NFA, Collaborative Intelligence

An agent swarm simulation engine for prediction market traders, with a decentralized marketplace of scenarios ranked and rewarded on forecasting accuracy.

Abstract

NFA is an agent swarm simulation engine that forecasts outcomes for prediction markets, powered by a decentralized marketplace of scenarios. Users submit a Polymarket or Kalshi market URL and receive a probability estimate produced by a swarm of AI agents simulating the actors, dynamics, and pressures that determine the market's resolution. Scenarios, reusable simulation templates published by domain experts, are the core primitive of the platform. The best scenarios, judged by historical accuracy against resolved markets, earn continuous royalties as traders use them.

The product addresses a specific gap in the prediction-market ecosystem: traders on Polymarket, Kalshi, and similar venues have access to price feeds, historical data, and news, but lack structured tools for forecasting outcomes driven by actor behavior under pressure: geopolitical events, regulatory decisions, policy outcomes, narrative dynamics, and behavioral markets. This class of markets represents 30 to 40 percent of active prediction-market inventory by count and a higher share of volume during periods of crisis, elections, and major news events. NFA serves this gap.

Three design principles shape the architecture. **First**, accuracy is the single metric that matters: all product, economic, and incentive decisions optimize for measurable forecasting accuracy. **Second**, the engine is production-grade from day one. Deterministic execution with full replay, per-call cost reconciliation, per-purpose LLM routing, prompt caching, and within-round parallelism are architectural primitives, not optimizations. **Third**, economic alignment is continuous: scenario authors earn royalties whenever their work is used, weighted by real-world accuracy, creating a compounding incentive for quality contribution.

NFA launches on **Solana**. The token, author settlement, and vesting are deployed as audited on-chain Anchor programs; the simulation engine is a proprietary stack of permissively licensed open-source components glued together by a custom orchestration layer.

This whitepaper describes the product, the technical architecture, the marketplace mechanics, the token model, and the path to launch.

1 · Problem

1.1 The prediction market category is growing fast

Prediction markets have matured from fringe experiments into a significant category of financial infrastructure. Polymarket crossed billions in volume during the 2024 US election cycle. Kalshi secured CFTC-regulated status and expanded across politics, economics, and weather. New markets continue launching around cultural events, crypto outcomes, corporate actions, and geopolitical developments. Both retail and institutional trading activity has grown consistently.

The category works because prediction markets aggregate information efficiently. Prices reflect the collective best estimate of thousands of participants, often outperforming polls, expert forecasts, and traditional surveys. But the efficiency of market-based aggregation has a ceiling set by the quality of information available to participants. Where participants have access to strong tools, markets price efficiently. Where participants rely on intuition, markets reflect that limitation.

1.2 Traders lack proper tools for actor-driven forecasting

Prediction markets broadly fall into three categories by what determines their resolution:

- **Statistical markets** where historical data and quantitative models produce strong forecasts: sports outcomes, base-rate questions, weather. Traders have mature tools (ELO ratings, xG models, weather models, actuarial data).
- **Microstructure markets** where price action over short horizons determines resolution: crypto price movements on 5-minute or hourly intervals. Traders use on-chain analytics, orderbook data, and funding rates.
- **Actor-driven markets** where outcomes depend on how specific actors behave under specific pressures: geopolitical events, regulatory decisions, policy outcomes, corporate actions, behavioral markets involving specific individuals or organizations. Traders have essentially no structured tools. They rely on news reading, intuition, and crowd-sourced takes on social media.

The third category is where NFA operates. Actor-driven markets constitute roughly a third of active prediction-market inventory and a higher share of volume during crisis periods (elections, wars, scandals, regulatory cycles). Traders in these markets know that quantitative tools built for statistical or microstructure markets don't apply. They need a different class of tool entirely.

1.3 Why existing approaches fall short

Expert panels and forecasting tournaments. Services like Good Judgment Project have demonstrated that trained human forecasters can outperform intelligence agencies on specific question classes. But expert panels are slow, expensive, and don't scale. They cannot be applied in real-time to the thousands of markets active on Polymarket or Kalshi at any given time.

Single-LLM forecasters. Direct LLM prompting ("will this peace deal happen?") produces shallow, unreliable forecasts. Single models lack the structured reasoning required for multi-actor dynamics, don't maintain consistent mental models of adversarial incentives, and hallucinate specifics that undermine credibility. LLMs are necessary but insufficient.

News aggregators and sentiment trackers. Tools that aggregate news coverage or track social sentiment provide useful signal but don't produce forecasts. A trader still has to translate "sentiment is 65 percent negative" into "probability is X percent." The translation is the hard part, and it's exactly what traders need help with.

Generic multi-agent frameworks. Running multiple LLM agents against a question improves on single-LLM forecasting but produces inconsistent results without structured simulation methodology. Without structured dynamics, action spaces, and resolution mappings, and without a market-specific cast grounded in the actual people whose decisions move the outcome, multi-agent outputs are arbitrary. They are not comparable across runs or auditable for accuracy.

1.4 The opportunity

A structured agent swarm simulation engine, designed specifically for actor-driven forecasting, with published accuracy data and a marketplace of domain-expert-built scenarios, fills a real gap. No incumbent occupies this position. The distribution channel (MCP, prediction-market frontends) is open. The technical substrate has matured to production-grade in the last 18 months. The audience is measurable and growing. The timing is right.

2 · Solution

2.1 Agent swarm simulation

NFA's core engine is an agent swarm simulator. Given a scenario and a set of initial conditions, the engine instantiates a population of AI agents representing the actors relevant to the forecasting question. Each agent has a defined role, personality, incentives, and memory. Agents interact over simulated time, exchanging messages, updating beliefs, reacting to events, and producing observable behaviors. The simulation runs for a configurable number of rounds, then outputs a probability distribution over possible outcomes.

