



HASHLOCK MARKETS • INTERFACE DESIGN

Product Design Preview

Eight screens from the Hashlock interface redesign, showing how an OTC counterparty finds a request, agrees a price, locks funds into an HTLC and settles — and how an agent reaches the same rail through the API and the MCP server.

These are design prototypes, not screenshots of the production application. They are high-fidelity interface designs for the current redesign programme. The settlement rail they describe is live on mainnet across all six chains today; the interface shown here is the design under which that rail will be presented, and some frames still carry prototype placeholder copy. Each screen below notes what is live and what is design intent.

Hashlock Markets • File #10561489 October 2026 Design prototype • not production UI

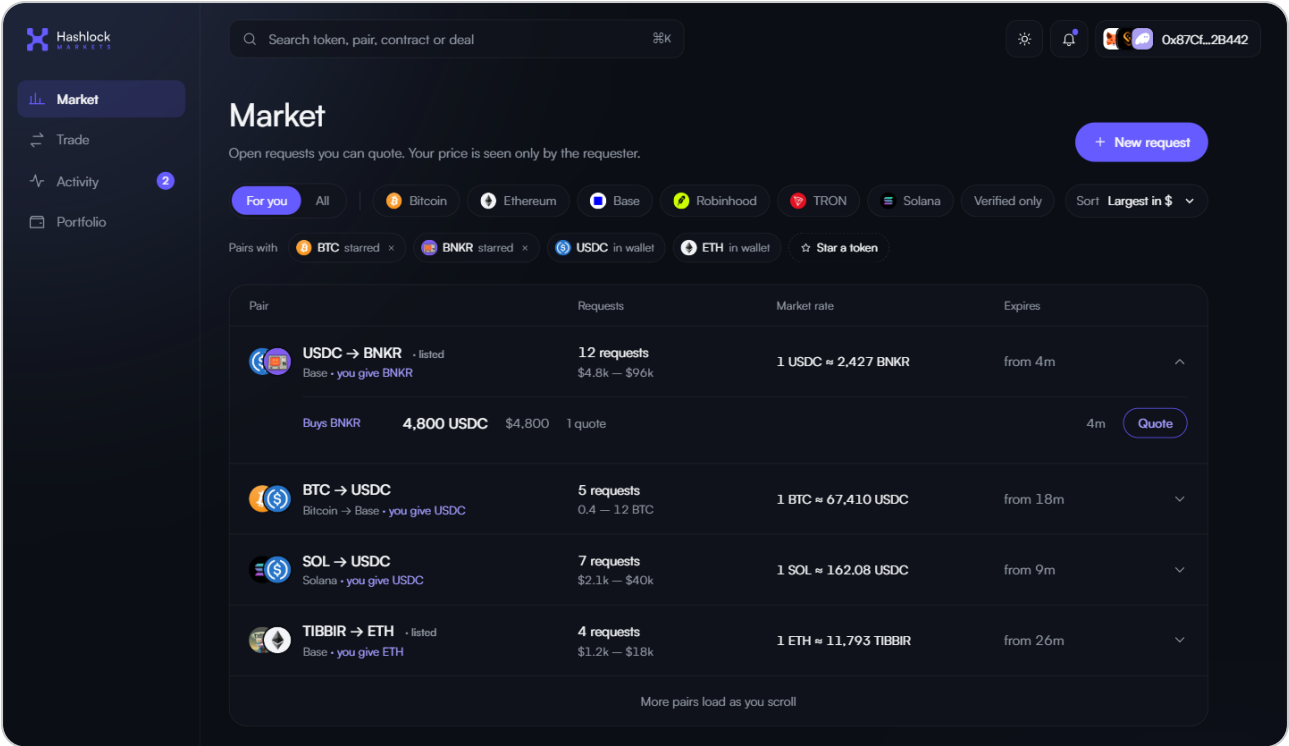
What is live behind these screens

The mechanics every screen depicts are shipped and running: dual hash time-locked contracts on Bitcoin, Ethereum, Base, Robinhood Chain, Tron and Solana, all on mainnet; a `/v1` REST API with webhooks and a maker feed; and an MCP server at `https://hashlock.markets/mcp`. **Hashlock never holds the funds at any point in these flows** — the contract is the escrow, and either both legs complete or both refund by timelock.

