조세훈

Frontend Engineer | React · React Native · Next.js

Tel: 010-8968-9274

Email: tpgnsjth@gmail.com

GitHub: github.com/951jth/pandytalk | **Google Play**: PandyTalk

---

경력

㈜델피콤 | Frontend Engineer | FE 3명, BE 3명

2021.05 ~ 현재 (약 **4년 10개월**)

- B2B 050 가상번호 통신 플랫폼(BizCall) 관리자 웹 및 랜딩 페이지 개발·운영
- 신규 기능 개발 및 프로젝트 단위 시스템 구축

프로필

5년 차 프론트엔드 개발자로서 문제를 스스로 정의하고 다양한 해결 방안을 제시해왔습니다.

성능 개선을 추상적으로 말하기보다 수치로 검증하며 구조 단위에서 개선하는 방식을 지향합니다.

스터디 활동과 개인 프로젝트 운영을 통해 다양한 직군과 소통하고 새로운 기술을 지속적으로 학습해왔습니다.

보유 기술

- **Performance**: Virtualization, Memory Profiling (Chrome DevTools), Long Task / Frame Drop 분석
- **Data & State**: Zustand, React Query, Redux, MobX, SQLite, Firebase (Firestore, FCM)
- **Frontend**: TypeScript, React, Next.js, React Native (CLI)
- **UI & Styling**: Tailwind CSS, MUI, Ant Design

회사 프로젝트 경험

---

1. Mozzle – 회원 관리 화면 페이지 개선

모임앱의 회원명부 관리 웹 서비스 (2026.01 ~ 2026.02

Mozzle의 모임 운영자를 위한 회원명부 관리 웹 서비스로서 회원 데이터를 조회·수정·관리하는 관리자 시스템입니다.

사용 기술: React, react-virtuoso (Virtualization), Chrome DevTools (Performance)

문제 정의

- **회원명부 편집 테이블**에서 Row 단위 수정 시 전체 리스트가 리렌더링되어 병목 현상 발생
- **Chrome Performance** 프로파일링을 통해 전체 리스트 re-render와 함께 DOM Node 및 메모리 사용량이 지속적으로 증가하는 현상을 확인
- 단일 페이지가 약 **1,200줄**에 달하는 구조로 유지보수 및 확장에 한계 존재

해결 전략

- **렌더링 범위 최적화:** react-virtuoso 기반 Virtualization을 도입하고 화면 내 Row만 렌더링되도록 Memoization 적용
- **로직 추상화:** 핵심 편집 로직을 커스텀 훅으로 분리하여 첫 페이지 코드량을 **1,200줄 → 140줄(88% 감소)**로 가독성 개선
- **정량적 검증**: 초기 렌더시간을 측정하는 커스텀 훅을 제작하여 프로덕션 빌드 기반의 최적화 유효성 검증

성과

- Total DOM Node 수 **85~90% 감소** (168,483 → 3,706)
- Worst Frame Duration 및 Long Task **약 90% 감소** (46건 → 1건)
- 첫 렌더링 시간 90% 개선 및 데이터 수정 시 상태가 변경된 **단일 Row**만 리렌더링되도록 컨텍스트 격리
- 텍스트 입력 → UI 반영까지 약 2초 지연 **→ 즉시 반영** 수준으로 개선 (1860ms → 28ms)
- 가상화 테이블을 **공통 컴포넌트화** 하여 재사용성 강화

※ 상세 측정 로그 및 아키텍처는 포트폴리오에서 확인 가능

---

2. 050 BizCall 관리자 웹 – 대용량 엑셀 다운로드 구조 개선

가상번호 관리 웹 서비스 (2023.05 ~ 2023.06)

050 가상번호를 활용한 개인정보 보호 및 마케팅 성과 측정 서비스로, 방대한 통화 내역(Call Log)과 녹취 데이터를 실시간으로 관리·분석하는 트래픽 처리 시스템

사용 기술: `JavaScript`, `SheetJS`

문제 상황

- 백엔드 개발자로부터 대용량 콜로그 Excel 다운로드 요청 시 서버 응답 지연 이슈가 공유됨
- 다운로드 요청이 동시에 여러 건 들어오면서, 엑셀 생성 작업이 서버의 다른 API 처리까지 방해하는 상황이 발생

해결

- 엑셀 생성 책임을 서버에서 클라이언트로 분산하는 구조로 재설계
- 서버는 대용량 JSON을 5000건 X 2건으로 스트리밍 응답하고, 클라이언트에서 10만 건 단위로 청킹 처리
- SheetJS 기반 워크북을 Web Worker를 활용하여 분할 생성 후 ZIP 압축하여 브라우저 메모리 피크를 분산
- 다운로드 진행률 UI 추가

성과

- 최대 60만 건 규모의 콜로그 다운로드 안정화
- 대용량 다운로드 실패 이슈 제거로 운영 안정성 개선 및 사용자 다운로드 UX 개선
- 서버 증설 없이 트래픽 부하 분산 구조 확보

---

3. Payking – 링크 기반 결제 앱 & 관리자 웹

간편 결제 앱 및 웹의 시스템 구축 (2025.06 ~ 2025.11 / FE 3명, BE 3명 등 총 8명)

