Dragon

Decentralizing token data to discover alpha.

Co-created by Harshan and Frog

Last updated: June 3, 2025

email: baddragonmygoodness@gmail.com

Table of Contents

1
Introduction

Everyday traders can discover alpha as composable data sourced by open intelligence.

Dragon is an open project powered by community-built token analyses. Each one sources on- or off-chain data and produces specific knowledge about one aspect of a token project's design. Think of the best analyses found on Bubblemaps, BullX, Solscan, GMGN and combine them with the best insights from KOLs, Dune analysts, and insider group chats. Meaningful token data are scattered among these platforms or gated by these chads who have custody of them. Retail traders lose by having to manage opening browser tabs or pinging telegram bots or paying subscription fees to get just some of the information they need to trade with intelligence. This is the stacked game in discovering **alpha**—yet nobody has approached it simply as identifying the volume signals in the data.

Dragon gives analysts the opportunity to contribute token analyses that they believe will signal volume changes, with building bite-sized **data modules** that get integrated into an open browser extension. Each Dragon user's extension is a customizable panel of these various analyses, allowing them to DYOR using the analyses they find most effective.

Dragon's AI trains on a user's on-chain transactions as it relates to their module use, learning their strategic trader profile or "analytical comfort zone" and then matches it to new tokens they haven't discovered yet. As the project grows, data modules achieve degrees of success by measuring their user demand and consistent impact on volume, creating a composable alpha-signal system through the collective intelligence of builders and traders. This open tool helps to build retail's contribution layer on the blockchain, enabling everyday consumers to impact future token design and development.

2

Market Dynamics & Opportunity

Trading bots across blockchains have seen over $80B in lifetime volume[1] and bot trading on Solana alone surpassed a volume of $770m one day earlier this May[2]. As the chain with the highest number of total network transactions[3], Solana's users have found value in a new category of tooling called **token scanners** such as Syrax, SarumAI, and trench.bot. These DYOR assistants currently see ~500k monthly active users with rugcheck.xyz leading the charge[4]. Even Solana's major trading platforms Photon, BullX, and Axiom are incorporating features that display token scanning data adjacent to the chart, acknowledging that data transparency is a top of mind concern for their users. Due to recent high profile scandals[5], extractive pump.fun launches[6], and even obscure behaviors at institutional levels[7], data transparency demands a clear standard for retail trading in crypto.

One of the misunderstood market dynamics of blockchain is **Maximum Extractable Value (MEV)**, a system feature that affects ~25% of all transactions on Ethereum, Solana, and Binance Smart Chain[8]. During periods of peak volatility and high network traffic, MEV can surge to impact over 66% of all block transactions[9]. It has been called the double-edged sword of crypto[10] because it incentivizes validators and drives block production, but it also creates information asymmetries that favor power players in the order flow over everyday traders. Infrastructure-level projects like Flashbots have emerged to address MEV on Ethereum, but there is no equivalent solution built for the retail layer; no way for traders to see how MEV impacts volume changes on the tokens they interact with day to day.

So far, most **Web3 data initiatives** have focused on building defense—like institutional compliance, threat detection, or top-down market reporting. Retail-facing analytics tools lock insights or lack composability to adequately inform trader decisions. Yet the market has proven their appetite. Dune Analytics validates the demand for community-driven dashboards and Nansen proves that trading data can be immensely valuable if made available to everyone. What doesn't exist is a project that combines the two: a community-driven, trader-focused, real-time token intelligence platform made by retail, for retail.