What is not yet live is the presentation: this visual system, the screen layouts and the copy on them. Treat the pictures as the design, and the document set elsewhere in this data room as the record of what the protocol does today.

Market — the RFQ board

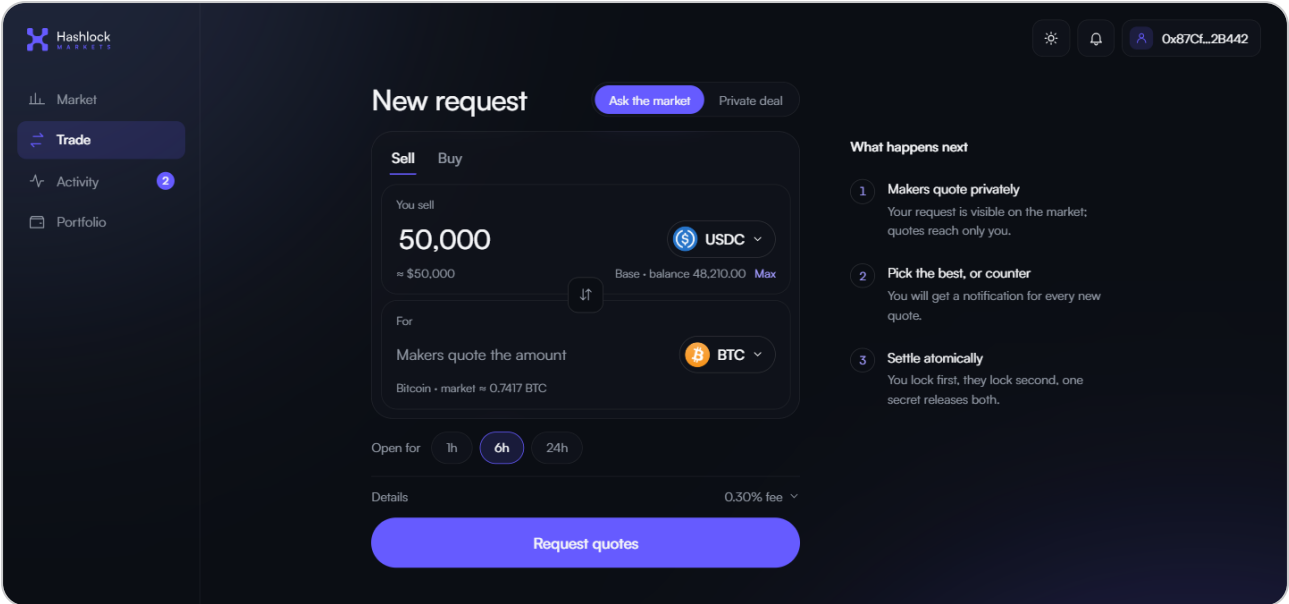
Open requests a maker can quote, grouped by pair. The chain filters across the top are the six chains that are live on mainnet today. A maker's price is seen only by the requester — the board shows the request, the size band and the time left, never the quotes.



Live today: the six-chain roster, the sealed-quote model, the request lifecycle. Design intent: this board layout, the token starring and the “For you” filter.

