유저의 입장에서 생각하며

최선의 방향을 고민하는 개발자 이정필입니다.

포트폴리오 싸이월드 버전 보기

About Me

이정필 | Lee Jung Pill

1999.06.18

성결대학교 컴퓨터 공학과

Contact

010-5628-7623

wjdvlf99@naver.com

경기도 안양시 석수동

Skills

Javascript Typescript React React-Query Redux Zustand

Tailwind Css Styled-Component Chart.Js Aws GoogleCloud

Notion Conflunce Jira Slack Github

PROJECT

jaychis 24.09~ 25.01

커뮤니티

- lightHouse를 통해 측정한 성능 점수 90점 이상
- 가상화된 스크롤을 통한 무한 스크롤 구현으로 메모리 최소화
- 카카오 Oauth, 공유하기 기능 개발
- useContext와 에러 모달을 통한 에러 관리
- Stage와 Production 서버 분리로 안정적인 사용자 경험을 보장

TypeScript, React, Redux, Ant-design, Styled-component,Jira, Conflunce

싹둑싹둑 24.02 ~ 24.06

의류 쇼핑몰

- 아토믹 디자인 패턴 적용을 통한 컴포넌트 분리
- Zustand, React-query를 통한 상태관리
- ICT 한이음 프로젝트 수료 및 졸업작품 경진대회 수상
- 깃허브 액션을 통한 CI / CD 파이프라인 구축
- 특허 출원