3

Dragon

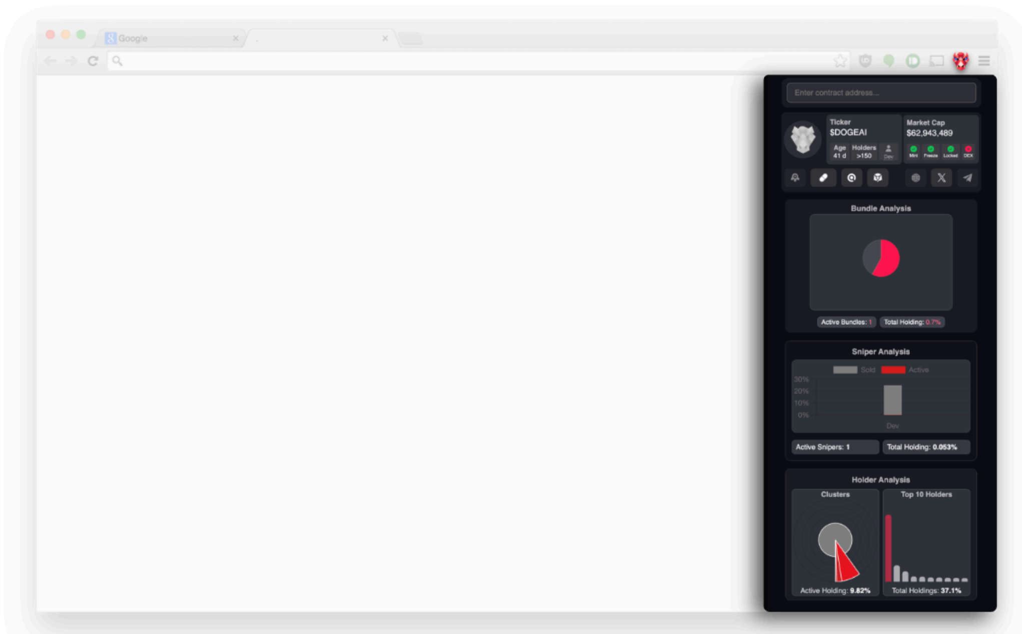


Figure 1: Dragon's extension stays open inside your browser session, staying with you as you navigate the internet, trade tokens, or conduct DYOR.

To use Dragon's browser extension, simply paste in a Solana token contract address. The panel becomes a dashboard of real-time information on that token using community-developed analyses drawing from on- and off-chain data. These analyses update and follow a user as they browse the internet, becoming a DYOR companion that stays with you.

Dragon uses wallet-based single sign-on (SSO) to sync settings across devices and track trading behavior. When a user purchases a token, Dragon **snapshots** the active analyses in their extension. Over time, this builds a behavioral profile that helps personalize recommendations and shape the trader's strategic identity. By using read-only methods, Dragon keeps wallet security risks low while generating rich contextual data for the trader.

Modularity

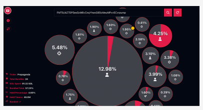
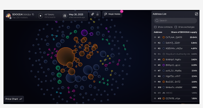
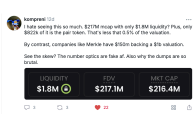
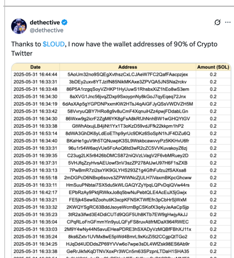
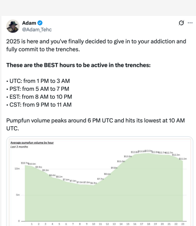
Dragon's key design feature is its modular architecture. Each token analysis is packaged as a bite-sized **data module** that evaluates one specific dimension of a token project such as liquidity flow, smart money inflow, team wallet behavior, or developer history.

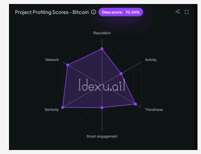
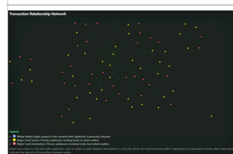
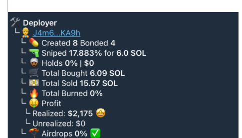
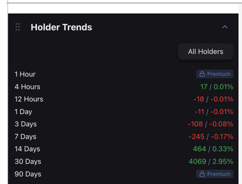
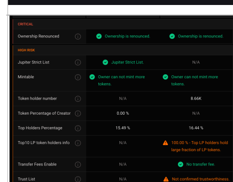
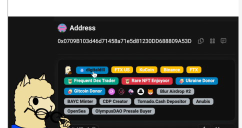
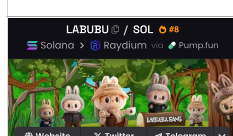
Dragon launches with four prototype modules, but the goal is an open platform for community-development where anyone can propose and submit a module sourcing on- or off-chain data. If their data module is accepted, a developer earns a bounty in $DRAGON to reward their contribution and increase their influence in building the project and community.