Trade — a new request

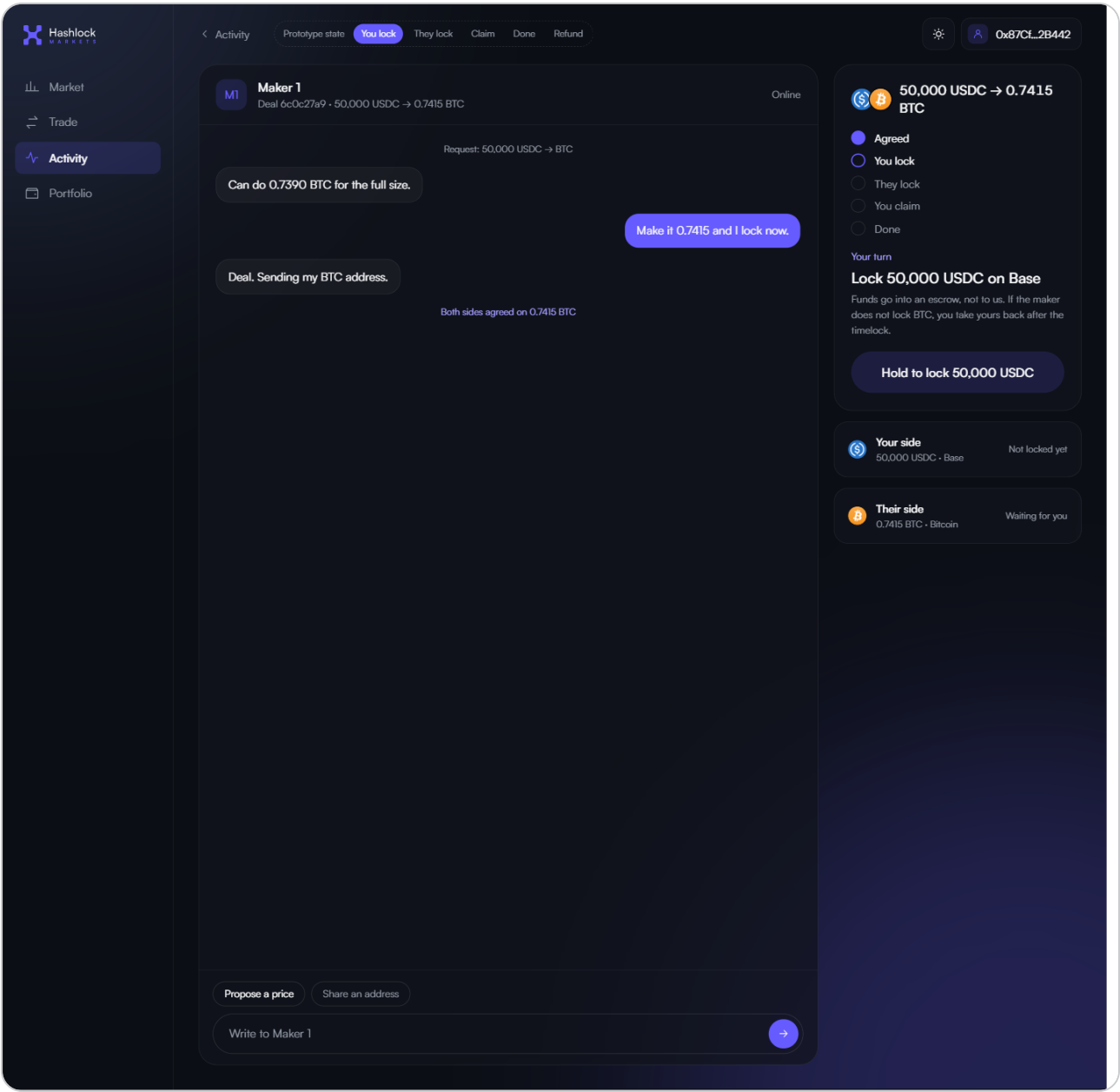
The requester states what they give and what they want, sets how long the request stays open, and sends it to the market — or opens it as a private deal with one counterparty. The panel on the right states the settlement promise in plain language: one side locks, the other locks, one secret releases both.



Live today: sealed RFQ, private deals, the lock-and-release settlement described on the right. Design intent: this composer. The “0.30% fee” on this frame is prototype placeholder copy — the commercial model in this data room is a volume-tiered fee of sub-10 bps, 8 bps in the plan.

