김귀영

FE (프론트엔드 개발자)

guiyoung22@naver.com

의미있는, 필요가 있는 개발자가 되겠습니다.

자기소개

"비즈니스 요구사항을 기술적 가치로 전환하는 개발자"

개발은 혼자가 아닌 팀이 함께 만드는 결과물임을 믿습니다. 기획과 디자인의 의도를 정확히 파악하고, 이를 사용자에게 가장 매끄러운 경험으로 전달하기 위해 기술적 해결책을 고민하는 과정에 즐거움을 느낍니다.

저는 단순히 기능을 빠르게 구현해내는 것에 그치지 않고, 작성하는 코드를 깊이 있게 생각하고 완벽히 이해하며 개발하는 것을 지향합니다.빠르게 변화하는 프론트엔드 생태계에서 특정 기술에 대한 고집보다는 유연한 사고를 바탕으로 프로젝트의 상황에 맞는 최선의 해결책을 찾아내겠습니다. 동료들과 활발히 소통하며 기술적 성과를 비즈니스의 성공으로 연결하는 프론트엔드 개발자가 되겠습니다.

기술 스택

JavaScript, React, HTML/CSS, TypeScript, Next.js, cursor

경력

아만타

개발팀 | 사원 | 정규직

2024. 02. ~ 2025. 09. (1년 8개월)

주요 업무로 **Vanilla JS와 TypeScript 기반의 AIDT(AI 디지털 교과서) 핵심 인터랙티브 콘텐츠와 교과서를 개발**했습니다. 프로젝트 진행 시 **교육부 가이드라인을 상세히 분석하여 공공 플랫폼에 최적화된 인터페이스를 설계**하고, 웹 접근성 인증 기준을 충족하는 표준화된 코드를 작성했습니다. 이에 더해 기존 시스템의 유지보수와 번들링 성능 최적화를 병행하며 서비스 안정성을 강화하였으며, 기획 및 영상팀 등 유관 부서와 직접 소통하며 다양한 디바이스에 최적화된 교육 서비스를 제작했습니다.

프로젝트

VibeBoard

개인

2025. 12. ~ 2026. 01.

프로젝트 설명 - 보드게임 추천·후기 작성·주변 보드게임 카페 검색을 제공하는 웹 서비스 (Supabase·카카오 API 연동, Vercel 배포)

기술 스택

React 19.2.0, TypeScript 5.9.3, React Query 5.90.12, Zustand 5.0.9, React Router DOM 7.11.0, Vite 7.2.4, Tailwind CSS 4.1.18, Supabase, 카카오 로컬 API, 카카오맵 JavaScript SDK, Vercel

---

기술적 특징 및 성과

-React Query (TanStack Query) 기반 서버 상태 관리

서버 상태와 클라이언트 상태를 명확히 분리해 Supabase 데이터 캐싱, 자동 리페칭, 리뷰 작성 후 invalidateQueries를 통한 목록 즉시 갱신을 구현했습니다. 또한 prefetchQuery를 React 마운트 전 실행해 컴포넌트 렌더 시점에 캐시에서 바로 데이터를 소비하도록 구성, LCP 요청 체인을 단축했습니다.

-Lighthouse 성능 점수 47 → 91점 개선

초기 성능 감사 결과 LCP 44.3s, CLS 0.362, 전체 payload 8.4MB라는 심각한 문제를 발견했습니다.

- LCP: Vite transformIndexHtml 플러그인을 직접 작성해 빌드 타임에 LCP 이미지 URL을 <link rel="preload">로 HTML에 자동 주입, JS 실행 전 브라우저가 이미지를 선제 요청하도록 구조 변경 (44.3s → 6.0s)
- CLS: 스켈레톤과 실제 컴포넌트 높이 불일치 원인 분석 후 레이아웃 예약 통일 (0.362 → 0)
- Payload: ttf 폰트(10MB+) → woff2 서브셋 전환, 이미지 WebP 변환, Vite manualChunks로 vendor 번들 분리 (8.4MB → 968KB)

- Zustand를 활용한 클라이언트 상태 관리

사용자 인증 정보(user, session, nickname)와 다크 모드 테마를 Zustand로 관리해 보일러플레이트 없이 간결한 전역 상태를 구현했습니다. initTheme()을 React 렌더 전에 실행해 다크 모드 초기화 시 발생하는 화면 깜빡임(FOUC)을 방지했습니다.

- Supabase BaaS 기반 풀스택 구현

별도 백엔드 없이 PostgreSQL DB, 이메일·Google·GitHub OAuth 인증, RLS(Row Level Security) 기반 권한 제어를 구현했습니다. 슬라이더 이미지 요청 시 필요한 컬럼만 select로 지정해 불필요한 데이터 전송을 줄이고, 슬라이드 순서를 고정해 prefetch 효과가 일관되게 적용되도록 설계했습니다.

- 카카오 API 연동 및 지도 서비스

카카오 로컬 API로 키워드·좌표 기반 보드게임 카페 검색, 카카오맵 SDK로 지도·커스텀 마커를 구현했습니다. SDK 스크립트를 동적으로 주입(useKakaoMapScript)해 지도 페이지 진입 시에만 로드되도록 구성, 초기 번들 부담을 줄였습니다.

