PORTFOLIO

FrontEnd Develeoper
주다원

Email
hwondaa@gmail.com

Phone
010-6518-8253

Link
Github
Blog

Contents

Values

핵심 가치

" 우리가 좀 더 빨리 답을 찾도록 "

First Mover

누군가 해야 할 일이라면 망설임 없이 먼저 움직입니다.

막연한 고민보다 협업과 이슈 정리를 통해 빠른 프로토타이핑으로 실질적인 가능성을 증명합니다.

Interface Builder

기획과 개발 사이의 언어를 연결합니다.

추상적인 요구사항을 논리적인 컴포넌트 구조로 정리해 기획과 개발이 같은 방향을 바라보며 소통할 수 있도록 돕습니다.

Efficiency Lab

팀 전체가 더 잘 협업할 수 있는 환경을 만듭니다.

UI 라이브러리 구축과 협업 도구 정착을 통해 동료들이 더 나은 환경에서 본질적인 문제 해결에 집중할 수 있도록 기반을 설계합니다.

Key Achievements

핵심 성과

(주)한국해양기상기술 2023.01 - 현재(3년 3개월)

(주)한국해양기상기술은 기상청, 국립수산과학원 등 주요 공공기관을 대상으로
지도 기반의 해양·기상 데이터 시각화 및 연구 지원 플랫폼을 구축하는 해양·기상 IT 전문 기업입니다.

1. **개발 공정 효율화**
   - React + TailwindCSS 기반 **UI 라이브러리를 구축**해 개발 리소스를 최대 98% 절감
   - 온프레미스 환경에서 **LLM 기반 자동 코드 리뷰 시스템 도입**으로 동료 검토 시간 50% 절감
2. **서비스 현대화 주도**
   - 기상청 대국민 서비스의 **UI/UX 주도적 설계로 사용자 만족도 2배 증가**
   - 2,000라인 이상의 레거시 컴포넌트를 분리하여 유지보수 시간을 60분에서 5분으로 단축
3. **비즈니스 성과 달성**
   - 스마트 양식장의 복잡한 데이터 구조를 시각화하여 **데이터 분석 시간을 최대 95% 개선**
4. **협업 체계 구축 및 기술 표준 수립**
   - Git 브랜치 전략 및 Commit convention 수립
   - Jira/Figma 등 협업도구 사내 정착
   - Claude code, Cursor 등 도입 및 사내 교육

Work Experience

직무 경험

1. 기상청 포털 서비스
2. AI Code Reviewer
3. 스마트 양식장
4. @koast/ui
5. Open Source Contribute

Project
기상청 서비스

Role
기획 50% · UI/UX 60% · Frontend 80%

Skills
Vue3 · TypeScript · Figma · Jira

대국민 서비스의 신뢰를 회복한 경험

기상청 포털 서비스

문제 파악

단순 유지보수가 아니라, 신뢰를 잃기 직전의 상황이었습니다.

Client

기상청 주무관

"반복된 요구사항 적체와 구닥다리 UI가 불만족스러워요"

임팩트 있는 개선 필요

한국해양기상기술

팀원

"과거 프론트엔드 외주로 인해 여전히 코드 파악이 힘들고, 코드리뷰 시간이 너무 깁니다."

한국해양기상기술

CTO

"고객사 불만사항이 계속 쌓이니 이대로면 내년에 프로젝트를 잃을 수도 있습니다."

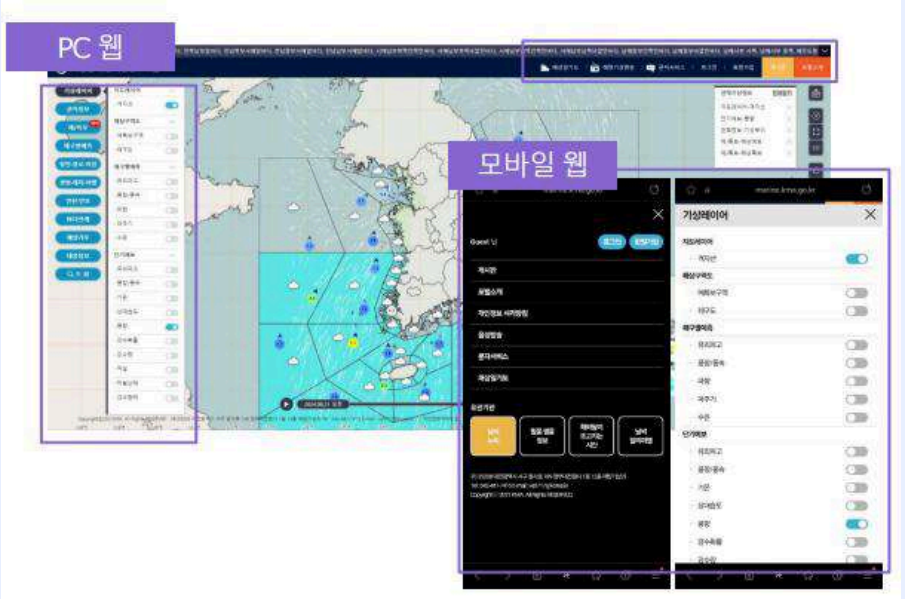
접근 방식

UI/UX 개선 TF팀을 만들어 메인페이지 개선을 주도했습니다.
대국민 조사로 사용자 불편을 확인하고, 기상청 실무진과의 커뮤니케이션을 통해 문제를 구조적으로 정의했습니다.

1. 사용자 정의

- 300여 명 대상의 대국민 설문 진행
- 사용성, UI/UX, 정보표출 속도 등 불만사항 접수
- **메인페이지 UI가 압도적으로 만족도 최저**