Figure 2: Token analyses are designed as customizable ***data modules****—each focusing on a single token design metric. Users can mix, match, and even build these modules themselves.*

Inspiration

Valuable token analyses today are scattered among various individuals and platforms. Dragon invites these data custodians into a shared, open-source research community where insights can reach broader audiences, market services, and remain open for continuous improvement. By going with an open platform, we also create friendly competition—incentivizing better, faster, and creative approaches to token analysis.

Inspiration	Example Analysis	Current Custodian
	KOL / smart money holders	StalkChain
	Sniper & insider buy counter	GMGN
	JITO bundle visualizer	trench.bot
	Top 10 holder awareness	Solscan
	Clustered wallets	Bubblemaps
	Liquidity - market cap ratio	@kompreni
	CT / whale holder % and movement	@dethective
	TikTok / X sentiment matching	@PixOnChain
	Time of day notifications	@AdamTehc

Inspiration	Example Analysis	Current Custodian
	Token narrative profiling	Dexu / @cryptokoryo
	Liquidity pool sandwich score	sandwiched.me
	Top holder previous transaction behavior	RugChecker
	Dev / deployer overview	Trenchy Bot
	Holder trends by time held	HolderScan
	Essential security heuristics	Birdeye (and others)
	Holder profiles / style tags	DefiLlama
	DEX paid? Project links and aesthetics	DEX Screener
	. . .	

Community Development

At launch, developers will propose new modules via pull requests on GitHub. These will be reviewed and integrated by Dragon's core team. As the project evolves, the team will introduce a **Module Forge**—a web-based drag-and-drop sandbox where developers (including non-JavaScript-native users) can build modules in Python or Rust, compile them to WebAssembly (WASM), and test them against historical token data in real time.

A complementary **Module Market** will also allow users to upvote or downvote modules, request new modules to be developed, and toggle modules on/off instantly without needing to reload their extension. This creates a fully composable and permissionless platform where community demand drives module development, visibility, and utility.

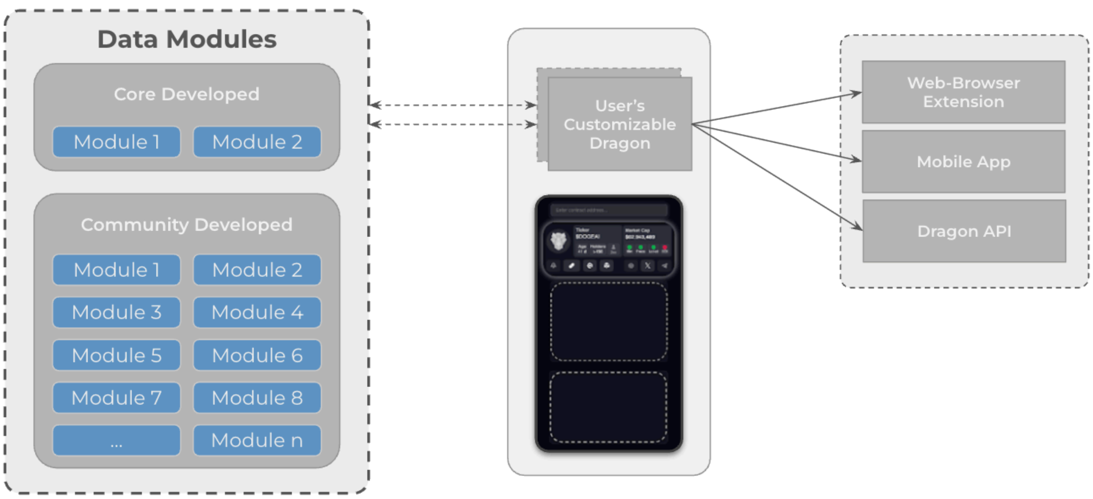


Figure 3: Dragon's architecture includes both core developed modules and community developed modules. The end product syncs across a user's browser extension, a mobile app, and a public API.