This approach reflects a decade of research in multi-agent systems, grounded in the observation that complex real-world outcomes emerge from interactions between heterogeneous actors with different goals, information, and constraints. Single-model forecasting collapses this complexity into a single point estimate. Swarm simulation preserves it and extracts probability distributions from the resulting dynamics.

The engine is general-purpose. It does not know about Polymarket or Kalshi. It does not know about specific geopolitical situations. It simulates whatever scenario it is given, according to whatever dynamics the scenario specifies. Specialization happens in the scenario layer.

2.2 Scenarios as the publishable primitive

A **scenario** is a reusable behavioral template. It describes the rules of a recurring kind of situation, a negotiation, a regulatory vote, a central bank decision, a corporate event, without naming who plays. It carries three things:

- **Dynamics**: how actors interact under pressure, what trade-offs they make, how positions evolve across rounds.
- **Action space**: what actions any actor in this kind of situation can take, with what costs and constraints.
- **Resolution mapping**: how the final simulation state translates into a market-specific probability.

The cast is discovered per run from the specific market. A "Will the Fed cut rates in March 2026" market gets the current FOMC voting members, the named senators on Banking, the Treasury Secretary in office, the people whose decisions actually move that outcome, sourced from public records at run time. The same central-bank-decision scenario applied to a different Fed market gets a different cast; applied to the ECB gets a different cast still. Same template, different actors every time. This is what makes a scenario a durable asset: it compounds in value across every market it is applied to, even as the world changes underneath it.

Scenarios are authored by domain experts. A Middle East policy analyst builds a negotiation scenario. A Fed watcher builds an FOMC-decision scenario. A crypto researcher builds a depeg-propagation scenario. When a user submits a market URL, the system matches the market to

appropriate scenario templates from the marketplace, researches the specific market to discover the relevant cast, and runs the simulation with those discovered actors plugged into the scenario's dynamics. Cast discovery is handled by an intermediate LLM layer that parses market metadata, executes targeted web research, and produces a market-specific cast and initial-conditions bundle.

2.3 Marketplace structure

The scenario marketplace is open to any author. Publishing a scenario costs a flat **$10 in $NFA**, paid in token and burned on publication, a deliberate anti-spam floor that keeps the marketplace free of throwaway listings without gatekeeping who may contribute. Once published, a scenario earns on every run it serves, priced by its measured forecasting skill on resolved markets (§5.1).

This creates a structural property: supply of scenarios grows automatically with trader activity. Every active user is a potential contributor. The best scenarios, judged transparently on accuracy, rise to the top of recommendations. Poor scenarios stay in the long tail, available but rarely surfaced. Quality emerges through competition rather than through central curation.

2.4 Distribution

NFA reaches users through three surfaces:

- **Web frontend** at `nfa.club`. Retail traders paste market URLs, run simulations, browse scenarios, publish their own. The primary consumer surface.
- **MCP server.** Machine-facing API. Trading agents, algorithmic desks, and AI assistants like Claude and ChatGPT consume NFA programmatically. MCP is the standardizing protocol for AI agent tooling; early positioning in public registries is a durable distribution advantage.
- **Authoring interface.** Where scenarios are built. Includes an AI copilot that helps non-expert users build structured scenarios by suggesting dynamics, action spaces, and resolution mappings appropriate to the kind of situation being modeled.

3 · Product

3.1 The user flow

A trader's primary interaction with NFA begins with pasting a Polymarket or Kalshi market URL into the web frontend or passing it to the MCP server. The system immediately fetches market metadata, analyzes the market to determine the appropriate category, retrieves candidate scenarios from the marketplace ranked by accuracy, and presents the user with three paths forward.

Path 1: top three scenarios ranked by accuracy. The most relevant scenarios from the marketplace are displayed, ranked by historical accuracy on markets similar to the one submitted. The top-ranked scenario is pre-selected as the default. Each option shows the scenario name, author identifier, overall accuracy, accuracy on the relevant market category, and number of prior uses. A single click runs the simulation. This is the zero-friction path.

Path 2: write your own scenario. Users who want to build their own scenario select this path. The authoring interface opens, optionally pre-filled from a top-ranked scenario as a starting point. An AI copilot assists by suggesting dynamics, action spaces, and resolution mappings appropriate to the kind of situation being modeled. The user customizes the scenario, publishes it to the marketplace for a flat $10 in $NFA (burned on publication), and runs it. This path is how the marketplace grows supply organically.

Path 3: be the first to write a scenario. For markets in categories where the marketplace has no relevant scenario, this path is shown prominently. Framing coverage gaps as opportunity rather than deficiency encourages contributors to fill uncovered categories. Early scenarios in new categories accumulate accuracy data fastest and often become dominant references as the category matures.

3.2 Simulation output

When a simulation runs, the output includes:

- **Probability estimate** for each market outcome, with confidence bands reflecting simulation variance.
- **Reasoning trace** showing how individual agents contributed to the consensus, what information each used, and how their positions evolved across simulation rounds.
- **Divergence summary** comparing NFA's probability to the current market price, highlighting where the simulation disagrees with market consensus and by how much.
- **Accuracy context** showing the scenario's historical accuracy on similar markets, giving the user calibrated confidence in the output.

> ▸ The reasoning trace is a critical product feature. Traders do not trust black-box forecasts. An auditable trace lets users evaluate not just the final probability but the quality of the reasoning that produced it.

Variance bundles

Standard simulations execute the scenario once and report a probability with confidence bands derived from in-run variance. Variance-bundle simulations execute the scenario multiple times with different deterministic seeds and report a probability with explicit robustness bands across runs.