Deal room — agreeing the price, then locking

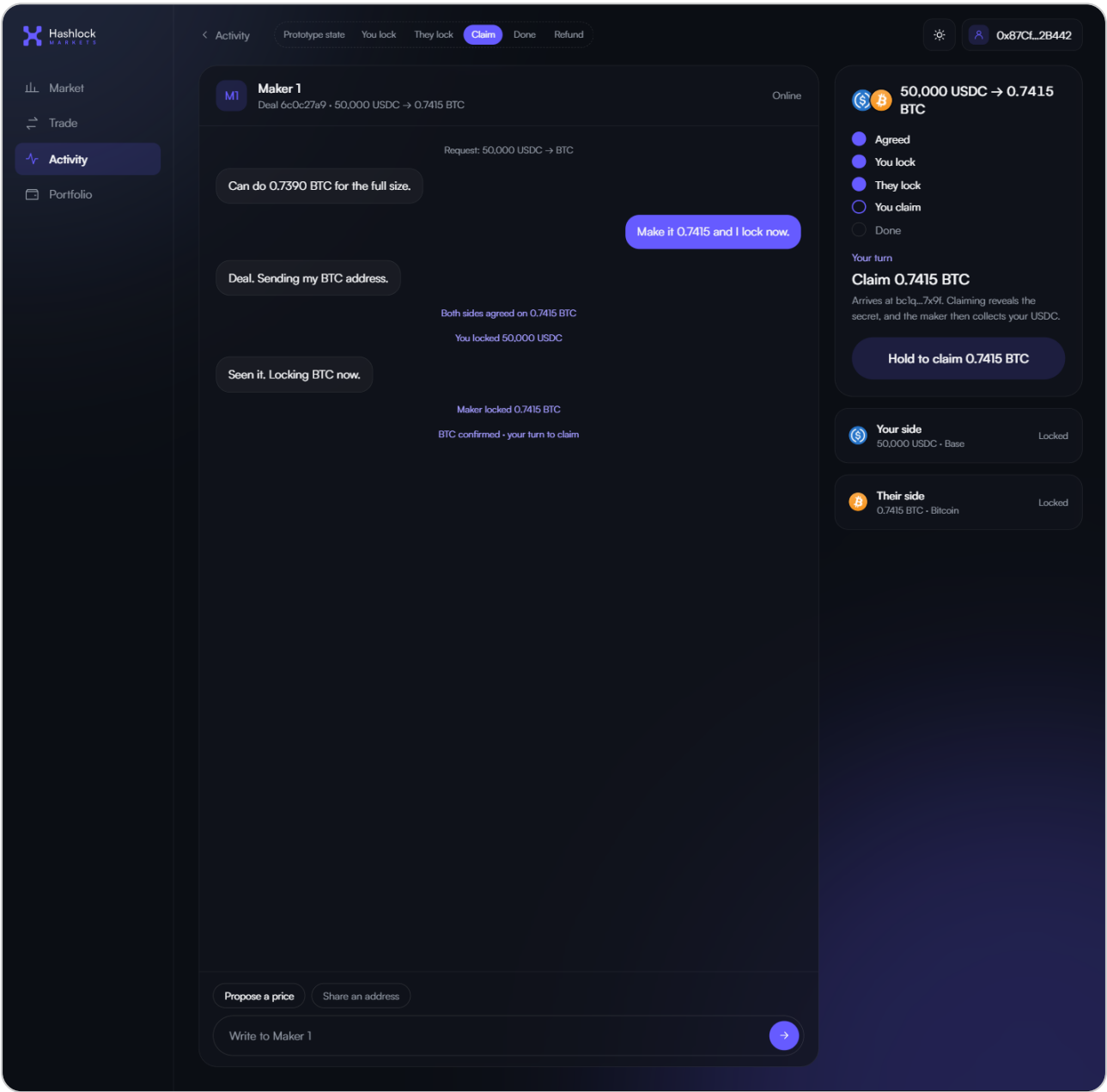
Negotiation and settlement in one view. The two sides agree a price in the thread; the panel on the right is the settlement state machine — agreed, you lock, they lock, you claim, done. The copy under the button is the non-custody promise stated to the user at the moment it matters: funds go into an escrow, not to Hashlock, and an unanswered lock is returned by timelock.



Live today: the full state machine and the timelock refund path. Design intent: the single-view deal room. The state switcher across the top of this frame is the prototype's own control for stepping through states, not part of the product.

