EvolveNP White Paper

An Autonomous, Rule-Based Fundraising Protocol for Nonprofits

Version: 2.0 | **Date:** January 2026 | **Network:** Ethereum (EVM-compatible support planned)

Table of Contents

1. **Motivation & Problem Statement**
2. **Design Principles**
3. **System Architecture (Overview)**
   3.1 Core Contracts & Actors
   3.2 Data Flows
   3.3 Execution Model (Permissionless)
   3.4 Break-Glass Emergency Architecture
4. **Launch Mechanics (Fair-Launch)**
5. **Components & Mechanics**
   5.1 FundRaisingToken (ERC-20)
   5.2 Hook (Impact Fee Routing)
   5.3 Vault (Swap + Distribution)
   5.4 EmergencyManager (NORMAL / ARMED / EMERGENCY_ACTIVE)
   5.5 IntegrationRegistry (Emergency-Only Uniswap Endpoints)
   5.6 Factory (Immutable Deployment)
6. **Flow of Funds**
7. **Security Model & Threat Considerations**
8. **Control & Upgrade Policy**
9. **Parameters (Immutable / Fixed at Deploy)**
10. **Compliance Posture (Informational, Not Legal Advice)**
11. **Observability & Public Reporting**
12. **Roadmap (Illustrative, Non-Binding)**
13. **Disclosures & Risk Factors**
14. **Glossary**
15. **Summary**

Developer Appendix

16. **Developer Appendix (Protocol & Audit Details)**
    16.1 Contracts in Scope
    16.2 Supporting Interfaces / Libraries
    16.3 Features & Risk Areas
    16.4 Upgradeability & Ownership (Summary)
    16.5 Timeline & Audit Goals

Abstract

EvolveNP is a fair-launch fundraising protocol that routes on-chain value to predefined beneficiary wallets through autonomous, rule-gated smart contracts. A small fixed **Impact Fee** (e.g., 1%) routes token inflows into a single **Vault** under deterministic rules, including a hard cap condition (e.g., the Impact Fee applies only while the Vault's token balance remains below a fixed threshold such as **30% of total supply**, after which it becomes zero by rule).

On a fixed cadence (e.g., monthly), the Vault can execute a **permissionless** "swap + distribution" event: it swaps a fixed percentage of its token holdings (e.g., **2%**) into **USDC** using allowlisted Uniswap endpoints, and distributes the resulting USDC to predefined beneficiary wallets using fixed splits. Execution is open to any caller and succeeds only when due and when predefined safety checks pass; otherwise it fails without state changes.

The protocol is designed to have **no normal-mode administrative control paths**: no discretionary pause/unpause, no parameter tuning, and no upgradeable proxy/beacon mechanisms. The only supported post-deploy change is a narrowly scoped, emergency-time repair mechanism for Uniswap integration endpoints via a single **IntegrationRegistry**, gated by a global **EmergencyManager** state machine and restricted to strict allowlists. All critical actions emit on-chain events for public auditability.

1. Motivation & Problem Statement

Modern fundraising is expensive, unpredictable, labor-intensive, and often opaque.

