김성주 Jake Kim

가장 트렌디한 금융 소프트웨어를 만들고 있습니다.

Product Engineer · Crypto-native Builder · AI-native Developer

jiovana.jake@gmail.com
github.com/howdyfrom2019
linkedin.com/in/sungjoo-kim-jake/
x.com/b_cryptojake
t.me/b_cryptojake
경기 용인 · 원격 근무
업데이트 2026-09-12

크립토에서 시작해 dApp, 온체인 게임, 그리고 기관용 트레이딩 터미널까지 만들어온 AI-native 프로덕트 엔지니어입니다.

지금은 크립토 파생상품 멀티 거래소 터미널 OrderX를 만들고 있습니다. Claude와 Codex를 planner/reviewer로 활용하고, 직접 설계한 시스템을 실제 서비스까지 운영하며 작은 팀으로 큰 제품을 만드는 것을 좋아합니다.

경력

SunLabs AG · OrderX FE Part Lead 2025.10 – 현재

크립토 파생상품에서 출발해 장기적으로 기관용 트레이딩 데스크를 지향하는 터미널입니다. Deribit, Hyperliquid, Lighter, Bybit, Binance 같은 크립토 거래소와 OPRA(미국 옵션), CME, IBKR 같은 전통 시장을 한 화면에서 다룹니다. 대형 멀티스트랫 펀드와 파일럿을 진행했고, 프라이빗 베타 단계에서 명목 거래 $10B를 처리하며 $1M 매출을 냈습니다. 프론트엔드는 초기 세팅부터 1인 주도로 개발하고, 필요한 구간은 페어 프로그래밍으로 풀었습니다. 18개 피처 도메인을 설계했습니다.

성과

- 파일럿 + 프라이빗 베타 단계에서 명목 거래 $10B 처리, $1M 매출. 대형 멀티스트랫 펀드 파일럿 운영.
- 주문 실행과 멀티 거래소 지원을 프론트에서 통합해, 트레이더가 거래소를 오가지 않고 한 화면에서 여러 계정으로 주문.
- 브라우저 메모리 상시 점유 3.8GB → 1GB 내외. 하루 종일 켜 두는 트레이더 워크플로우에서 리로드 없이 운용.
- AI 워크플로우 정착: 단일 에이전트 작업에서 방치되던 엣지 케이스를 planner/reviewer 단계에서 잡아, 기능당 작업일 1~3일 이상 절감. 소수 인원으로 MVP 기능 플로우 전체를 완성. 다국어는 Figma → i18n 에이전트 파이프라인으로 자동화.

프로젝트

주문 실행과 멀티 거래소 지원 2025.12 – 2026.09

문제 · 트레이더는 Deribit, Hyperliquid, Bybit 같은 거래소마다 계정을 여러 개 갖고 있고, 같은 BTC 옵션이라도 거래소마다 종목 이름·소수점·증거금 방식이 다릅니다. 기존에는 거래소 화면을 오가며 따로 주문해야 했고, 실수하면 엉뚱한 계정이나 종목으로 돈이 나갔습니다.

- 하나의 주문 폼에서 여러 종목을 골라 한 번에 제출하는 배치 주문 구조. 종목마다 앵커 가격을 정해 미리보기에서 예상 체결가를 보여주고, 제출 전 검증을 통과해야 나가도록 설계.
- 종목·거래소·사용자 선택을 조합해 주문이 갈 계정을 자동으로 고르는 계정 해석 규칙을 만들어, 계정을 잘못 고르는 실수를 UI에서 차단.
- 거래소별 종목 표기(Deribit 'BTC-26DEC25-90000-C' vs OPRA 'IBIT_271217P00048000')를 문자열이 아닌 구조화된 필드(기초자산, 만기, 행사가, 콜/풋)로 다뤄 거래소가 늘어나도 주문 로직이 깨지지 않게 함.
- 이 위에 전문 트레이더용 주문 방식을 하나씩 올림: TP/SL 예약 주문, RFQ 견적 주문, Spreader 스프레드 주문, Arb 빌더, Hedge 시뮬레이션, TWAP 분할 주문, One-tap buy.
- AI 채팅으로 주문할 때 화면 상태를 AI에 넘기는 UIContext가 배치 주문과 TP/SL을 표현하지 못하는 문제를 발견하고, 개선안을 RFC 문서로 써서 백엔드 팀과 프로토콜을 바꿈. '@계정' 태그로 종목별 계정을 지정하는 방식도 제안.
- 여러 계정의 포지션을 한 표에 합칠 때 서버 메시지에 계정 정보가 없어 안전하게 구분할 수 없다는 점을 분석해, 백엔드 변경 요구사항으로 정리.