2. Pain Point 도출(AS-IS)

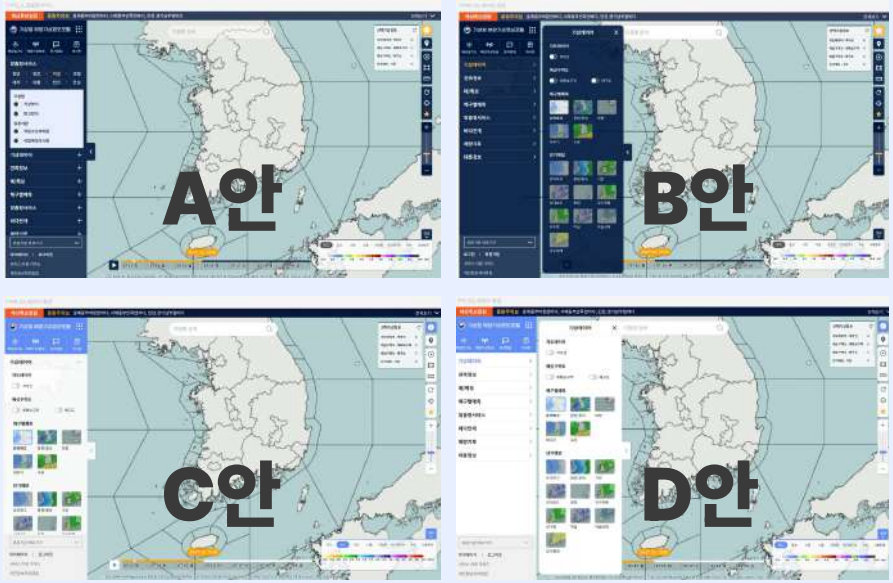


- 화면의 각 **섹션을 나누어 개선사항을 정리**
- PC/Mobile의 레이아웃과 톤앤매너 불일치
- 저대비 색상, 분산된 메뉴, 주요 기능 노출 실패

접근 방식

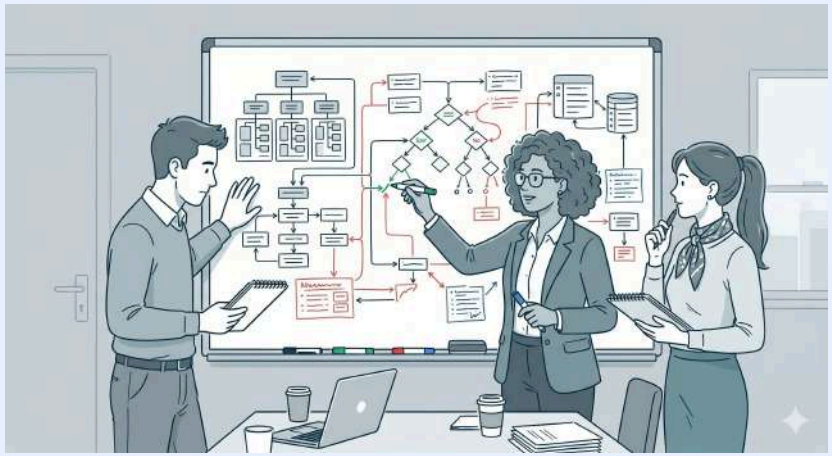
다른 프로젝트와 병행해야했기 때문에 시간을 많이 할애할 수 없었습니다.
빠른 시안 제작으로 화면을 정의하고, 코드 리팩토링 관련하여 애자일하게 회의를 진행하였습니다.

3. Figma 기반 빠른 시안 제작(TO-BE)



- 기상청 실무진과 직접 소통
- 디자이너와 함께 Figma 기반 시안 A~D 제작
- **회의 단계에서 빠른 의사결정 지원**

4. 구조 재설계



- UI/UX 개선과 병행할 리팩토링 논의
- 컴포넌트 구조 분리
- 상태 관리 단순화
- 타입 체계 구축
- 렌더링 성능 개선

UI/UX 개선

예쁘게만 만든 게 아니라, 사용자가 **덜 고민하게** 만들었습니다.

AS-IS

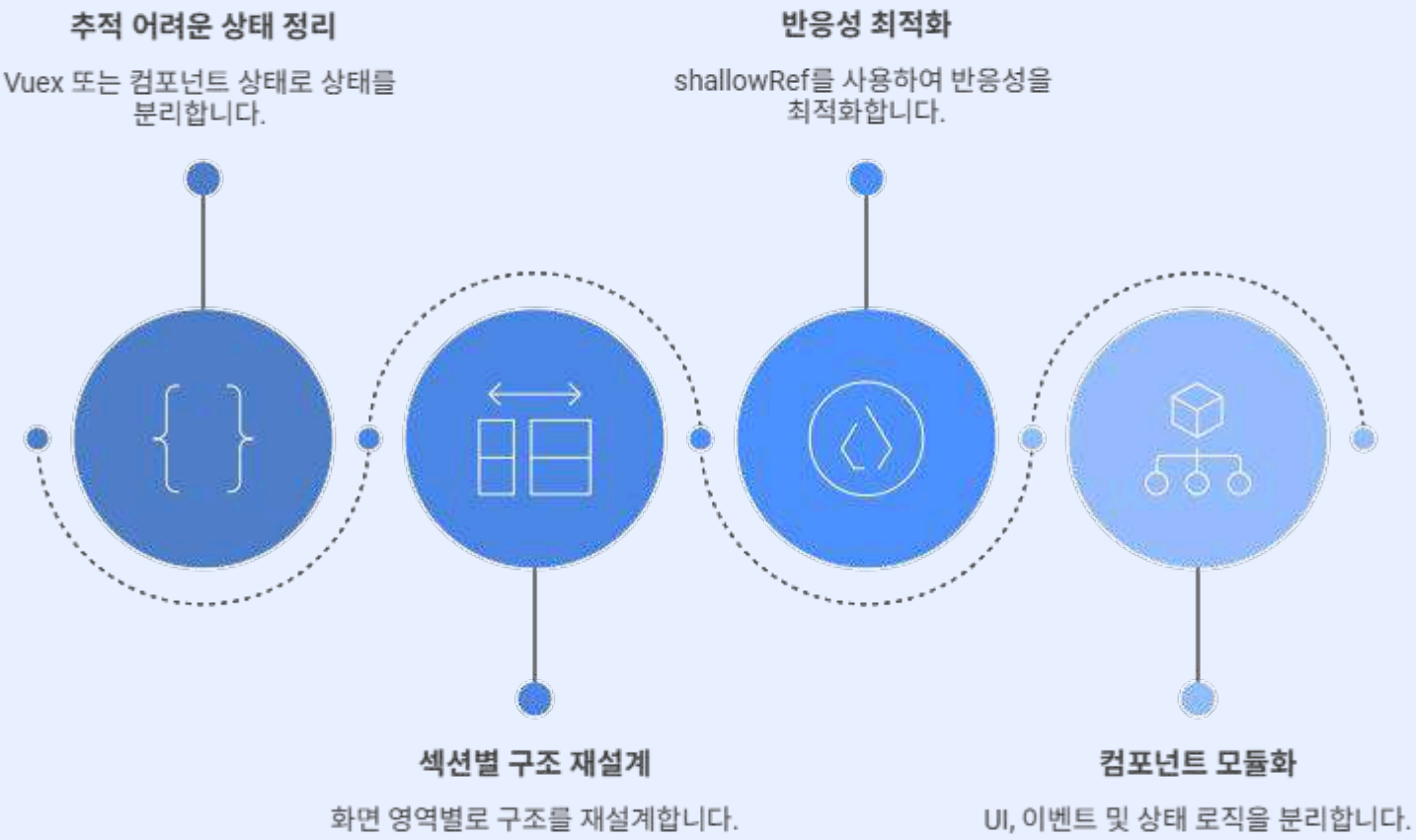
- PC/Mobile의 레이아웃 차이
- 저대비 색상으로 가시성 부족
- 메뉴가 분산되어 탐색 흐름 단절
- 주요 기능이 화면에서 드러나지 않음

