Email	seunghwan7700@gmail.com
Mobile	010-5752-7305
Github	link
Blog	link

Frontend Engineer

이승환

소개

- End-to-End 프로젝트의 전체 개발 프로세스에 대한 이해를 기반으로 디자이너, 백엔드 개발자와 협업합니다.
- Next의 SSR환경을 이해하며 프로젝트를 진행한 경험이 있습니다.
- 확장성과 유지보수를 위해 상황에 맞는 다양한 **컴포넌트 패턴**을 사용하도록 노력하고 있습니다.

프로젝트

Commit Body / Frontend Engineer (Frontend: 2명, Backend: 1명, Designer: 1명) (2024.08 ~ 진행중)

사용자가 자신만의 운동 루틴을 만들고 기록을 관리하며, 운동 성과를 공유하고 커뮤니티에서 소통할 수 있는 서비스

기술 스택	NextJs, Tanstack-Query, Zustand, Tailwinds, ShadnUI, PWA, Github Actions
개발 내용	**[서버컴포넌트를 활용해 초기 렌더링 및 SEO 개선]** → **이슈** : CSR환경에서 운동 리스트API를 호출하면서 첫 렌더링시간이 **1.36초**로 지연 → **해결** : 서버컴포넌트로 전환 후 첫 운동 리스트API를 prefetch, 그 결과 첫 렌더링시간을 **1.36초 → 0.5초**로 단축 → **고민** : 서버컴포넌트를 사용해 JS번들 크기와 초기 렌더링속도를 감소 시켰지만 상황에 따라 **서버부담이 커진다고 판단** → **해결** : SSR환경의 fetch에서 제공하는 **캐시**를 활용해 서버부담을 최소화 **[Stepper 기반 회원가입 폼 개발]** → **1차 이슈** : 작성해야하는 폼이 많아 사용자가 도중이탈하고 싶다는 피드백을 받음 → **해결** : 회의를 통해 **Stepper** 기반의 회원가입 폼을 개발하여 UX를 개선하고 도중 이탈을 방지하도록 변경 → **2차 이슈** : 회원가입 절차가 여러 페이지로 나누어지면서 데이터 관리와 디버깅의 어려움이 생기는 문제 발생 → **해결** : **Funnel패턴**을 사용해 한 페이지에서 데이터 관리 **[좋아요 버튼을 누를 때 사용자에게 즉각적으로 응답이 오는 것처럼 적용하기 위한 낙관적 업데이트 적용]** → **이슈** : 좋아요 API 요청을 보냈을 때 데이터 최신화를 위해 운동 리스트를 매번 refetch 해줘야하는 문제 발생 → **고민** : 방법 1. 데이터 최신화를 위해 refetch시 좋아요API 요청을 하면 운동 리스트도 매번 다시 가져와야해서 http 요청을 두번하는게 부적절하다고 판단 방법 2. 좋아요API가 성공한다면 운동 리스트의 캐시를 조작, http 요청은 한번에 끝나지만 좋아요 API 응답이 늦어지면 사용자 경험 저하 방법 3. 좋아요API 실행 전 캐시를 미리 업데이트해 사용자에게 변경된 UI제공, 실패한다면 롤백 → **해결** : 좋아요 버튼 같은 경우 실패하기 어렵고 빠른 응답이 이상적이기때문에 낙관적 업데이트 (방법 3) 선택 그 결과 약 **0.7s** 딜레이 단축으로 사용자 경험 증가 **[반복되는 서버 요청에 대한 캐싱을 위해 리액트 쿼리 사용]** → **이슈** : 자주 변경되지 않는 데이터의 반복되는 요청으로 인한 네트워크 트래픽 발생 → **고민** : staleTime과 gcTime을 어떤 기준으로 설정해야하는지 → **해결** : staleTime이 cacheTime보다 짧게 설정해 응답 시간을 단축하고, 네트워크 요청을 최소화 **[사용자의 간편한 로그인을 위해 OAuth 사용]** → **이슈** : 사용자 인증 과정에서 복잡한 로그인 절차로 인한 사용자 경험 저하 → **고민** : 간편하고 안전한 인증 방식을 제공하기 위해 어떤 방법을 도입할지 → **해결** : OAuth(next-auth)를 사용해 Google, Kakao 소셜 로그인 구현, 인증 절차를 간소화하고 사용자 경험을 개선

