Crypto Trading AI Agent

How It Works — Explained for Everyone

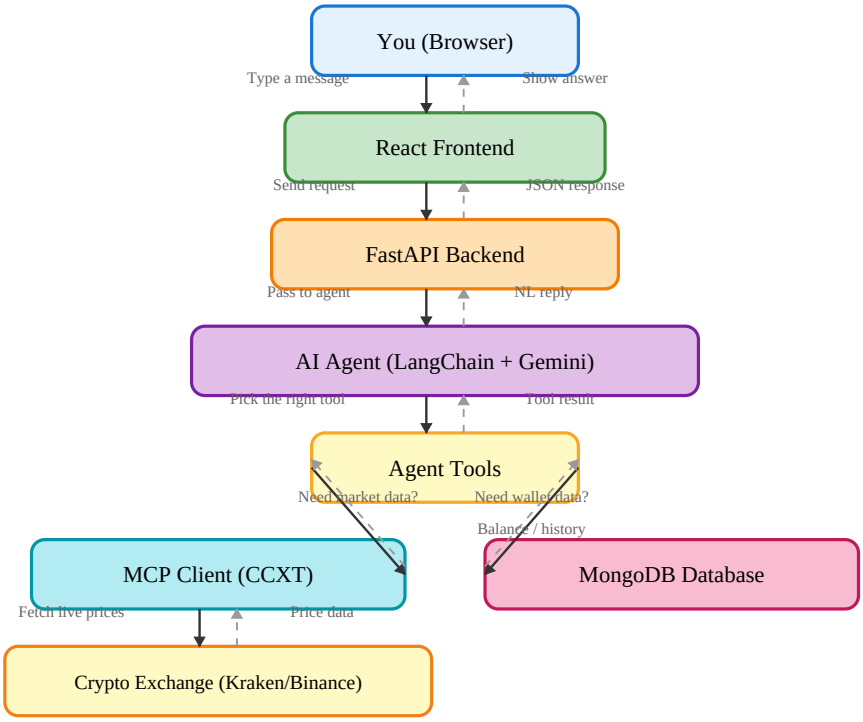
Overview

This app lets you **trade cryptocurrency using natural language**. Instead of clicking buttons on an exchange, you just tell the AI what you want in plain English — like texting a friend who happens to be a trading expert.

Step	What Happens
1. You chat	Type a message like “Buy $100 of Bitcoin” or “What’s the price of ETH?”
2. AI understands	The AI agent figures out what you want and picks the right tool.
3. Real data, simulated trades	Prices come from a real exchange (Kraken in prod, Binance locally). Trades use a simulated wallet — no real money. You start with $10,000.
4. See results	The AI replies in plain English. Wallet and transaction history update automatically.

System Architecture

Follow the arrows from top to bottom. Solid = request, dashed = response. Left side handles *market data* (prices), right side handles *user data* (wallet).



Component	Description
Frontend	React app (Vercel). Chat UI, wallet display, transaction history. REST API calls to backend.
Backend + AI Agent	Python FastAPI (Render). LangChain agent + Gemini 2.5 Flash interprets your language and calls tools.
MCP Client (CCXT)	CCXT library wrapper. Kraken in production (US-friendly), Binance locally. No API key needed.
MongoDB Atlas	Free M0 cloud database. Persists simulated wallet and transaction history across sessions.

What Happens When You Chat

Key takeaway: The AI agent acts as a smart router. It reads your message, understands the intent, picks the right tool, and translates raw data into a human-friendly response. You never need to know which API to call.

Example: you ask *"What is the price of BTC?"*

1. You type your question in the chat box and hit send.
2. The React frontend sends a *POST /chat* request to the FastAPI backend.
3. The AI agent (LangChain) passes your message to the Gemini LLM to understand intent.
4. The LLM determines it needs price data and calls the *get_crypto_price* tool.
5. The tool uses the CCXT library to query the exchange's public API for BTC/USDT.
6. The exchange returns real-time market data (price, bid, ask, volume, 24h high/low).
7. The LLM formats a natural-language reply and sends it back through the chain.
8. The frontend displays the answer — the whole round trip takes 3–8 seconds.