The variance-bundle output reports four robustness dimensions: **action-pattern robustness** (do the same action sequences emerge across runs), **coalition robustness** (do the same coalition formations emerge), **pivot robustness** (do the same narrative shifts trigger), and **stance robustness** (do agents converge to the same final positions). Outlier runs that diverge significantly from the bundle median are flagged separately, and the resolution-mapping consistency across runs is reported.

A scenario reporting *"35% YES with 90% of runs landing within ±3 percentage points and consistent coalition formation"* is materially more useful than a single 35% point estimate. Variance-bundle simulations are unlocked by staking $NFA.

3.3 The frontend

Live market gallery. A continuously updated feed of active Polymarket and Kalshi markets that have corresponding NFA simulations. Each entry shows the current market price, NFA's simulated probability, and the divergence between them. Sortable by divergence magnitude, scenario accuracy, recent activity, or market volume. This is the content engine, every large divergence is a shareable piece of content.

Simulation runner. The URL-paste interface and three-option flow. Simulations stream results in real-time, showing agents reasoning as the swarm runs. Final output includes the probability estimate, reasoning trace, divergence summary, and accuracy context.

Marketplace browser. Scenarios browseable by category, author, accuracy, usage volume, and recent activity. Each scenario has its own page showing its full accuracy history, markets it has been applied to, and detailed breakdown by market category. Top authors have reputation pages showing their portfolio of scenarios and aggregate earnings.

Author hub. Where contributors manage their scenarios, view earnings, tune performance, and publish new work. Earnings dashboards show royalty accrual in real-time. Accuracy dashboards show how each scenario is performing across recent resolved markets.

3.4 The MCP server

For programmatic consumption, NFA exposes an MCP (Model Context Protocol) server. Trading agents, AI assistants, and custom bots integrate via standardized tool calls. The server is listed in public MCP registries.

Exposed tools include:

```
run_simulation(market_url, scenario_id)         // run a specific scenario
forecast_market(market_url, auto_select=true)   // best-fit scenario
list_scenarios(category, min_accuracy)          // filter the marketplace
get_scenario_accuracy(scenario_id, category)    // detailed track record
compare_markets(market_urls)                    // run across related markets
explain_consensus(simulation_id)                // agent-level reasoning trace, replayed
```

Access is billed in USDC. Trading agents that integrate NFA pay per simulation at the same run price, compute fee plus scenario fee, as the web frontend.

3.5 The authoring experience

Any thoughtful trader with domain knowledge can build a functional scenario in 15 to 30 minutes, without simulation-theory expertise. The AI copilot handles the structural scaffolding. The human contributor provides the domain knowledge.

```
Scenario:      [name]
Category:      [market category]
Template type: [e.g. regulatory vote / central bank decision /
               negotiation / corporate event / electoral race]

Dynamics:
  - [how actors in this kind of situation interact]
  - [what pressures move their positions]
  - [how rounds progress and when they end]

Action space:
  - [what any actor can do at each round]
  - [costs, constraints, signaling effects]

Resolution mapping:
  - [how the final simulation state translates to market probability]

# Cast is discovered per run by the binding layer from
# the specific market context. The author does not name actors.
```

Scenarios can also be published privately. Private scenarios are not visible in the marketplace, earn nothing, and are usable only by their author. Private authors pay only the compute fee on their own

runs, with no scenario fee, supporting professional traders who use NFA as a personal forecasting tool without revealing methodology.

4 · Technical architecture

4.1 Stack overview

NFA is built on a stack of permissively licensed open-source components **plus a proprietary simulation engine**, each selected for production maturity and alignment with the product's specific requirements.

FIG. 1 · STACK ARCHITECTURE · PROPRIETARY ENGINE BOUNDARY IN GREEN

- **Application, Next.js.** Frontend, API routes, MCP server. Apache 2.0. Open source.
- **Settlement, Anchor (Solana).** Token, author settlement, buy-and-burn, and vesting deployed as on-chain programs on Solana. Apache 2.0. Open source. Audited.
- **Binding, custom LLM layer.** Parses market metadata, performs guided research, customizes scenarios at runtime. Proprietary.
- **Coordination, LangGraph.** Workflow orchestration. Built-in checkpointing, streaming, time-travel debugging. MIT.
- **Agent swarm, CAMEL-AI.** Multi-agent substrate. NeurIPS 2023 pedigree. Apache 2.0.

- **Memory, Graphiti.** Temporal knowledge graph. Bitemporal model handles time-aware reasoning. Apache 2.0.
- **LLM gateway, LiteLLM.** Provider abstraction with cost tracking and per-purpose routing. MIT.
- **Inference, OpenRouter.** LLM routing across multiple providers. Cost optimization and provider-agnostic.
- **Data, Postgres + pgvector.** Structured data, scenarios, accuracy history, earnings, embeddings. Neon hosted.
- **Durability, Temporal (Phase 2).** Long-horizon workflow orchestration for market resolution tracking. MIT.
- **Infrastructure, Vercel + R2.** Edge-deployed application layer with object storage for simulation artifacts.

> ▸ The simulation engine itself, the binding layer, agent orchestration, memory substrate, determinism layer, and cost tracking, is **proprietary**. The contracts, MCP protocol implementation, and frontend are **open-source**. This split provides the auditability traders need where it matters (settlement math, royalty calculations, protocol integration) while protecting the engine that produces forecasting accuracy as competitive moat.

4.2 Determinism and replay

Simulations are bit-for-bit reproducible where the underlying providers support it. Every run carries a stable seed context, pinned model versions per call class, and a captured trace that can be replayed end-to-end. The marketplace economics depend on this: when an author disputes their accuracy score, the platform must replay the exact run and produce an identical result. Without genuine determinism, accuracy scores would be advisory rather than authoritative.