- 환경별 설정 및 기능 제어

Vite 환경 변수(VITE_)로 API 키·Supabase URL 분리, 개발/배포 환경에 따른 리뷰 작성 허용 여부(VITE_ALLOW_REVIEW_CREATE) 제어로 배포 시 테스트 데이터 등록 방지

- 환경별 기능 제어 및 에러/로딩 UX

VITE_ALLOW_REVIEW_CREATE 환경 변수로 배포 환경에서 테스트 데이터 등록을 제어했습니다. 전역 ErrorBoundary, 스켈레톤 로딩 UI, NetworkStatus 컴포넌트로 네트워크 끊김 감지 및 재연결 시 쿼리 자동 무효화를 구현해 안정성과 사용자 경험을 강화했습니다.

-배포 및 운영 자동화

Vercel로 SPA 배포, GitHub Actions로 Supabase 주기 호출 워크플로 구성하여 무료 플랜 7일 비활성 일시정지 방지

자세한 상세 설명 및 안내는 깃헙에 있습니다 - https://github.com/guiyoung2/VibeBoard

리뷰 블로그 (LogOfReview)

개인

2025. 12. ~ 2025. 12.

프로젝트 설명 - 리뷰들을 작성하는 블로그

기술 스택

React 19.2.0, TypeScript, React Query 5.90.11, Zustand 5.0.9, Axios, Styled Components, React Router DOM

기술적 특징 및 성과

- React Query (TanStack Query) 기반 서버 상태 관리

서버 상태와 클라이언트 상태를 명확히 분리해 관리했습니다. queryKey에 카테고리·검색어·정렬 옵션을 모두 포함시켜 조건이 바뀔 때 자동으로 재요청이 발생하도록 설계했고, 댓글 작성/수정/삭제 후 invalidateQueries로 해당 리뷰

의 댓글 목록을 즉시 갱신했습니다. useMutation의 onSuccess/onError 콜백으로 성공·실패 처리를 선언적으로 구성해 각 컴포넌트의 비동기 로직을 일관성 있게 유지했습니다.

- Zustand + persist 미들웨어로 인증 상태 영속화

사용자 인증 정보(user, token, isLoggedIn)를 Zustand로 관리하고, persist 미들웨어를 적용해 새로고침 후에도 로그인 상태가 유지되도록 구현했습니다. Axios 요청 인터셉터에서 useUserStore.getState().token을 직접 참조해 모든 API 요청에 JWT 토큰을 자동 주입하고, 응답 인터셉터에서 401 에러 발생 시 logout()을 호출해 자동 로그아웃 처리를 한 곳에서 일괄 관리했습니다.

- 환경별 데이터 전략 분리

import.meta.env.PROD로 실행 환경을 감지해 개발 환경에서는 JSON Server Mock API, 배포 환경에서는 정적 JSON 파일을 사용하는 이중 전략을 구현했습니다. 쓰기 작업(작성/수정/삭제)은 배포 환경에서 명시적으로 차단해 데모 포트폴리오에 테스트 데이터가 누적되는 문제를 방지했습니다. Axios 인스턴스의 baseURL도 환경에 따라 자동 전환되어 별도 설정 변경 없이 배포·개발 환경이 각각 동작합니다.

- 렌더링 최적화 (useMemo, useCallback)

상세 페이지에서 댓글 작성자 정보 매핑(userId → nickname)을 useMemo로 캐싱해 댓글 목록이 바뀔 때마다 전체를 재계산하지 않도록 했습니다. 댓글 CRUD 핸들러와 이미지 네비게이션 함수를 useCallback으로 감싸 불필요한 자식 컴포넌트 리렌더링을 방지했습니다.

- 클라이언트 사이드 검색·필터·정렬

JSON Server가 복합 검색 쿼리를 지원하지 않는 한계를 파악하고, 제목·내용·태그를 동시에 검색하는 로직을 클라이언트에서 직접 구현했습니다. 카테고리 필터와 정렬(최신순/오래된순/별점순)을 URL 쿼리스트링으로 관리해 브라우저 뒤로가기 시에도 필터 상태가 유지되도록 설계했습니다.

자세한 상세 설명 및 안내는 깃헙에 있습니다 - https://github.com/guiyoung2/LogOfReview

emotionDiary

개인

2025. 11. ~ 2025. 11.

프로젝트 설명 - 투두리스트 형식처럼 하루 감정을 기록하는 프로젝트입니다

기술 스택

React 19.2.0, TypeScript, React Router DOM, Vite

기술적 특징 및 성과

- Context API + useReducer로 전역 상태 관리 및 localStorage 동기화

DiaryStateContext(읽기)와 DiaryDispatchContext(쓰기)를 분리해 상태를 필요로 하는 컴포넌트만 각각 구독하도록 설계했습니다. useReducer의 reducer 함수 내에서 CREATE/UPDATE/DELETE 액션 처리 후 localStorage.setItem을 호출해 상태 변경과 영속화를 한 곳에서 원자적으로 처리했습니다. 앱 초기화 시 localStorage 데이터를 파싱해 INIT 액션으로 복원하고, 저장된 id 최댓값을 useRef로 추적해 새로고침 후에도 id 중복이 발생하지 않도록 구현했습니다.

