

박중섭

☎ 010-3160-6198 @crusub0306@gmail.com

React / TypeScript 기반 프론트엔드 개발과 실시간·스트리밍 처리 화면 구현에 집중해왔습니다.

- WebSocket, SSE, Fetch Stream으로 시뮬레이션 진행률, AI 챗봇 응답, LLM council 과정을 단계별 UI로 구현한 경험
- React Query, Zustand, Web Worker, virtual scroll로 긴 작업과 대용량 데이터를 화면 상태로 안정적으로 다룬 경험
- Astro, React, Firebase로 글 작성, 이미지 업로드, 발행 확인까지 처리하는 브라우저 CMS를 직접 구축한 경험
- Canvas API, requestAnimationFrame, Socket.IO로 실시간 게임 화면과 대전 이벤트 동기화를 구현한 경험

경력 신입

LLM Yatcha | 개인 프로젝트

2026.01 - 2026.01

"Andrej Karpathy의 LLM-Council을 fork해 provider 연동, SSE 진행 표시, 컨텍스트 압축, 합의 시각화를 구현한 다중 LLM 비교 도구"

1. 여러 LLM provider를 하나의 사용 흐름으로 통합

- 사용자가 보유한 API Key를 직접 등록해 원하는 모델 조합을 구성할 수 있게 수정하고, OpenAI, Gemini, Anthropic, xAI, Ollama를 `provider:model` 형태의 단일 인터페이스로 통합
- `asyncio.gather()` 기반 병렬 호출과 부분 실패 허용 구조로 한 모델 실패가 전체 응답 실패로 번지지 않게 처리

2. 긴 AI 처리 흐름을 단계별로 표시

- 3단계 council 진행 과정을 SSE로 노출해 사용자가 긴 응답 대기 시간을 진행 상황으로 이해할 수 있게 구성
- 각 Stage 완료 시 즉시 UI를 갱신하고, 불만족한 경우 직전 설정으로 재생성 가능하게 구현

3. 긴 대화를 위한 Context 압축 기능 구현

- 30K 토큰 임계치 기준으로 오래된 대화를 LLM 요약으로 압축하고, 최근 2개 Q&A는 원문 유지하여 맥락 보존
- 토큰 사용량과 압축 결과를 UI에서 바로 확인할 수 있도록 가시화

4. LLM 합의 수준 정량화 및 해석 보조

- 4개 LLM이 서로의 답변을 익명 평가한 결과를 바탕으로 합의 수준을 0~100점으로 산출
- High/Medium/Low 3단계 분류와 분기 분석으로 답변 신뢰도를 더 쉽게 판단할 수 있게 구성

기술 스택

Python, FastAPI, React, httpx(async), SSE

Stock Lab | 5인 팀프로젝트

2025.10 - 2025.11

"20~30초 이상 걸리는 시뮬레이션과 AI 챗봇 응답을 진행률, 스트리밍, 실패 복구 흐름으로 풀어낸 퀀트 시뮬레이션 플랫폼"

1. 긴 백테스트 실행 시간을 진행 상태 UI로 전환

- 서버에서 비동기로 실행되는 백테스트의 준비 단계와 진행률을 WebSocket으로 전달해, 사용자가 긴 계산 시간을 진행 흐름으로 인지할 수 있도록 개선
- 초기 Polling 방식의 반복 조회 한계를 바탕으로, 서버가 상태 변경 시점에 진행 데이터를 push하는 구조로 전환
- 초기 전체 상태로 화면을 동기화한 뒤 변경된 필드만 병합해, 누적 데이터 반복 전송을 줄임

2. Fetch API 기반 SSE 스트리밍 구현

- EventSource의 인증 헤더와 요청 제어 한계를 보완하기 위해 `Fetch + ReadableStream`으로 AI 챗봇 응답 스트림을 직접 처리
- `TextDecoder` 기반 SSE 파싱과 `AbortController`를 적용해 스트리밍 중단, 타임아웃, 재시도 흐름을 제어

3. Virtual Scroll + Web Worker 기반 대용량 데이터 최적화

- @tanstack/react-virtual 기반 가상화로 대량 row 렌더링 시 DOM 노드 수를 화면에 필요한 수준으로 제한
- Web Worker에서 실시간 tick 데이터를 100ms 단위로 배치 집계해 메인 스레드의 잦은 상태 갱신 부담을 완화

4. React Query 기반 즉각적 UI 반응과 실패 복구

- 삭제/추가 액션에서 서버 응답을 기다리지 않고 UI를 먼저 갱신하고, 실패 시 이전 캐시 스냅샷으로 롤백
- 백테스트 상태 조회 결과와 WebSocket 수신 데이터를 하나의 화면 상태로 합쳐, 완료/실패/진행 중 UI를 일관되게 처리

