Bai: The Full Introduction

The community collaboration assistant "Bai"

Summary

The community collaboration assistant "Bai" is an open-source platform for community collaboration and, at the same time, a contribution ledger that records what people give to a community. It gives every community member a digital identity they can sign in with using semi; inside it, the community posts tasks, makes announcements and runs events, while members claim tasks, submit proof and earn community points. Every bit of effort given to the community becomes a traceable record — public and trustworthy. It grows alongside the community digital identity Semi: Semi certifies "who we are", and Bai records "what we have done for the community". It is both an incentive model that captures community contribution as it happens and an exploration of what digital community governance might look like in the future.

1. It Starts with a Community

In our hands-on experience with local communities, we found that a community always has more work than hands: someone needs to help in the kitchen, someone needs to run an event, someone needs to fix a stretch of fence, someone needs to sort out a pile of documents, someone needs to lead a round of online co-learning. These jobs are warm and mundane at once, and they depend on people doing them of their own accord. What keeps a community running over the long term is not who has the loudest voice, but whether every small thing can be seen, remembered and answered.

Yet the moment a task moves online, it tends to slip: a request posted in a group chat is buried after two scrolls; sign-ups happen by chain message, and whether someone has signed up has to be checked by calling out names; when the work is done, all that is said is "I've finished it" in the group. The larger the community and the more people come and go, the harder this "running things from memory" approach becomes. With people coming and going, there is no shared yardstick. Some people have lived here for years, others stay only two weeks; some are right in the village, others are a thousand miles away. In a world of familiar faces, "I keep it in my head" is enough; once there are more people and they change often, you need rules that are public and records that can be checked.

Communities do not lack enthusiasm. What they lack is a place where that enthusiasm can be seen and kept.

Bai was built to move this everyday collaboration into a place everyone can see.

2. What It Is

The community collaboration assistant "Bai" is a collaboration and management platform for communities.

You can think of it as a shared hall for the community:

- Announcements are posted on the wall, so no one misses what is happening in the community;
- A task wall sits on the table, where anyone who needs help and anyone who can help is listed out in the open;
- In the corner is a contribution ledger that notes what each person has done for the community.

Behind it, Bai connects to the community digital identity Semi — with the same identity, members can move freely between multiple community apps without registering an account everywhere; and once a contribution is completed, the record lands back under that same identity. Bai grew out of real community collaboration, but it does not belong to any single community: any village, cooperative, community or interest group can use it to organize its own work.

3. What It Can Do Today

1. Sign in with a semi identity and join a community. After signing in with semi, the system automatically creates an account; through Semi's digital identity, one account can sign in to multiple community apps. On first entry, you can browse the community plaza and join the communities that interest you, then switch between them from the top bar at any time.
2. Post and claim tasks — the core of Bai. Posters spell out what the task involves, how many points it pays, the claim and submission deadlines, what kind of proof is required and whether several people may work on it together; members browse the task plaza, filter by tags, keywords, deadlines and points, and claim with one click. A task can be limited to one person or completed by several together.
3. Submit proof, closing the review loop. The claimer uploads evidence of the work, and the poster can "approve", "reject and resubmit" or "terminate". Tasks move through "unclaimed → claimed → awaiting submission → under review → completed", and a rejected submission can be sent again. The whole process stays on the task timeline — who did what, for the community, and when.
4. Community points, so that every contribution counts. Points are issued as soon as a task is completed. Different communities can set up their own points under their own rules to record local members' contributions. Points are not a bonus; they are a shared contribution ledger.
5. Run events. Create an event with its time, place, capacity and sign-up method, open registration and generate a sign-up proof; members can see what is coming up in the community calendar and register on the event page. Publishing, registration and the participant list for an event are all managed in one place.
6. Community shop and billing. Communities can list goods (with variants, stock and prices), and members can add them to a cart, check out and pay. The billing center keeps every order, every status and its on-chain proof, and supports resuming payment, requesting a refund, and viewing receipts and fulfilment progress; payments are reconciled automatically from on-chain records, with no manual "confirm receipt" step.
7. Announcements and updates. A community's announcements, updates, members and introduction are shown together on its home page; admins can publish and pin announcements to maintain the community's public information.

8. Messages and notifications. When a task is claimed, a review reaches a result, an event moves along, or a bill changes, the person involved is notified in the app.

9. Personal profile. A member's avatar, communities, community points, wallet address and payment QR code, along with the tasks and updates they have posted and taken part in, together make up their storyline in the community.

4. Why We See Points and Records the Way We Do

In a community, points are not only for transactions; they are more like a shared contribution ledger: a pot of food someone cooked and a line of code a distant partner wrote can both be seen fairly in this ledger. What it aims to do is turn contribution from the memory of a few into a shared record — and in a community, that record is also an important basis when people take part in discussions about community affairs and make decisions together.

In this system we place particular value on the storyline: a points entry states only a number, but with the task, the proof, the review and the timeline it tells a concrete story — who did what for the community, when they did it, and whether they got it done. Years later, when many of these records are looked over again, they are still there; they do not disappear because the person in charge changed or the system changed.

Our original intent was to make everyone's contribution visible and recorded, and at the same time to explore what community governance might become. Recording the present is also recording the future.

5. How It Works (Technical Section)

This section is for friends who are interested in the technical side. If you are not familiar with technology, feel free to skip it. All of it can be found in the public repositories on GitHub.