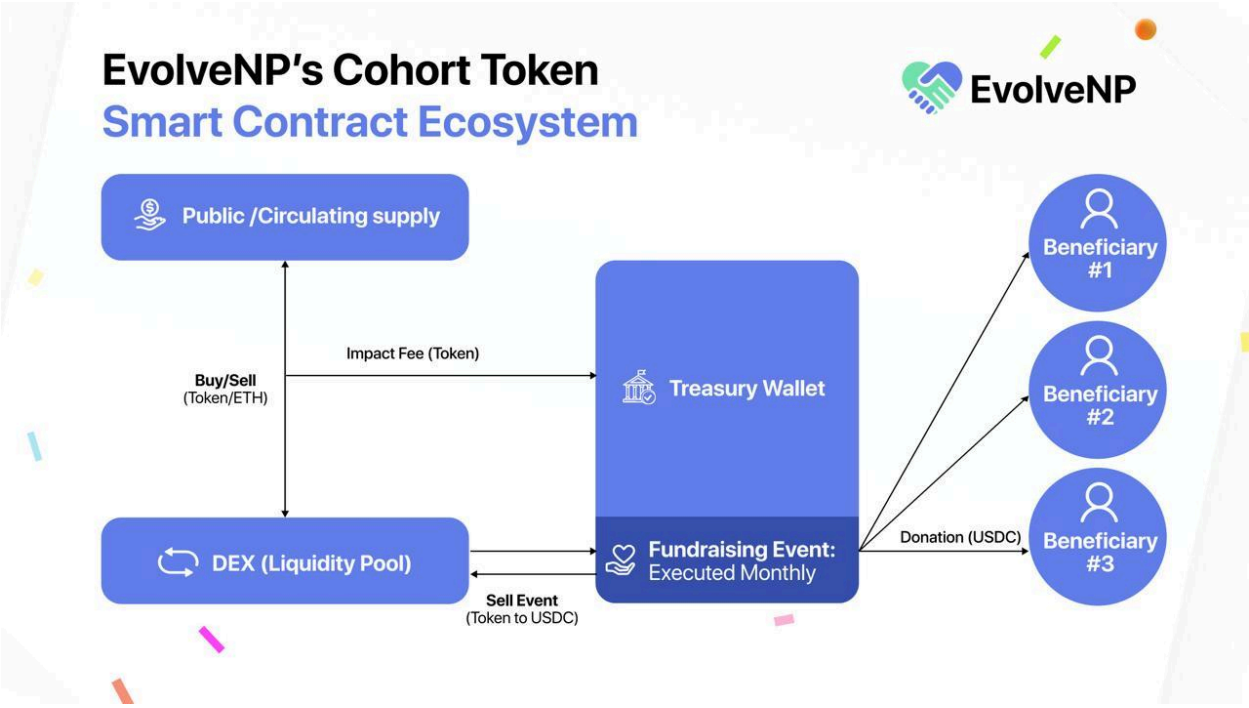
- **High cost to raise one dollar:** Intermediaries and processors add fees and overhead; a meaningful share of each dollar is consumed by acquisition, processing, and reconciliation.
- **Unpredictable capital flows:** Episodic campaigns create revenue spikes—not streams—hindering planning.
- **Heavy labor burden:** Each campaign requires fresh creative, coordination, and manual reporting; staff time shifts from mission to execution.
- **Transparency & trust gaps:** Supporters struggle to verify flows and outcomes; fragmented tools erode confidence.

Goal: Provide nonprofits and supporters with an automated, simple, and auditable mechanism where community activity routes value on-chain to **predefined beneficiary wallets** (typically nonprofit-controlled) — lowering effective fundraising overhead, smoothing inflows, improving end-to-end transparency, and reducing operational labor.

2. Design Principles

- **Non-custodial by design:** No privileged party can withdraw or redirect funds. The system distributes to predefined external recipient wallets by immutable rules.
- **Determinism over discretion:** In normal operation, there are no administrative controls to pause/unpause, tune parameters, or change routing/execution venues.
- **Permissionless execution:** Core operations are callable by anyone when due and safe; reliability does not depend on a privileged operator.
- **Fair launch:** No insider allocations, whitelists, or preferential access.
- **Minimized mutable surface:** No proxies/beacons. The only post-deploy mutable surface is emergency-only allowlisted Uniswap endpoint repair via a single registry.
- **Transparency:** Critical actions emit events and can be independently verified via on-chain data and public dashboards.

3. System Architecture (Overview)



3.1 Core Contracts & Actors

- **FundRaisingToken (ERC-20):** Standard token used for market activity and protocol routing.
- **Hook (Impact Fee Router):** Enforces the **Impact Fee** and routes token inflows to the Vault under deterministic rules, including a hard-coded cap rule (e.g., fee applies only while Vault < 30% of total supply).
- **Vault (Swap + Distribution):** Holds routed tokens and, on a fixed cadence, swaps a fixed percentage to USDC and distributes the swap proceeds to predefined beneficiary wallets under fixed splits.
- **EmergencyManager:** Global emergency state machine with modes **NORMAL**, **ARMED**, and **EMERGENCY_ACTIVE**, where EMERGENCY_ACTIVE is time-bounded and only reachable after objective on-chain arming.
- **IntegrationRegistry:** Stores the minimal set of Uniswap endpoints required for quoting/swapping; updatable **only during EMERGENCY_ACTIVE** and **only to allowlisted endpoints**.
- **Factory:** Deploys immutable instances of the protocol stack; does not retain runtime powers to tune economics or routing.

