Cookie Chat — Whitepaper

The AI terminal for Cookie Chain

v1.1 · August 2026 · Living document

1. Executive Summary

Cookie Chat is a chat terminal for **Cookie Chain**. You ask a question in your own words – about a token, a wallet, the network, staking, the bridge, an NFT collection, an ecosystem app – and you get a short written answer together with live data rendered inline as cards, tables and charts. It is listed in the Cookie Chain ecosystem apps registry as an official ecosystem app, and it is live in production at https://cookiechat.net.

One question, every source: Cookie Chat reads all of Cookie Chain's data so you don't have to.

What separates it from a general-purpose assistant with a crypto prompt is a single architectural decision: **figures are computed from live sources, never written by a language model.** A deterministic engine reads the question, fetches from the chain and its data providers directly, builds the cards and selects the verified facts that question needs. Only then is a language model given those facts – and nothing else – to write the reply from. The numbers exist before it is called; its output is forced through one validated contract before it reaches the screen.

That division is what the whole product rests on, and it cuts the other way too. An engine that both computes *and* phrases can only phrase what was written into it in advance, which means it answers the topic rather than the question. Keeping the two apart is what allows an answer to be exact and specific at the same time.

Three commitments define the product:

- **Answers, not links.** A question returns an answer with the relevant cards inline – not a list of places to go and check.
- **Computed, then written.** Every figure is computed before any model runs; a model contributes wording, never a number.

- **Structured output.** Every answer conforms to the same schema-validated contract, so the interface is always well-formed.

$CHAT, the Cookie Chat utility token on Cookie Chain, is what opens the tiers above the free terminal. Its mechanics are deliberately out of scope here and are covered in a dedicated tokenomics paper; nothing in this document is an offer of any kind.

2. The Problem

Cookie Chain's data is real, public and scattered. Answering one ordinary question meant visiting a different place for each part of it: the cookiescan explorer for the chain and its tokens, the swap for pooled prices and charts, the chain's RPC for anything on-chain, a DAS indexer for NFTs, CoinGecko for $COOK's price, DexScreener for its liquidity, DefiLlama for TVL, Solana itself for holder counts, Bake Your Stake for the liquid staking pool, the team's updates feed for news, the apps registry for what exists in the ecosystem, and the $COOK whitepaper for everything the chain wrote down about itself. Each is accurate. None of them is an answer.

General-purpose assistants cannot see this chain – and don't admit it. Cookie Chain is young. There is very little written about it for a model to have learned, and none of it is live. Ask a general assistant for a price, a vault address or a validator count and it will produce something confident and wrong, because producing something is what it does. On a chain where a wrong address costs real money, a plausible answer is worse than no answer.

A block explorer has the data but not the answer. Explorers are built for people who already know what they are looking for and how to read it. "Is this token healthy", "what does this wallet actually hold", "how do I bridge" are not queries an explorer takes.

The result was that the people most likely to be asking – someone who just heard about Cookie Chain, someone holding COOK, someone building on it – were the least equipped to get an answer.

3. The Solution

Cookie Chat resolves a question in two stages that are deliberately kept apart: **what is true** is computed, and **how it is said** is written.

Oven 2 – the default engine. Oven 2 reads a question along two axes: the *topic* it is about, and *what is being asked* about that topic. "What is staking", "how do I stake", "how much is staked" and "is staking safe" share a topic and are four different questions. It resolves both axes, fetches the live data from the sources that hold it, builds the cards, and selects exactly the verified facts that ask needs – figures already computed and already formatted.

Those facts, and only those facts, are then handed to a language model to write the reply. It is given what a complete answer to that ask must contain, told to quote every figure exactly as computed, and instructed to state a gap plainly rather than fill it. Its output passes the same validated contract as everything else, and the cards rendered beneath it are the engine's own – not the model's.

Why the wording is written rather than stored. Cookie Chat originally composed the prose too, from stored copy assembled per topic and per ask. That copy cannot vary with the question: any two questions reaching the same topic and ask receive the same paragraph, however differently they were asked. Relevance scoring caught the worst of it, but a filter over a generator that does not vary has a ceiling – "not obviously wrong", never "answers the question you asked". Readers said so consistently, and they were right. Moving the wording to a model over verified facts removes that ceiling without giving up a single figure.

Oven 1 – the original fast path. A single ordered chain of keyword rules with one stored answer per topic, and **no model anywhere in it**. It is instant, has very few moving parts, and remains selectable for anyone who wants stored copy and nothing else. It sees the topic rather than the question, which is precisely why Oven 2 replaced it as the default.