브라우저 메모리 최적화 (3.8GB → 1GB) 2026.04 – 2026.08

문제 · 실시간 스트림, 위젯 상태, TradingView iframe, 쿼리 캐시가 누적되며 크롬 메모리가 상시 3.8GB까지 올라가 장시간 사용 시 UI가 느려지고 탭이 죽었습니다.

- 오더북·포지션 테이블의 갱신 단위를 행/심볼 단위로 쪼개 불필요한 리렌더와 객체 생성을 제거.
- AG Grid를 자체 DataTable로 교체하고 행 메모이제이션, 가상화, 구독 해제 경로를 정리.
- 3단계 런타임 복구 시스템으로 백그라운드 탭 복귀 시 부분 재구독 → 캐시 무효화 → 리로드를 단계적으로 수행. 주문 입력 중에는 리로드 차단.

translate 기반 자체 그리드 시스템 2026.06 – 2026.09

문제 · react-grid-layout이 리사이즈 중 매 mousemove마다 레이아웃을 재계산하고 React 리렌더를 유발해 CPU를 잡아먹었습니다.

- dnd-kit + 명령형 DOM transform(translate) 업데이트로 드래그 중에는 React를 거치지 않도록 설계.
- grid ↔ pixel 변환을 순수 함수로 분리하고 테스트로 고정.
- 위젯 stretch, 공유 가능한 페이지 레이아웃(OG 이미지 자동 생성) 추가.

AI 워크플로우 도입 2026.01 – 2026.09

문제 · 소수 인원으로 18개 도메인의 MVP 기능을 동시에 밀어야 했습니다. 단일 모델·단일 에이전트로 만들면 당장은 돌아가도 엣지 케이스가 방치돼, 며칠에서 몇 주 뒤 같은 기능을 다시 열어봐야 하는 일이 반복됐습니다.

- 레포 컨벤션·아키텍처·도메인 규칙을 context injection으로 고정해 어떤 에이전트가 붙어도 같은 기준으로 작업하게 함.
- Claude(planner) – Codex(reviewer) 역할 분리로 설계 → 구현 → 리뷰를 나눔. Plan/Design 문서와 Gap 분석을 남겨 검증.
- react-doctor 기반 클린업을 정기 루틴으로 돌려 기능 통합 시 누적되는 부채를 제거.
- 결과: 속도보다 커버리지. 리뷰 단계에서 잡힌 엣지 케이스 덕분에 기능당 작업일 기준 1~3일 이상의 재작업이 사라짐.

Figma MCP + i18n 에이전트 파이프라인 2026.06 – 2026.09

문제 · 디자인 변경과 다국어 문구 반영이 수작업이라 릴리스마다 누락과 불일치가 생겼습니다.

- Figma MCP로 디자인 소스를 직접 읽어 컴포넌트 변경을 반영.
- i18n 에이전트가 누락 키를 찾아 번역·등록하고 Lokalise와 동기화. next-intl 기반 다국어 서비스 운영.

차트 라이브러리 마이그레이션 (lightweight-charts → TradingView) 2026.03 – 2026.09

문제 · 백엔드가 표준 규격이 아닌 Protobuf over HTTP + WebSocket이라 TradingView의 기본 Datafeed를 쓸 수 없었습니다.

