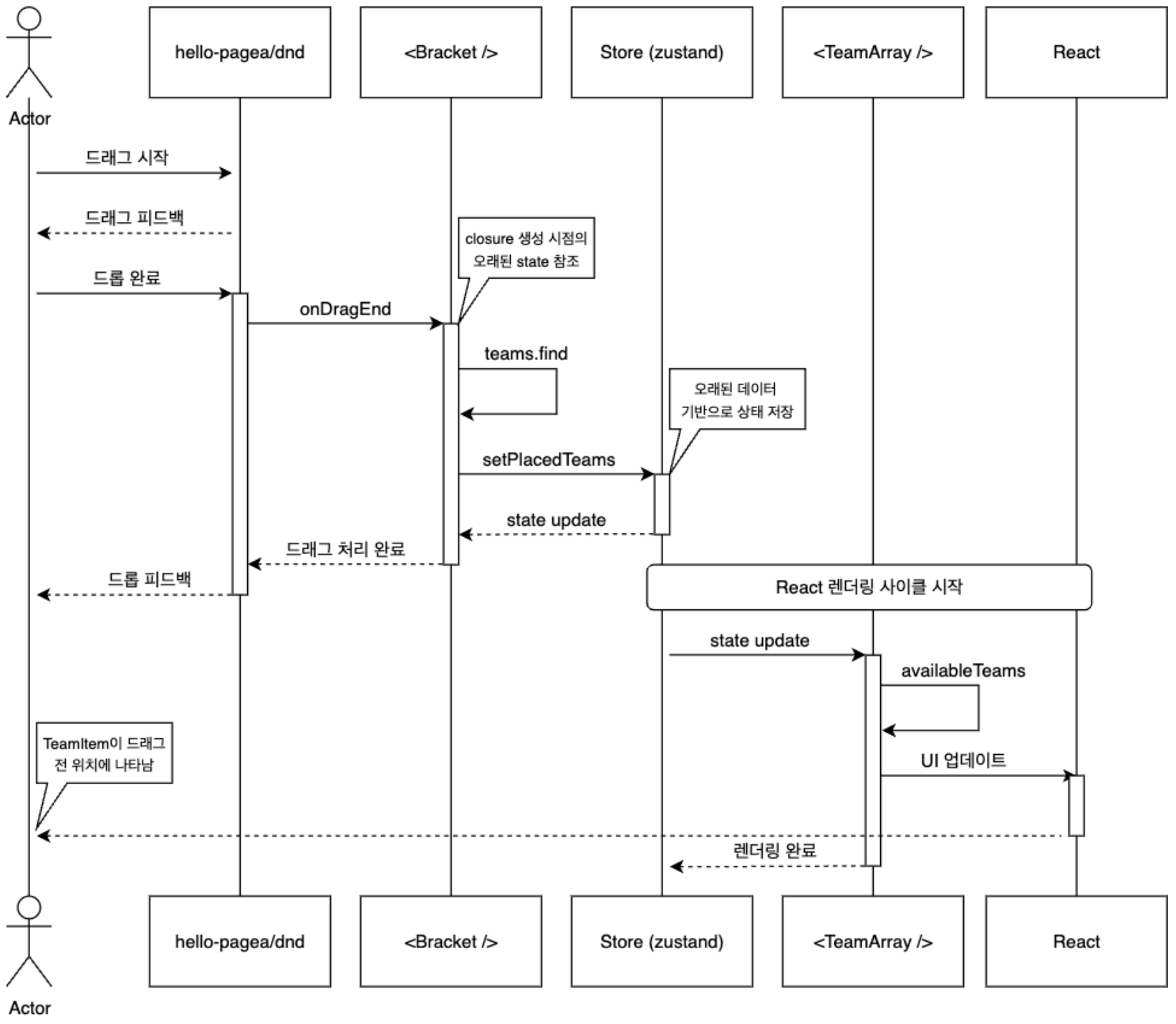
## 빠르고 연속적인 비동기 환경(DND)에서 stale closure 문제 해결

### 문제 상황

- 여러 컴포넌트 간에서 드래그를 해도 드래그 전 상태로 돌아갔고, 리렌더링을 해야 반영되었습니다.