4.3 Per-purpose LLM routing

LLM calls within a simulation serve different purposes with different cost-quality elasticity. NFA routes each call class to the appropriate model tier rather than running everything on a single model, yielding a meaningful cost reduction at no measurable quality loss versus all-frontier execution. Routing is configurable per simulation; advanced traders can opt into all-frontier routing for marginally higher accuracy at higher compute cost. Authors can specify routing requirements for scenarios where specific call classes need frontier models for the scenario to function correctly.

4.4 Prompt caching

A typical simulation reuses the same scenario context, agent personalities, dynamics, and instructions across hundreds of LLM calls. Prompt caching exploits this structurally on every supported provider, substantially reducing both input-token cost and time-to-first-token after the

first call in a window. Combined with per-purpose routing, this is the difference between optimized and unoptimized engine economics.

4.5 Within-round parallelism

Simulation rounds proceed sequentially: round N+1 cannot begin until round N completes. Within a round, however, agent reasoning is parallel by design, per-agent action proposals, plausibility validation, scoring, reflection, and coalition detection are dispatched concurrently with bounded provider concurrency. Rate-limit and transient-failure handling are first-class. The net effect is the difference between simulations that finish in tens of minutes and ones that take hours.

4.6 Cost record persistence and reconciliation

Every LLM call emits a cost record with full provenance, provider, model, call class, and the source from which the cost was computed. A reconciliation job runs on a regular cadence comparing engine-reported simulation totals against provider billing API records, with any discrepancy flagged for manual review before author payouts settle. This is essential for marketplace integrity: the compute fee and author settlement are both derived from metered model cost, which must accurately reflect actual spend.

4.7 Adversarial defense via plausibility validation

Every agent action passes through an independent plausibility validation step before being accepted into the simulation. The validator catches institutionally implausible actions (a regulator taking actions outside their mandate), contradictions with prior agent commitments, scenario-constraint violations, and patterns that suggest manipulation rather than honest forecasting (e.g., agents converging on author-preferred outcomes through implausible means).

Plausibility validation is intentionally routed to frontier models because its accuracy directly determines simulation integrity. The cost of frontier-model plausibility is justified by the marketplace integrity it protects.

4.8 Data flow

A simulation proceeds through six stages.

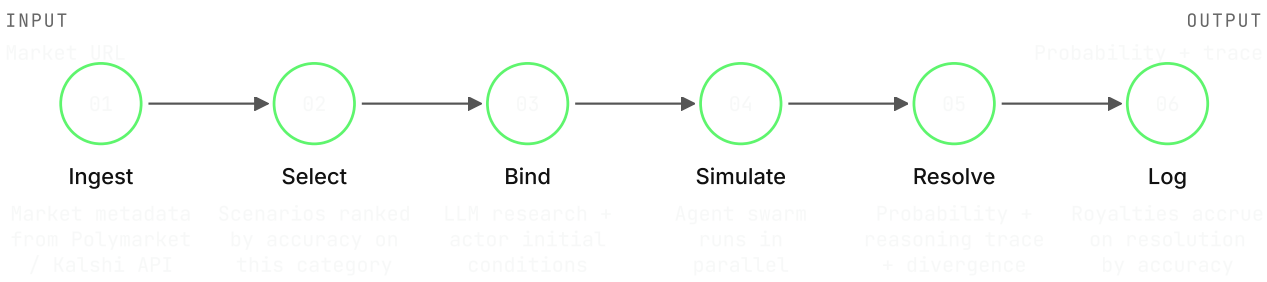

FIG. 2 · SIMULATION DATA FLOW · SIX STAGES FROM URL TO RESOLUTION

Stage 1: Market ingestion. User submits a market URL. System pulls market metadata from the Polymarket or Kalshi API: rules text, resolution criteria, deadline, current prices, trading volume, market category, resolver address, related markets. Parsed into structured representation.

Stage 2: Scenario selection. System queries marketplace for scenarios matching the market category, ranked by accuracy on similar resolved markets. Top three candidates surfaced. User selects, opens authoring, or enters first-scenario flow.

Stage 3: Binding. Intermediate LLM layer researches the specific market and discovers the relevant cast, the actual people whose decisions move this market's outcome, along with their public positions, prior actions, and voting history. The discovered cast is bound into the scenario's dynamics and action space alongside market-specific initial conditions. Output is a fully parameterized scenario instance, market-specific cast included, ready for simulation.

Stage 4: Simulation. LangGraph orchestrates the agent swarm. CAMEL agents are instantiated for each actor discovered in binding. Graphiti provides each agent temporal memory. Swarm runs for scenario-specified rounds with within-round parallelism.

Stage 5: Resolution. Simulation outputs aggregated into probability distribution. Resolution mapping translates raw simulation states into market-specific probabilities. Returned with reasoning trace, accuracy context, divergence summary.

Stage 6: Logging and tracking. Simulation logged with full state for replay. Scenario usage count increments. Royalties accrue. When the underlying market later resolves, accuracy metrics update, feeding back into ranking and earnings.

4.9 Run pricing and compute margin

Simulation cost scales with the number of agents, the number of rounds, and the routing choices made by the user. With per-purpose routing and prompt caching enabled, the metered model cost per run stays well below all-frontier execution.

Every run is billed in **USDC** at a single quoted price with two components. The **compute fee** covers the run's model cost plus a margin that keeps the platform profitable on every production run regardless of token price. The **scenario fee** is paid to the author and priced by the scenario's forecasting skill (see §5.1). The user sees one price with the breakdown beneath it. Because model cost varies with market complexity, the compute fee is quoted as a band before the run and settled within it. One hundred percent of the compute margin funds the $NFA buy-and-burn described in §7.1.