A question, end to end. Take *"how are staking rewards distributed?"* Oven 2 recognises the topic as staking and the ask as a figure, reads the liquid staking pool and the chain's own staking totals from their respective sources, computes the rate, builds the staking card, and hands over the rate, the pool total, the epoch state and the written mechanics as facts. The reply is then written to that question – that rewards accrue into the exchange rate rather than being paid on a schedule – quoting the computed figures verbatim, with the card beneath it and the link to where staking actually happens.

When the model cannot run. No key, a cost switch, a spent budget, a provider outage: the engine's own composed answer ships instead, with its cards. A failure of the writing

stage costs phrasing, never facts, and never an error page.

4. Architecture

Every question enters one pipeline. The deterministic engine runs first and always: it routes the question, fetches the data, builds the cards and assembles the facts. The writing stage follows. The result is the same validated contract either way, and the interface renders it identically.

Inside the ovens. The operating principle is public: read the question, identify what is being asked, fetch the live data that answers it, assemble a structured response. The topic library, the matching and scoring rules, the facet model and the grounding logic are Cookie Chat's core intellectual property and are not published.

One output contract. Every answer is the same object: a short written body, an ordered list of typed blocks, follow-up suggestions and any disclaimers that apply. It is validated before rendering. That single contract is why the interface is always well-formed regardless of which layer produced the answer – and why a model that forgets to emit a card cannot degrade the result: the cards are rebuilt from the data the tools already returned.

Conversation memory. A follow-up inherits the topic of the turn before it, so "and how do I do that?" resolves against what was just discussed rather than starting from nothing. History is loaded server-side from the conversation store.

Stack. Next.js (App Router) and React on Cloudflare Workers via OpenNext; Tailwind for the interface; Cloudflare D1 for conversation history and feedback, KV for caching, feature flags and rate limits; schema validation on every answer.

5. Data & Accuracy

The product rests on one commitment:

> **Figures are computed from live sources, never generated by a model.**

Prices, balances, supply, holders, TVL, staking totals and network state are read from the sources that hold them and passed through untouched. When a model is involved, it reasons *around* those figures – it does not produce them.

Attribution as a trust mechanism. Every answer names the sources it was built from, in the note beneath it: where that note already says "from CoinGecko", the name itself is the link, and anything it doesn't mention is named on a line of its own. The list is collected while the answer is assembled, so what is cited is what was actually read rather than a list maintained by hand – a question about the price names the price feed, and an explanation drawn from the chain's documentation says so. Where a number came from is something you can check rather than something you have to take on faith.

Three layers, named honestly. Not everything Cookie Chat knows is live, and pretending otherwise would be the easiest way to lose the point of the document:

LAYER	WHAT IT IS	HOW FRESH
Live	Read from the chain and its data providers at the moment you ask	Current, every time
Curated registries	The chain's own public repositories – the ecosystem apps registry, the updates feed	As fresh as the team keeps them; labelled as curated
Static knowledge	Written text sourced from the published $COOK whitepaper and official documentation	Fixed until re-checked – guarded by an automated fact audit that re-verifies addresses, links and claims against their sources

Only the third layer can rot, which is why it is the one with a drift guard pointed at it.

Where the data comes from.

SOURCE	USED FOR
CookieScan explorer	The token universe, chain overview, supply and epoch, bridge statistics
CookieScan swap	Pooled tokens, accurate per-token prices, charts
Cookie Chain RPC	Network state, accounts, transactions, token holders, programs, validators
DAS indexer	NFT assets and collections
CoinGecko	$COOK price, market cap and price history
DexScreener	$COOK open-market liquidity
DefiLlama	Chain TVL
Solana RPC	$COOK holder count on its Solana listing
`cookiechain/chain-updates`	News and official updates
`cookiechain/apps`	The official ecosystem apps registry
$COOK whitepaper	Tokenomics, governance, reserves, history

When there is no answer. A name that appears in none of these – not a tracked token, not an ecosystem app, not in the documentation – is answered with *"I don't know"* and a list of what was checked. That is a deliberate feature, not a shortfall: an assistant that fills gaps is one you cannot use for the questions that matter.

6. Product & Capabilities

Cookie Chat serves three jobs with one interface.