In time, modules will be ranked by their performance based on their output correlation to real token volume changes and adoption across Dragon's user base. High-impact and high-visibility community developers will gain influence in the project's direction, aligning incentives between module quality and community value.

Mobile Application

Dragon's mobile app mirrors the extension, giving users access to their modules, dashboards, and alerts on the go. Push notifications can be configured to trigger when selected modules detect major shifts like volume anomalies, social spikes, or liquidity surges.

Because Dragon tracks usage patterns across both desktop and mobile, its personalization function (powered by the Magic engine) adjusts for behavior deltas between devices, refining token match opportunities to traders wherever they are.

4

AlphaSwarm & Magic

An orchestration agentic approach to selective pattern-recognition model training that captures Web3 alpha with a user's modules, sourcing on- and off-chain data and historical volume changes.

Introduction

Unlike traditional on-chain scanners that only surface static metrics, **AlphaSwarm** is a multi-agent architecture that generates scenario-based scores paired with clear, human-readable explanations—so you can instantly spot genuine market anomalies rather than scan through raw data. As a modular ensemble observatory, it fuses on-chain signals (DEX volume, wallet flows, MEV activity) with off-chain indicators (social sentiment, news, alpha-group chatter) to deliver truly actionable trading insights.

The system doesn't rely on full-streaming data. Instead, it learns from significant market moments—volume spikes, rug pulls, pumps—and builds a behavioral map of modular signal combinations during those key events.

These snapshots of module data and volume fluctuations become training data for a **central orchestrator** (numeric pattern recognize) that holistically associates patterns, rather than isolating the effect of a single signal. It rates new events on a 0–10 scale (quantitative) based on how much they resemble past scenarios and provides a narrative explanation (qualitative)—offloading the cognitive burden from traders.

Event-Curated Training on Meaningful Windows

Rather than training a Large Language Model (LLM) on an exhaustive stream of real-time data—much of which is noisy or uninformative—Dragon adopts a more efficient and insightful approach: event-curated training. Specifically, the system identifies periods in the dataset with significant volume changes and extracts the full set of module outputs corresponding to those timeframes. These time-bounded, high-impact segments—referred to as event windows—serve as the training corpus for the orchestration framework. Moreover, all event-window training is conducted in an offline pipeline—so we avoid the complexity, latency and cost of continual LLM fine-tuning in production—while still ensuring our patterns stay up to date through scheduled batch retrains.

Each data module in Dragon is handled by a dedicated agent that monitors and logs its respective features over time. A numeric pattern recognizer learns how different combinations and permutations of these module trends correlate with actual market movements. Rather than attempting to assign linear causality to any single feature, the model captures multi-dimensional patterns that characterize specific market scenarios. The LLM summarizes these for the user.

During inference, when new data arrives, the orchestrator computes a similarity score to past events using the numeric model. The LLM then explains why a given situation may be unfolding, offering a more robust and context-aware form of predictive analytics.

Agentic Confounding Resolution via Orchestration

One of the fundamental challenges in financial modeling is the issue of confounding, where multiple variables change simultaneously, making it difficult to isolate the effect of any single factor. AlphaSwarm avoids the faulty attempt of strict real-time causality per module, which is typically unreliable in the stochastic and noisy environment of crypto markets.

Instead, the system relies on a team of agentic modules, each tasked with monitoring a specific metric—such as Sniper % or Bundle % in a token—independently over time. These agents feed their observations into the orchestration hub which enable agents to validate, enrich, or challenge each other's findings.