5 · Marketplace mechanics

5.1 Scenario pricing and author earnings

Every scenario carries a **skill score from 0 to 100**: a measure of how much better than the market its forecasts resolve, not raw accuracy. The score is the scenario's Brier skill against the market-implied probability at run time (see §6), so a scenario is only rewarded for beating the crowd, never for confidently calling outcomes the market already prices as near-certain. The scenario fee scales with that score:

```
scenario_fee = $1 + (skill_score / 100) × $19

where skill_score ∈ [0, 100]
// $1 floor for unrated or no-edge scenarios, ~$20 ceiling at top skill
// skill_score is the scenario's Brier skill vs the market baseline
```

A newly published scenario launches **unrated** at the $1 floor and earns its price as the markets it forecasts resolve. Early scenarios are therefore cheap to run, rewarding the traders who discover them, while proven scenarios command up to twenty times the floor and pay their authors accordingly. This rewards **demonstrated forecasting edge**, not publication or hype: a scenario that does not beat the market stays at the floor no matter how often it is used.

Authors are paid in **$NFA**. When a user pays the scenario fee in USDC, the protocol automatically buys $NFA on the open market and routes it to the author, so every paid run is direct buy pressure on the token and authors are aligned with the asset their work drives. Earnings are claimable on demand with no vesting; authors typically recycle them into new publications, each costing the $10 $NFA publishing burn (§7.1), which closes the loop.

5.2 Accuracy measurement

Accuracy is measured in two layers.

Backtest layer. When a scenario is published, it runs automatically against a historical event corpus with known outcomes covering 18 months of resolved Polymarket and Kalshi markets. A portion of the corpus is a blind test set authors cannot see during development. Published accuracy uses only the blind set, detecting overfitting.

Live performance layer. When a scenario is used to simulate a currently-unresolved market, the simulation's probability output is recorded. When the market later resolves, the Brier score is computed and added to the scenario's accuracy record.

Live performance weighting is determined by sample size. A scenario with 5 resolved markets has its accuracy estimate weighted toward the backtest baseline. A scenario with 100 resolved markets

is weighted almost entirely on live performance.

5.3 Anti-gaming defenses

- **Blind test sets** prevent scenarios from being tuned to known historical outcomes.
- **Unique-wallet counting** ensures earnings come from distinct wallet usage rather than raw call volume.
- **Similarity detection** identifies near-duplicate scenarios. Publishes with similarity above threshold require differentiation or inherit reduced earnings.
- **Accuracy weighting** is the most fundamental defense. Pumped usage produces no earnings if forecasts are inaccurate.
- **Plausibility validation** (see §4.7) catches manipulation attempts during simulation runtime, not just post-resolution. Bad scenarios fail at the action-generation step, not weeks later when markets resolve.
- **New scenario labeling** marks scenarios with insufficient resolved-market data as unproven. Recommendations always include at least one established scenario.

5.4 Private scenarios

Authors can publish scenarios privately. Private scenarios are not visible in the marketplace, do not appear in recommendations, and earn nothing. They are usable only by their author. Private authors pay only the compute fee on their own runs, with no scenario fee on top, so personal use costs just the metered compute price.

Private scenarios support professional traders who use NFA as a personal forecasting tool without revealing methodology.

5.5 Bootstrap via usage-mining

NFA launches without commissioned seed scenarios. The marketplace opens to the community from day one, anyone can publish (for the $10 $NFA burn, §7.1) or run scenarios from the moment public testnet is live. Quality emerges from the same skill-weighted economics that govern the steady state, not from curated launch partners.

For the first **five months** following public availability, the protocol emits **1% of total supply per month** as a usage-mining reward, roughly **0.0333% of supply per day**. Each day's emission is split equally across every simulation run that day: on a day with three runs, each run earns 0.0111% of supply; on a busier day each run earns proportionally less. The reward goes to the wallet that funded the run, turning early usage directly into token distribution. Five months at 1% per month draws the entire **Community / airdrop allocation** (§7.2), with no separate budget line.

This substitutes self-funded community contribution for upfront commissioning. Early users are over-compensated for showing up when the marketplace is thinnest, and the over-compensation comes from token issuance the protocol controls, not treasury cash it does not. Because the daily pool is fixed, the per-run reward is richest exactly when usage is lowest, a built-in incentive to be

early, and the open-market activity it drives feeds the same buy-and-burn and author-settlement flows that run in steady state.

For market categories where swarm simulation does not add value (short-horizon price action, pure statistical sports markets) NFA continues to honestly display "no coverage" rather than producing low-quality simulations.

6 · Accuracy as the north star

6.1 Why accuracy matters more than other metrics

Prediction-market trading is fundamentally about expected value. A trader who can identify mispriced markets earns money over time; a trader who cannot does not. The tool that helps traders identify mispricings is only valuable to the extent it is more accurate than the market it is being used against.

Accuracy is the only metric that matters at the foundational level. All other product decisions, marketplace design, royalty structure, cold-start strategy, coverage decisions, pricing, are downstream of accuracy. The product is designed to produce accurate forecasts and to measure accuracy transparently.

6.2 Public track record

All scenario accuracy data is public by default. Every scenario has a dedicated page showing complete accuracy history, broken down by market category, with full details of every simulation run that was later resolved.

Transparency serves multiple purposes: **user trust** (traders use tools they can verify), **market efficiency** (better information surfaces faster when performance is visible), **author accountability** (authors cannot hide behind selective reporting), **regulatory defense** (transparent accuracy data distinguishes NFA from opaque forecasting services).

6.3 Scoring methodology

Brier scores are the primary accuracy metric. For a binary market with resolution outcome `o ∈ {0, 1}` and predicted probability `p ∈ [0, 1]`, the Brier score is `(p - o)²`. Lower is better; `0.0` is perfect, `0.25` is the score achieved by always predicting 50/50.

For multi-outcome markets, scoring generalizes to mean squared error across all outcome categories.