React, JavaScript, Tailwind, Vite, React-query, Zustand, Ant-design

제이치스

가장 많이 검색된 주제, 가장 많은 댓글이 달린 게시글 등을 보여주는 백오피스 커뮤니티

개발 기간

- 24.09 ~ 25.01

개발 인원

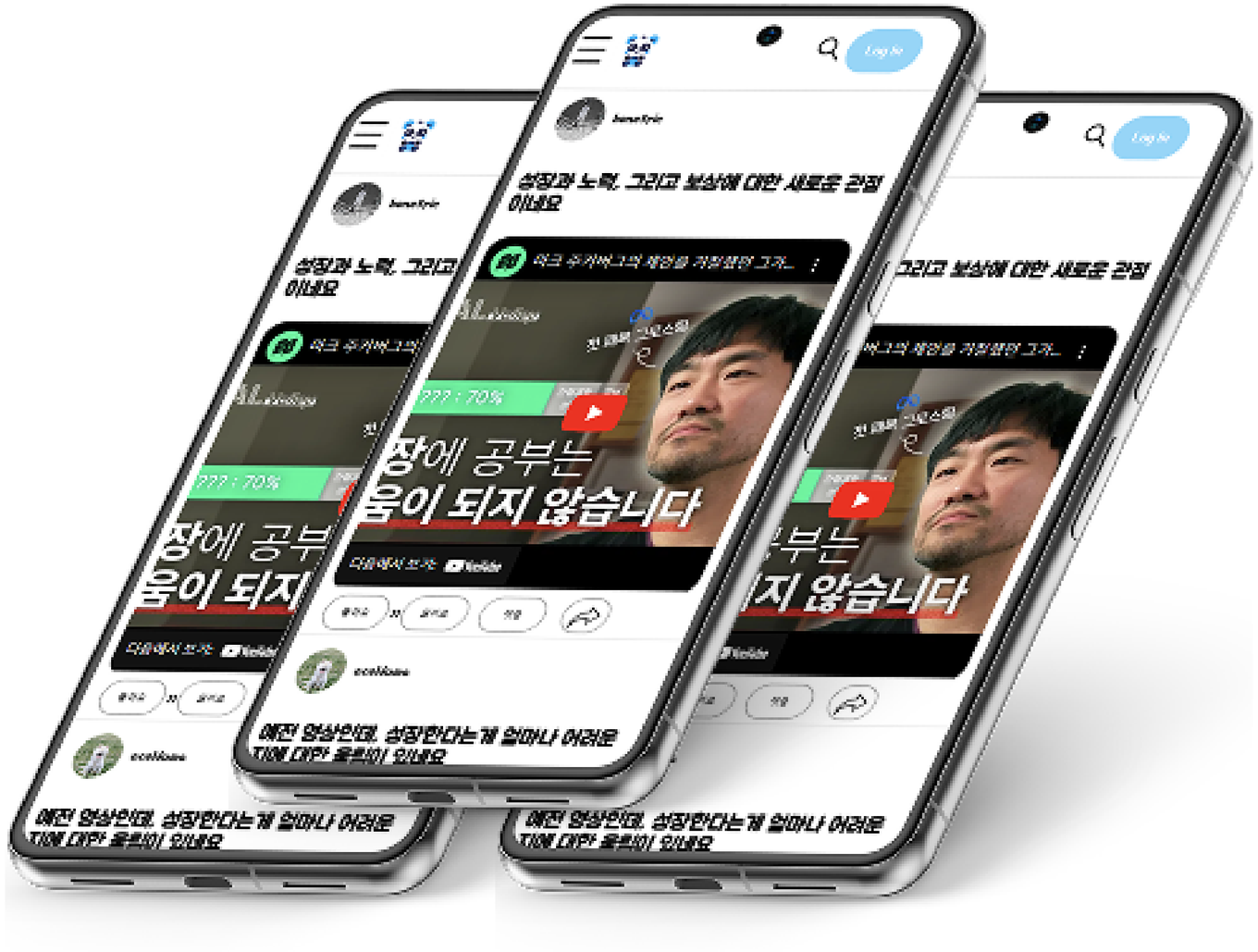
- Front 1명, Back 1명, Full 1명

협업

- Git Hub
- Jira
- Conflunce

기술 스택

- React
- Styled-Component
- Redux
- ant-Design



Jaychis

기여 내용

[카카오 API]

- **카카오** OAuth **인증**을 기반으로 기존 계정 존재 여부에 따라 로그인 성공 또는 회원가입 페이지로 유도하여 초기 진입장벽을 낮췄습니다.
- **카카오 공유하기**를 통해 사용자가 게시글을 쉽게 공유할 수 있도록 하였습니다.

[에러 전역 관리]

- 에러 Context와 에러 Modal을 통해 **에러를 전역적으로 관리**하였습니다.
- 에러의 종류를 크게 2가지로 개발자의 실수로 인한 에러와 네트워크 오류로 인한 예측 불가한 에러가 있다고 생각하고 이를 **일관성 있게 처리하게 위해** 에러가 발생했을 때 useContext에 에러에 대한 상태 값을 전달하고, useContext와 연결된 에러 모달에 이를 표시하여 에러를 관리하였습니다.

[무한스크롤]

- 기존의 무한 스크롤은 DOM 요소가 증가함에 따라 렌더링 성능 저하가 발생하는 것을 확인했습니다. 이를 개선하기 위해 react-virtualized 라이브러리를 활용하여 가상화된 스크롤을 통해 **화면에 보이는** DOM **요소만 렌더링** 하는 방식으로 무한 스크롤을 구현하였고 메모리 사용량을 최소화할 수 있었습니다...
- **그 결과** React Developer Tools**를 통한 속도 측정에서 렌더링 시간이** 15ms**에서** 2.8ms**로 단축**되며 성능이 크게 향상되었습니다.

[협업]

- Jira를 활용해 이슈를 관리하고 업무 우선순위를 조정하여 우선순위를 명확하게 파악할 수 있도록 했습니다. 또한 Confluence를 통해 개발 규칙, 회의록, 공통 문서 등을 정리하여 팀원 간의 원활한 정보 공유와 업무 연속성을 유지할 수 있도록 했습니다.

Jaychis

기여 내용

[초기로딩속도 개선]

- lighthouse로 측정한 **성능 점수** 49**점 -> 92점으로 약** 88% **개선**
- LCP 점수 개선을 위해 메인 페이지에서 **사용되지 않는 컴포넌트들을** Reacy.lazy를 이용하여 **동적으로 로드**하였습니다. 이를 통해 **기존의 번들이 여러 개의 청크로** 나누어져 초기 로딩 시간이 단축되었습니다. 이후 LCP 점수를 추가로 개선하기 위해 tree shaking**를 이용하여 사용되지 않는 코드들을 정리**하여 1.6mb**였던 메인 번들의 크기를** 730kb**로 감소**시켰고 LCP **점수 또한 향상**되었습니다.

[2개의 서버를 통한 사용자 경험 개선]

- 로컬 환경에서는 괜찮았지만 **배포 이후 버그가 발생**하거나 예상과 다른 UI가 적용되는 경험을 하였습니다.
- 배포 환경과 로컬 환경의 차이로 배포 상태에서 예기치 못한 버그가 발생할 수 있다는 점을 인지하고 이를 방지하기 위해 **서버를** stage**와** prod**로 분리**하여 stage 서버에서 테스트를 거친 후 prod로 배포하여 더 **안정적인 사용자 경험을 제공**할 수 있도록 했습니다.