1. Overall structure: three layers, three jobs

Bai consists of three parts:

- Frontend (mycoseed-frontend): a web app written with Nuxt 3 + Vue 3, with Pinia for state, Nuxt UI and Tailwind CSS for the interface, deployed on Vercel. It handles display and navigation and holds no keys.
- Backend (mycoseed-backend): an API service in Node.js + Express + TypeScript, with data stored in Supabase (PostgreSQL) and images and proof also stored in Supabase Storage; deployed on Fly.io with a Dockerfile, so it can be moved to any server that supports Docker.
- Smart contracts (contracts-project): written in Solidity and compiled and deployed with Hardhat. What is on-chain today is the remark contract RemarkLogicV1, wrapped in an upgradeable proxy, so that a bug can later be fixed by swapping the implementation while the contract address stays the same.

The on-chain logic is concentrated in the backend's services/billing directory; the frontend never deals with the chain directly.

2. The life of a task: a state flow with an audit trail

- Tasks live in the two tables tasks / task_info: title, description, points, claim deadline, submission deadline, proof required, whether multiple people are allowed, tags, and so on.
- The state moves through "unclaimed → claimed → awaiting submission → submitted → under review → completed"; a rejected submission can be resubmitted, and a task can also be terminated. Every change appends an entry to the task timeline, recording the status, the operator and the time, without overwriting earlier records.
- For tasks with multiple people, records are kept per person: who claimed, who submitted and who was approved are tracked separately and never mixed up.
- When proof is submitted, the backend compresses the image, stores it in Supabase Storage and calculates a SHA-256 digest to keep for verification. The review endpoint is open only to the poster, and the server checks the operator's identity every time.
- Whenever the status changes, a notification is written to the notifications table, and whether it is sent depends on each user's own notification settings. The frontend reads it from the API, so no one has to track things by hand.

So a task does not rest on someone saying "it's done" out loud; every step has a record, an operator and a time.

3. Working with Semi: one identity, a whole ecosystem

Bai and Semi have clear roles: Semi is the digital identity, and Bai is the digital record. The two are connected through unified sign-in and on-chain remarks.

- Signing in is a single OAuth authorization. Bai's backend joins Semi as an OAuth 2.0 client, using "authorization code + PKCE" (so that even if the code is intercepted in transit it cannot be exchanged for a sign-in credential), with state to prevent requests from being forged or mixed up. On callback it exchanges the code for a token, reads the user's information, then finds or creates the same person by their Semi user ID and records their smart account address. The same Semi identity is the same user in Bai; when other community apps connect to Semi later, they share that one identity and no one has to register again.
- Points are issued in the Semi smart account. Every Semi account is a Safe smart account (in line with the ERC-4337 account abstraction standard), and gas fees are covered by the platform's paymaster service. The user enters a 6-digit payment code, and the browser unlocks the key and signs locally. Bai never touches the private key or stores the payment code at any point.
- Leaving Bai carries a "coordinate". When the poster clicks "issue points" on the task page, Bai opens Semi's transfer page and passes along the task pool ID, the task ID, a default remark (for example, "Task: “Title” + time", which can be edited) and the note the claimer wrote.
- Semi converts the two IDs into on-chain numeric IDs with keccak256 and calls the remark contract's saveRemark(poolId, taskId, senderRemark, receiverRemark). "Issuing points" and "writing the remark" are placed in the same on-chain operation: either both succeed or neither happens. The contract only

keeps proof and never touches money, and emits a RemarkSaved event when it is done.

- The record can be found again, and checked. Bai's backend queries on-chain logs through Alchemy by event signature and ID hash, retrieving the two remarks and the time they were written, matched one-to-one with the task. When the frontend syncs the result back, the backend does not take it on trust: it reads the receipt for that operation and confirms that the sending address really is the person's smart account and that the recipient really was part of the operation; success or failure and the fee are all taken from the chain.

This means identity comes from Semi and contribution goes back to Semi; every issuance in between is recorded by Bai and proven on-chain.

4. If you want to deploy it yourself

The database migration scripts are all in db/migrations, run in numerical order and come with rollback scripts; community-level switches can be rolled out gradually per community or rolled back with one click. The backend has a Dockerfile and Fly.io configuration; the frontend can be deployed straight to Vercel or any Node environment. Settings for the chain, token, OAuth, SMS and so on are kept in environment variables, and a missing value causes an immediate error at startup.

Other community tools that want to connect to Semi only need to register an OAuth application in Semi and set the callback URL and scopes.

The code is all on GitHub (github.com/nantang-dao); you are welcome to look at it, use it and help improve it.

6. Next Steps

Bai's money flow today is a transitional implementation of "peer-to-peer direct payment": points and payments are completed through Semi, with automatic reconciliation based on on-chain transfers and remarks. This is the part we deliberately got working first — complete in structure and safe to use, but not yet involving escrow or split payments.

Next, we plan to deploy the escrow contracts for real, so that money in the community genuinely "goes into escrow first and is released later":

- Task staking and escrow. The poster locks the bounty in the contract first, and it is released to the claimer only after acceptance; any part not completed is returned proportionally.
- Escrowed payouts for the shop and events. For the shop, funds are released after the buyer confirms receipt; for events, funds are released to the organizer after the cancellation deadline, and before that the event can be cancelled with a refund.
- Platform fee and community fee. Rates can be set by category and by community, with the fees kept inside the contract and settled in batches each month.
- On-chain refunds and reconciliation. Payouts, refunds and fee accrual are all driven by contract events, so reconciliation no longer depends on manual work.

By then, every contribution in the community will have not only a clear storyline but also a money flow that can be held in escrow, traced and settled.