Accuracy is displayed as a 0-to-1 score where 1.0 represents perfect forecasting and 0.0 represents random baseline (0.25 raw Brier). More intuitive for non-expert users while preserving statistical properties. For marketplace pricing and staking access, this is re-expressed as a **0–100 skill score** measuring forecasting performance relative to the market baseline (§5.1): 0 means no edge over the market, 100 the demonstrated-skill ceiling.

6.4 Category-specific accuracy

A single aggregate accuracy number is misleading. NFA tracks and displays accuracy per market category. When a user submits a market URL, the accuracy shown for candidate scenarios is specifically the accuracy on that market category.

6.5 Comparative benchmarking

NFA publishes monthly comparative benchmarks showing how NFA simulation accuracy compares to:

- Polymarket / Kalshi market consensus at simulation time;
- Naive baselines (always 50/50, category base rate);
- Single-LLM forecasts (frontier models on identical questions).

These benchmarks serve as the honest check on whether the platform is adding value. If NFA simulations are less accurate than market consensus, the platform publishes that fact rather than hiding it.

6.6 Continuous improvement mechanics

- **Automatic category expansion.** When a category reaches sufficient activity without good scenario coverage, the platform flags the gap and promotes it to commissioning.
- **Scenario staleness detection.** Scenarios whose recent accuracy trends down get flagged to authors for refresh.
- **Winning pattern extraction.** Analysis of top-performing scenarios informs platform-level improvements.
- **Governance feedback.** Accuracy data surfaces marketplace parameters that may need adjustment.

7 · Token economics

7.1 Token utility

$NFA is a Solana SPL token. Sub-second finality and low transaction cost make per-run author settlement and continuous open-market buy-and-burn economically viable at retail price points an EVM L1 cannot match. NFA prices compute and scenario access in **USDC**, which keeps margins legible and removes token-acquisition friction for new users. $NFA is the value-capture and access layer on top of that economy, with five distinct roles.

Buy-and-burn from platform profit. Every run's compute fee carries a margin over its model cost. **100% of the resulting profit**, net simulation revenue after model cost, is used to buy $NFA on the open market and burn it. Operations are funded separately from the project treasury, so platform profit flows entirely to the burn. The burn amount is non-discretionary, fixed at 100% of profit; the cadence will be finalized ahead of launch.

Author settlement. Scenario fees are paid in USDC, automatically used to buy $NFA on the open market, and routed to the scenario's author. Every paid run is direct buy pressure, and authors earn the token their work drives (§5.1).

Publishing. Listing a scenario costs **$10 of $NFA** (USD-pegged), burned on publication. It prices out spam and turns every new listing into a small permanent supply reduction. Authors typically fund it from their settlement earnings, recycling token straight back into the burn.

Staking for premium access. Once a scenario proves an edge, a **skill score above 10**, running it requires at least **$1,000 of $NFA staked**. Unrated and low-scoring scenarios stay open to everyone, so discovery is cheap, but the proven catalog requires skin in the game.

Governance. $NFA holders govern platform parameters: the skill-price curve, accuracy weighting formulas, the staking-access threshold, treasury policy, category weighting, and protocol upgrade decisions. Governance is weighted by staked tokens.

Together these close a loop: users spend USDC, authors earn $NFA, authors recycle it into new listings (burned), and platform profit continuously buys and burns supply. Token demand scales with real usage rather than speculation.

7.2 Supply and distribution

Total supply is fixed at **1,000,000,000 NFA** tokens.

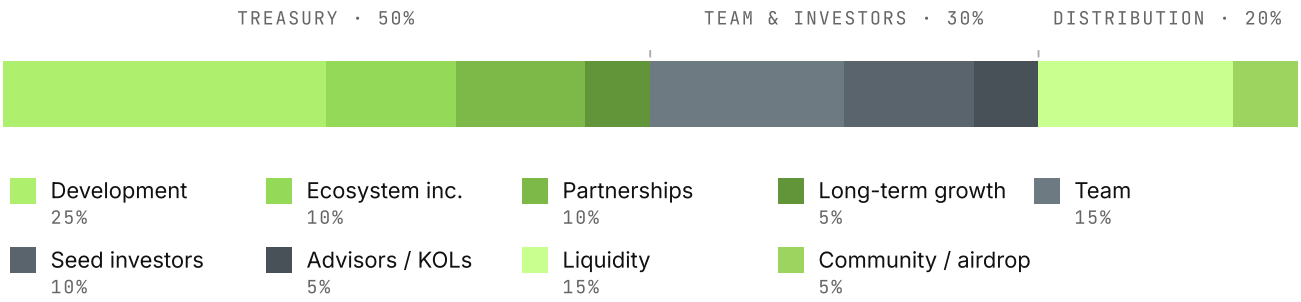

FIG. 3 · TOKEN ALLOCATION · 1,000,000,000 NFA

- **Team, 15%.** Core contributors, long-term alignment, governance and operations.
- **Seed investors, 10%.** Honor commitments from initial raise.
- **Liquidity, 15%.** Initial liquidity provisioned across Solana AMMs at TGE (Raydium, Orca, Meteora), ensuring efficient price discovery and depth for the continuous buy-and-burn and author-settlement flows.
- **Community / airdrop, 5%.** Funds the five-month usage-mining program (§5.5); five months at 1% of supply per month draws the full allocation.
- **Advisors / KOLs, 5%.** Strategic contributors and distribution partners.
- **Development, 25%.** Engineering, infrastructure, audits, and team scaling beyond initial MVP.
- **Ecosystem incentives, 10%.** Accuracy bounties, trader onboarding incentives, and integration bounties for third-party developers.
- **Partnerships, 10%.** Strategic relationships with prediction-market venues, AI agent platforms, data providers, and exchanges.
- **Long-term growth fund, 5%.** Reserve for acquisitions, vertical expansion, and ecosystem programs not anticipated at launch.