별도의 단말기 없이 스마트폰만으로 대면(NFC·카메라) 및 비대면(SMS·URL) 결제를 지원하는 소상공인용 모바일 통합 결제 솔루션

사용 기술: `Next.js` (SSR), `React.js`, `React Native`, `CSS Grid`

역할

- React.js, Next.js 및 React Native 기반의 온/오프라인 통합 결제 솔루션(Web/App) 프론트엔드 개발

- SMS/URL 기반 결제 페이지 단독 구현 및 BFF 레이어 기반의 리다이렉트 결제 사용자 흐름(Flow) 제어
- 기획·PM과 요구사항을 재분석하고 관리자 입력 플로우를 단순화하는 방향으로 구조 개선
- 디자인팀과 협업하여 오류 안내 방식을 인라인 Validation 구조로 전환

문제 상황

- 관리자 입력 페이지의 초기 3단계 플로우(**상세 → 수정 → 저장**) 구조로 인해 상태 관리 복잡도 증가 및 일정 리스크 존재
- Alert 기반 오류 안내 방식으로 사용자 흐름 단절 발생
- 디자인 확정 전 개발 착수 상황에서 반복 JSX 구조는 향후 변경 범위 확장 위험 존재

해결

- 기획·PM과 협의하여 입력 플로우를 **2단계(수정 → 저장)** 구조로 재정의하고 상태 관리 범위를 축소
- Alert창 기반 오류 안내를 인라인 Validation UI 구조로 제안하고 전환하여 사용자 입력 흐름 단절 제거
- **JSON 설정 기반 선언적 폼** 구조를 도입하여 validation·필드 노출 조건·레이아웃을 설정화하고, **CSS Grid**(24분할) 기반 반응형 구조로 변경 대응 비용을 최소화

성과

- 입력 단계 축소로 상태 관리 복잡도 감소 및 일정 내 안정적 프로젝트 완료
- 관리자 주요 입력 폼 관련 페이지 LOC 평균 약 **23%** 감소
- 디자이너 교체로 시안이 전면 수정되었으나, 구조 변경 없이 설정값 수정만으로 대응 가능한 구조를 유지

---

4. 기타 프로젝트

- **050 BizCall 랜딩페이지 SEO 개선** | 2026.02 (1주)

  Next.js 기반 시맨틱 마크업 및 메타데이터 구조 개선, 코드 분할 적용으로

  Lighthouse Performance 59 → 91(+32), Accessibility 83 → 96(+13), SEO 91 → 100(+9) 개선
- **델피콤 홈페이지 및 관리자 페이지 (Next.js) |** 2024.07. ~ 2024.08 (1개월)

  Next.js API Route를 활용하여 단순 관리자 페이지 DB 조회 및 응답 가공까지 처리하는 풀스택 구조 구현 (백엔드 서버 없이 운영)
- **OCC 호텔·여행사 중계 플랫폼 |** 2024.02 ~ 2024.06 (4개월)

  react-i18next 기반의 다국어 리소스를 기능 및 페이지 도메인 단위로 분리하는 네임스페이스(Namespace) 방식으로 모듈화

개인 프로젝트

---

1. PandyTalk – 오프라인 퍼스트 기반 AI 채팅 앱 개인 프로젝트 (React Native)

개인 프로젝트 / 1인 개발 (2025.06 ~ 운영 중)

네트워크 연결이 끊겨도 소통이 중단되지 않는 채팅 환경을 위해 Offline-First 설계를 도입했으며, 실시간 AI 스트리밍 기술을 더해 마치 대화하듯 생동감 있는 답변 시스템을 구현한 프로젝트

문제 상황

- 장기간 미접속 사용자가 채팅방에 재진입할 경우, 과거 메시지를 일괄 조회해야 하면서 네트워크 지연 및 대량 Read 비용 증가 문제 발생
- 이전에는 Firestore 업데이트 방식에 의존하여 AI 답변이 완성될 때까지 사용자가 긴 시간을 대기해야 했고, 실시간 스트리밍 체감이 낮음.
- 로컬 DB, 메모리 캐시, 원격 데이터가 혼재되어 데이터 흐름 추적 및 오류 관측이 난해함.

해결 방향

- 메시지 순번(SEQ) 기반의 '데이터 간극 탐지(Data Gap Detection)' 로직 설계. 부족한 구간만 20건 단위로 페이징 조회하여 초기 로딩 속도 및 비용 최적화
- HTTP 기반 SSE(Server-Sent Events) 엔드포인트를 구축하여 AI 답변을 실시간 스트리밍으로 전환. 체감 대기 시간을 줄임과 동시에 완료 시점에만 DB에 영구 저장하는 하이브리드 전략 취득
- SQLite를 Source of Truth로 하는 단일 데이터 흐름 구조 설계. Service → Data(Local/Remote)로 이어지는 레이어드 아키텍처를 강제하여 데이터 접근 경로를 단일화하고 로깅 체계 정립

※ Mozzle 회원명부 관리 시스템 및 PandyTalk 프로젝트의 자세한 수치 및 코드, 시연영상은 아래링크를 통해 보실 수 있습니다.

👉 포트폴리오(Notion) 링크 바로가기