For example, if past event data shows that sharp increases in Sniper % combined with flat Bundle % and rising liquidity often lead to significant frontrun-pump scenarios, the orchestrator learns to recognize that pattern. When similar conditions reappear in new data, it surfaces the match—not as a hard statistical claim, but as a scenario match based on prior outcomes. In this way, AlphaSwarm resolves confounding using historical pattern coordination.

Qualitative & Quantitative Output Layer

AlphaSwarm delivers a scenario-based scoring mechanism ranging from 0 to 10. A score of 0 indicates that current conditions bear no resemblance to known high-volatility scenarios from the training data, whereas a 10 reflects a strong historical match to a past pattern with significant market impact. This scoring system reflects confidence in the model's learned memory rather than a strict statistical likelihood.

Each score is accompanied by a breakdown of per module historical precedence—indicating which components contributed most to the pattern match—and a natural language explanation. The explanation is generated through an embedded NLG (Natural Language Generation) pipeline, allowing traders to quickly interpret the rationale behind each signal. This dual output of numeric score and qualitative insight ("AI-Hub" =

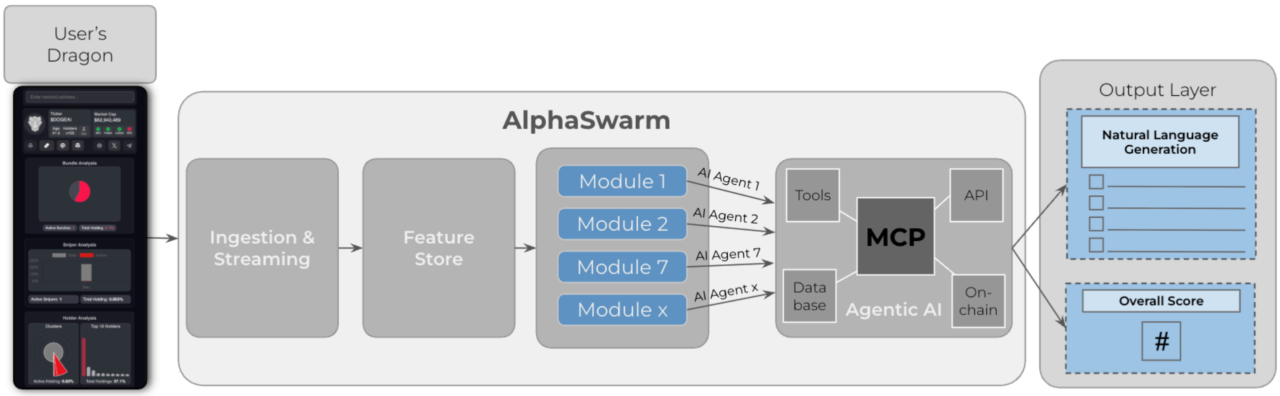


Figure 4: Dragon's Core AlphaSwarm Module

numeric + NLG) helps traders build intuition and make informed changes in their trading focus, even in situations where pure mathematical certainty is lacking.

AlphaSwarm Architecture

The AlphaSwarm is built as a robust, real-time data analytics system that integrates a full-stack modular AI pipeline. It sources data from the various community developed modules that source both on-chain and off-chain events.

What is Model Context Protocol (MCP)?

MCP is an open standard, open-source framework introduced by Anthropic to standardize the way LLMs integrate and share data with external tools, systems, and data sources. Inspired by agentic AI systems like Microsoft's AutoGen, MCP ensures real-time adaptability without retraining the full pipeline, echoing a broader shift in corporate AI towards modular, context-aware intelligence.

For more information on MCP, visit Anthropic's announcement here.

Data streams are ingested through a high-throughput infrastructure using AWS Kinesis or Apache Kafka, which organize signals into time-aligned buffers. These are written into a dual-layer feature store: a hot layer (e.g., Redis or KeyDB) for sub-second latency access using compacted 32–64 byte features, and a cold layer (e.g., Apache Iceberg on S3) for long-term analytics and backtesting.

The core of AlphaSwarm is the modular analysis stack, composed of discrete agents working on each module. These module outputs are passed to the Agentic AI hub, or **Model Context Protocol (MCP)**, where agents using LangGraph or AutoGen frameworks validate, enrich, or challenge one another's findings to orchestrate the final output.