3.2 Data Flows

- DEX activity → Hook applies Impact Fee (when applicable) → tokens routed to Vault.
- Monthly (or interval-based): Vault executes swap + distribution if due and safe.
- Emergency: objective failures can arm EmergencyManager; only then can a short emergency window be opened to update allowlisted Uniswap endpoints via IntegrationRegistry.

3.3 Execution Model (Permissionless)

The monthly event is not dependent on Chainlink for correctness. Anyone can call execution when due and when safety checks pass. A monitoring bot can be used for reliability but has **no special permissions**.

3.4 Break-Glass Emergency Architecture

In normal operation, no privileged actions exist. If objective on-chain conditions indicate integration failure, the system can enter ARMED. Only then may a time-bounded EMERGENCY_ACTIVE window be opened to repair allowlisted Uniswap endpoints via IntegrationRegistry. No other parameters or behaviors can be changed post-deploy.

4. Launch Mechanics (Fair-Launch)

- **Initial Liquidity:** Seed a DEX pool with ~3–7 ETH (or equivalent) paired with tokens.
- **Initial Allocations: 75%** of supply to the public LP at launch; **25%** to the **Vault**. No private/insider allocation.
- **LP Receipts:** Burn (or permanently lock) launch LP receipts to reduce rug risk and increase trust.

Anti-Sniping (first ~1 hour):

- Short hold for the first **N blocks** after liquidity is added.
- **1-minute** per-wallet buy cooldown.
- **0.333%** of total supply max buy per transaction.

5. Components & Mechanics

5.1 FundRaisingToken (ERC-20)

- **Standard ERC-20:** The token itself is a non-upgradeable ERC-20 used for open-market activity on public DEXs.
- **Impact Fee enforcement (via Hook):** Any protocol fee logic is enforced externally via the **Hook** and routed to the **Vault** under deterministic rules (not via mutable token-owner controls).
- **No admin toggles:** The token contract does not include pause/unpause controls or admin switches to enable/disable the Impact Fee. In normal operation, transfers behave as a standard ERC-20, with any fee behavior applied at the protocol layer (Hook) when applicable.

5.2 Hook (Impact Fee Routing)

- Applies a fixed **Impact Fee** to qualifying activity per deterministic rules.
- **Hard cap rule:** Impact Fee applies only while the Vault's token balance is below a fixed threshold (e.g., **< 30% of total supply**).
- Once the threshold is reached, the Impact Fee becomes **zero by rule** until the balance falls below the threshold again.
- There is **no admin switch** to toggle the Impact Fee on/off.

5.3 Vault (Swap + Distribution)

- Receives token inflows from the Impact Fee.
- On a fixed cadence (e.g., monthly), swaps a fixed percentage of its token holdings (e.g., **2%**) to USDC using Uniswap endpoints sourced from IntegrationRegistry.
- Distributes the USDC produced by the swap to predefined beneficiary wallets using fixed splits.
- Execution is permissionless and succeeds only when due and when safety checks pass.

5.4 EmergencyManager (Global Emergency State Machine)

- **NORMAL:** default operation; no privileged actions exist.
- **ARMED:** objective on-chain conditions indicate integration failure risk.
- **EMERGENCY_ACTIVE:** time-bounded window where the only privileged action allowed is updating allowlisted Uniswap endpoints in IntegrationRegistry.

5.5 IntegrationRegistry (Emergency-Only Uniswap Endpoints)

- Stores minimal Uniswap endpoints used by protocol code paths (router/quoter/state view/permit2/pool manager as applicable).
- Updates are permitted only when EmergencyManager is **EMERGENCY_ACTIVE** and only to strict allowlists (and optionally allowlisted codehashes).
- No economic parameters are stored here.