TO-BE

- PC/Mobile 화면 레이아웃 및 톤앤매너 통일
- 높은 명암비로 가시성 문제 해결
- 좌측 네비게이션으로 탐색 구조 단일화
- 주요 기능(검색, 시간 슬라이더) 메인에서 바로 노출
- 텍스트 대신 시각적 단서(썸네일)로 이해 유도

기술적 해결

화면만 바꾸지 않았습니다. **상태와 구조를 추적 가능하게** 만들어 신규기능 추가 시간을 대폭 줄였습니다.



결과

측정 가능한 변화만 남겼습니다.

×2

사용자 만족도
672명 기준

4점 → 8점

4개월

개편 완료
PC·Mobile

메인페이지 UI/UX

100%

요구사항 충족
약 20개 Tasks

Jira를 통한 스케줄링

×2

평균 인터렉션 시간
배포 후 1개월 기준

3분 → 6분

기상청 프로젝트에서는
메인페이지 만족도만 올리는 것이 목표가 아니었습니다.

저대비 색상, 분산된 메뉴 구조, 기술 부채, 느린 리뷰 프로세스 등 여러 문제가 함께 얽혀 있었기 때문에
사용자 경험과 개발자 경험을 동시에 개선해야 했습니다.

그 결과 만족도와 체류시간 같은 사용자 지표뿐 아니라,
적체된 요구사항을 처리할 수 있는 개발 구조도 함께 회복할 수 있었습니다.

“ 코드리뷰만 3시간...
너무 힘들어요. ”

— 기상청 팀 동료의 한 마디

이 한마디에서 다음 프로젝트가 시작됐습니다.

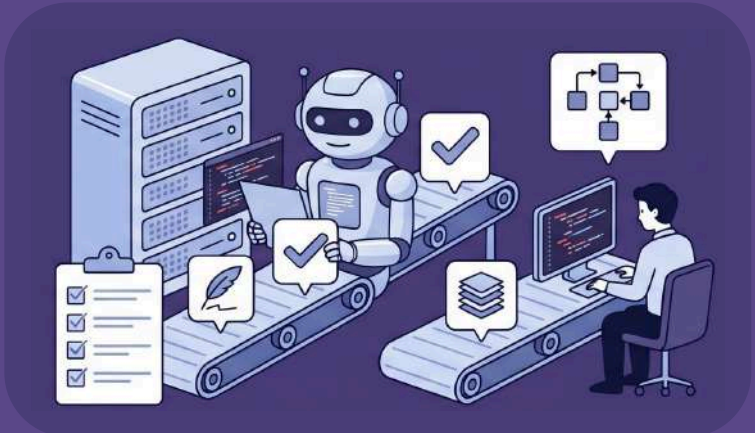
AI Code Reviewer

한 번 시작하면 끝을 모르고 진행되던 코드리뷰. 병목의 이유는 **<남이 짜던 코드 파악하기 />**였습니다.
로컬 LLM으로 **리뷰어가 변경 목적을 즉시 이해할 수 있는 요약/설명 생성 체계**와 **단순한 1차 리뷰 시스템을 자동화**하면 병목이 많이 해소될 것이라 생각했습니다.

Problem

Gitlab MR 리뷰 — 코드 파악만 30분+

보안 정책상 외부 LLM API 사용 불가
시니어가 오탈자 · 컨벤션까지 수동으로 확인



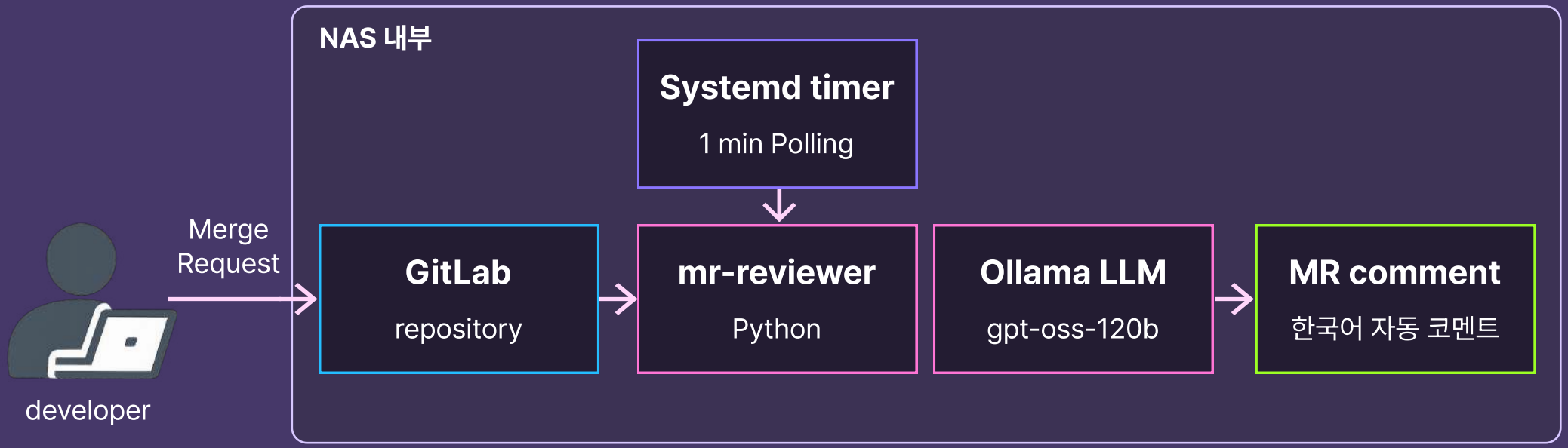
Hypothesis

1차 검수만 자동화해도 충분

정형화된 항목(오탈자 · 컨벤션 등) 은 LLM이 충분히 처리
시니어는 비즈니스 로직만 확인

How It Reviews

보안 제약 안에서 **가장 단순한 통합**을 골랐습니다.



