

## 한상일 | 팀과 협업하며 서비스 품질을 향상시키는 프론트엔드 개발자



Email [bsc7417@gmail.com](mailto:bsc7417@gmail.com)

Github <https://github.com/sangil94s>

Portfolio <https://www.fronthan.dev>

---

### Introduce

성과와 이슈를 먼저 개선 제안하고 실행한 프론트엔드 개발자 한상일입니다.

기존 서비스의 성능 이슈(**LCP 30초, 리소스 1,400개, Lighthouse NO\_FCP**)를 확인 후, React 기반으로 마이그레이션을 주도하고 5분 캐싱과 기능별 컴포넌트 분리 등으로 **LCP 1초, 리소스 50개 이하, Lighthouse 76점으로 개선**했습니다. MVP 배포 후 실 사용자에게 **7개월간 속도 관련 피드백 0건**을 유지했습니다.

필요성을 판단한 API 문서화 추가 와 디자인 가이드 수정을 먼저 제안, 협의 후 수정된 프로세스를 정착시켰습니다.

---

### Work Experience

#### 김스인더스트리 | 2022.12 - 2025.03 (약 2년 4개월)

한국/북미 약 300개 대리점에 수제 담배를 제조하는 기계를 판매하는 기업 (약 30명, 개발팀 10명)

- EJS → React 마이그레이션 주도, (유효성 검사, 전역 상태 관리, API 캐싱 추가 등) 개선 병행.
- API 문서화가 없던 환경에서 먼저 제안, Swagger/Notion 중 협의 후 Notion 도입, 작성/수정 후 PDF로 변환하여 팀원들과 공유하는 프로세스를 구축
- 반응형 기준이 부재한 상황에서 PM과 협의 후, PC는 FHD / 모바일은 (한국: 갤럭시 S20 Ultra, 미국: 아이폰 14 Pro Max) 기준으로 협의 후 2개 프로젝트 진행하며 프로세스 정착.

#### Compact-Machine | 2024.01 - 2025.03 (약 1년 2개월) | [링크: 포트폴리오 바로 가기](#)

- React(Vite), TypeScript, Tanstack/React-Query, TailwindCSS, Express, MySQL, GCP
- 전국 180개 대리점의 생산량 등 대리점 정보를 내부 부서가 조회하는 목적의 백오피스 프로젝트
- 총 7명(PM 1, 디자이너 1, HW 3, SW 2) 중 웹 개발과 GCP Compute Engine로 배포 전담.
- 이슈 - 레거시(APEHEX)의 주요 사례
  - 각 페이지에 필요한 코드가 분리 없이 작성, Header의 경우 로그인 제외 모든 페이지마다 작성되었고 관리자 페이지는 단일 파일 기준 약 2,000줄 등 프로젝트 유지보수 난이도 증가
  - 크롬 Performance 기준 LCP 약 30초, 리소스 요청 1,400개, Lighthouse NO\_FCP 발생
  - Ajax 8개를 forEach로 판매된 기계 수만큼 반복 호출, API의 캐싱 미적용으로 과도한 DB 접근

- 관리자 3개 권한 별 페이지 파일이 별도 분리, Header 등 중복 코드가 많아 유지보수 난이도 증가
- API 문서화 없음, 업데이트 또는 이슈 발생 시 소통 누락으로 API 확인에 지연 사례 발생
- 해결 - 컴팩트머신에서 개선한 사례
  - LCP 30초, 리소스 1,400건 등 이슈 인지 후, API에 페이지네이션, max-age=300 캐싱, Vite와 loadable-components로 코드 스플리팅, Tanstack/react-query에서 staeltime, retry 등 적용
  - **LCP 30초 → 1초 이하, 리소스 1400건 → 50건 이하** 등으로 APEHEX 대비 성능을 개선
  - 기존 관리자 페이지에서는 **Header 등 중복 코드가 각 파일마다 반복**, 유지 보수 난이도 높음. 로그인 후 권한 값을 Zustand에 저장 후 이를 기준으로 권한 별 조건부 렌더링을 적용.
  - 지도/표 등 기능별 컴포넌트화, Zustand의 persist로 새로고침 시 상태를 유지, 관리자 페이지 기준 **APEHEX 약 2,000줄의 코드가 Compact-Machine에서는 100줄 미만으로 축소.**
  - Swagger/Notion 중 후보군이 팀 내부에서 축소, 협의 결과 Notion으로 작성 후 PDF로 주고받는 게 선정되었고 해당 방식으로 문서화 프로세스 시작.
- 성과 - 컴팩트머신에서 개선 성과
  - **Performance** 탭 기준 LCP 30초 → 1초 이하, 리소스 요청 1,400개 → 50개 이하로 개선
  - Lighthouse (Desktop 기준) NO\_FCP → 3회 측정 시 평균 76점으로 개선
  - API 문서의 문서화/최신화 전담하며, 이후 프로젝트도 문서화를 진행하며 해당 프로세스를 정착
  - **MVP 배포 후 7개월간 속도 피드백 0건을 유지**, 배포 전 주 사용 부서로부터 레거시보단 빠르다란 평가를 얻었고 이후 주 사용 부서를 통해 팀 내부의 요청을 받고 추가 기능을 개발.