Deal room — the claim

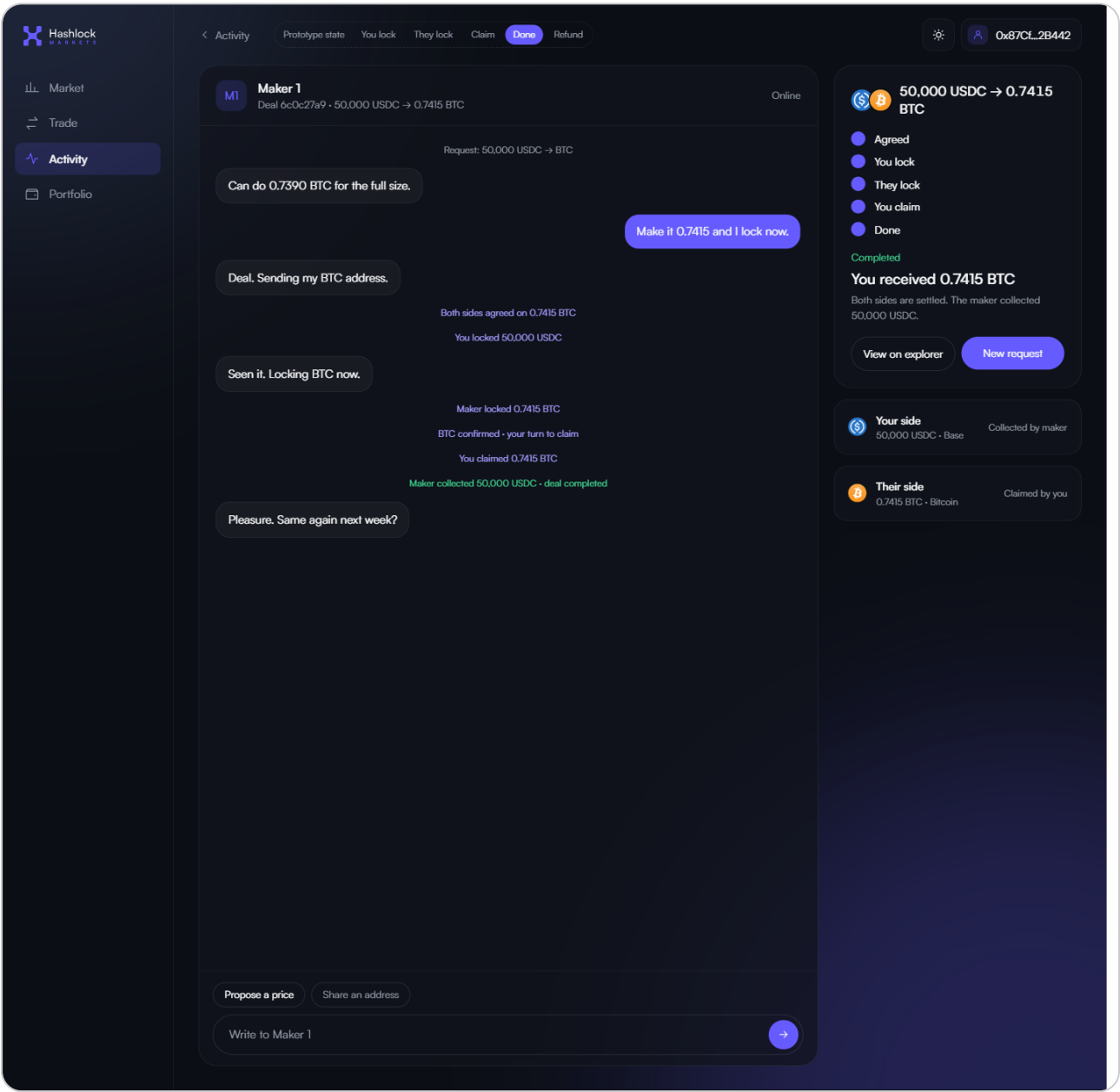
Both legs are funded and the receiving side claims. Revealing the preimage to claim one leg is what makes the other claimable — the single cryptographic act that makes the trade atomic.