Q. Why local LLM(Ollama)?

- 보안 정책상 코드 외부 전송 불가
- NAS 안에서 모델을 호스팅하는 게 유일한 옵션

Q. Why Systemd, not webhook?

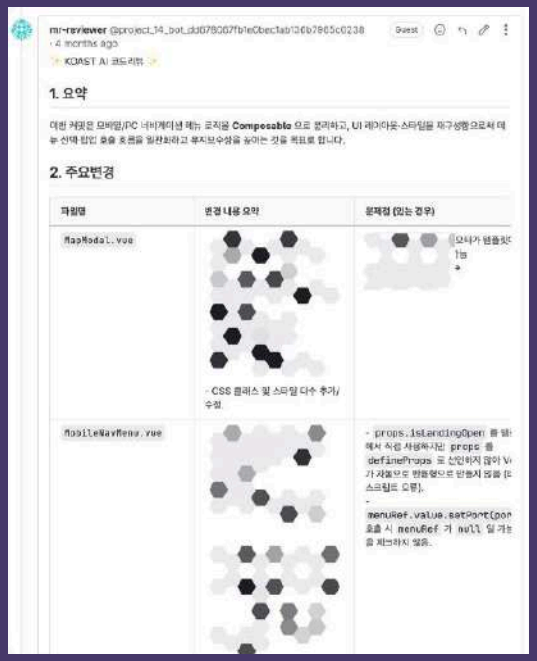
- Gitlab webhook은 외부 IP노출 필요
- 1분 간격 폴링이 보안 + 운영편의 양쪽에서 단순

Q. Why gpt-oss-120b?

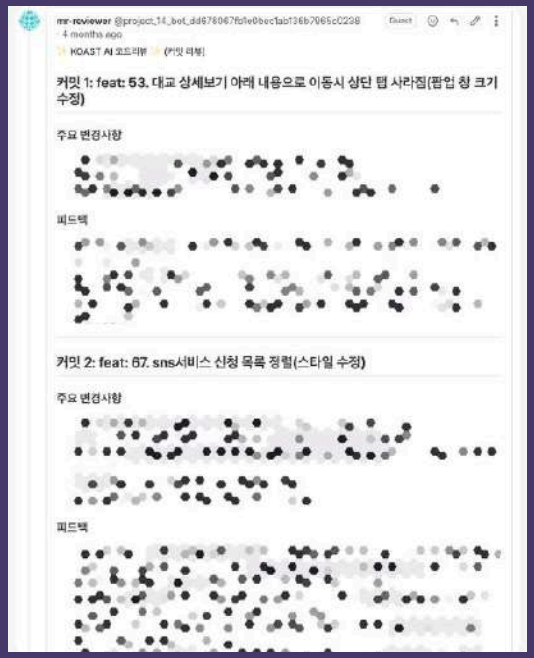
- NAS 리소스 한계의 균형
- 10개 이상 LLM 비교 후 한국어 코드 리뷰 품질이 가장 좋은 모델

Result

모든 MR에 대해 전체 리뷰와 커밋별 리뷰를 자동으로 생성합니다.
리뷰어는 LLM 1차 리뷰를 기반으로 검토를 진행하며, **코드 리뷰 시간이 약 50% 단축** 됐습니다.
현재 해당 방식은 **사내 표준 리뷰 프로세스**로 자리잡아 여러 프로젝트에 확대 적용되고 있습니다.



MR 전체 리뷰



MR 커밋별 리뷰

Review Time

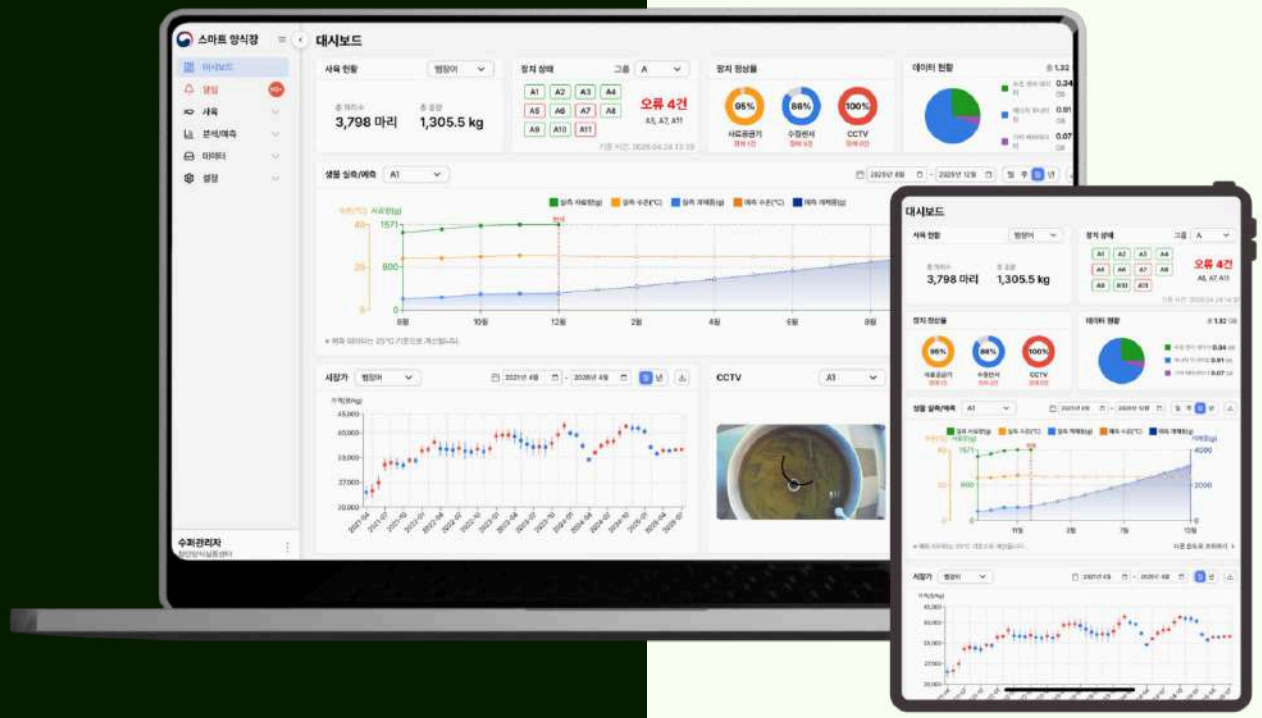
3 hr → 1.5 hr

효율 50% 향상 (기상청 팀원 5명 사용 기준)

회사가 시키지 않은 도구가 표준이 됐습니다.