### 원인 파악

- onDragEnd 함수가 생성될 때, 내부에서 참조하는 teams와 placedTeams는 그 시점의 스냅샷으로 closure에 캡처됩니다. DND 이벤트 발생 시 호출되는 onDragEnd가 생성 시점의 stale state를 참조하여 zustand에 저장하는 것을 확인했고, 다음 렌더링까지 이 지연된 상태가 전파된다고 추측했습니다.
- availableTeams 함수와 TeamArray 컴포넌트 등 다른 곳에서 오염된 teams 데이터를 참조하여 추가적인 문제가 발생했습니다.




### 해결

- CustomEvent API 기반의 이벤트 버스를 도입하여 드래그 상태를 실시간으로 저장하고, 이벤트 발행 시 각 gameId별로 데이터를 독립적으로 관리하도록 설계했습니다.

## 드래그 대상이 부모 container의 스타일 영향을 받는 문제 해결

### 문제 상황

- overflow: hidden을 적용해야 하는 TeamArray의 하위 component인 TeamItem이 드래그 대상으로 사용되었는데, TeamArray의 범위를 벗어나면 hidden 처리되는 문제가 있었습니다.

### 원인 파악

- TeamArray 컴포넌트가 클리핑 컨텍스트가 되었기 때문에 DOM 하위에 있던 TeamItem이 클리핑에 걸렸기 때문이었습니다.

### 해결

- page component가 mount될 때 portal 전용 DOM 엘리먼트를 생성하여 document.body에 직접 추가한 뒤, 해당 portalRef를 TeamArray 컴포넌트에 props로 전달했습니다. TeamArray 렌더링 시 createPotal 함수를 이용해 드래그 중인 요소를 portalRef가 참조하는 DOM 요소로 렌더링했습니다.

```
return (
  <>
    {snapshot.isDragging && portalRef.current
      ? createPortal(draggableContent, portalRef.current)
      : draggableContent}
    {provided.placeholder}
  </>
);
```

## 가상 스크롤 라이브러리 직접 제작 및 적용을 통한 대규모 랭킹 UI 최적화

### AS IS

- 기존 IntersectionObserver 기반 무한 스크롤은 스크롤할수록 DOM 노드가 계속 누적되는 구조였고, 데이터가 증가할수록 초기 렌더링 비용과 스크롤 성능 저하가 발생할 수 있었습니다.
- 특히 랭킹 페이지는 학생들이 자신의 순위를 탐색하기 위해 반복적으로 스크롤하는 화면이기 때문에, 데이터 로딩 최적화보다 렌더링 자체의 비용을 줄이는 구조 개선이 필요했습니다.

### TO BE

- 가상 스크롤 라이브러리 scrolloop을 도입해 랭킹 페이지를 가상화 기반 리스트로 전환했습니다.
- hidden="until-found"와 beforematch 이벤트를 활용한 오프스크린 사이드카를 구현해, 전체 데이터를 브라우저 검색(ctrl+f) 대상에 유지하면서도 실제 화면 렌더링은 가상화된 리스트로 처리하도록 개선했습니다.