- TypeScript 타입 시스템 설계

ReducerAction을 Discriminated Union 타입(type: "INIT" | "CREATE" | "UPDATE" | "DELETE")으로 정의해 reducer 함수에서 각 액션의 data 타입을 컴파일 타임에 정확히 추론하도록 했습니다. DiaryStateContext, DiaryDispatchContext를 각각 DiaryItem[] | undefined, DiaryDispatch | undefined로 타입 지정해 Context를 Provider 외부에서 잘못 사용할 경우 런타임 이전에 감지하도록 설계했습니다.

- 커스텀 훅으로 관심사 분리

- useDiary(id): URL 파라미터로 받은 id에 해당하는 일기를 Context에서 조회하고, 존재하지 않을 경우 알림 후 홈으로 리다이렉트하는 가드 로직을 캡슐화했습니다. Diary, Edit 두 페이지에서 동일하게 재사용해 중복 코드를 제거했습니다.
- usePageTitle(title): document.title을 페이지별로 동적으로 변경하는 로직을 훅으로 분리해, 각 페이지 컴포넌트에서 한 줄 호출만으로 타이틀을 관리할 수 있도록 했습니다.

자세한 상세 설명 및 안내는 깃헙에 있습니다 - https://github.com/guiyoung2/emotionDiary

틱택토 (Tic-Tac-Toe)

개인

2025. 11. ~ 2025. 11.

프로젝트 설명 - 리액트 자습서 틱택토를 참조하여 다른 방식으로 구현

기술 스택

React 19.2.0, JavaScript (ES6+) , Vite

기술적 특징 및 성과

- 턴 기반 데이터 구조 설계

React 공식 튜토리얼의 1차원 배열 방식 대신, 각 턴을 { square: { i, j }, player } 객체로 저장하는 배열 구조로 설계했습니다. 이 구조 덕분에 보드 상태를 별도로 저장하지 않고 gameTurns를 순회해 2D 배열로 파생(derive)하는 방

식이 가능해졌고, checkWinner와 GameBoard 컴포넌트 모두 같은 턴 배열 하나를 소스로 사용해 상태 불일치가 발생하지 않습니다.

- 게임 로직 유틸 함수 분리

- checkWinner(gameTurns): 턴 배열에서 3×3 보드를 파생한 뒤 가로·세로·대각선 승리 조건을 순회 검사하고, 9턴 소진 시 무승부를 반환합니다. 상태와 완전히 분리된 순수 함수로 구현해 입력값만 있으면 단독 테스트가 가능합니다.
- getCurrentPlayer(turns): 턴 수의 홀짝(turns.length % 2)으로 현재 플레이어를 계산합니다. 별도 플레이어 상태를 두지 않고 단일 소스인 gameTurns에서 파생해 상태 동기화 문제를 원천 차단했습니다.

- 히스토리 분기 처리

history를 턴 스냅샷 배열로 관리하고 turnIndex로 현재 위치를 추적했습니다. 과거 턴에서 새로운 수를 두면 history.slice(0, turnIndex + 1)로 이후 분기를 제거한 뒤 새 스냅샷을 추가합니다. 이 방식으로 사용자가 특정 시점으로 점프(handleJumpTo)한 뒤 게임을 재개할 때 이전 분기가 자동 정리되어 항상 일관된 히스토리 상태를 유지합니다.

자세한 상세 설명 및 안내는 깃헙에 있습니다 - https://github.com/guiyoung2/tic-tac-toe

과거 포트폴리오 작업 1

개인

2023. 05. ~ 2023. 08.

과거 취업하기 전 작업했던 포트폴리오 1의 작업물입니다.

포트폴리오의 경로 - https://guiyoung2.github.io/portFolio/

포트폴리오의 작업물 코드 - https://github.com/guiyoung2/Project_group

과거 포트폴리오 작업 2

개인

2023. 09. ~ 2023. 12.

과거 취업하기 전 작업했던 포트폴리오 2의 작업물입니다.

포트폴리오의 경로 - https://guiyoung2.github.io/portFolio2/

포트폴리오의 작업물 코드 - https://github.com/guiyoung2/Project_group_2

포트폴리오

링크

github

교육

대림대학교

졸업 | 대학교(전문학사) | 전자과,경영학과

2017. 03. ~ 2022. 02.

대외활동

스마트웹 & 콘텐츠 개발

라인컴퓨터학원

2023

- HTML, CSS , Javascript, Jquery 등 기초 지식부터 학습.
- HTML5:미디어쿼리를 이용한 반응형웹 크로스 브라우징 구현가능
- CSS:CSS3로 다양한 효과와 움직임 구현 가능.
- Javascript:기본적인 슬라이드와 탭 안의 탭 구현 가능
- jQuery:코드의 작성과 수정, 플러그인 활용가능
- React.js : 기초적인 컴포넌트 작성 , State 관리 , Bootstrap 사용 가능