5.6 Factory (Immutable Deployment)

- Deploys immutable instances wired to the same EmergencyManager and IntegrationRegistry.
- No beacon/proxy plumbing; no retained runtime powers to tune behavior.

6. Flow of Funds

1. DEX activity occurs.
2. Hook applies the Impact Fee when the Vault is below the hard cap threshold and routes tokens to the Vault.
3. When due, any caller may invoke the Vault's monthly execution function.
4. Vault checks time + safety conditions; if satisfied, it swaps a fixed percentage to USDC and distributes to predefined beneficiary wallets using fixed splits.
5. If conditions are not satisfied, execution fails and can be retried later.
6. Events are emitted for all critical actions for verification and reporting.

7. Security Model & Threat Considerations

- **Non-custodial by design:** There is no privileged key or admin function that can withdraw, seize, or redirect protocol funds. The Vault distributes value only through its fixed, rule-gated execution path to predefined beneficiary wallets.
- **No normal-mode administrative controls:** In **NORMAL** mode there are no pause/unpause controls, no parameter knobs, and no ability to change routing or execution behavior. Deployed contracts are immutable (no proxy/beacon upgrade mechanisms).
- **Narrow, break-glass-only repair surface:** The only supported post-deploy change is emergency-time replacement of **allowlisted Uniswap integration endpoints** via the **IntegrationRegistry**, permitted only during a **time-bounded EMERGENCY_ACTIVE** window governed by the EmergencyManager. All emergency transitions and registry updates emit on-chain events.

- **Permissionless execution (no privileged operator):** Monthly execution is callable by anyone when due and when safety checks pass. A monitoring bot can be used for reliability, but it has **no special permissions** and cannot change rules or funds flow.
- **Swap safety gating:** Monthly execution is protected by deterministic safety checks (e.g., TWAP/tick deviation bounds and/or other predefined price-integrity constraints). If checks fail, execution does not proceed and can be retried later.
- **Allowance & integration hardening:** Uniswap endpoints used for quoting/swapping are sourced from the IntegrationRegistry and restricted to strict allowlists (and optionally codehash allowlists) to prevent pointing to arbitrary contracts.
- **Launch integrity:** Launch LP receipts are burned or permanently locked to reduce rug vectors and increase trust.
- **Audits & bounties:** Code is verified on-chain; the protocol targets a third-party security audit and a standing bug bounty.
- **Incident transparency:** Emergency arming/activation and integration updates are publicly observable via on-chain events, with a published runbook describing emergency modes and response procedures.

Threat examples

- **DEX price manipulation during swaps:** Mitigated by deterministic safety checks (e.g., TWAP/tick deviation bounds), predefined constraints, and rule-gated execution that fails rather than forcing a swap in unsafe conditions.
- **Malicious "routing" / unauthorized recipients:** The Vault has no admin withdrawal path and distributes only to **predefined beneficiary wallets** with fixed splits; there is no runtime ability to redirect recipients.
- **Integration breakage (router/quoter/state-view changes or failures):** Objective failures can **ARM** the system; a short **EMERGENCY_ACTIVE** window enables emergency-time replacement of **allowlisted Uniswap endpoints** via the IntegrationRegistry, with on-chain events for auditability.
- **Execution/liveness failure (no one calls the monthly event):** Execution is permissionless — anyone can call it when due. A monitoring bot can improve reliability, but it has no special permissions; if it fails, others can execute instead.
- **Unexpected token/USDC transfers to the Vault:** (Optional line, only if you keep the delta accounting detail elsewhere) Direct deposits cannot change recipients or rules; distribution logic remains rule-gated.

8. Governance & Controls

Control & Upgrade Policy

- **No admin controls in normal operation:** Contracts run deterministically with no administrative controls to change parameters, routing, or execution behavior. **Permissionless execution:** Anyone can execute the monthly event when it's due and safety checks pass.
- **Emergency-only integration updates:** The only post-deploy change is emergency-time replacement of allowlisted Uniswap integration endpoints via the IntegrationRegistry, allowed only during a time-bounded emergency mode and recorded via on-chain events.