What Happens When You Buy Crypto

Key takeaway: The buy operation is *atomic* — the tool checks your balance, fetches the price, updates the wallet, and records the transaction all in one step. If any step fails (e.g. insufficient funds), the agent reports the error.

Example: you say *"Buy $100 of ETH"*

1. The AI recognizes you want to buy and calls *buy_crypto* with symbol and USD amount.
2. The tool fetches the current ETH price from the exchange (e.g. $1,972.25).
3. It calculates how much ETH $100 buys: 100 ÷ 1972.25 = 0.05070 ETH.
4. Your MongoDB wallet is updated: $100 deducted, 0.05070 ETH added.
5. A transaction record is saved with price, amount, USD value, and timestamp.
6. The AI confirms the trade in plain English with a breakdown.
7. Your Wallet and Transactions pages refresh automatically via Redux state updates.

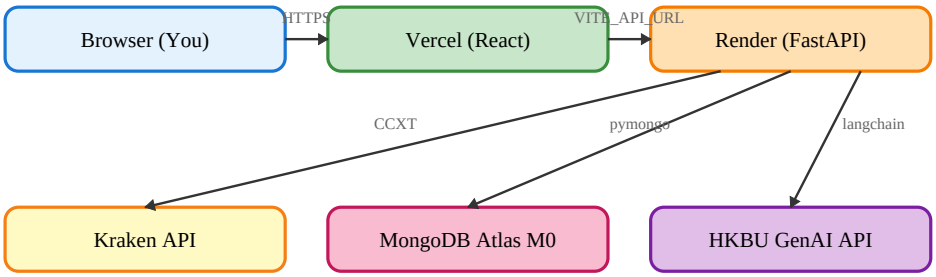
The AI Agent's Toolbox

The agent has 5 tools. Based on what you say, it automatically picks the right one (or multiple). Think of them like apps on a phone — the AI opens the right app.

Tool	Description
Get Crypto Price	Fetches real-time price from the exchange (Kraken in prod, Binance locally).
Get Order Book	Shows current buy/sell orders. Useful for understanding market depth and liquidity.
Buy Crypto	Simulates buying crypto with USD. Fetches real price, calculates amount, updates wallet.
Check Balance	Returns wallet holdings with USD values. Shows how much of each crypto you own.
Transaction History	Lists recent trades: type, amount, price, and timestamp.

Deployment Architecture

The entire stack runs on **free-tier services** ($0/month). Frontend is a static build on Vercel; backend is a web service on Render.



Why Kraken instead of Binance? The backend runs on Render's servers in Oregon, USA. Binance blocks API requests from US IPs, so production uses Kraken (no geo-restrictions). Locally you can still use Binance. Configurable via *DEFAULT_EXCHANGE* env var.

Service	Details
Vercel (Free)	Hosts React static build. Auto-deploys on git push. Global CDN.
Render (Free)	Runs FastAPI backend. Sleeps after 15 min; cold start ~30–50s.
Kraken API	Public market data, no geo-restrictions. No API key required.
MongoDB Atlas M0	Free cloud database (512 MB). Wallet + transaction storage.

Normal Bitcoin Buying vs This App

Aspect	Normal Bitcoin Buying	This App (AI Agent)
How you interact	Log into an exchange, navigate menus, click Buy, enter amount and price	Type in plain English: "Buy $100 of Bitcoin"
Understanding	Must know where to click and what each field means	AI understands natural language — no training needed
Price lookup	You check the price yourself	AI fetches real-time price automatically
Execution	You confirm the order; real money is spent	Simulated trade; no real money (demo wallet with $10k)
Wallet & history	On the exchange's servers; log in to view	In MongoDB; visible in app's Wallet and Transactions pages
Best for	Actual investing with real funds	Learning, testing strategies, trying crypto risk-free

In short: Normal buying = you do everything manually. This app = you chat, the AI does the work, and everything is simulated so you learn safely.

Tech Stack

Layer	Technology
Frontend	React, TypeScript, Redux Toolkit, Vite
Backend	Python, FastAPI, LangChain
AI Model	Gemini 2.5 Flash (via HKBU GenAI API)
Market Data	CCXT → Kraken (prod) / Binance (local)
Database	MongoDB Atlas (cloud, free M0 tier)

Hosting	Vercel (frontend) + Render (backend) — $0/month