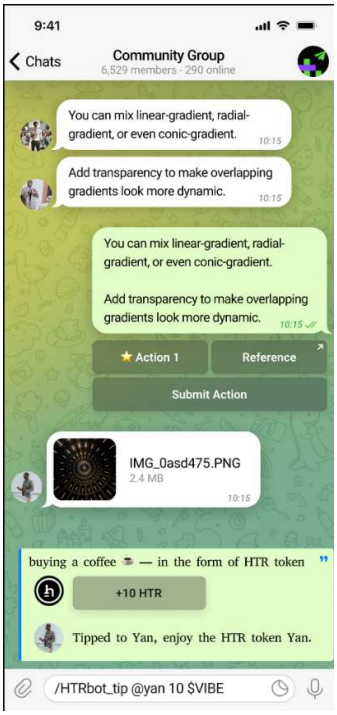
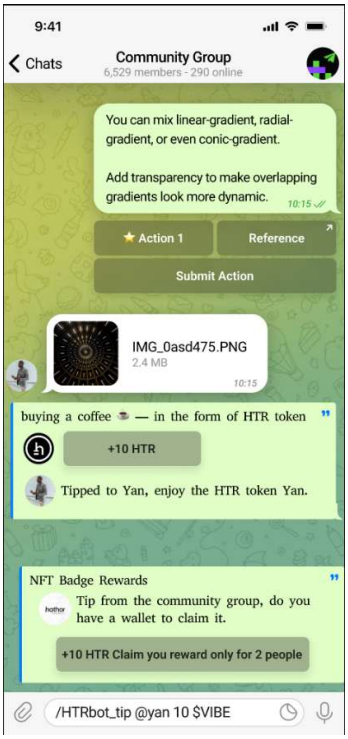
HathorChat: Tokenized Communities in Telegram

Telegram Mini App



1. Sending

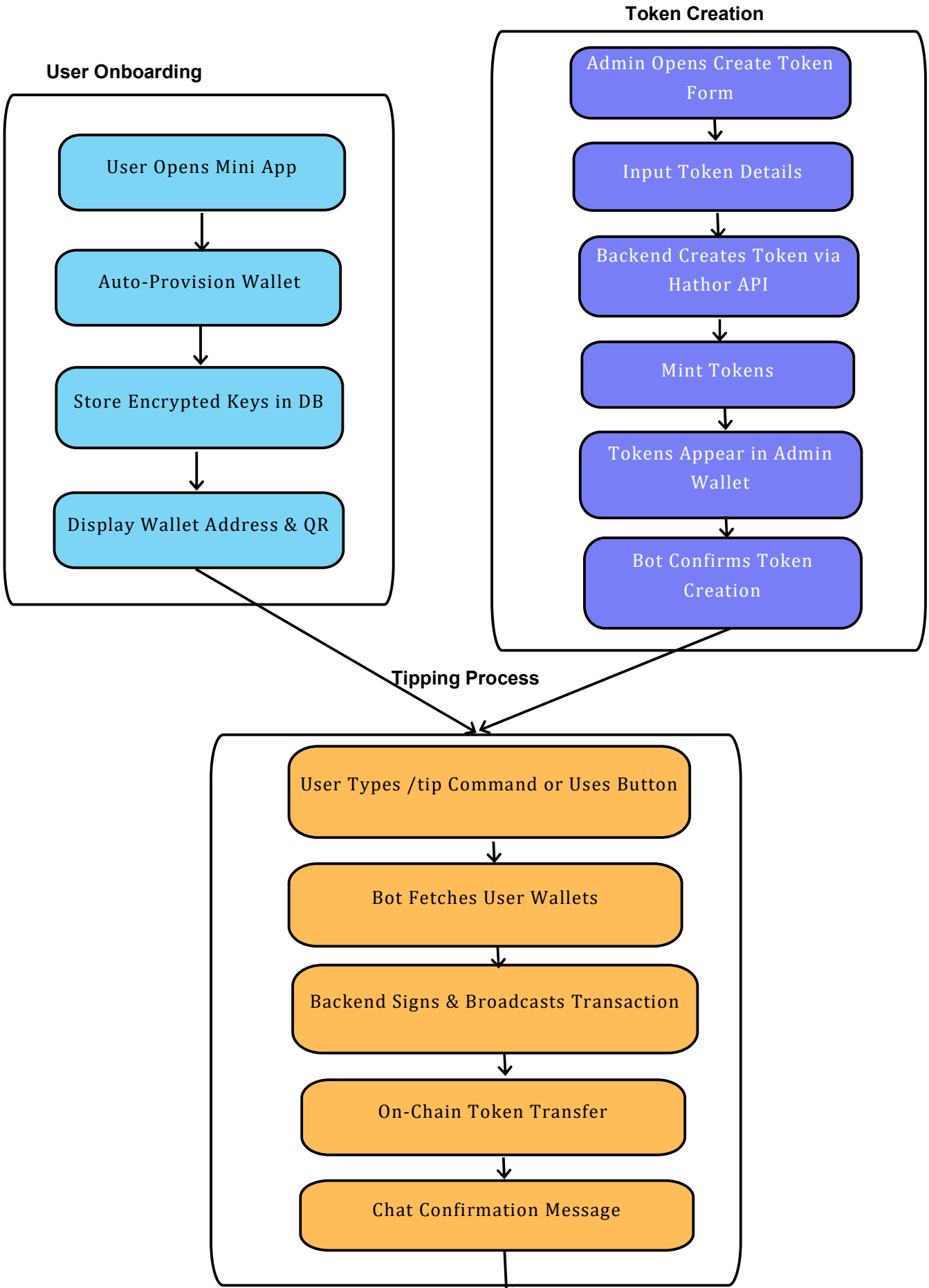
1. **User Action**: `/HTRbot_tip @yan 10 $VIBE` or click “Send” in UI
2. **Backend**:
    - Builds transaction with Hathor API (inputs: sender priv key, recipient address, token ID, amount)
    - Broadcasts to network
3. **Confirmation**: Bot replies “Tipped to Yan, enjoy the HTR token Yan. 🎉”