9. Parameters (Immutable / Fixed at Deploy)

- **Impact Fee:** fixed rate and deterministic application rules (including hard cap threshold, e.g., Vault < 30%).
- **Cadence:** fixed interval (e.g., 30 days).
- **Swap percentage:** fixed (e.g., 2% of Vault token holdings).
- **Beneficiary set + splits:** fixed at deployment (weights sum to total).
- **Safety checks:** fixed constraints (e.g., TWAP/tick deviation bounds; slippage constraints if applicable).
- **Emergency window duration:** fixed time-bounded EMERGENCY_ACTIVE length.
- **Allowlists:** predefined endpoint allowlists (and optionally codehash allowlists) for Uniswap integrations.

10. Compliance Posture (informational, not legal advice)

- **Open DEX trading**: no trader KYC on the protocol.
- **Rules-based execution:** swap + distribution is permissionless and rule-gated; no operator privileges required.
- **Control limits:** no normal-mode admin controls; only emergency-time allowlisted Uniswap endpoint repair is supported, and only within a time-bounded emergency window.

- **Transparency:** public addresses and event-based reporting enable independent verification.

Disclaimer: informational only; consult counsel as appropriate.

11. Observability & Public Reporting

- **On-chain events:** Impact Fee routing activity, Vault execution attempts (executed / skipped / reverted), swap executions (when due and safe), USDC distribution events to predefined beneficiary wallets, emergency state transitions (NORMAL → ARMED → EMERGENCY_ACTIVE), and IntegrationRegistry endpoint updates.
- **Public dashboard:** Vault inflows (Impact Fee), Vault token balance, monthly execution history (due vs executed vs failed), swap outputs (USDC produced), distribution totals by beneficiary wallet, and a full log of emergency events and integration updates — all derived from on-chain data.
- **Change transparency:** Core parameters are immutable in normal operation. Any emergency-time IntegrationRegistry updates are publicly visible via on-chain events and reflected in a public changelog.
- **Audit / bounty status:** Links to verified source code, third-party audit reports, issue disclosures, and the bug bounty program (when live).

12. Roadmap (Illustrative, Non-Binding)

- **Audit & launch readiness:** Third-party security audit, remediation, final spec freeze, and public documentation updates.
- **Cohort launches (multi-beneficiary first):** Deploy initial cohort pilots where a single Vault distributes to multiple predefined beneficiary wallets with fixed splits. Validate execution reliability, reporting, and recipient operations.
- **Scale onboarding & observability:** Expand partner onboarding, improve dashboards and event indexing, and harden monitoring and incident runbooks.
- **Single-beneficiary launches (then singular):** Deploy single-beneficiary instances using the same immutable architecture and reporting standards, informed by cohort pilot learnings.
- **Category index research (optional):** Publish methodology and design constraints for category index constructs prior to any deployment.

Note: Timing depends on audit outcomes, partner readiness, and market conditions.

13. Disclosures & Risk Factors

Smart-contract risk; market/volatility risk; liquidity risk; automation/liveness risk; operational/configuration risk; third-party dependency risk (AMMs, oracles); evolving regulatory risk. Audits/bounties mitigate but cannot eliminate risk.

14. Glossary

DEX: Decentralized exchange where tokens trade via smart contracts (e.g., Uniswap).

LP (Liquidity Pool): The pool of your token and a paired asset on a DEX that enables trading.

TWAP: Time-Weighted Average Price — a pricing reference used to reduce the impact of short-lived spikes/dips and help detect manipulated conditions.

Impact Fee: A small, fixed protocol-level fee applied under deterministic rules that routes tokens into the Vault. It cannot be manually toggled on/off.

Vault: The protocol contract that receives Impact Fee inflows and, on a fixed cadence, swaps a fixed percentage of its token balance to USDC and distributes it to predefined beneficiary wallets according to fixed splits.

Beneficiary Wallet: A predefined recipient address that receives USDC distributions from the Vault (commonly a nonprofit-controlled multisig). These wallets are external to the protocol.

USDC: A widely used U.S. dollar–denominated stablecoin used as the distribution asset.