# 시민화폐 광산

**광산구 주민 화합을 위한 지역 화폐 거래 관리 서비스** 2025.04. - present


레포지토리(웹) 레포지토리(앱)

## 개요

- 시민화폐 광산은 광주광역시 광산구의 외주 프로젝트로, 지역 내에서 가상의 화폐 ‘광산’을 활용해 서비스와 물품을 거래할 수 있는 서비스입니다.
- 개발 기간동안 데일리 스크럼을 주최하며 팀 전원의 스크럼 단위 목표 공유 및 이슈 파악 체계화
- iOS 파트 개발자가 갑작스럽게 이탈하는 상황이 발생했고, 비록 제 담당은 어드민 웹 개발이었으나 원활한 진행과 원청과의 신뢰 유지를 위해 React Native를 활용하여 공백을 신속히 메우고 **App store 릴리즈**까지 진행

## 주요 역할

- 모바일앱: 거래 게시글 및 채팅으로 물건/서비스를 교환할 수 있는 MVP 플로우 구현
- 웹(어드민): 원청의 요구사항에 따라 가입 승인, 신고 처리, ‘광산’ 지급·회수 같은 운영 기능 개발 및 운영

## 크로스플랫폼 앱 테스트 인프라 구축

### AS IS

- 아키텍처 변경을 포함한 대규모 리팩토링이 예정되어 있었으나, QA 과정이 수동으로 이루어져 팀원들의 생산성과 일정에 부담이 있었습니다.
- 실제 사용자 흐름을 검증하는 E2E 테스트는 매번 시뮬레이터를 직접 조작해야 했습니다.

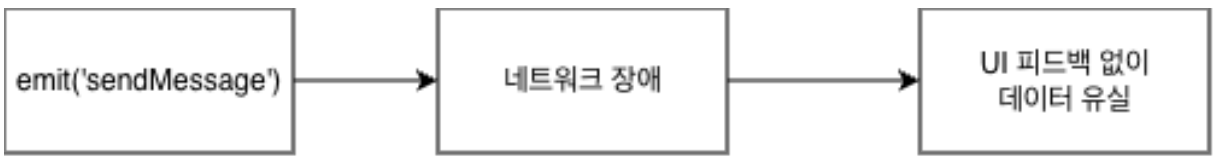
### TO BE

- Jest와 React Testing Library 기반에 MSW로 API 통신에 대한 핸들러를 구현하고, 테스트마다 동적으로 핸들러를 교체해 에러 시나리오까지 독립적으로 검증했습니다.
- Detox로 iOS/Android 양 플랫폼 E2E 테스트를 구성하고, 앱 시작·로그인·프로필 화면의 핵심 사용자 흐름을 자동 검증하도록 했습니다.
- Unit 테스트는 PR마다 Jest 커버리지 리포트가 코멘트로 자동 게시되고, E2E 테스트는 babel-plugin-istanbul을 빌드에 삽입해 시뮬레이터 실행 결과에서 커버리지를 추출함으로써, 두 테스트 레이어 모두 CI에서 커버리지를 자동으로 보고합니다.
- CocoaPods·Detox 프레임워크·Xcode 중간 산출물 캐싱을 적용해 CI 실행 시간을 최소화했으며, 테스트 결과는 Discord 웹훅으로 팀에게 즉시 전달됩니다.

## 웹소켓 데이터 전송 시 메시지 유실률 개선 및 오프라인 대응

### AS IS

- 기존에는 웹소켓 연결 상태(connectionState)만 확인하고 데이터를 전송했기 때문에, 도중에 연결이 끊길 시 전송중이었던 메시지를 유실하는 문제가  있었습니다.
- 오프라인 전송, 재시도 기능, 상태 피드백이 없어 UX가 저하되었다는 피드백이 있었습니다.




### TO BE

- 우선 zustand 기반 Persistent Queue를 도입하여 사용자 테스트 결과 일반적인 상황에서 데이터 유실률을 1% 이내로 감소시키는데 성공했습니다.
- 이를 근거로 낙관적 업데이트를 도입하여 UI가 즉시 반영되도록 개선했습니다.
- isSocketConnected를 useEffect로 감시하며 네트워크 재연결 시 zustand에 저장된 ‘PENDING’ 또는 ‘FAILED’ 상태의 메시지를 자동으로 재전송하도록 했습니다.