In Dragon, the MCP is a coordination layer that dynamically assigns trust scores to analytic modules based on how well their signals align with actual volume changes. It acts like an ensemble controller—boosting modules with strong predictive value. The MCP serves as a shared trust layer, dynamically updating the weight w_i of each module based on its historical signal quality. A final composite hype index $H = \Sigma_i(w_i \times signal_i)$ is computed for each token, where:

$$H = total\ hype\ score$$
$$\Sigma_i = sum\ over\ modules$$
$$w_i = module\ trust\ weight$$
$$signal_i = module\ output\ signal$$

AlphaSwarm's results are rendered in two formats through its output layer: a quantitative score from 0-10 (and perhaps a future radar chart!) that summarizes the weighted module contributions and a natural language summary that explains the reasons why a token may have a significant shift in volume soon. These outputs are delivered via the AlphaSwarm API to both the browser extension and mobile app, offering an immediate and accessible interface for traders. The entire system is also integrated with the **Magic** engine, enabling one-click new token discovery based on a user's trading patterns.

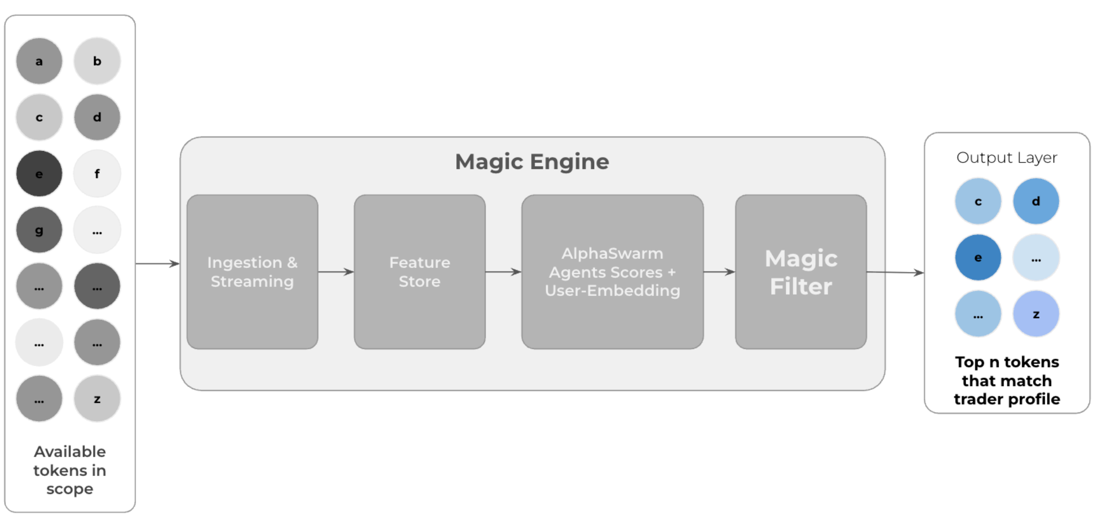


Figure 5: Dragon's Core Magic Module

A Personalized Magic Module

Dragon enhances AlphaSwarm's framework with **Magic**— a personalized module designed to increase your action potential as a trader. Every time you interact with another data module or make an on-chain transaction, your behavior is captured and embedded into a dynamic user profile. This profile encodes not just your preferences, but your actual trading behaviors—the risk levels, liquidity patterns, and market sentiment signals at which you engage with tokens at the most—forming a “behavioral fingerprint” of your strategic profile based on the data analyses that you use.

This fingerprint becomes the input to a two-tower architecture at the core of the Magic engine. The first tower ingests your usage history: module activation patterns, past transactions, and clickstream events across the extension. The second tower encodes AlphaSwarm’s fused analytics that have been passed through the trust-weighted MCP. This means every token embedding already reflects the best-matching AlphaSwarm-derived signals.