Emergency Modes (NORMAL / ARMED / EMERGENCY_ACTIVE):

- **NORMAL:** Default operation. No admin controls to change parameters, routing, or behavior.
- **ARMED:** Objective on-chain signals indicate an integration issue; the system becomes eligible to open a break-glass window.
- **EMERGENCY_ACTIVE:** A short, time-bounded emergency window where the only privileged action allowed is repairing allowlisted Uniswap endpoints via the IntegrationRegistry.

EmergencyManager: The protocol's non-upgradeable emergency state machine that determines when break-glass mode is eligible/active, based on objective on-chain conditions.

IntegrationRegistry: The protocol's only mutable surface, used solely to store and (in emergency only) update allowlisted Uniswap integration endpoints used for quoting/swapping.

Allowlist: A predefined set of approved contract addresses (and optionally bytecode hashes) that endpoints must match before they can be used/updated.

On-chain Events: Logs emitted by contracts that make activity publicly verifiable (e.g., monthly execution, distributions, emergency transitions, integration updates).

Safe (Multisig): A wallet that requires multiple approvals to move funds (commonly called a Gnosis Safe).

15. Summary

EvolveNP combines fair-launch distribution and autonomous, rule-gated mechanics to route on-chain value from community activity to predefined beneficiary wallets (typically nonprofit-controlled). A small fixed **Impact Fee** routes tokens into a **Vault** under deterministic rules, and on a fixed cadence the Vault can be executed by anyone when due and when safety checks pass. When executed, the Vault swaps a fixed percentage of its token holdings into **USDC** and distributes the swap proceeds to predefined recipients using fixed splits. Every critical action is transparent through on-chain events and public reporting.

Disclaimer: *This white paper is for informational purposes only and does not constitute legal, tax, investment, or financial advice. Core parameters and behaviors are intended to be immutable in normal operation; the only supported post-deploy change is emergency-time replacement of allowlisted Uniswap integration endpoints within a time-bounded emergency window, recorded via on-chain events. Recipient wallets are external to the protocol. All actions are subject to contract code and prevailing network conditions.*

16. Developer Appendix

16.1 Contracts in Scope (estimated LOC; provide exact counts from repo)

FundRaisingToken (ERC-20, ~200–300 LOC)

- Standard ERC-20 used for open-market activity.
- **No embedded "admin fee toggle" or pause/unpause paths** required for protocol operation.
- Any protocol fee behavior is enforced externally (via **Hook**) rather than through mutable token-owner controls.
- Non-upgradeable immutable instance per deployment.

Hook (Impact Fee Router, ~200–350 LOC)

- Applies the **Impact Fee** under deterministic rules and routes token inflows to the **Vault**.
- **Hard cap rule (example):** the Impact Fee applies only while the Vault's token balance is below a fixed threshold (e.g., **< 30% of total supply**); once reached, fee becomes **zero by rule** until back below the threshold.
- No admin switch to toggle the Impact Fee on/off.
- If the Hook requires Uniswap integration touchpoints (e.g., state view/quoter), it must read endpoints from **IntegrationRegistry** (no mutable local endpoints).

TreasuryDistributionVault / Vault (Swap + Distribution, ~300–500 LOC)

- Receives token inflows from the Hook (Impact Fee routing).
- Implements a **permissionless**, rule-gated monthly (interval-based) execution function:
    - Callable by anyone when due and safe.
    - Swaps a **fixed percentage** of Vault token holdings (e.g., **2%**) to **USDC** using Uniswap endpoints from IntegrationRegistry.
    - Distributes swap proceeds to **predefined beneficiary wallets** using fixed splits.
- Deterministic safety gates (e.g., TWAP/tick deviation bounds and/or min-out constraints as implemented).
- No admin withdrawal path and no ability to change recipients, splits, cadence, or swap percent post-deploy.

EmergencyManager (Global Emergency State Machine, ~250–450 LOC)

- Single global emergency controller with modes:
    - **NORMAL** (default): no privileged actions exist.
    - **ARMED**: objective on-chain failure conditions detected (e.g., repeated swap/quote failures, liveness failure, endpoint integrity failure, oracle invalid/stale data if used).
    - **EMERGENCY_ACTIVE**: time-bounded emergency window that can only be activated from ARMED.
- Emits structured events for arming, activation, and exit.