## 바텀시트에서 슬라이더 드래그 시 깜빡임 문제 해결 블로그 링크

### 문제 상황

- 슬라이더 드래그·드롭 시 바텀시트가 순간적으로 초기화되며 슬라이더 값이 원래 위치로 돌아갔습니다.
- 바텀시트 내부 컴포넌트(Input)들의 값을 변경할 때 마다 지속적인 깜빡임이 발생했습니다.

### 원인 파악

- 슬라이더 드래그 시 발생하는 연속적인 상태 업데이트가 상위 컴포넌트의 리렌더링을 유발하며, 이로 인해 바텀시트의 레이아웃과 제스처 애니메이션이 반복적으로 재계산되어 깜빡임이 발생했습니다.
- JS 스레드와 네이티브 UI 스레드 간의 비동기적 타이밍 차이가 누적되어 렌더링 타이밍이 어긋나면서 불안정성이 심화되었다고 생각했습니다.

### 해결

- 드래그 중에는 로컬 상태만 사용하고, 사용자가 명시적으로 작성 완료 버튼을 누를 때 부모 상태를 업데이트하도록 전략을 변경했습니다.

## 채팅 메시지 리스트 렌더링 성능 최적화

### AS IS

- 채팅방 화면은 메시지 입력, 새 메시지 수신, 소켓 이벤트로 리렌더가 빈번했지만, 매 렌더마다 전체 메시지 정렬과 timestamp 파싱이 반복되고 있었습니다.
- Array.sort 비교 콜백 내부에서 new Date()가 반복 호출되어 N=500 기준 렌더당 약 9,000회의 Date 객체 생성이 발생했고, FlatList 기본 설정으로 인해 메시지 누적에 따라 마운트 셀 수와 메모리 사용량이 증가했습니다.

### TO BE

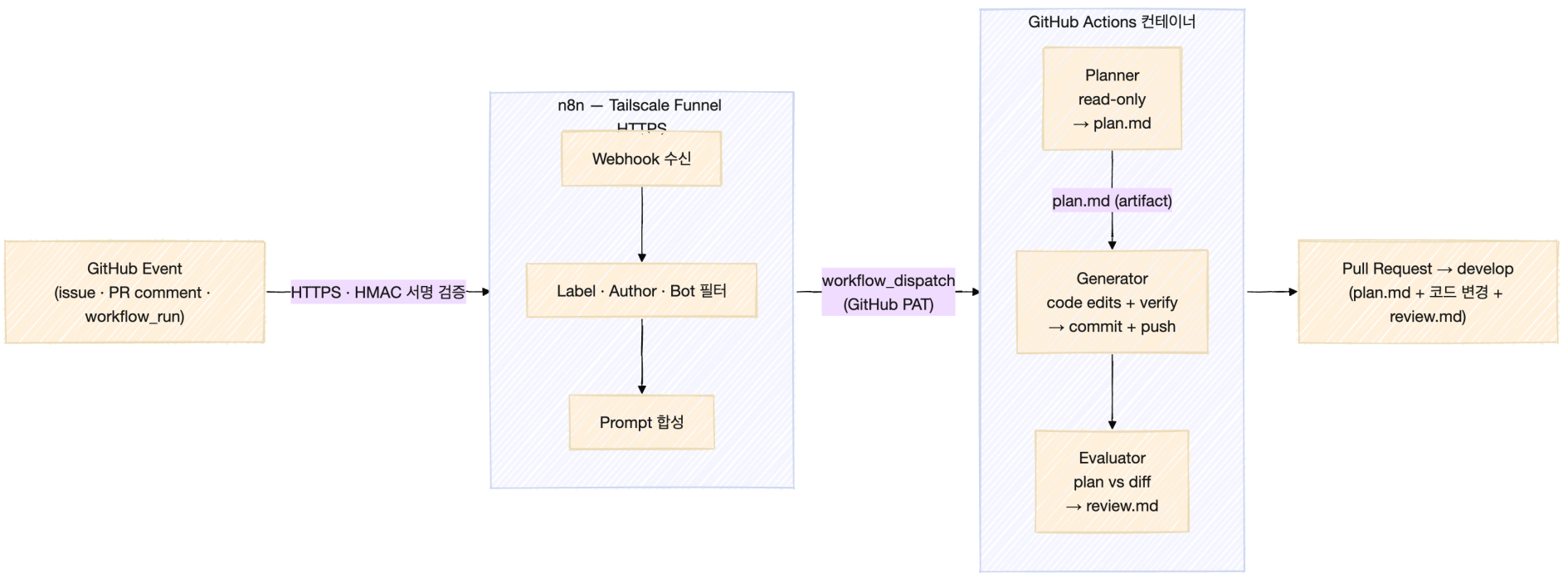
- FlatList 파라미터를 VirtualizedList의 viewport / window / batch 모델을 기준으로 튜닝했습니다. 블로그 링크
- 서버와 React Query 단계에서 메시지가 이미 ASC 정렬된다는 불변식을 확인하고, 전체 정렬을 제거한 뒤 findIndex + splice 기반 sorted insertion으로 변경했습니다.
- ISO 8601 timestamp의 lexicographic ordering을 활용해 Date 객체 생성을 제거하고, 메시지 가공 로직은 useMemo로 deps 변경 시에만 재계산되도록 개선했습니다.
- N=300 기준 마운트 셀 수를 약 210개에서 110개로 줄이고, 메모리 사용량을 약 1.6MB에서 0.9MB 수준으로 개선했습니다.