- Provider 패턴의 커스텀 Datafeed 작성. 포지션/체결 오버레이, 실시간 호가 push, 계정별 오버레이.
- 커스텀 인디케이터(OI 프로파일, visible range 프로파일, 크로스 거래소 스터디)와 범위 프리셋 툴바. FedWatch, 경제 캘린더, SVI 파라미터 같은 분석 위젯.

Frontend: Next.js 15, React 19, TypeScript, Bun, Tailwind v4, Radix UI, Jotai, TanStack Query, React Hook Form, Zod, dnd-kit, framer-motion, react-virtuoso · Realtime: Protocol Buffers (ts-proto) over HTTP + WebSocket · Chart: TradingView Charting Library v31, TitanCharts, ECharts · AI: Vercel AI SDK, assistant-ui, ElevenLabs · Tooling: Playwright, Bun test, Lokalise, PostHog, OpenTelemetry, AWS Amplify

YGG · Waifu Sweeper Frontend & Contract Engineer 2025.08 – 2026.07

Yield Guild Games(YGG) 산하 Waifu Sweeper 팀. 프로젝트 초기명은 Princess Sweeper, 이후 Waifu Sweeper / Playpsweeper로 리브랜딩했습니다. 웹 클라이언트와 스마트 컨트랙트 전체를 혼자 맡았고, 기획 변경(보상 배율, 시즌 리더보드, Gold Mine 모드, revive 모드)을 매주 단위로 배포했습니다. 서비스는 종료됐으며, 링크는 최초 프론트엔드 코어 로직 기반의 빌드입니다.

성과

- 결제 컨트랙트 19,939건 무실패 처리. 합계 약 $1.5M 상당: USDC 72,500 · USDT 25,000 · ETH 104.65 · YGG 50,616,446. 표시 금액은 2026.09.12 시세 환산이며, 마우스를 올리면 현재 시세로 다시 계산합니다.
- 외부 감사(2025.12) 반영 후 Abstract 메인넷 배포. 94개 Hardhat 테스트로 구매·오라클·NFT·세션 트래커 시나리오 검증.
- 업그레이더블 프록시 운영으로 보상 배율, 시즌 리더보드, Gold Mine, revive 모드 같은 토큰 경제 변경을 재배포 없이 반영. 재구매·재방문율 개선에 기여.
- 웹 클라이언트와 컨트랙트 전체를 단독 개발·운영. Cloudflare Turnstile로 봇 트래픽 차단.

프로젝트

운영 방법론: 업그레이더블 프록시와 context injection 2025.09 – 2026.07

문제 · 토큰 경제(보상 배율, 재구매 유도, 시즌 이벤트)를 매주 바꿔야 했지만, 결제 컨트랙트를 건드릴 때마다 사용자 자금 리스크가 생겼습니다.

- ShopManager·NFT·PriceFeed를 UUPS 프록시로 두고 storage layout 검토를 배포 체크리스트로 고정. 새 기능(세션 트래커)은 프록시에 슬롯을 추가하지 않고 별도 컨트랙트로 분리.
- 게임 규칙·엔티티·보상 로직을 context injection으로 명세화해 인게임 로직 변경을 빠르고 일관되게 구현.
- 결과: 19,939건 결제 무실패, 기획 변경 주 단위 배포.

AGW 세션 키 기반 게임 세션 2026.05 – 2026.06

문제 · 타일 오픈, 몬스터 처치, 아이템 획득, 부활 같은 반복 액션을 매번 지갑 서명 없이 온체인에 기록해야 했습니다.

- 24시간 세션 키(수수료 한도, 컨트랙트·셀렉터 단위 call policy)를 생성·로컬 저장·복구하는 훅 설계.
- WaifuGameSessionTracker 컨트랙트에서 플레이어별 세션 ID와 액션 nonce로 순서 검증.
- 이미 배포된 ShopManager 프록시의 storage layout을 건드리지 않도록 tracker 주소를 인자로 넘기는 설계를 택하고 문서화.