[이미지 최적화]

- 유저들이 업로드하는 다양한 이미지로 인해 **초기로딩속도가 저하**되는 문제가 발생하였습니다. 이를 해결하기 위해 **이미지를** s3**에 업로드하기 전** webp **형식으로 변환하는** 기능을 구현하여 이미지 용량을 평균 30% 감소시켜 성능 악화 방지와 비용 개선 효과까지 얻을수 있었습니다.

[커스텀 Hook]

- **반복적으로 사용되는 로직을 효율적으로 관리**하고 코드의 재사용성을 높이기 위해 커스텀 Hook을 개발하였습니다.

[테스트 코드 작성]

- 안정적인 코드 품질을 위해 Jest와 testing-library를 활용하여 단위 테스트 코드를 작성하였습니다. 이를 통해 코드 변경 시 발생할 수 있는 버그를 사전에 예방하고 유지 보수 효율성을 향상시켰습니다.

싹둑싹둑

판매자와 디자이너의 매칭을 통해 의류 구매 및 리폼을 동시에 진행 가능한 웹 플랫폼

개발 기간

- 24.02 ~ 24.06

개발 인원

- Front 2명, Back 2명

협업

- Git Hub
- Notion
- Swagger

기술 스택

- React
- tawind-css
- Zustand
- react-query

싹둑싹둑

기여 내용

[메인 페이지]

- React-Query를 이용하여 **반복적인 비동기 데이터 호출을 방지**하며 성능 최적화 및 리소스 낭비를 줄였습니다.
- 페이지 네이션을 구현하여 초기 데이터 로드량을 최소화했습니다

[처리속도 개선]

- 상품 상세 페이지의 좋아요 API 평균 응답 시간이 351ms로, **사용자 인터랙션 후** UI **반응이 지연되는 문제가 발생**했습니다. 이를 해결하기 위해 사용자가 **좋아요 버튼을 클릭할 때 서버 응답을 기다리지 않고 로컬 상태에서 즉시 좋아요 수를 증가 / 감소시켜** API **호출 수를 최적화하고 처리 속도를** 99% **향상시켰습니다.**

[아토믹 디자인 패턴]

- 컴포넌트 **재사용성을 높이기 위해 아토믹 디자인 패턴을 적용**했습니다. 하지만 일부 페이지에서는 아토믹 디자인과 상충되는 구조적 요구사항이 있어 이를 변형해 사용했습니다.
- 컴포넌트를 최대한 Atoms, Molecules, Organisms로 구분하였고 복잡한 UI를 가진 폼의 경우 단일 Template으로 통합 하였습니다.

[액세스 토큰 재발급]

- 보안을 위해 **엑세스 토큰의 시간을 짧게 설정**하였지만 이로 인해 네트워크 요청이 실패하며 발생하는 불편함이 있었습니다.
- 이를 개선하기 위해 **인터셉터를** 사용해 **액세스 토큰이 만료된 상태로** API **요청 시** 401 에러를 감지하여 리프레시 토큰을 이용해 새로운 액세스 **토큰을 발급받은 후 이전에 실패했던** API **요청을 동일한 조건으로 다시 시도**하는 기능을 개발하였습니다.

싹둑싹둑

기여 내용

[깃허브 액션을 통한 CI / CD 파이프라인]

- Aws를 통해 수동으로 배포를 하는것 보다는 **배포 자동화**를 통해 개발 **효율성**을 향상 시킬 필요가 있다고 생각 하였습니다.
- 이를 위해 GitHub Actions을 통한 CI/CD **파이프라인 구축**으로 **배포 자동화와 개발 효율성을 향상** 시켰습니다.

[전역 상태관리]

- 프로젝트의 규모가 크지 않아 복잡한 상태 관리나 다단계 상태 변화가 필요한 상황이 아니었기에 Redux나 Recoil보다는 **보일러 플레이트 코드가 적고 가벼운** Zustand**가 더 적합하다고 판단하였습니다**. 이를 통해 **전역 상태 관리를 보다 간결하게 처리**할 수 있었고, 필요한 상태를 효율적으로 관리하면서 성능을 최적화할 수 있었습니다.

끝까지 읽어주셔서 감사합니다.

포트폴리오 싸이월드 버전 보기

Email: wjdvlf99@naver.com

GitHub: https://github.com/jungpill

PhoneNumber: 010-5628-7623