IntegrationRegistry (Emergency-Only Uniswap Endpoints, ~150–300 LOC)

- Stores the minimal set of Uniswap endpoints required by the protocol (router/quoter/state view/permit2/pool manager as applicable).
- **Only mutable surface** in the system.
- Endpoint updates are permitted **only** when `EmergencyManager.isEmergencyActive() == true`.
- Enforces strict allowlists (address allowlist and optionally codehash allowlist).
- Emits events for every endpoint update.

Factory (Immutable Deployment + Wiring, ~200–350 LOC)

- Deploys immutable instances of the protocol stack per deployment (Token + Hook + Vault), and wires shared references to EmergencyManager + IntegrationRegistry.
- No beacon/proxy deployment pathways.
- After initialization, the factory should not retain powers that change runtime behavior (i.e., “freeze semantics”).

16.2 Supporting Interfaces / Libraries

Interfaces

- **IERC20** (OpenZeppelin) — standard token interface.
- **Uniswap endpoints (as actually used):** e.g., router/quoter/state view/pool manager/permit2 (exact set depends on your v4 integration).
- **Protocol interfaces:** IHook, IVault, IEmergencyManager, IIntegrationRegistry, IFactory (as applicable).

Libraries / External Contracts

- Uniswap v4 types/helpers used by your code paths (PoolKey, Currency, StateLibrary, etc., as applicable).
- Safety/math helpers for TWAP/tick deviation and/or slippage/min-out enforcement (implementation-dependent).
- Reentrancy guard and safe transfer helpers.
- Tooling: forge-std/console (tests/dev only; not deployed).

Out of Scope (unless requested)

- Uniswap v4 core/periphery, Permit2, Safe implementation, and standard OpenZeppelin libraries (treated as vetted dependencies).

16.3 Features & Risk Areas

Features / behaviors

- **Permissionless monthly execution:** anyone can call execution when due; succeeds only when deterministic safety checks pass.
- **Impact Fee routing:** Hook routes inflows to Vault under hard-coded threshold rules (no toggles).
- **Fixed distributions:** Vault distributes swap proceeds to predefined recipients with fixed splits.
- **Break-glass integration repair:** EmergencyManager + IntegrationRegistry enable emergency-only allowlisted Uniswap endpoint updates within a time-bounded window.

Risk surfaces

- **No-privilege guarantees in NORMAL mode:** prove there are no callable admin actions to pause/unpause, change parameters, change recipients, or update endpoints in NORMAL/ARMED.
- **Emergency correctness:** objective arming conditions, counter thresholds, expiration, and activation gating (ARMED → EMERGENCY_ACTIVE only).
- **Allowlist enforcement:** IntegrationRegistry must prevent non-allowlisted endpoints (and optionally enforce codehash allowlists).
- **Swap safety correctness:** TWAP/tick deviation logic and/or min-out constraints; verify these checks cannot be bypassed and fail safely.

- **Monthly execution correctness:** due gating, idempotency, reentrancy protection, failure handling (revert/no partial state).
- **Distribution correctness:** split math, rounding, total conservation, event integrity.
- **Endpoint integrity checks:** extcodesize (and optional extcodehash) validation for endpoints.
- **MEV/slippage exposure:** evaluate swap pathways under volatile conditions; confirm deterministic guards and caps as implemented.
- **Allowance hygiene:** approval scope and deadlines (especially if Permit2 is used); avoid lingering infinite approvals.

16.4 Upgradeability & Ownership

Upgrade pattern

- **No proxies/beacons.** All deployed contracts are immutable instances (no implementation swap mechanism).
- The protocol's only mutable surface is **IntegrationRegistry endpoint updates**, gated to emergency mode and strict allowlists.

Runtime permissions (enforced on-chain)

- **NORMAL:** no privileged actions exist (no admin toggles, no endpoint updates, no parameter changes).
- **ARMED:** still no privileged actions (arming is objective; eligibility only).
- **EMERGENCY_ACTIVE:** the **only** privileged action is updating allowlisted Uniswap endpoints in IntegrationRegistry (event-logged and time-bounded).

Changing core behavior requires

- **Redeploy + migrate** to a new immutable instance set (conservative posture), rather than upgrading logic in-place.