5. 실시간 연결 장애 대응 흐름 정리

- WebSocket/SSE 연결 실패 시 재시도 간격을 지속적으로 늘려 불필요한 재접속 부담을 줄이는 구조 적용
- 실시간 연결이 불안정한 상황에서도 Polling 기반 상태 조회로 화면을 계속 유지할 수 있도록 fallback 경로 설계
- 이후 Polling/SSE/WebSocket, 전체 상태/변경분 전송, reconnect/fallback 차이를 로컬 sandbox로 재현해 구현 의사결정을 문서화

기술 스택

Next.js, TypeScript, React Query, Zustand, TailwindCSS, Web Worker

Cruz Lab | 개인 프로젝트

2025.12 - 2026.01

"글 작성, 이미지 업로드, 즉시 발행까지 브라우저에서 처리하는 개인 콘텐츠 저작·운영 도구"

1. 브라우저 기반 콘텐츠 저작 흐름 구현

- 로컬 파일 수정과 수동 배포에 묶여 있던 운영 흐름을 줄이기 위해, markdown 편집기, 로컬 초안 저장, import/export를 갖춘 브라우저 CMS 구축
- 작성, 미리보기, 이미지 삽입, 발행까지 한 화면 흐름으로 묶어 콘텐츠 운영 병목을 줄임

2. Firestore 즉시 발행과 markdown 백업 분리

- 브라우저에서 작성한 글을 Firestore에 바로 반영하고, markdown은 GitHub 백업 경로로 분리해 즉시 반영과 기록 보존을 함께 처리
- Firebase Storage 업로드 결과를 본문 삽입 흐름과 연결해 이미지가 포함된 글도 브라우저 안에서 발행 가능하게 구성

3. 공개 페이지와 관리자 화면 책임 분리

- Astro SSG와 React 관리자 화면을 분리하고, 접근성 보완과 View Transitions 적용으로 읽는 흐름과 체감 완성도 개선

기술 스택: Astro, React, TypeScript, Firebase, CodeMirror 6, Framer Motion, TailwindCSS

Jungle Tetris | 3인 팀프로젝트

2025.07 - 2025.07

"Canvas와 Socket.IO로 3박 4일 안에 구현한 실시간 대전 테트리스 웹 게임"

1. DOM 렌더링을 Canvas 기반 게임 루프로 전환

- DOM 블록 렌더링을 Canvas로 전환하고, requestAnimationFrame 기반 루프로 화면 갱신을 제어
- 고정 타임스텝과 보간 전략으로 입력 처리와 렌더링 흐름을 분리

2. Socket.IO 기반 실시간 대전 이벤트 설계

- 전체 게임 상태가 아닌, 점수/방해 라인 등 필수 이벤트만 송수신하는 경량 프로토콜 설계
- 3박 4일 해커톤 기간 내 실시간 대전 시스템 완성

3. 솔로 리플레이 시스템 구현

- 입력 로그와 랜덤 시드를 저장해 동일한 플레이를 다시 재생하는 리플레이 페이지 구현
- 0.5x/1x/2x 배속 재생 기능으로 플레이 분석 지원

4. 클라이언트 예측 및 저지연 점수 동기화

- 입력 지연을 줄이기 위해 로컬 반영 후 서버 상태로 보정하는 흐름을 적용하고, 점수 동기화 경로를 다시 정리
- WebRTC DataChannel과 Socket.IO 폴백 구조를 적용해 저지연 전송과 안정성 사이의 균형을 시도

기술 스택

JavaScript, Canvas API, Socket.IO, WebRTC, Flask, MongoDB

학력



단국대학교

2014.03 - 2020.02 | 졸업 | 화학공학과

스킬

TypeScript

JavaScript

React

React Native

Next.js

Zustand

Python

Git

Figma

Web Socket



Krafton Jungle 10기 수료

2025.12 | 교육

- 5개월간 합숙 환경에서 주 100시간 이상 몰입하며, 어려운 문제를 끝까지 파고드는 습관을 체득
- C, 자료구조/알고리즘, 운영체제, 네트워크를 집중적으로 학습
- Pintos OS 프로젝트에서 스레드 스케줄링, 시스템 콜, 가상 메모리 구현 경험
- 웹 프록시 서버 구현으로 HTTP 요청 파싱, 캐싱, 연결 흐름을 학습하고 WebSocket/SSE/fallback 설계의 기초로 활용

링크

blog

<https://www.cruzlab.dev/>

github

<https://github.com/crucial-sub>

LLM Yatcha 프로젝트 기술블로그 정리

<https://www.cruzlab.dev/projects/llm-yatcha>

Stock Lab 프로젝트 기술블로그 정리

<https://www.cruzlab.dev/projects/stock-lab>

Cruz Lab 프로젝트 기술블로그 정리

<https://www.cruzlab.dev/projects/cruz-lab>

Jungle Tetris 프로젝트 기술블로그 정리

<https://www.cruzlab.dev/projects/jungle-tetris>