피그마	Figma
깃허브	GitHub

Auction / Frontend Engineer (Frontend: 1명, Backend: 1명) (2023.12 ~ 2024.4)

물품 거래 서비스

기술 스택	Next.js, TypeScript, React Hook Form, Scss, Tanstack query, Recoil, RTL, Cypress
개발내용	**[Next.js의 SSR을 통해 웹 페이지의 SEO 및 초기 로딩 속도 개선]** **[여러 컴포넌트에서 상태를 공유하기 위해 Recoil 라이브러리 활용]** → **이슈 :** 여러 컴포넌트에서 공통된 상태를 사용해야하는 문제로 **Props Drilling** 발생 → **해결 :** 기존의 **Lifting State Up** 패턴의 코드 대신 **Recoil**를 활용해 복잡도 개선 **[합성 컴포넌트 패턴을 사용해 컴포넌트 재사용]** → **이슈** : 비슷한 Input UI가 많아서 컴포넌트로 만들면 props가 많아져 확장성이 떨어지는 상황 발생 → **해결** : **합성 컴포넌트 패턴**을 사용해 Input UI컴포넌트를 만들어 재사용 **[비제어 컴포넌트를 사용해 불필요한 리렌더링 최적화]** → **이슈** : 많은 Input의 유효성 검사의 어려움과 불필요한 리렌더링이 비효율적이라고 판단 → **해결** : 비제어 컴포넌트를 사용하는 react hook form을 활용해 유효성검사 및 **불필요한 리렌더링 최소화** **[댓글 무한 스크롤 개발]** [blog] → **1차 이슈** : onScroll방식을 사용해 무분별하게 이벤트가 호출되는 문제 발생 → **해결** : **Throttling**을 적용해 이벤트 호출 최소화 → **2차 이슈** : **Throttling** 적용 후에도 이벤트 호출이 잦다고 판단 → **해결** : onScroll방식보다 성능이 좋은 **Intersection Observer**를 활용해 개발 **[타입스크립트 도입 및 활용]** → **1차 이슈** : 자바스크립트로 진행시 예상치못한 타입 오류 발생 → **해결** : 컴파일 과정에서 에러 체크를 해주는 타입스크립트 도입 → **2차 이슈** : 타입이 많아져 비슷한 타입을 여러번 선언하는 상황 발생 → **해결** : Recode, Pick, Omit, Partial 등 유틸리티 타입을 활용해 타입 재사용 **[서버 부하 최소화 및 사용자 입력 처리 최적화를 위한 Debounce 기능 구현]** → **이슈** : 검색창에 검색어 입력 시 API 요청이 무분별하게 발생하는 문제 발생 → **해결** : 사용자 입력 처리 과정에서 서버 부하를 줄이기 위해 입력 텍스트에 500ms의 디바운스 적용 **[프론트엔드 개발 환경 개선을 위해 MSW도입]** [blog] → **이슈** : 백엔드 개발이 늦어져 API 연동이 늦어지는 상황 발생 → **해결** : MSW를 도입해 API 개발 속도에 영향 받지 않고 프론트엔드 개발 스프린트 진행
깃허브	GitHub

챌린지

동아리에서 진행한 기능 구현 챌린지/ Frontend Engineer (개인) (2024.05.03 ~ 2024.05.18)