✓ 기상청 해양정보기상포털	최초 도입
✓ 스마트양식장 플랫폼	확대 적용
••• 사내 신규 프로젝트	확대 적용 중

실제 업무 화면을 활용했으며, 보안상 세부 내용은 일부 비식별화했습니다.



기획부터 개발까지 E2E 리딩 프로젝트

스마트 양식장 · 연구데이터 분석 플랫폼

Role

기획 100% · UI/UX 100% · Frontend 100%

Skills

React · TypeScript · Vite · Redux · RTK-Query · Figma · Jira

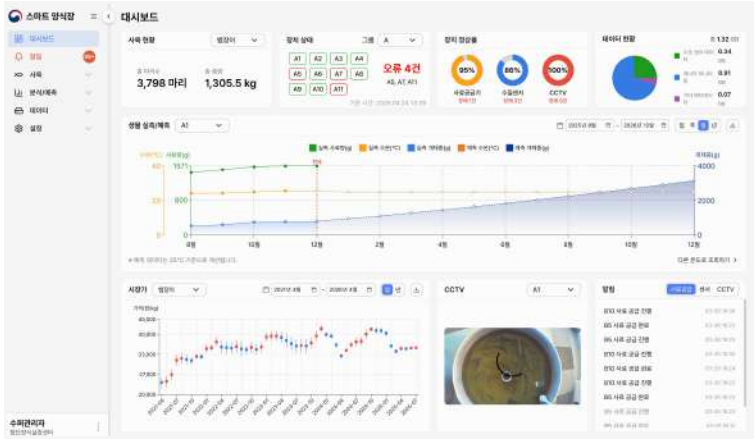
1. 개발 목표: 무엇을 만들어야 했는가

기존 기능과 신규 기능을 아우르는 연구데이터 분석 플랫폼

- 기존

- 지원 종료된 Vue.js 2로 제작
- 메뉴 구조 중복이 많고, 정보 구조가 불명확함
- 화면별 구현 방식이 제각각이라 확장이 어려움
- 톤앤매너 일관성 부족
- 불필요하거나 동작하지 않는 기능 다수

- 신규



- 기존 기능 35개 React 마이그레이션
- 신규 기능 25개 설계 및 구현
- 데이터 시각화 중심 기능
- 연구원이 직접 해석할 수 있는 분석 흐름 필요
- 단순 데이터 출력이 아니라 비교·탐색 가능한 UI 필요

→ 기존 구조를 정리하면서, 새로운 분석 기능까지 동시에 설계해야 했습니다.

2. 개발 전에 한 일: 요구사항을 먼저 구조화

고객사도 모르는 **'원하는 것'을 기다리지 않고, 먼저 보이게 만들어 합의**했습니다.

01 | 다양한 채널로 요구사항 수집

- 고객사 대상 **네이버 폼 조사**
- **화상회의**를 통한 맥락 확인
- **스프레드 시트** 기반 빠른 피드백 수집

02 | 기존 서비스 전체 화면 정리

- 기존 양식장 서비스의 전체 화면과 기능 흐름 인벤토리화
- **중복 메뉴, 단절된 흐름 및 불명확한 구조 확인**

03 | Miro 기반 CJM, Jira 기반 WBS 작성

- 연구진의 작업 흐름과 pain point 시각화
- **어떤 화면이 왜 필요한지** 공통 기준 정리
- 기준에 따른 기능 명세화

04 | Figma 시안으로 선택지 제시

- **추상적인 요구사항을 시안으로** 보여줌
- 방향성을 빨리 선택하게 함

실제 업무 화면을 활용했으며, 보안상 세부 내용은 일부 비식별화했습니다.

3. 개발 진행(1) 기술 선택

왜 이 스택을 골랐나

React + TailwindCSS + TypeScript

- 위 스택이 **회사 내 표준 Stack**으로 정해졌습니다.
- @koast/ui(사내 UI 라이브러리)를 위 스택으로 만들었고 바로 도입하기 위해서 사용했습니다.
- 빠른 통합과 톤앤매너 정리, 타입 명확화가 필요했습니다.

Redux

- 전역 상태를 명확하게 관리할 필요가 있었습니다.
- Context API는 간단한 전역 상태 공유에는 적합하지만, 상태 구조가 많아지고 비동기 데이터나 UI 상태와 함께 얽히면 복잡해집니다.
- Zustand는 가볍고 간결하지만, **상태 흐름을 명확히 보여주고 예측 가능한 구조**를 만들어야 해서 Redux Toolkit을 선택했습니다.

Vite

- 그래프, 레이아웃, 색상, 반응형 UI를 빠르고 반복적으로 확인해야 해서 **개발 서버 구동과 HMR이 빠른** Vite를 선택했습니다.
- 또한 빠른 개발과 가벼운 프로젝트 구성을 위해, 웹팩보다 간단한 설정이 마음에 들었습니다.

Rechart.js (그래프 라이브러리)

- React 컴포넌트 방식으로 차트를 구성할 수 있어 프로젝트 구조와 잘 맞았습니다.
- 레퍼런스를 참고해 **다중 그래프를 빠르게 구현**하기에 적합하다고 판단했습니다.
- highchart는 유료라, 무료 라이브러리가 필요 했습니다.

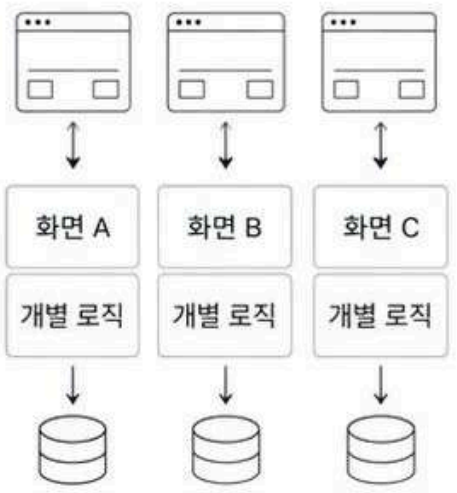
4. 개발 진행(2) 구조 재설계와 화면 체계화

기존 화면을 옮기는 데 그치지 않고, **새 기능이 계속 붙을 수 있는 구조**를 먼저 만들었습니다.