Understanding the chain. *"What is Cookie Chain?"* · *"What is Gorbagana?"* · *"How does governance work?"* · *"Is COOK safe?"* – explanations of the chain, its origin, its

tokenomics, its risks and its governance, each ending at the place where you can act on it rather than at a full stop.

Following the money. *"COOK price"* · *"Show me this wallet"* · *"COOK vs SOL"* · *"What's trending?"* · *"How much is staked?"* – live prices, market caps, liquidity, holders, wallet holdings and allocation, NFT collections, token comparisons, network health and staking figures.

Building on it. *"How do I connect to the RPC?"* · *"What programs are deployed?"* · *"Where are the reserves?"* · pasted addresses and transaction signatures – resolved, classified and summarised.

Every answer renders as ordered blocks rather than a wall of text:

BLOCK	RENDERS
`text`	The written answer
`tokenCard`	One token: market cap, price, change, liquidity, holders, links
`tokenList`	A ranked or grouped set of tokens
`priceChart`	Market cap over time
`walletCard`	An address, its total value and top holdings
`portfolioAllocation`	A wallet's allocation breakdown
`nftList`	NFT holdings or a collection
`newsCard` / `newsList`	One update, or a feed of them
`networkHealthCard`	Slot, block height, TPS, fees, TVL
`validatorList`	The validator set
`bridgeStatusCard`	Solana ↔ Cookie Chain bridge state and reserves
`programList`	Deployed programs
`stakingCard`	Staking, the pool and rewards
`projectList`	Ecosystem apps, marked as curated
`table`	Anything tabular

Follow-up chips accompany every answer and are written to resolve to topics the engine routes squarely, so a suggested question is never one the app answers poorly. Chat history is kept per device, answers can be rated, and the interface works on a phone.

7. Positioning & Principles

Cookie Chat sits between assistants that talk fluently about a chain they cannot see, and tools that hold the data but not the answer.

	GENERAL AI ASSISTANT	BLOCK EXPLORER	MANUAL RESEARCH	COOKIE CHAT
Knows Cookie Chain	Barely – too new to have been learned	Completely	Completely	Completely
Figures	Generated	Computed	Computed	**Computed**
Output	Prose	Raw data	Whatever you assemble	**Answer plus live cards**
Effort	One question	You must know where to look	Your time, every time	**One question**
Audience	Everyone	People who already know	The determined	**Everyone**

Five principles govern the product.

1. **Answers, not links.** A question gets an answer with the data inline, and a door at the end of it – the place where the thing being discussed actually happens.
2. **Deterministic where possible, generative where necessary.** Facts are computed. Only judgement and explanation reach a model.
3. **"I don't know" beats a good guess.** Cookie Chat says what it checked and stops. It does not invent addresses, figures, dates or announcements, and it does not repeat a claim back as though it had confirmed it.
4. **No keys, no accounts, no custody.** Cookie Chat holds no signing key and produces no signature; where it builds a transaction, the user's own wallet signs it. No sign-up, no identity beyond a per-device cookie.
5. **An official app, not a spokesman.** Cookie Chat is listed in the ecosystem apps registry and points to official sources – it does not speak for the chain and cannot confirm what the chain has not made public.

8. Security & Privacy

No keys, by construction. Cookie Chat holds no signing key, on any server, for any user. It cannot produce a signature, and no configuration change would let it. Wallet

connection is optional; reading a balance takes no signature at all.

The custody model, now that it prepares transactions. An earlier version of this document promised that if Cookie Chat ever became able to prepare a transaction, the custody model would be stated explicitly rather than assumed. It can, so here it is.

Oven 3 – the default engine – can attach a card that completes what a question asks for. Today that is a **swap**, the registration of a `.cook` **name**, **staking and unstaking COOK**, a **bridge transfer to Solana**, and **buying, listing and making or accepting offers on NFTs**. The sequence is:

1. The action is priced live – a swap route against the Cookie Chain aggregator, a name against the registry's own on-chain config, a stake against the pool's own exchange rate and fees, an NFT against the marketplace's current listing.
2. The transaction is **built unsigned**, with the connected wallet as its fee payer, and those bytes are passed to the browser.
3. **The wallet on the user's own machine signs them**, in its own prompt, showing its own summary of what is about to happen.
4. The browser sends the signed transaction and waits for confirmation.

The server's role begins and ends at step 2. It never holds a key, never receives one, and the transaction it hands over carries an empty signature slot – a fact anyone can check by decoding what the endpoint returns. Nothing moves without a signature that only the user can produce, and an action the user does not approve simply does not happen.