한국임상정보 페이지의 검색영역을 클론하기

기술 스택	Vite, React, TypeScript, Tanstack query v5, Styled-components, Vitest, StoryBook, Github actions
개발 내용	**[검색 결과리스트 무한스크롤 구현]** → **1차 이슈** : 검색 결과리스트의 요소가 많아 한번에 데이터를 불러왔을 때 페이지 로딩 시간이 길어짐 → **해결** : **Intersection observer**를 활용해 무한스크롤 적용, 사용자가 스크롤을 바닥까지 내렸을 때 데이터를 부분적으로 가져와 페이지 로딩 시간 개선 → **2차 이슈** : 스크롤을 내릴 때마다 **DOM요소가 누적되어 성능 문제 발생** → **해결** : react-virtuoso 라이브러리를 사용해 **가상 스크롤 구현** **10,000개**의 데이터를 처리할 때, 렌더링 시간을 **약 800ms에서 100ms로 최적화** → **3차 이슈** : 스크롤을 내린 뒤 다른 페이지를 갔다가 돌아왔을 때 **스크롤 위치가 초기화되어 사용자 경험 저하** → **해결 방안** : 브라우저의 **세션 스토리지를 활용해서** 사용자의 **스크롤 위치를 저장**하고, 페이지 재방문 시 해당 위치를 복원 → **고민** : 반응형 UI에 대응하려면 ScrollTop을 어떻게 조절해야 할까

	[검색창에서 키보드만으로 추천, 최근검색어들로 이동 가능하도록 구현] → **고민** : 드롭박스로 구성된 추천검색어, 최근검색어를 마우스로 접근하는게 사용자 경험에 안좋다고 판단 → **해결** : **키보드를 활용**해 추천,최근 검색어들로 이동 가능하게 구현해 사용자 경험 향상 **[스토리북을 활용한 UI 테스트]** → **이슈 :** 새로운 기능이나 변경된 기능이 도입되었을 때, 예상치 못한 버그로 일관된 UI를 제공할 수 없을 수도 있다고 판단 → **해결 : 스토리북**을 적용해 컴포넌트의 동작을 쉽게 확인하고 **디버깅 및 시각화** **[Github actions를 통해 스토리북 배포 자동화]** → **이슈** : 스토리북을 수정할 때마다 매번 빌드를 해줘야하는 상황발생 → **해결** : Github actions를 활용해 push할시에 스토리북 **배포 자동화** **[Render props 패턴을 사용해 컴포넌트 재사용]** → **이슈** : 동일한 UI지만 다른 기능을 하는 요소가 있어서 재사용하기 어려운 상황 발생 → **해결** : **Render props 패턴**을 사용해 다른 기능을 가진 UI만 props로 넘겨줘서 해결
깃허브	GitHub

기술 스택

FrontEnd	BackEnd	DevOps
Language : JavaScript, TypeScript **Framework :** NextJs **Library :** React, Recoil, React-Query **Style :** Styled-components, Scss	**Language :** JavaScript, TypeScript **Framework :** Nest	**Devops :** Docker **Cloud :** AWS

경험

- 성결대학교 **(컴퓨터공학과)** 2018 ~ 2024.02 **(졸업)**
- [티타임즈 x techeer]실리콘밸리 sw 부트캠프 **2023.07 - 2023.08**
- 원티드 프리온보딩 프론트엔드 인턴쉽 과정 수료 **2023.04 - 2023.05** → 관련자료
- 실리콘밸리 개발자 멘토링 프로그램 Techeer 4 기 **2022.09 - 현재**
  - 실리콘밸리 엔지니어의 SW 개발자 커리어 그룹
    기술 세션, 프로젝트, 스터디 등 다양한 개발 및 네트워킹 활동에 집중하는 그룹
- [티타임즈 x techeer]실리콘밸리 sw 부트캠프 **2022.12 - 2023.2**
- [티타임즈 x techeer]실리콘밸리 sw 부트캠프 **2022.08 - 2022.10**