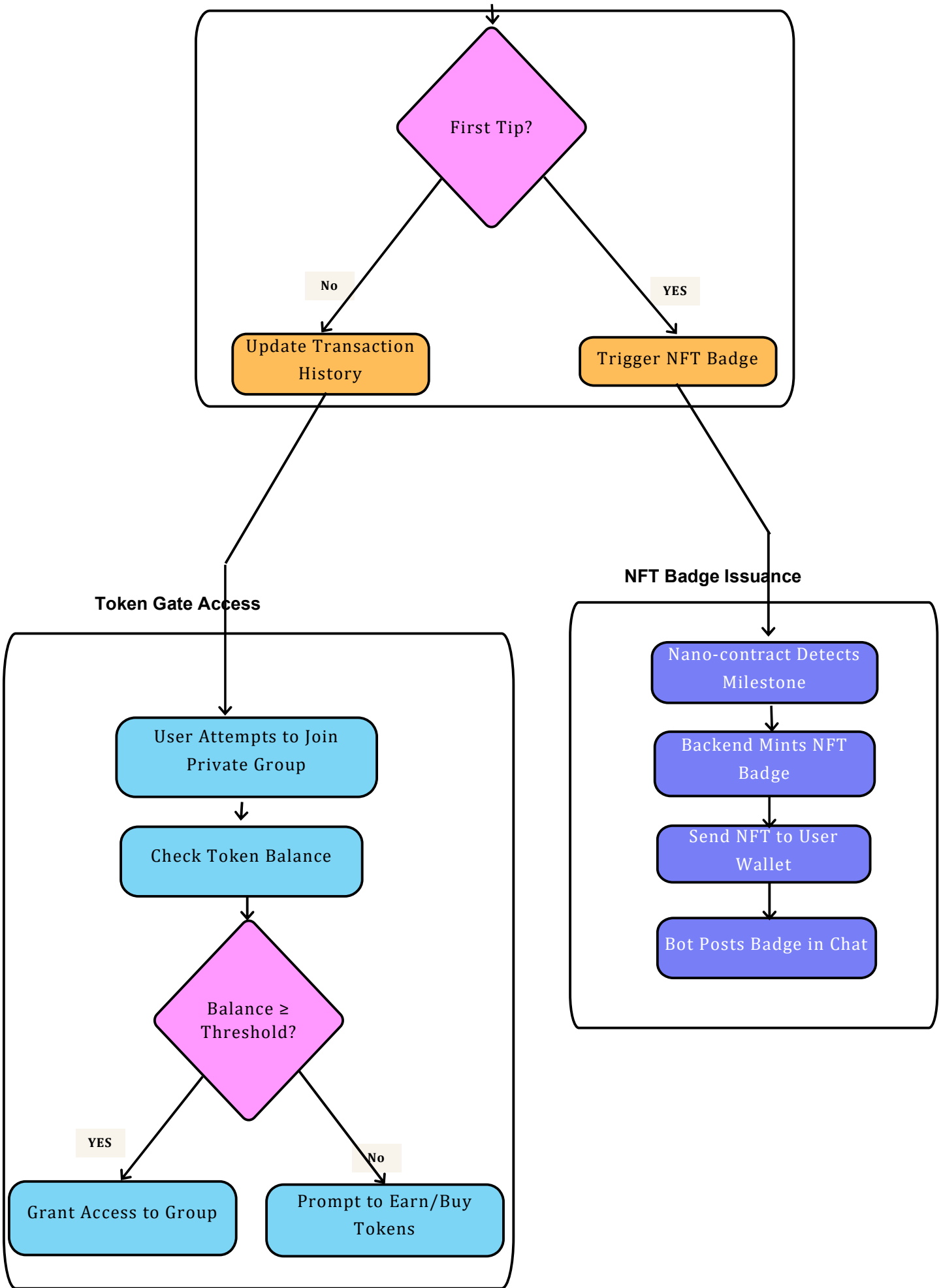


2. Tipping

1. **User Action**: `/HTRbot_tip @yan 10 $VIBE` or click “Send” in UI
2. **Backend**:
    - Builds transaction with Hathor API (inputs: sender priv key, recipient address, token ID, amount)
    - Broadcasts to network
3. **Confirmation**: Bot replies “Tip from the community group, do you have a wallet to claim it. 🎉”

HathorChat Flow Diagram.




First Tip?
No
YES
Update Transaction History
Trigger NFT Badge
Token Gate Access
User Attempts to Join Private Group
Check Token Balance
Balance ≥ Threshold?
YES
No
Grant Access to Group
Prompt to Earn/Buy Tokens
NFT Badge Issuance
Nano-contract Detects Milestone
Backend Mints NFT Badge
Send NFT to User Wallet
Bot Posts Badge in Chat

HathorChat: Tokenized Communities in Telegram

Frictionless tipping, rewards & gated access—Web3 made as easy as chat.

At HathorChat, we turn any Telegram group into a living, breathing token economy—no crypto expertise required. In a single Mini App, users get an auto-provisioned wallet, can mint or tip in native HTR or custom tokens, and earn on-chain NFT badges for participation. Communities unlock paid content or private channels simply by holding or earning tokens, while admins tap powerful engagement tools—leaderboards, rewards, access control—all running on Hathor's ultra-low-fee nano contracts. The result? Deeper engagement, new monetization streams, and a seamless Web3 experience inside the world's favorite chat platform.

Technical Feasibility

Frontend: React + Telegram Mini App SDK

Backend: Node.js/TypeScript + Hathor Wallet Library

Data layer: PostgreSQL (user profiles, encrypted keys, token metadata)

Blockchain:

- On-chain: token mint/transfer/NFT issuance
- Off-chain: nano-contract logic (leaderboards, thresholds)

Security & Key Management: Encryption, KMS, HTTPS + Telegram auth

🧩 Telegram Mini App + Bot Architecture in Detail

- **Frontend**:
  Built using **React + Telegram Mini App SDK**, the Mini App runs inside Telegram's native WebView. This means users don't leave the chat, and everything feels instant and familiar.
- **Bot Integration**:
  A connected Telegram Bot handles message-based commands like `/tip`, `/balance`, or `/create_token`, ensuring full conversational control alongside the Mini App UI.