Two further guards sit around a swap. The transaction enforces a **minimum received** amount, so a route that moves against the user fails rather than filling at any price; and the price is re-checked at build time, with the wallet prompt withheld if the new route pays materially less than the figure on screen – nobody is asked to sign a number they were not looking at.

A registration has a guard of its own: it is **simulated against the chain before the wallet is opened**. The two things that actually go wrong – somebody registered the name a moment earlier, or the wallet cannot cover the fee – are answered on the card, in plain words, rather than at a signature prompt. Nothing is signed or sent when that simulation fails.

One consequence of reading the same registry is worth stating, because it is about other people's money rather than the user's: a `.cook` name that is **listed for sale** is refused wherever an address is expected. While a name is listed, the registry reports its owner as the marketplace's escrow account – a program-owned address that cannot sign – so treating it as a wallet would send funds somewhere nobody can spend them.

The bridge asks a question the source chain cannot answer, so it asks the destination. A transfer to Solana is dispatched over the Hyperlane warp route: COOK is locked on this side and a relayer releases it on the other, usually within a minute. Two things decide whether that release can happen, and neither is visible to a simulation on Cookie Chain – whether the route's Solana-side escrow still holds enough COOK, and whether the recipient has an account there to be credited at all. Both are read before anything is signed, because a transfer that fails either one does not fail: it succeeds here, locks the money, and then hangs undelivered with no error anywhere. When either check comes back short, the card refuses and says so. When a check cannot be run at all, the transfer is allowed – an unreadable account is an unknown, never a zero.

That is also why a bridge receipt has two states rather than one. A confirmed signature means **dispatched**; the card then reads the destination mailbox's own record of the message and only says **delivered** once that record exists. Cookie Chat completes the outbound direction only. The return leg is signed and broadcast on Solana, and this application's wallet connection is to Cookie Chain – so it names the Hyperlane bridge interface instead of quietly building the wrong thing.

NFT trades settle on the marketplace's own auction house, the same program behind Baked Bazaar, and the same custody rule applies with one addition worth stating: **the price a buyer pays is never taken from the browser.** A question names an NFT; the listing price is read from the marketplace at build time and put into the instruction. The only figure the user supplies is one about their own money – what to list at, or what to offer. Royalties are read from the NFT's own metadata account rather than from an index, because the settlement pays each creator directly and a wrong list is somebody's royalty going missing. The marketplace takes 1%, which the card shows before anything is signed.

Two actions need a second signature, and it is not ours either. Both staking instructions require a throwaway key beside the user's wallet: a deposit debits an

ephemeral funding account, a withdrawal burns through an ephemeral transfer authority. That key is **generated in the browser and never leaves it** – the server is told its public half, builds a transaction naming it, and the browser signs with it before the wallet signs. Generating it here would have made this a service that holds signing keys, which is the one thing it is not. The key is also worth nothing outside its own transaction: a transaction is atomic, so the funding account is emptied by the pool in the same transaction that fills it, and the delegate's allowance is spent by the burn in the same transaction that grants it. A bridge transfer needs one for a different reason and on the same terms: Hyperlane derives its replay protection from a one-shot key that has to sign, and that key is generated in the browser too. It authorises nothing and holds nothing – its only job is to be unrepeatable.

Sending, approving and launching remain outside the boundary. They are refused structurally, exactly as a swap once was, and each will be described here before it ships rather than after.

No accounts. There is no sign-up and no identity. A browser is identified by one `httpOnly` cookie; conversations belong to that cookie and are not visible across devices or to other users. All secrets are server-side and never reach the browser.

What is stored, and what it is used for. Conversations and 👍 / 👎 ratings are stored, and they are read. Real questions asked in production are what the routing and answer test suites are built from – every misrouted or badly answered question is a case that gets added and then has to keep passing. Cookie Chat is better than it was because of questions people asked it, and that is worth stating plainly rather than burying.

Limits. Only the writing stage is metered – per session, per address and against a daily ceiling – because it is the only part that costs anything to run. The ceilings are set well above ordinary use and exist to stop abuse, not to ration answers. Reaching one does not produce an error: the question is answered by the engine's own composed reply, with the same cards and the same figures. The model layer can be switched off entirely without taking the product down.

9. Monetization & Token

Cookie Chat is **free**. No subscription, no per-question fee, no payment details, no wallet required, no invite code and no sign-up.