NFT 가챠박스 · 에너지팩 상점 2025.09 – 2025.12

- ShopManager (UUPS): ETH/USDC/USDT/YGG 다중 결제, 토큰별 decimals 명시 관리, EIP-712 서명 검증 + nonce 리플레이 방지, 슬리피지 처리.
- PriceFeed (UUPS): Pyth ETH/USD 피드 통합, 신뢰도 검증, 센트 단위 팩 가격 관리.
- WaifuSweeperNFT (UUPS ERC721): 최대 30개 배치 민팅, gas limit 회피용 페이지네이션 tokensOfOwner.
- 프론트: Pyth 가격 업데이트 데이터를 트랜잭션에 동봉, NFT 결제 수단, YGG 할인.

라이브 운영 도구와 시즌 시스템 2026.01 – 2026.07

- 시즌 리더보드, Gold Mine 리더보드, 보상 대시보드, revive 모드, 레퍼럴 코드 검증.
- 점검 모드 페이지 + 서비스 스케줄 설정, 기능별 feature flag / 날짜 플래그.
- GA4 이벤트 설계, Twitter Pixel, SEO/OG 최적화, BGM/효과음 시스템, 이미지 프리로드.

Frontend: React 19, TypeScript, Vite 7, Tailwind v4, Radix UI, Jotai, TanStack Query, Framer Motion, react-router 7 · Web3: Wagmi, Viem, Ethers v6, RainbowKit, Abstract Global Wallet (sessions) · Contracts: Solidity, Hardhat, OpenZeppelin Upgradeable v5, Pyth SDK, TypeChain · Infra: Vercel, Cloudflare Turnstile, GA4

111퍼센트 · 곰블 Frontend Engineer 2025.04 –

국내 게임 스튜디오 111퍼센트의 Web3 사업부 곰블 프로젝트의 웹/앱 플랫폼을 개발했습니다.

성과

- 하루 1만 건 이상의 스파이크 트래픽이 몰리는 TGE 상황에서 안정적인 지갑 연결과 트랜잭션 전송을 보장하는 dApp 설계.

프로젝트

광고 보상 리워드앱

- 모바일 네이티브 앱(React Native, Expo)과 PWA 개발.

NFT 커뮤니티 웹사이트

- NFT 인증·스테이킹, NFT 게임 제작 의뢰 프론트엔드.

베팅 기반 블록체인 게임 컨트랙트

- Proxy 패턴을 적용한 업그레이드 가능한 스마트 컨트랙트.

$GM 토큰 클레임 웹사이트

- Binance Alpha, 코인원, Bitget 상장 토큰 클레임 사이트.

Frontend: React, TypeScript, Vite, Bun, Vite-pwa, React Native, Expo · Web3: Viem, Ethers, Wagmi, WalletConnect, Hardhat, Solidity · Infra: AWS S3 + Route53, Vercel

지엑스씨 · xCBT Blockchain Engineer 2024.10 – 2025.04

한 달마다 출시되는 블록체인 게임을 미리 해보고 경쟁적인 보상을 받는 플랫폼 xCBT의 스마트 컨트랙트 개발·검증과 프론트엔드 상호작용을 담당했습니다.

성과

- 10K+ USDT 이상 수익을 낸 SNS 인증 미션 플랫폼 기능 개발.

프로젝트

SNS 인증 미션 플랫폼

- 미션 기능 개발.
- NFT 관련 스마트 컨트랙트 작성과 업데이트.
- 클라이언트·백오피스 통합 모노레포(pnpm, Turborepo).

Frontend: TypeScript, Next.js, Tailwind CSS, Framer Motion · Web3: Solidity, Hardhat, Viem, Thirdweb, Abstract Global Wallet

캣제랩스 · Yooldo Frontend Engineer 2023.04 – 2024.10

Web2 게임의 Web3 전환을 돕는 블록체인 게이밍 플랫폼 Yooldo와 게임 온보딩을 지원했습니다.