Backend Logic

- **Node.js + TypeScript Backend**
  The backend handles:
    - Wallet creation (using the official Hathor Wallet Library)
    - Token minting and transfer logic
    - NFT badge minting and metadata management
    - Business logic (e.g., tip history, milestone triggers, gating rules)
- **Nano-Contracts (Off-Chain Rules Engine)**
  All rules—like “first tip → badge,” “hold 100 $VIBE → unlock chat”—are enforced off-chain using nano-contract logic written in Node. This allows full control with zero smart contract complexity.

Hathor Integration

- **Hathor Wallet Library (TypeScript)**
  Used server-side to generate user wallets, sign transactions, and interact with the network.
- **HTR Testnet**
  We develop on Hathor’s testnet for cost-free testing, then deploy to mainnet when ready.
- **Public API or Self-Hosted Node**
  Transactions are broadcast via either a hosted Hathor node or official API endpoints.

👥 Use Cases & User Journeys

HathorChat transforms regular Telegram users into active participants in token-based economies. Here's how it works under the hood — in clear, real-life examples.

🪙 1. Tipping as Social Currency

Use Case: A user tips someone for being helpful in a Telegram group.

Frontend (Mini App or Bot):

- User types `/tip @Alice 5 $VIBE` or uses the “Send Tip” button.
- Frontend collects recipient username, amount, and token type.

Backend:

- Resolves both users' wallet addresses from Telegram IDs via database.
- Decrypts sender's private key securely.
- Constructs and signs transaction using Hathor Wallet Library.

Blockchain:

- Transaction is broadcast to Hathor network via public API or self-hosted node.
- Network confirms token transfer from sender to recipient wallet.

Database:

- Logs the tip (sender, recipient, token, amount, timestamp) for analytics and user history.
- Triggers milestone check: “Was this the first tip?”

🎖 2. NFT Badge Reward (Milestone)

Use Case: A user earns a badge after sending their first tip.

Frontend:

- User receives an in-chat bot message:

  “🎉 You earned the *First Tip Sent* NFT badge!”

Backend:

- Nano-contract logic detects milestone triggered from previous tip.
- Calls Hathor’s NFT creation API using admin wallet (or badge contract wallet).
- Generates badge metadata (image, title, rarity) and signs mint transaction.

Blockchain:

- NFT is minted and sent directly to the user’s wallet address.

Database:

- Logs badge: {user_id, badge_id, timestamp, token_id}

- Updates “My Badges” gallery in the Mini App

💼 3. Creating a Custom Token

Use Case: A user wants to create a token called “CoffeeCoin” ($BREW) with a 10,000 supply.

Frontend:

- User opens “Create Token” form in the Mini App.
- Fills in: name, symbol, supply, and optional logo.

Backend:

- Validates input (no offensive names, supply limits, etc.).
- Uses admin wallet to call Hathor’s `createToken()` API.
- Mints full supply and transfers to the user’s wallet.

Blockchain:

- Token is minted as a fungible token on the Hathor network.
- Appears in the user’s wallet with its own token ID.

Database:

- Stores token metadata: name, symbol, supply, creator_id, logo URL
- Logs mint event under the user’s history
- Adds token to available tip/send options in Mini App

📊 4. Viewing Wallet & Transaction History

Use Case: A user wants to check how much HTR and custom tokens they hold.

Frontend:

- User taps “My Wallet” → “View Balance & History”

Backend:

- Calls `getWalletBalance(user_id)` and `getTransactionHistory(user_id)`
- Fetches and formats on-chain data

Blockchain:

- Queries real-time balances and transaction history using Hathor’s explorer API.

Database:

- Caches basic wallet info for quick display.
- Merges history with local logs (tips, mints, badges) for richer view.

Output:

- User sees full balance in HTR and all tokens
- Transaction history includes tips, rewards, and mints with timestamps

---