// SECURITY AND ARCHITECTURE

# Dappster technical overview

Architecture, transaction flows, security controls, and trust boundaries for dappster.fun

| VERSION | PUBLISHED | CANONICAL DOMAIN |
| --- | --- | --- |
| 1.0 | July 31, 2026 | https://dappster.fun |

**KEY CUSTODY STATEMENT**

Dappster does not request, collect, or store EVM or user Solana seed phrases and private keys. Users approve wallet connection, payments, credit burns, and deployment-related signatures in their own wallet.

01 / EXECUTIVE SUMMARY

## What Dappster does

Dappster is an AI-assisted application builder that generates smart-contract source, frontend source, deployment instructions, and optional automated audit reports. Users can review and preview generated artifacts before deployment.

On EVM networks, deployments remain non-custodial: the connected wallet signs and broadcasts contract creation. On Solana, a disclosed, user-funded technical wallet performs program deployment only after wallet-signed funding and authorization, then transfers program upgrade authority to the user.

02 / ARCHITECTURE

## System components

| WEB APPLICATION | GENERATION AND AUDIT |
| --- | --- |
| Next.js and TypeScript on Vercel. wagmi/viem and WalletConnect-compatible EVM connectors; Solana wallet adapter. | Server-side structured AI output. Restricted Solidity compilation and isolated Solana builds. |
| **DATA AND AUTHORIZATION** | **PUBLISHING** |
| Supabase authentication, linked wallets, Postgres Row Level Security, owner-scoped records, and idempotent transaction synchronization. | Verified frontends are packaged with deployment metadata, pinned through Pinata, and served from IPFS gateways. |

03 / EVM DEPLOYMENT

## User-signed contract creation

| | |
| --- | --- |
| 1 | The authenticated user selects a supported EVM network and requests a generated contract and frontend. |

| | |
|---|---|
| 2 | Dappster compiles Solidity with solc, permits bundled OpenZeppelin imports, and returns ABI and creation bytecode. |
| 3 | The exact contract-creation payload is simulated against the selected chain RPC and gas is estimated before the wallet prompt. |
| 4 | The connected wallet signs and submits deployment. Dappster never receives the wallet private key or seed phrase. |
| 5 | The backend verifies receipt success, contract address, contract-creation form, exact 0.001 ETH value, fee recipient, and emitted fee event. |
| 6 | Only after verification may the generated frontend be published to IPFS. |

| REQUIRED DEPLOYMENT VALUE | FEE RECIPIENT |
|---|---|
| **0.001 ETH** | `0x5D69C42A3a481d0CCFd88CFA8a2a08e2BF456134` |

The zero-argument payable constructor requires exactly 0.001 ETH, forwards it atomically to the disclosed recipient, and emits DappsterDeploymentFeePaid. If forwarding fails, deployment reverts. Network gas is separate.

04 / SUPPORTED EVM NETWORKS

# Explicit allowlist

| NETWORK | CHAIN ID | ENVIRONMENT |
|---|---|---|
| **Base** | `8453` | Mainnet |
| **Ethereum** | `1` | Mainnet |
| **Arbitrum One** | `42161` | Mainnet |
| **OP Mainnet** | `10` | Mainnet |
| **Linea** | `59144` | Mainnet |
| **Robinhood Chain** | `4663` | Mainnet |
| **Ethereum Sepolia** | `11155111` | Testnet |
| **Base Sepolia** | `84532` | Testnet |

05 / SOLANA DEPLOYMENT

# Authorized, user-funded relayer

| | |
|---|---|
| 1 | Dappster compiles the generated Anchor program and calculates rent and deployment costs. |
| 2 | The linked user wallet signs funding to the disclosed technical wallet with a unique deployment-job memo. |
| 3 | The backend verifies cluster, sender, recipient, minimum amount, memo, signature status, and a separate deployment authorization. |
| 4 | A job queue and cluster lock serialize deployment wallet usage. |

| | |
|---|---|
| 5 | The technical wallet deploys with Solana's Upgradeable Loader, verifies the executable account, and transfers upgrade authority to the user. |
| 6 | The frontend is published only after Program ID verification. |

The technical wallet is not used for EVM deployment. Mainnet SOL required for rent and deployment comes from the requesting user. Funding references cannot be reused across jobs.

06 / PAYMENTS AND CREDITS

# On-chain settlement on Base

- Credit packages and membership settle in native Circle USDC on Base.
- Production membership contract: 0xea7e37d45b6f75ae6826c1925d7b0ac314c7ecae.
- The backend credits an account only after verifying the successful receipt and exact USDC Transfer event.
- Credits are non-transferable ERC-1155 units. The normal flow asks the linked wallet to sign burnOwnCredits.
- Unique usage IDs and payment references make purchases and credit consumption idempotent.

07 / SECURITY CONTROLS

# Validation and access boundaries

- Schema-validated and size-limited API inputs; explicit chain allowlist; restricted Solidity imports.
- Wallet-to-account linkage for payment, credit, and deployment actions.
- Backend verification of chain, sender where applicable, destination, amount, success, events, and deployed address.
- Postgres Row Level Security for profiles, private projects, audits, transactions, and deployment jobs.
- Replay resistance using usage IDs, memos, job IDs, transaction references, and unique database constraints.
- Same-origin, bounded client-error telemetry that excludes wallet secrets.

08 / TRUST BOUNDARIES

# What users must still verify

AI-generated code can contain defects. Compilation, transaction simulation, and automated audit output do not replace an independent professional security audit. Users can inspect source before deployment and should test high-value applications on a test network first. Wallet simulation and independent security providers remain additional protection layers.

09 / INDEPENDENT VERIFICATION

# Public references

| | |
|---|---|
| **Production** | `https://dappster.fun` |
| **Technical overview** | `https://dappster.fun/technical-overview` |
| **Marketplace** | `https://dappster.fun/explore` |
| **Base membership contract** | `https://basescan.org/address/0xea7e37d45b6f75ae6826c1925d7b0ac314c7ecae` |
| **Deployment fee recipient** | `https://basescan.org/address/0x5D69C42A3a481d0CCFd88CFA8a2a08e2BF456134` |
| **Security contact** | `mikforlani@gmail.com` |