성과

- 블록체인 게이밍 플랫폼에 90만 명 이상의 유저 온보딩.
- NFT 구매·검증 UX 설계 참여 등으로 월 최대 15만 달러 이상의 프로젝트 수익화.
- MetaMask를 운영하는 Consensys로부터 투자 유치에 기여.
- 한화 3억 원 이상 수익을 낸 8종 이상의 자체 NFT 민팅 페이지 구축.

프로젝트

Yooldo 플랫폼 프론트엔드

- 플랫폼 개발과 운영, 온체인 이벤트 대응.
- 트랜잭션 영수증 분석을 통한 게임 재화 획득 검증 로직.
- 백오피스 기능 개발, 공통 UI 컴포넌트 npm 라이브러리 출시 및 관리.

Frontend: TypeScript, Next.js, TanStack Query, Framer Motion, Storybook · Backend: Node.js, Nest.js, Prisma · Web3: Viem, Wagmi · Infra: Vercel, GitHub Actions

블루포인트파트너스 Frontend Engineer 2021.07 – 2022.03

시드·pre-A 단계 테크 스타트업 전문 액셀러레이터의 인하우스 투자 관리 툴을 개발했습니다.

성과

- 3일 이상 소요되던 투자계약 준비 운영 작업 자동화.
- 팀마다 분산 관리되던 투자 포트폴리오를 인하우스 툴로 통합.

프로젝트

비상장 스타트업 투자 관리 툴

- 투자 심사역 사용자 인터뷰를 통한 스타트업 발굴 UX 개선.
- recharts 기반 시계열·전환율 시각화.

Frontend: TypeScript, React, Jest, recharts, styled-components · Infra: AWS S3 + Route53

테이스팅벤처 · FOUND Founder 2018.12 – 2019.06

GPS 기반 분실물 습득/보상 모바일 플랫폼 FOUND를 개발하고 용인·경기 지역에서 운영했습니다.

성과

- 플레이스토어 5천 다운로드.

프로젝트

FOUND

- 습득물 리포트 작성 화면 흐름(Android, Java).
- 카카오톡 챗봇 분실물 신고 + Flask/MongoDB 챗봇 서버.

Android: Java, Retrofit2, Glide · Backend: Python, Flask, MongoDB

학력 · 자격

단국대학교 전자전기공학부 학사 · 3.92/4.5 2016–2023
정보처리기사 한국산업인력공단 2020
특허 · 버스 요금 결제 방법 및 이를 수행하는 버스 요금 결제 서버 특허청 2020
창업 아이디어 해커톤 대상 경기창조경제혁신센터 2018
Alchemy University Ethereum Bootcamp 수료 Alchemy 2024

용어 설명

파생상품
기초자산(예: BTC) 가격에서 값이 파생되는 계약. 옵션, 선물, 무기한 선물(perpetual)이 대표적이며 현물보다 레버리지와 리스크가 큽니다.

멀티스트랫 펀드
여러 전략(옵션, 차익거래, 헤지 등)을 동시에 운용하는 헤지펀드. OrderX 파일럿 고객 유형입니다.

멀티 거래소
Deribit, Hyperliquid, Lighter, Bybit, Binance 같은 크립토 거래소와 OPRA(미국 옵션), CME, IBKR 같은 전통 시장을 한 화면에서 다루는 것. 거래소마다 종목 표기, 소수점, 마진 방식이 달라 통합이 어렵습니다.

배치 주문
여러 종목 주문을 하나의 묶음으로 동시에 제출하는 것. 예를 들어 콜 옵션 3개를 한 번에 사는 스프레드 전략에 필요합니다.

계정 해석
사용자가 같은 거래소에 여러 계정을 가질 때, 이 주문을 어느 계정으로 보낼지 결정하는 규칙. 종목·거래소·사용자 선택을 조합해 자동으로 고릅니다.

RFQ
Request for Quote. 큰 물량을 오더북에 바로 던지지 않고 마켓메이커들에게 '이 조건에 가격 줘'라고 견적을 요청해 최적가로 체결하는 방식. 기관 거래에서 흔합니다.