## 사이드 프로젝트

### Scrolloop | framework-agnostic 경량 스크롤 라이브러리 배포 링크

리스트를 windowing 기법으로 렌더링하는 VirtualList와 이를 이용한 무한 스크롤을 지원하는 InfiniteList를 제공하며, Trbopack 기반 모노레포 구조로 5개 프레임워크(라이브러리)를 동시에 지원합니다.

n8n 워크플로우 3종 + GitHub Actions 3-agent harness + composite action + Tailscale Funnel로 issue에 라벨을 다는 것만으로 AI가 PR을 생성하는 end-to-end 개발 파이프라인을 구축했습니다. 블로그 링크

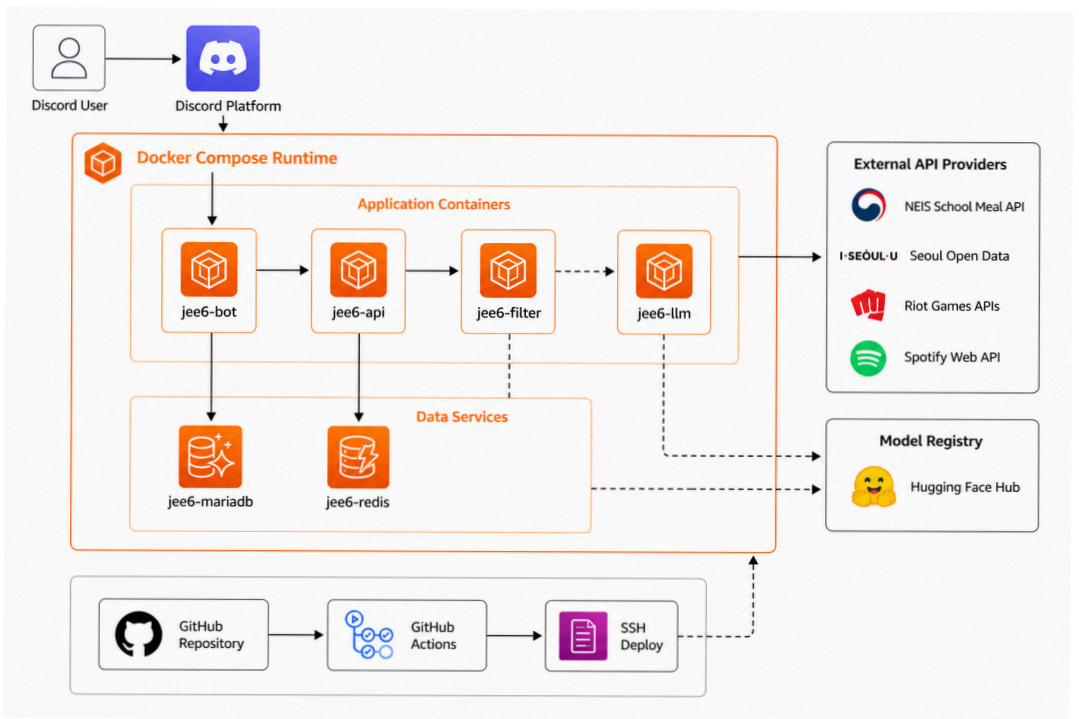



최소 설계를 목표로 약 45kb, 디펜던시 0개로 구현했으며, 현재 **누적 다운로드 2700+**건을 기록하고 있습니다.

### JEE6 | 교내 Discord 편의성 봇

서버 관리, 급식 조회, 욕설 메시지 필터, 미니게임 등을 이용할 수 있는 Discord 봇입니다.

자체 Ubuntu server에 Docker Compose 기반으로 하단 다이어그램과 같이 6개의 서비스를 분리 운영하고 GitHub Actions + SSH로 배포를 자동화했습니다. FastAPI Gateway와 Redis TTL 캐싱으로 외부 API 호출 경로를 통합하고, 반복 조회 데이터를 캐싱해 응답 지연과 rate limit 부담을 줄였습니다.




학교 공식 디스코드 서버를 포함한 40+개의 서버에 도입되었으며, **DAU 100+**명을 보유한 서비스 입니다.