Live today: preimage reveal and relay, and the keeper that submits claims and refunds on the EVM chains, Tron and Solana. On Bitcoin the spend needs the owner's own signature, so the Bitcoin leg is signed by the user.

Deal room — settled

The terminal state. Both legs completed, each on its own chain, with no intermediary having held either asset at any point.



The alternative terminal states — refund and refunded — are designed in the same frame set and are the path taken when a counterparty does not fund their leg.

Activity

Every request and deal the account is party to, on both sides of the book — requests sent and waiting on quotes, quotes given and waiting on an answer, and deals in settlement.

Hashlock
MARKETS

Market

Trade

Activity 2

Portfolio

Activity

Your requests, quotes and deals in one place.

Needs you 2

In progress

History

Q Token, counterparty or deal id

50,000 USDC → 0.7415 BTC

Deal with Maker 1 • Base → Bitcoin

Lock your USDC

3h 41m left

240 SOL → USDC

Your request • 3 quotes, best 38,904 USDC

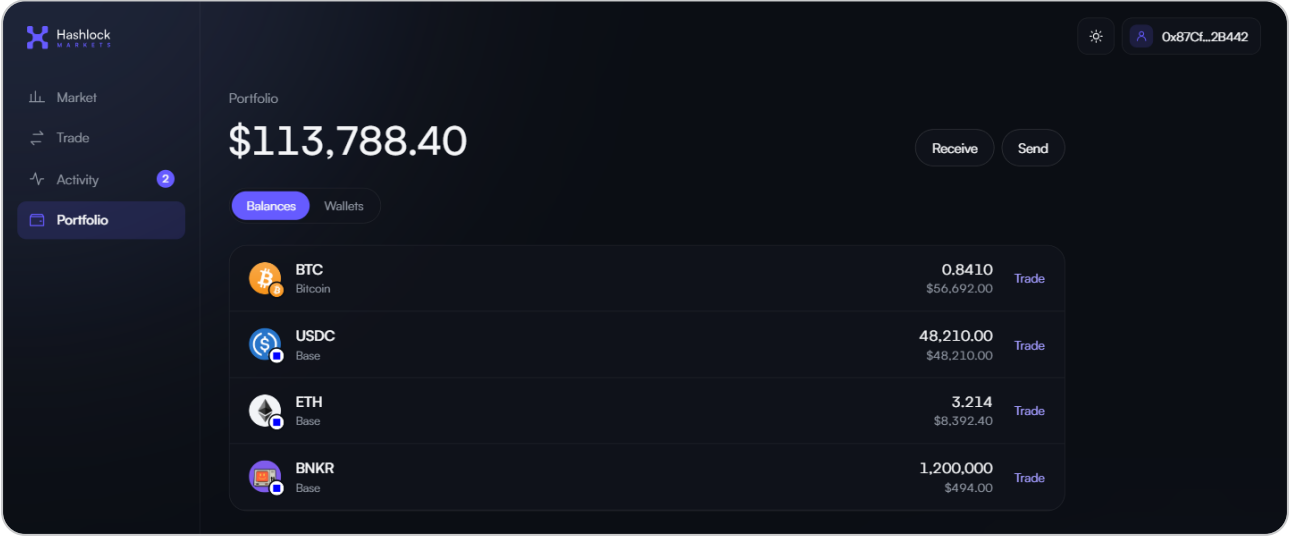