트레이딩 데스크
기관에서 여러 거래소·상품의 주문을 한곳에서 집행하고 리스크를 관리하는 팀과 그 도구. OrderX는 이 업무를 소프트웨어로 만드는 것을 목표로 합니다.

명목 거래대금
거래된 계약의 액면 총액. 실제 오간 돈이 아니라 '얼마짜리 포지션이 체결됐는가'를 뜻합니다. $10B notional은 100억 달러어치 계약이 터미널을 거쳤다는 의미입니다.

주문 폼
수량, 가격, 주문 종류(지정가·시장가), 조건을 입력해 거래소로 주문을 보내는 화면. 실수하면 바로 돈이 나가는 곳이라 상태 관리와 검증이 가장 중요합니다.

앵커 가격
배치 주문에서 각 종목의 기본 가격을 어디서 가져올지(현재가, 호가 중간값, 사용자가 찍은 호가)를 정하는 기준. 주문 미리보기에 표시되는 예상 체결가의 근거입니다.

TP/SL
Take Profit / Stop Loss. 목표가에 도달하면 이익을 확정하고, 손실 한도에 닿으면 자동으로 청산하는 예약 주문입니다.

Spreader
두 개 이상의 종목을 동시에 사고팔아 가격 차이(스프레드)를 노리는 주문을 행렬로 짜는 위젯. 예: 다른 만기의 선물 두 개를 반대 방향으로 동시에.

Arb 빌더
Arbitrage(차익거래) 빌더. 같은 자산이 거래소마다 가격이 다를 때, 싼 곳에서 사고 비싼 곳에서 파는 주문을 한 번에 설계·실행하는 도구입니다.

TWAP
Time-Weighted Average Price. 큰 주문을 시간에 걸쳐 잘게 나눠 체결해 시장 충격을 줄이는 알고리즘 주문입니다.

UIContext
AI 채팅으로 주문할 때 '지금 사용자가 화면에서 무엇을 보고 있는가'(선택한 종목, 계정, 주문 폼 상태)를 AI에 전달하는 데이터 구조. 이게 빈약하면 AI가 엉뚱한 종목에 주문합니다.

오더북
현재 시장에 걸려 있는 매수·매도 주문을 가격별로 쌓아 보여주는 표. 초당 수십~수백 번 바뀌어 화면 갱신 성능이 가장 까다로운 위젯입니다.

런타임 복구
브라우저 탭을 오래 방치했다 돌아왔을 때 끊긴 연결과 낡은 데이터를 되살리는 절차. 가볍게(재구독) → 중간(캐시 무효화) → 강하게(새로고침) 순으로 단계를 올립니다.

translate 기반 그리드
드래그 중에는 React를 거치지 않고 DOM의 transform: translate 값만 바꿔 움직이는 방식. 화면 재계산이 없어 60fps가 유지됩니다.

planner / reviewer
하나의 AI가 설계와 구현을 다 하지 않고, Claude가 계획·구현을 맡고 Codex가 독립적으로 리뷰하는 역할 분리. 서로 다른 모델이 검토해 오류가 걸러집니다.

Figma MCP
AI 에이전트가 Figma 디자인 파일을 직접 읽게 해주는 연결(Model Context Protocol). 디자이너가 바꾼 화면을 에이전트가 보고 코드에 반영합니다.

Protobuf
Protocol Buffers. JSON보다 작고 빠른 이진 데이터 형식. OrderX는 HTTP와 WebSocket 모두 이 형식으로 통신하며, 타입은 .proto 파일에서 자동 생성합니다.

Datafeed
TradingView 차트 라이브러리에 '이 종목의 캔들 데이터를 이렇게 가져와라'를 알려주는 어댑터. 백엔드 규격이 표준과 다르면 직접 작성해야 합니다.

FedWatch
선물 가격에서 역산한 '다음 FOMC에서 금리가 오를/내릴 확률'. 매크로 이벤트 전에 트레이더가 보는 지표입니다.