7.3 Vesting schedules

All vesting is enforced on-chain through immutable contracts. No party has governance discretion to alter vesting after launch.

- **Team** · 6-month cliff, 18-month linear vesting.
- **Seed investors** · 12-month linear vesting from TGE (no cliff).
- **Liquidity** · locked 12–24 months.
- **Community / airdrop** · released over the 5-month usage-mining program (§5.5).
- **Advisors / KOLs** · 3-month cliff, 9-month linear vesting.
- **Development · Ecosystem incentives · Partnerships · Long-term growth** · voting-controlled release.

7.4 Circulating supply at TGE

Total circulating supply at TGE is approximately **20 to 25 percent** of total supply. Team and advisors are fully cliffed at TGE. Seed investors carry no cliff and begin linear unlock from TGE day one over 12 months, visible, predictable, and small in any given week. The Development, Ecosystem incentives, Partnerships, and Long-term growth allocations are voting-controlled, so the protocol can release tranches as legitimate operational needs justify rather than mechanically unlocking on a schedule. This produces tighter float at launch than a typical cliff-and-vest schedule and avoids the supply overhangs that distort price discovery in the first months post-TGE.

7.5 Value accrual

Compute-profit burn. 100% of simulation profit, the compute margin defined in §4.9, is used to buy $NFA on the open market and burn it. This is the primary, usage-scaled deflation mechanism, observable on-chain and tied to platform profitability rather than speculation.

Author-settlement buy pressure. Every scenario fee is converted from USDC into an open-market $NFA purchase routed to the author. This is continuous buy-side demand that scales directly with marketplace volume; authors who hold or stake their earnings reduce velocity further.

Publishing burn. Each new scenario costs $10 of $NFA, burned on publication. A growing marketplace is a steady, permanent supply sink, and authors funding it from settlement earnings recycle token straight back into the burn.

Staking lockup. Accessing proven scenarios (skill score above 10) requires $1,000 of $NFA staked, and advanced features stake more. This removes float from circulation and scales with the most engaged users.

Net supply. In the bootstrap phase, treasury operational spend and usage-mining emissions partly offset these sinks; as simulation volume grows, compute-profit burn outpaces treasury spend and net supply turns deflationary. The crossover is a function of usage, not of token-price speculation.

8 · Governance

8.1 Governance philosophy

NFA progressively transitions from core-team-managed operations to DAO-governed protocol over 18 to 24 months post-launch. Pragmatic rationale: early period requires rapid parameter iteration based on real-world data; mature period broadens to token-holder base.

8.2 Governance scope

- Scenario skill-price curve (floor, ceiling, slope)
- Accuracy weighting formula and time windows
- Per-purpose LLM routing defaults
- Staking access threshold (currently $1,000 of $NFA for skill > 10)
- Publishing fee and usage-mining parameters
- Category taxonomy and weighting
- Verification badge thresholds
- Treasury allocation decisions
- Partnership and integration decisions over a defined threshold
- Protocol upgrade approval

Core mechanics affecting user/author funds require supermajority approval (two-thirds of voting tokens). Parameter adjustments within defined ranges require simple majority.

8.3 Governance structure

Voting power is held by NFA token holders, weighted by tokens held or staked. Staked tokens receive a multiplier (up to 1.5×). Proposals can be created by any token holder meeting a minimum threshold. Three phases: **discussion** (7 days), **voting** (7 days), **execution** (48-hour timelock).

8.4 Transition timeline

- **Months 0–6.** Core team retains decision-making. DAO advisory only.
- **Months 6–12.** Non-core parameters transition to binding DAO governance.
- **Months 12–24.** Core mechanics progressively transition to DAO.
- **Month 24+.** Full DAO governance. Core team operates infrastructure under protocol-defined parameters.

9 · Roadmap

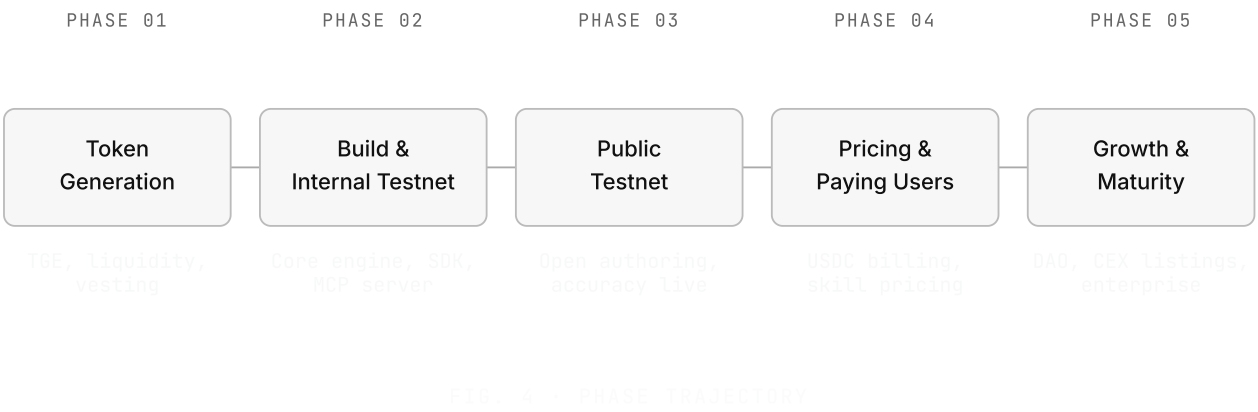

FIG. 4 · PHASE TRAJECTORY

9.1 Phase 1: Token Generation

TGE executes on Solana. Liquidity deployed across Solana AMMs. Vesting programs activate. Legal opinion finalized.