BEFORE

Vue2 + JavaScript

- 화면별 구현 방식 제각각
- 반복 화면 재사용 어려움
- 메뉴 구조 혼재
- 톤앤매너 불일치



AFTER

React + TypeScript + Tailwind

- 공통 Wrapper 구조로 반복화면 통일
- 공통 컴포넌트, 화면 컴포넌트 설계 분리
- 메뉴 구조 재정의로 탐색 구조와 톤앤매너 통일

01 구조 재설계

단순 이관이 아니라, 이후 기능 확장을 고려한 구조로 boiler plate 제작

02 UI 체계화

다크 테마와 Color set, 반응형 레이아웃까지 함께 정비

03 개발 생산성 향상

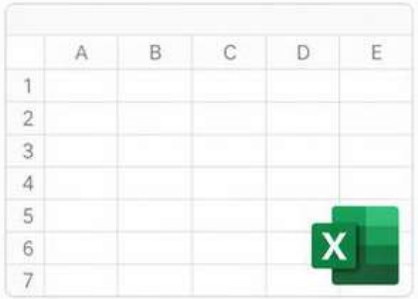
새 기능이 계속 추가될 수 있도록 기반을 확보

5. 개발 진행(3) 핵심 신규 기능 — 어류 성장 시뮬레이션

연구원이 가장 필요로 했던 기능은 '계산'이 아니라 **'변화'**를 보는 일이었습니다.

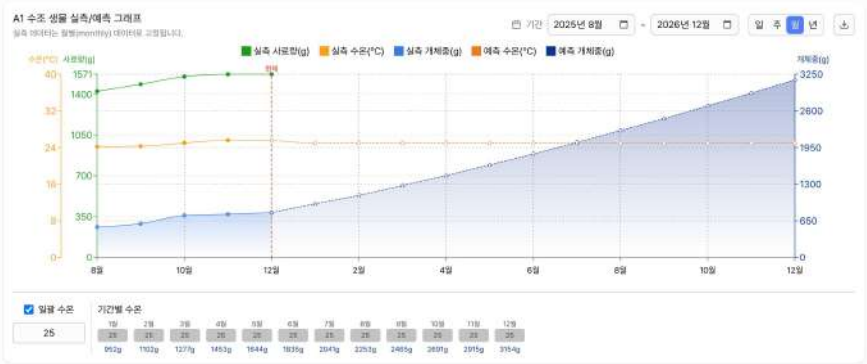
BEFORE · Excel 기반 분석

- 예측 수온에 따른 개체중 계산
- 설정값 변경이 어려움
- 시각화 및 그래프 비교 어려움
- 일/주/월/년 단위 파악 어려움



기존에는 엑셀로 예측 수온에 따른 개체중을 계산했습니다.
하지만 설정값을 바꾸며 흐름을 비교하기 어렵고,
시각화도 매번 엑셀 차트를 만들어야 했습니다.

AFTER · 어류 성장 시뮬레이션



- ✓ 실측 + 예측 데이터를 하나의 흐름으로 연결
- ✓ 예측 수온 입력 시 즉시 예측 개체중 계산
- ✓ 기간 단위 비교가 가능한 UI 설계
- ✓ 다크테마 적용으로 어두운 작업 환경에서 작업 피로도 감소

6. 결과: 무엇이 달라졌나

기존 양식장의 Pain Point & 신규 요구사항 100% 해결

주요 성과

40분 → 2.3분

어류 성장 데이터 평균 분석 시간

만족도 4.8점

실사용 연구진 3명 기준

100%

마이그레이션 + 신규 개발

6개월

기획에서 산출물까지

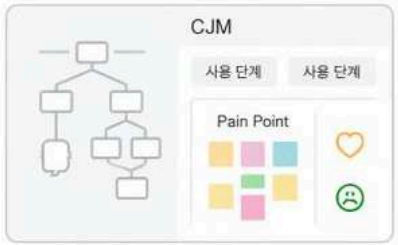
7. 인사이트

좋은 화면은 결과였고, 먼저 만든 것은 **의사결정 구조**였습니다.

1 수집을 먼저 했습니다.

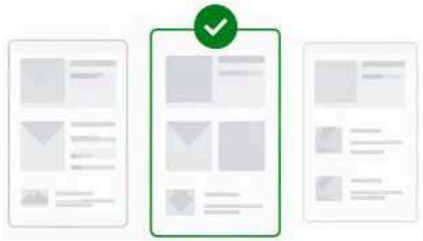
설문, 화상회의, Figma 시안 등
다양한 채널로 요구를 빠르게
수집했습니다.

2 공통 기준을 만들었습니다.



기존 화면과 흐름을 정리하고,
연구진의 작업흐름과 Pain Point를
기반으로 공통 기준을 만들었습니다.

3 선택지를 제시하고 정렬했습니다.



추상적인 요구를 시안으로 보여주고
빠르게 선택하게 하여, 개발 방향을
정확하게 정렬했습니다.

4 재작업 없이 빠르게 구현했습니다.

정리된 기준 위에서 개발했기 때문에
핵심 기능을 빠르게 구현하고,
사용자 만족도를 높일 수 있었습니다.

처음으로 맡은 PM 역할, 사용자 입장을 먼저 고려하였습니다.

실제 사용자인 연구진은 **자신들이 무엇을 원하는 지** 명확하지 않았습니다.
요구사항이 추상적일수록 바로 구현에 들어가는 대신 고객을 집요하게 파고들어,
실행 가능한 기준으로 먼저 정리하는 방식이 더 효과적이라 생각합니다.

이번 프로젝트는 제가 중요하게 생각해 온 방식이
실제 현장에서도 유효하다는 걸 보여준 사례였습니다.

> 원하는 그대로
> 100% 나왔습니다.

실사용 연구진 3명 피드백

@koast/ui

한국해양기상기술의 React UI 라이브러리

도메인 특화 컴포넌트로 **반복 UI 개발 시간을 최대 8h → 10min 으로 단축**하고,
비 FE 직군도 독립적으로 UI를 구현할 수 있는 표준 환경을 만들었습니다.

```
$ npm install @koast/ui
```

ROLE 1인 개발

STACK React · TS · Tailwind

Publish on NPM

Documented on Storybook

5 Components

1인 설계 · 배포

Why I Built It?

사내 UI 라이브러리를 만들기까지의 의사결정 과정

01 PROBLEM

스택 분산으로 프론트엔드 병목

JSP · Typmeleaf · Vue 등 사내 서비스가 이질적 스택으로 흩어져 있어 FE 의존도가 높음. 단순 UI 반복에도 공수가 많이 들어 납기 리스크가 누적됨.

02 HYPOTHESIS

표준 스택 + 공통 라이브러리

표준 스택을 정의하고, 도메인 특화 UI를 라이브러리로 분리하면, 비 FE 직군도 독립적으로 UI를 구현하고 FE 병목 현상도 해결될 것.

03 VALIDATION

CTO 협의 후 러닝커브 검증

CTO와 표준 스택을 React + TypeScript + Tailwind 로 합의. 러닝커브 대비 사내 React 스터디를 주최하며 도입 타당성 확인.