Abstract 체인
이더리움 위에 올라간 소비자용 L2 블록체인. AGW(Abstract Global Wallet)라는 내장 지갑과 세션 키를 제공해 게임 UX에 유리합니다.

Turnstile
Cloudflare의 캡차 대체 봇 차단. 사용자에게 퍼즐을 풀게 하지 않고 브라우저 신호로 봇을 걸러냅니다.

세션 키
지갑 주인이 '이 컨트랙트의 이 함수만, 이 수수료 한도 안에서, 24시간 동안' 허용하는 임시 키. 게임 중 매 액션마다 지갑 서명 팝업을 띄우지 않아도 됩니다.

Pyth 오라클
실시간 ETH/USD 같은 가격을 블록체인에 올려주는 서비스. 달러로 표시된 상품을 ETH로 결제할 때 환율 근거가 됩니다.

Hedge
보유 포지션의 위험을 반대 방향 거래로 상쇄하는 것. 헤지 시뮬레이션은 '이 계약을 추가하면 전체 위험이 어떻게 변하는가'를 미리 보여줍니다.

One-tap buy / Quick order
미리 정한 수량으로 클릭 한 번에 주문을 내보내는 빠른 주문 UI. 속도가 중요한 상황용이라 오조작 방지 장치가 함께 필요합니다.

RFC
Request for Comments. 변경 제안을 문서로 써서 다른 팀(여기서는 백엔드)의 검토를 받는 절차. 프로토콜을 바꾸기 전에 합의를 만드는 방식입니다.

포지션 테이블
지금 보유 중인 계약과 손익, 위험 지표(델타·감마 등)를 계정별로 보여주는 표. 가격이 바뀔 때마다 모든 행의 손익이 다시 계산됩니다.

react-grid-layout
위젯을 드래그·리사이즈하는 대시보드용 React 라이브러리. 마우스가 움직일 때마다 React 상태를 갱신해 무거운 화면에서 CPU를 많이 씁니다.

Context injection
AI 코딩 에이전트에게 레포의 규칙, 아키텍처, 도메인 지식을 매번 자동으로 넣어주는 방식(CLAUDE.md, AGENTS.md, 스킬 문서 등). 누가 어떤 에이전트로 작업해도 같은 기준이 적용됩니다.

react-doctor
React 코드의 불필요한 리렌더, 잘못된 훅 사용, 성능 문제를 찾아주는 진단 도구. 기능을 합칠 때마다 정기적으로 돌려 부채를 지웠습니다.

i18n
internationalization(다국어 지원). 화면 문구를 언어별 사전으로 빼고, 번역 누락을 찾아 채우는 파이프라인을 에이전트로 자동화했습니다.

WebSocket
서버가 브라우저로 데이터를 계속 밀어 보내는 연결. 시세·체결·잔고가 이 통로로 실시간 도착합니다.

OI 프로파일
Open Interest(미결제 약정) 프로파일. 가격대별로 열려 있는 계약이 얼마나 되는지 차트 옆에 그려, 큰 포지션이 몰린 가격을 보여줍니다.

SVI
Stochastic Volatility Inspired. 옵션 내재변동성 곡선을 몇 개 파라미터로 표현하는 수식. 옵션 데스크가 가격 왜곡을 볼 때 씁니다.

UUPS 프록시
배포된 스마트 컨트랙트의 주소와 저장된 데이터는 그대로 두고 로직만 교체할 수 있는 업그레이드 패턴. 결제 컨트랙트를 매주 바꿔야 할 때 필수입니다.

storage layout
업그레이더블 컨트랙트에서 변수가 저장되는 순서. 새 버전에서 순서가 어긋나면 기존 사용자 데이터가 덮어써지므로 배포 전 반드시 검토합니다.

EIP-712
사람이 읽을 수 있는 구조로 서명하는 이더리움 표준. 서버가 '이 가격에 이 상품을 사도 된다'고 서명하면 컨트랙트가 검증하고, nonce로 같은 서명의 재사용을 막습니다.