---

## AFO | 2023.03 - 2023.09 (약 6개월) | [링크: 포트폴리오 바로 가기](#)

- React(Vite), TypeScript, Tanstack/React-Query, TailwindCSS, Express, MySQL, GCP
- 레거시 기계 업그레이드와 동일 시기에 React로 백오피스 재구축과 UI/UX 개선을 병행하는 프로젝트
- 총 7명(PM 1, 디자이너 1, HW 3, SW 2) 중 웹 개발과, GCP Compute Engine로 배포 전담.
- 이슈 - 레거시(APEHEX)의 사례
  - 유효성 검사 부재하여 데이터 등록 시 필수 값 누락되어도 DB에 데이터 등록.
  - 9개 Log Page마다 CSV 다운로드, 필터 등 배치된 기능, 표/카드 등이 달라 UI/UX 일관성 부족
  - **Bootstrap + 인라인 스타일 + 약 3만 줄의 CSS 파일 모두 혼용**하여 유지보수 난이도 높음
  - 디자인 가이드가 존재하나, 사용 색상 정보가 부재하여 추가 확인 후 정확한 색상 정보 파악 가능
- 해결 - AFO에서 개선한 사례
  - 레거시의 유효성 검사 부재로 필수 값이 들어오지 않고 저장, React-Hook-Form과 Joi로 프론트/백엔드 각각 유효성 검사 도입, 이후 필수 값이 누락된 상태로 DB에 저장되는 사례 개선.
  - UI/UX 통일성 확보를 제안하여 PM과 협의 후 9개 Log Page → 5개로 축소, CSV 다운로드, 필터, 검색, 페이지네이션 4개 기능 배치 와 Log Page의 UI를 표로 통일하며 UI/UX 일관성 확보

- 약 3만 줄의 CSS + Bootstrap + 인라인 스타일 혼용 → TailwindCSS + DaisyUI로 통일.
  - 디자인 가이드 (AI 파일)에 사용 색상 정보가 없어 추가적인 소통 후 정확한 사용 색상 정보가 확인, 협의 후 가이드에 HEX 코드를 포함하여 불필요한 반복 소통 추가 발생을 개선.
  - 성과 - AFO에서 개선 성과
    - 프론트엔드/백엔드 유효성 검사 추가로 필수 값 누락 시 DB 등록을 방지하여 안정성 개선
    - **5개의 Log Page의 UI/UX를 먼저 통일성 확보를 제안하고 반영**하여 사용자에게 일관성 유지
    - 디자인 가이드에 **사용 색상 HEX 코드 값 추가**, 색상 확인을 위한 중복 소통 필요성 축소
- 

## Personal Project

Steamrader | 2025.04 - 2025.05 (약 1개월) |  [링크: 포트폴리오 바로 가기](#)

- Next.js(App Router), TypeScript, Prisma, Tanstack/React-Query, TailwindCSS, Supabase
  - Next.js로 프론트/백엔드를 개발, Steam의 API로 약 150개의 게임 중 할인 게임을 보여주는 프로젝트
  - 이슈
    - Steam API로, 데이터를 Supabase에 저장 시도 → 요청 후 약 10분간 Access Denied 발생
    - Google Search Console 등록 후 5건 미만으로 노출, 게임 리스트 API 응답에 1초 이상 소모
  - 해결
    - 모든 게임 리스트에서 appid 추출 후 요청한 기존 방식 → 유명 게임 appid 150개 분리 후 **할인율 15% 이상 + 리뷰 5천 개 이상만 호출**로 변경
    - metadata.title이 steamrader로 노출 어려움 인지, 스팀 할인정보로 변경 후 총 노출수 증가
  - 성과
    - CSR → SSR과 배포 리전 변경으로 큰 변화 없을 가능성과 Cold Start를 고려할 필요성 확인.
    - Next.js 메타데이터 수정을 통해 구글 총 노출 수 **약 5건 → 약 40건 증가**, SEO 중요성 인지.
- 

## Skill

**Frontend** : React, Next.js(App Router), Typescript, TailwindCSS, Zustand, Tanstack/React-Query

**Backend** : Express, Prisma, MySQL, Supabase

**Deploy** : GCP (Google Cloud Platform) , Vercel

---

## Education

- 오산대학교 산업공학과 졸업 | 2014.03 - 2018.02