04 DECISION

@koast/ui 자체 라이브러리

MUI 등 외부 라이브러리로 커버 안 되는 도메인 특화 UI (TimeSlider, 범례)를 자체 라이브러리로 분리해 NPM에 배포.

What I Built?

도메인 특화 UI 라이브러리

AS-IS

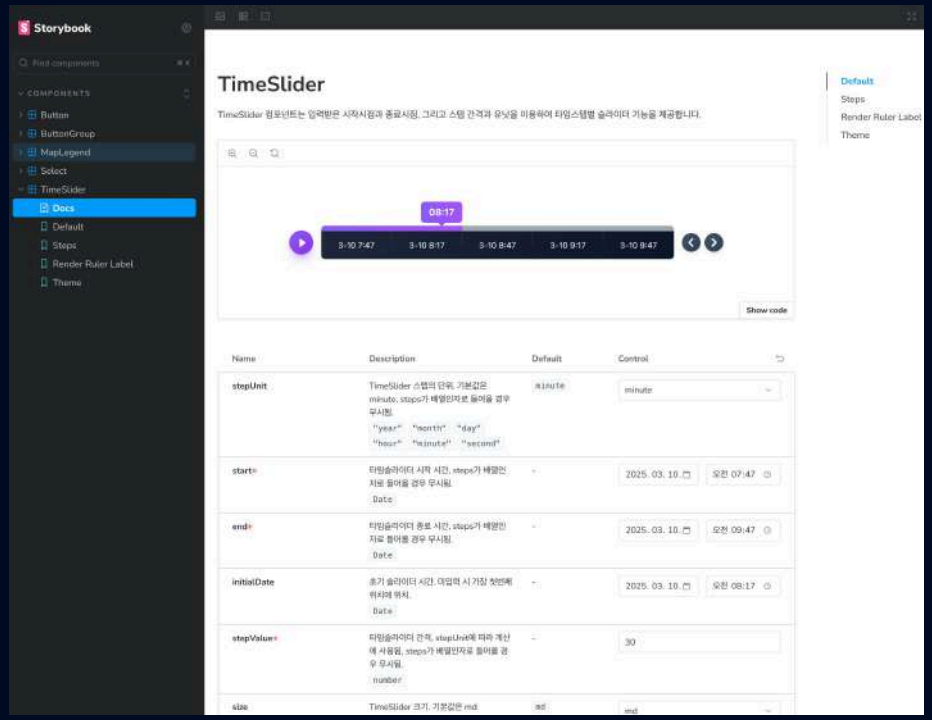
- 서로 다른 스택으로 제작되어 프로젝트마다 새로 UI를 제작해야 함
- 지도 기반 특화 UI 제작에 공수가 많이 소요됨
- 프로젝트마다 톤앤매너가 맞지 않음

TO-BE

- **동일한 스택으로 제작해 재사용 편리**
- **사내 프로젝트 컴포넌트 특성에 맞는 기능 구현**
- **프로젝트에 동일한 톤앤매너 제공**

What I Built?

Storybook Docs, Release Note



스토리북 문서화

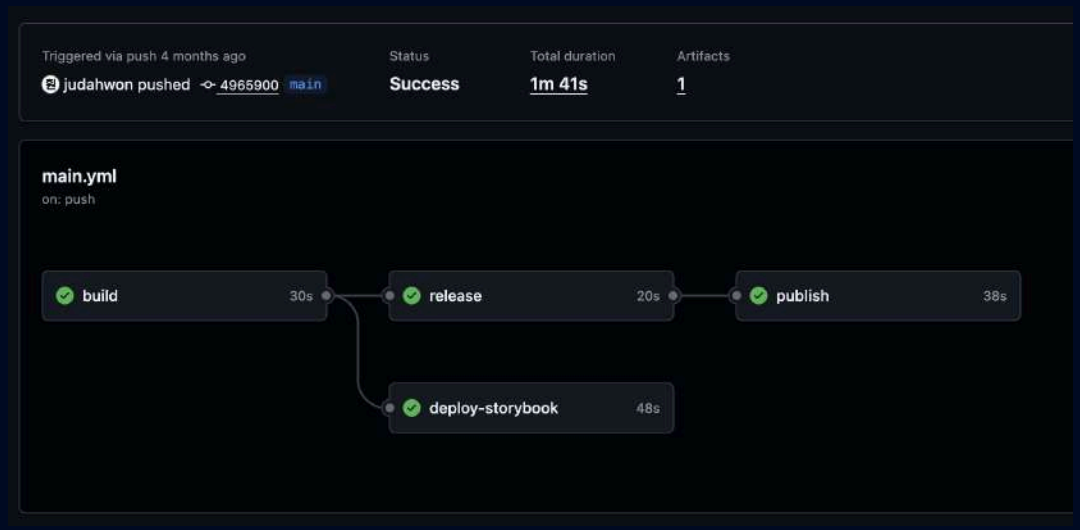
- 컴포넌트 사용법을 표준화
- 디자인/개발 해석 차이를 줄임
- 백엔드나 다른 팀도 FE 도움 없이 일정 수준의 UI를 만들 수 있음

릴리즈 노트

- 배포된 버전에서 무엇이 추가/수정/변경됐는지 팀이 공통으로 알고 안전하게 따라가게 하려는 운영 장치
- github actions로 changelog.md를 자동으로 파악하여 태그 브랜치 및 릴리즈 자동화

How It ship?

1인 라이브러리의 지속가능성, **자동화**



▶ **main.yml** 배포 자동화 github actions

build

type/lint 체크 + vite 빌드

release

버전 태그 + 체인지 로그

deploy-storybook

stroybook docs 배포

publish

NPM 배포

Q 왜 NPM을 선택했나요?

초기엔 사내 NAS에 Verdaccio 배포를 검토하고 테스트했습니다만, scope 분리의 까다로움과 npm 공개 배포 시의 CI 단순화 이점이 커서 선택 했습니다.

Q 왜 혼자 만들었나요?

당시 프론트엔드 인력이 사실상 저만 남아 있어, 기술 스택 통합과 공통 UI 기준 정립을 직접 주도했습니다. 처음에는 반복 구현이 많고 공수가 큰 UI부터 공통화해, 빠르게 검증 가능한 형태로 라이브러리를 만들었습니다.

Q 만들면서 가장 어려웠던 점은 무엇인가요?

어느 수준까지 공통화해야 실제로 도움이 되는지를 판단하는 일이었습니다. 너무 범용적으로 만들면 쓰기 어려워지고, 너무 특정 프로젝트에 맞추면 공통 라이브러리 의미가 없어지기 때문에, 반복 구현이 많고 재사용성이 분명한 부분부터 우선적으로 공통화했습니다.