Pick a quote

12m left

Older activity loads as you scroll

Portfolio

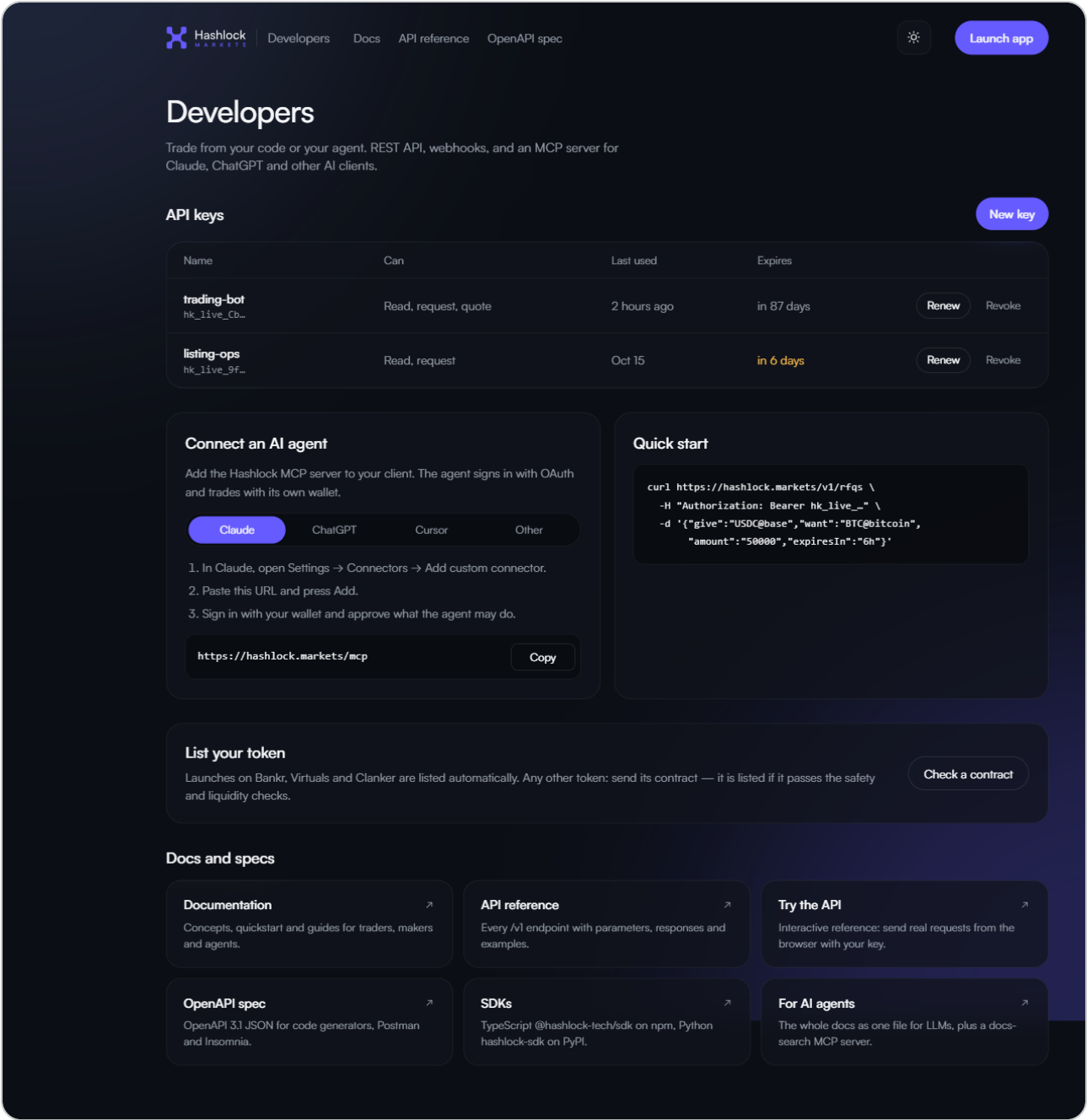
Balances across the wallets the user has linked to the account, with the chain each balance sits on.



These are the user's own wallets, read on-chain — not a Hashlock account balance. Nothing on this screen is held by Hashlock; unlinking a wallet removes it from the view and moves no funds. The non-custodial architecture is what keeps this screen a reading of the chain rather than a ledger Hashlock owes against.

Developers — the API and the agent rail

API keys with scoped permissions and expiry, the quick-start call against `/v1/rfqqs`, and the MCP server address an agent runtime connects to. This is the screen behind the claim that the same settlement primitive is reachable by a human desk and by an agent without a different product underneath.



Live today: `hk_live` keys, the `/v1` REST API with webhooks and a maker feed, and the hosted MCP server. Design intent: this console, the per-client connection guides and the token-listing check.