## 오픈소스 기여

**typescript-eslint** | no-floating-promises 규칙에서 allowForKnownSafeCalls 옵션이 함수 이름을 올바르게 인식하지 못하는 문제 해결 블로그 링크

**TanStack/query** | eslint-plugin-query를 TS가 없는 환경에서도 사용할 수 있도록 TS를 optional peer dependency로 선언

**nest** | SSE, 캐시, swagger 관련 기능 검증을 위한 unit 및 e2e 테스트 추가

**fast_float** | double/float 파싱의 edge case 검증을 위한 unit 테스트 추가

**toss/es-toolkit** | 누락되어있던 keyBy export 구문을 추가하여 문서와의 불일치 해결

**typescript-eslint** | no-unnecessary-condition에서 타입 가드 검사 시 불필요한 조건문까지 감지하도록 수정

## 학력

- **광주소프트웨어마이스터고등학교 인공지능(AI) 학과** | 2024. 03. ~ 2027. 01.

## 자격 · 성적

- **TOPCIT 수준 3 (512점)** | 2025.05.24. 23회 정기평가
- **정보처리산업기사** | 2025.12.24. 정기 4회 (과정평가형)
- **TOEIC 785점 (LC 445, RC 340)** | 2026.01.25. 제561회차

## 수상

**2024**

- 제 10회 빛가람 에너지밸리 SW 작품 경진대회 장려상
- 2024 SW 동행 해커톤 장려상
- 2024 SW 동행 데모데이 장려상
- GSM 아이디어 페스티벌 3위 입상
- GSM 역량인증대회 3위 입상
- 제 7대 학생부회장 취임

**2025**

- 제 10회 HIGHTON 장려상
- GSM 역량인증대회 3위 입상
- 제 2회 GSM Dev Fest 2위, 3위 입상
- 제 8대 학생부회장 취임
- 1개년 개근상

**포트폴리오에 담지 못한 부분은 인터뷰에서 보여드리겠습니다.**

산업기능요원으로 복무 가능하며, 10월 이후부터 출근이 가능합니다.
학교와 산학협약 체결 시 7점, 선도기업 선정 5점, 총 12점의 산학협약 가산점 획득이 가능합니다.