ADOPTION & IMPACT

개인 사이드 프로젝트가 아니라, **사내 표준**으로

98%

리소스 절감

- TimeSlider 도입 사례 기준 **8시간 작업 → 10분 이내**로 단축
- 반복 UI 개발 비용을 라이브러리 한 번으로 영구 흡수

75%

프로젝트 도입률

- 2025년 신규 프로젝트 4개 중 3개에 적용
- 자체 라이브러리가 첫 도입 후 '자연스러운 사내 표준'으로 정착

5+

배포 컴포넌트

- 가장 많이 사용하는 공통 UI 설계
- 도메인 특화 2개(TimeSlider, MapLegend)
- 범용 3개(Button, ButtonGroup ,Select)

사용자 이슈 분석 및 버그 수정
그리고 핵심 로직 유닛 테스트 설계 및 작성

Open Source Contribute

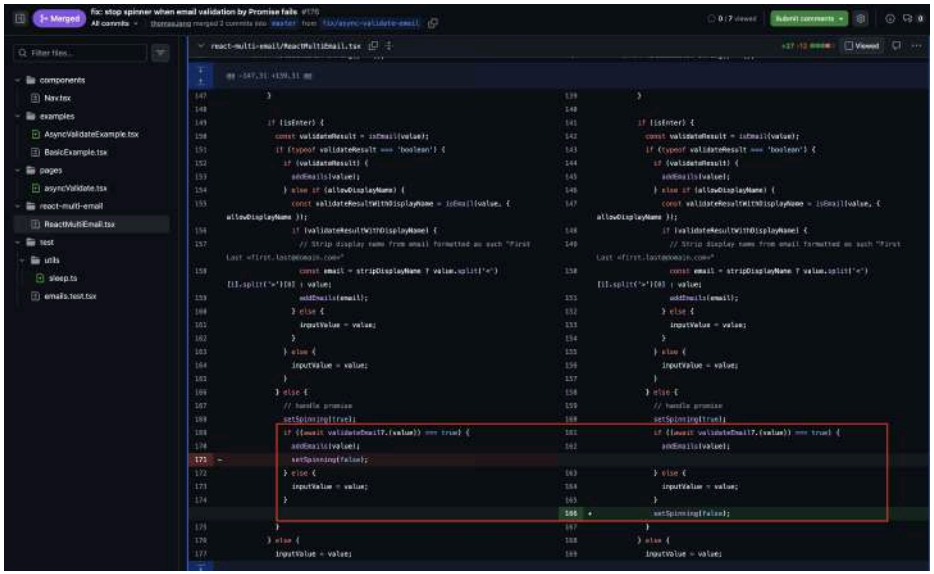
개요 react 기반 이메일 입력 오픈소스

skills React, TypeScript, Jest, Github Actions, NPM

Github stars 300+

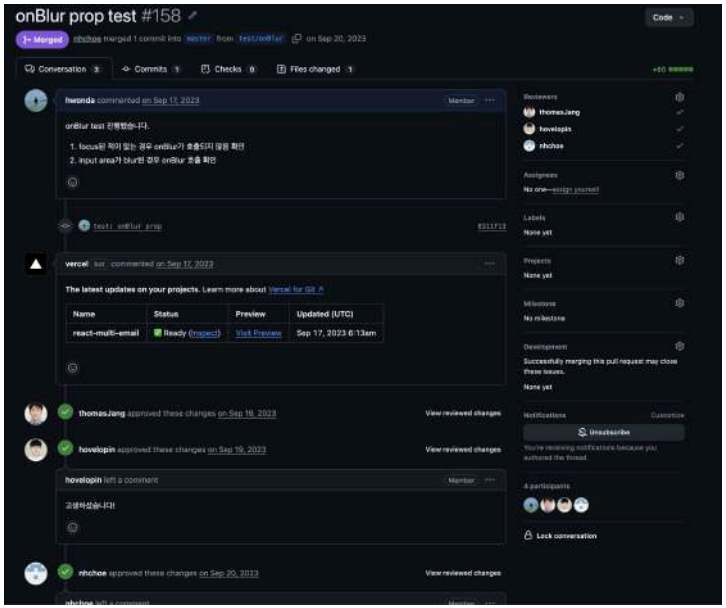
NPM weekly downloads 100,000+

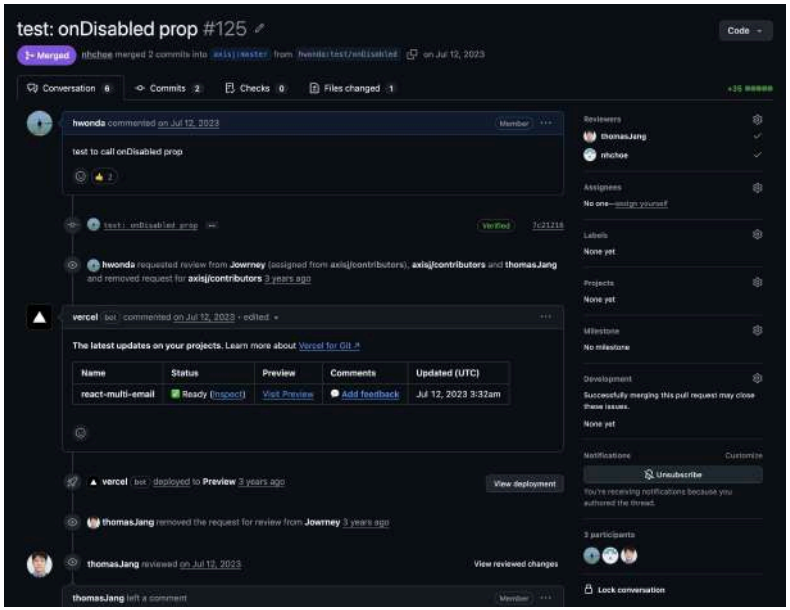
What I do



Bug Fix

- email 입력 후 엔터키 클릭 시 validateEmail이 false를 반환하거나 의도치 않은 오류가 발생하면 스피너가 종료되지 않는 버그 수정 및 PR Merged
- 수정 관련 내용 Test code 작성





Test Code

- 테스트 코드 부재로 런타임 에러 발생 가능성 존재
- onBlur, onDisabled 등 주요 시나리오 중심으로 Jest Unit Test 작성

주 다 훤

FrontEnd Develeoper
Portfoilo

Email
hwondaa@gmail.com

Phone
010-6518-8253

Link
Github
Blog