The architecture is what makes that possible: most questions are answered without ever reaching a language model, which is where the cost of an assistant usually concentrates. The data providers behind those answers are not free, and that cost grows with use – which is precisely what a paid tier is for.

$CHAT, the Cookie Chat utility token on Cookie Chain, launched in August 2026, and it is what opens the premium tier: higher limits, alerts and premium capability unlocked by **holding the token** rather than by a card subscription. Holding, not spending – the balance is read on-chain from the wallet connection this terminal already has, with no signature and no transfer, and the tokens never move. The free tier stays genuinely useful – that is a consequence of the architecture, not a concession.

The token's supply, allocations, vesting schedules and utility are deliberately out of scope for this document and are covered in the dedicated **tokenomics paper** at https://cookiechat.net/tokenomics, which also publishes every address: the mint, the pool, the project wallet and the three on-chain escrows.

> **There is exactly one $CHAT contract**, published in the tokenomics paper and returned by this terminal whenever it is asked. Any other address claiming to be $CHAT is not ours and should be treated as a scam – impersonation is at its most common right after a launch. Announcements come through the official channels named in §12 and nowhere else.

Nothing in this document is an offer, a solicitation, or a promise of any token, in any jurisdiction.

10. Current Scope & What's Next

Several things sit deliberately outside today's boundary. Each is a choice with a next step; the full picture is in the companion roadmap.

- **Non-custodial, and now able to act.** Cookie Chat holds no key and produces no signature – that guarantee has not moved and will not. What changed is what it can build: **Oven 3**, the default engine, answers exactly as Oven 2 does and, where a question asks for something to be done rather than reported, attaches a card that

completes it. Today that is a **swap**: the route is priced live, the transaction is built unsigned, and **your own wallet signs it on your machine**. The server never sees a key and could not sign if it wanted to. Ask for a swap on another engine and you are told which one can do it. *Next:* the remaining actions – staking, bridging, NFTs and `.cook` names – built the same way, from the same source. The instruction-building comes from **Cookie MCP**, the ecosystem's MCP server (MIT), ported rather than hosted: that server signs with a local key, which is right on the machine of the person who owns it and impossible behind a public site.

- **Answers follow your language; the interface does not yet.** Ask in German and the answer comes back in German, with the card labels translated too for eighteen of forty-two languages, and any language at all for the prose – the figures are computed identically whatever you asked in, so only the wording changes. The chrome around the conversation, and the step-by-step walkthroughs, are still English. *Next:* the rest of the surface.
- **Request and response.** Answers are produced when asked, which keeps the system simple and predictable. *Next:* watchlists and alerts, then cards that update while you watch them.
- **Per-device history.** The cookie identity keeps the product frictionless and free of accounts. *Next:* optional sign-in layered on top of it, never replacing it.
- **One surface.** Cookie Chat is a website. *Next:* a public API and an embeddable widget, so other ecosystem apps can answer their own users' questions with the same engine – and then a native mobile app on top of it.

11. Vision

Cookie Chat begins as a terminal, but the ambition is to become **the default interface for Cookie Chain** – the layer through which people arrive at the chain, understand it, follow it, and eventually act on it, all by asking.

The chain does not need another dashboard. It needs a place where a question gets an answer, where the numbers are real, and where the honest reply to something unknown is that it is unknown. The same engine that computes an exact answer today can tomorrow serve other ecosystem apps, drive agents, and extend from reading the chain toward acting on it under the user's control.

The thesis holds the whole way up: one question, every source – and figures that are computed, not guessed.

12. Contact

Cookie Chat's own channels are **@askCookieChat** on X – the handle listed beside it in the ecosystem apps registry – and **t.me/askCookieChat** on Telegram. Those two are the only channels that speak for this app.

For the chain itself: **cookiechain.wtf** · **docs.cookiechain.wtf** · **cookiescan.io** · **@TheCookieChain**.

13. Disclaimers & Legal

This document is for informational purposes only. It is not financial advice and is not a recommendation to buy, sell or hold any asset. Cookie Chat does not provide personalised financial advice and makes no price predictions.

Nothing here is an offer or solicitation to sell securities or a token, in any jurisdiction.

This document contains forward-looking statements. Roadmap items describe direction and intent, not commitments; features, scope and timing may change.

Cookie Chat is an official Cookie Chain ecosystem app. It does not speak for Cookie Chain, is not a channel for official announcements, and cannot confirm anything the chain has not made public. Data drawn from curated registries is labelled as curated within the product. Cookie Chat can be wrong – verify anything that matters against the official sources it links to.