9.2 Phase 2: Build & Internal Testnet

Core engine operational with deterministic execution end-to-end. Scenario authoring SDK functional. URL-paste flow with binding layer. MCP server with initial tool surface. Backtesting harness with blind test set. Repository with clean license structure. Initial scenarios authored by the team and early community in development.

9.3 Phase 3: Public Testnet

Public testnet launch. Open authoring for any user. Full accuracy tracking live. MCP server listed in public registries. Bug bounty opens. Tier-1 audit begins on settlement contracts.

9.4 Phase 4: Pricing & Paying Users

Metered USDC billing begins, run price is compute fee plus scenario fee. Skill-based scenario pricing and $NFA author settlement go live. Staking gates the proven-scenario catalog. First paying users, first author earnings. The five-month usage-mining program (§5.5) opens with public availability and runs through this phase.

9.5 Phase 5: Growth & Maturity

Coverage expansion through organic authoring. Tier-2 CEX listings, with Tier-1 targets contingent on volume. First DAO governance proposals, transitioning to full DAO governance per §8.4. Enterprise tier development. Long-term accuracy track record becomes the primary marketing asset.

10 · Risks and mitigations

This section exists because most token launches treat risk disclosure as boilerplate. Done properly, it is a competitive advantage. The risks below are the real ones.

10.1 Accuracy risk

The product thesis depends on NFA producing forecasts meaningfully better than market consensus on a durable basis.

Mitigations. The five-month usage-mining program seeds early scenario supply and accuracy data; transparent accuracy tracking and honest reporting including underperformance; category-specific accuracy display; continuous feedback loops; honest coverage gaps rather than forced low-quality simulations.

10.2 Regulatory risk

Prediction-market tooling operates in complex regulatory environments. SEC, MiCA, CFTC interactions. Influencer promotion, token distribution, cross-border access introduce surface.

Mitigations. Platform operates as forecasting infrastructure not as a market; liability for specific forecasts sits with scenario authors; geographical restrictions at hosted-frontend level (US blocking); KYC on token claims above thresholds; legal opinion procured before TGE; advisor and KOL allocations publicly disclosed and vested.

10.3 Licensing and intellectual property risk

Mitigations. Open-source components (frontend, MCP server, settlement contracts) under Apache 2.0 / MIT; proprietary engine clearly delineated; `NOTICE` files and attribution maintained; license compliance reviewed in audit scope.

10.4 Marketplace quality risk

Open marketplaces attract gaming, spam, low-quality contributions.

Mitigations. Accuracy-weighted earnings structurally disadvantage bad actors; blind test sets prevent overfitting; similarity detection prevents plagiarism; unique-wallet counting prevents farming; plausibility validation catches manipulation at runtime; new scenario labels prevent unproven work from displacing established scenarios.

10.5 Compute economics risk

Simulation costs scale with LLM inference pricing.

Mitigations. Per-purpose routing optimizes cost-quality tradeoff per call class; prompt caching materially reduces input-token costs on supported providers; OpenRouter provides provider-

agnostic routing; the platform's competitive position improves as LLM prices fall over time.

10.6 Cost tracking risk

Author payouts and the compute margin both depend on accurate per-call cost tracking. Production experience in adjacent systems shows cost-tracking failures can underreport spend by orders of magnitude when fallback chains are hit silently.

Mitigations. Per-call cost-record persistence with provenance for how each cost was computed; recurring reconciliation against provider billing APIs with discrepancies hard-flagged for manual review before author payouts settle; settlement gated on reconciliation confidence.

10.7 Key-person and operational risk

Mitigations. Comprehensive documentation; open-source codebase for non-engine layers; progressive DAO governance reduces concentration; operational playbooks; ability to run alternative hosted implementations once protocol is DAO-governed.

10.8 Adversarial scenario authorship

A malicious actor could publish scenarios designed to manipulate markets rather than forecast them.

Mitigations. Plausibility validation as runtime defense (§4.7) catches institutionally implausible action patterns during simulation, not after market resolution; accuracy-weighted earnings make manipulation economically unviable; transparency of scenario authorship; divergence monitoring catches anomalous outputs; on-chain record of scenario changes; reputation accumulates over many markets.

10.9 Market category coverage risk

Different categories have different accuracy characteristics.

Mitigations. Honest coverage decisions; category-specific accuracy display; recommendations only surface scenarios with demonstrated accuracy on the relevant category; clear platform communication about which categories swarm simulation suits.

11 · Closing

NFA sits at the intersection of three things that have matured in the last 18 months: prediction markets as a credible financial category, multi-agent AI systems as production-grade infrastructure, and MCP as the standardizing protocol for AI agent distribution. The product occupies a specific niche, actor-driven forecasting for prediction-market traders, that no incumbent serves.

The platform's design reflects a deliberate choice: **accuracy is the north star, everything else follows**. The marketplace structure, royalty mechanics, governance transition, coverage decisions, token economics, and engine architecture all optimize for measurable forecasting accuracy. Opaque metrics and narrative-first positioning are rejected in favor of public data and honest disclosure.

The economic engine is a scenario marketplace where domain experts contribute simulation templates, earn royalties based on real-world accuracy, and compete to serve a growing market of prediction-market traders. The technical engine is **production-grade from day one**: deterministic execution with full replay, per-call cost reconciliation, per-purpose LLM routing, prompt caching, and within-round parallelism, because marketplace integrity demands it.

The risks are real and described openly. The mitigations are concrete and reflected in architecture, policies, and planned execution. The cost of doing this properly is higher than shipping a typical token launch. The cost of doing it improperly is unbounded.

NFA is the collaborative-intelligence brand made literal as product. A coordinated mind composed of specialists, each contributed by domain experts, deliberating to produce accurate forecasts, compensated by a transparent economy. What the brand has been promising for a year, now delivered.

NFA

COLLABORATIVE INTELLIGENCE · NFA.CLUB