During training, the system optimizes a learning objective that prioritizes tokens you’ve historically interacted with or traded. Over time, your preferences become more refined and context-aware as AlphaSwarm agents re-weight signal quality, ensuring that your new tokens are not just personalized, but also responsive to changing market dynamics.

For fast and scalable retrieval, Dragon indexes all token embeddings using FAISS’s HNSW (Hierarchical Navigable Small World) algorithm, enabling near-instantaneous nearest-neighbor lookups. Your personal embedding is updated in real-time using lightweight online learning steps and consolidated with a nightly batch retrain to capture longer-term evolution. For new users without enough interaction history, the system applies clustering algorithms like DBSCAN to assign a cold-start profile by grouping you with traders showing similar early-stage signals.

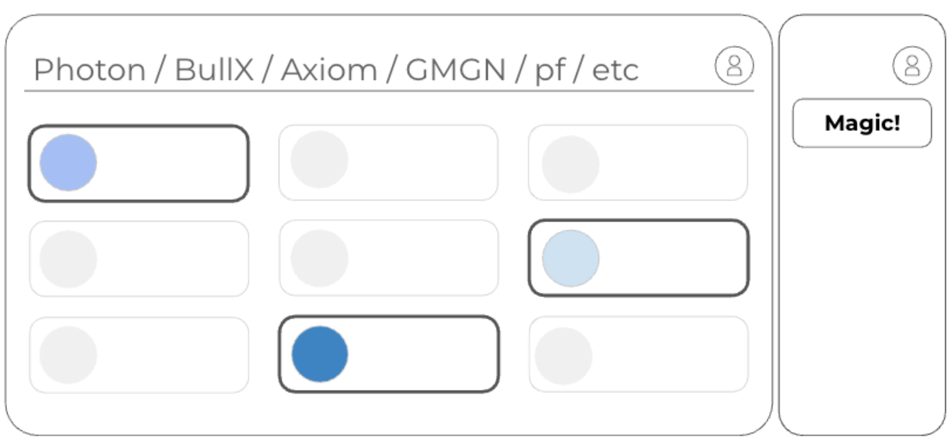


Figure 6: The Magic button interacts with your browser, eliminating the tokens onscreen that do not match your trading profile.

The Magic Button Experience

Once the Magic engine identifies a set of top-matching tokens based on a fusion of your behavioral fingerprint and AlphaSwarm's signals, the Magic button offers a one-click bridge to execution. With a single tap, the Magic Button filters out all non-matched tokens on your selected platform (e.g., DEX Screener, pump.fun), and preloads parameters. In effect, AlphaSwarm fuels the intelligence, Magic delivers the personalization, and the Magic Button enables an action.

This integration of insights, preferences, and execution expedites the loop for intelligent trading—giving users an advantage during new token cycle iterations.

Parallel Enhancements to AlphaSwarm & Magic

Dragon will eventually include a built-in conversational chatbot—your on-demand trading assistant embedded in the extension and mobile app. This chatbot retrieves the most relevant docs (Wikipedia, module snapshots, on-chain traces and real-time agentic analytics) to ground its answers. LangChain agents will orchestrate tool calls for signal queries, while Dragon connects to your encrypted profile and transaction history so every answer is personally tailored—becoming your true AI companion.

5

Project Token

CA:

Utility

There are three utilities the $DRAGON token provides to holders.

1. Rewards a developer for a successful contribution of a token analysis module.
2. Enables a user's access to the core AI modules (AlphaSwarm and Magic).
3. Facilitates a DAO governance of the project's development and treasury allocation.

These three functions will be enabled gradually and with increased detail as the project progresses.

Rollout

I. $DRAGON is launched using Believe and a portion of all trading fees go to the project's treasury, half in SOL and half in $DRAGON

II. Project contributors are given tokens from the $DRAGON treasury to promote community growth and project participation

III. Project overhead and administrative costs are paid out of the SOL treasury to streamline transactions and allow for the widest scope of development opportunities

IV. When trading fees and project growth have stabilized, the core team will build a smart-contract that automates treasury distributions and payments

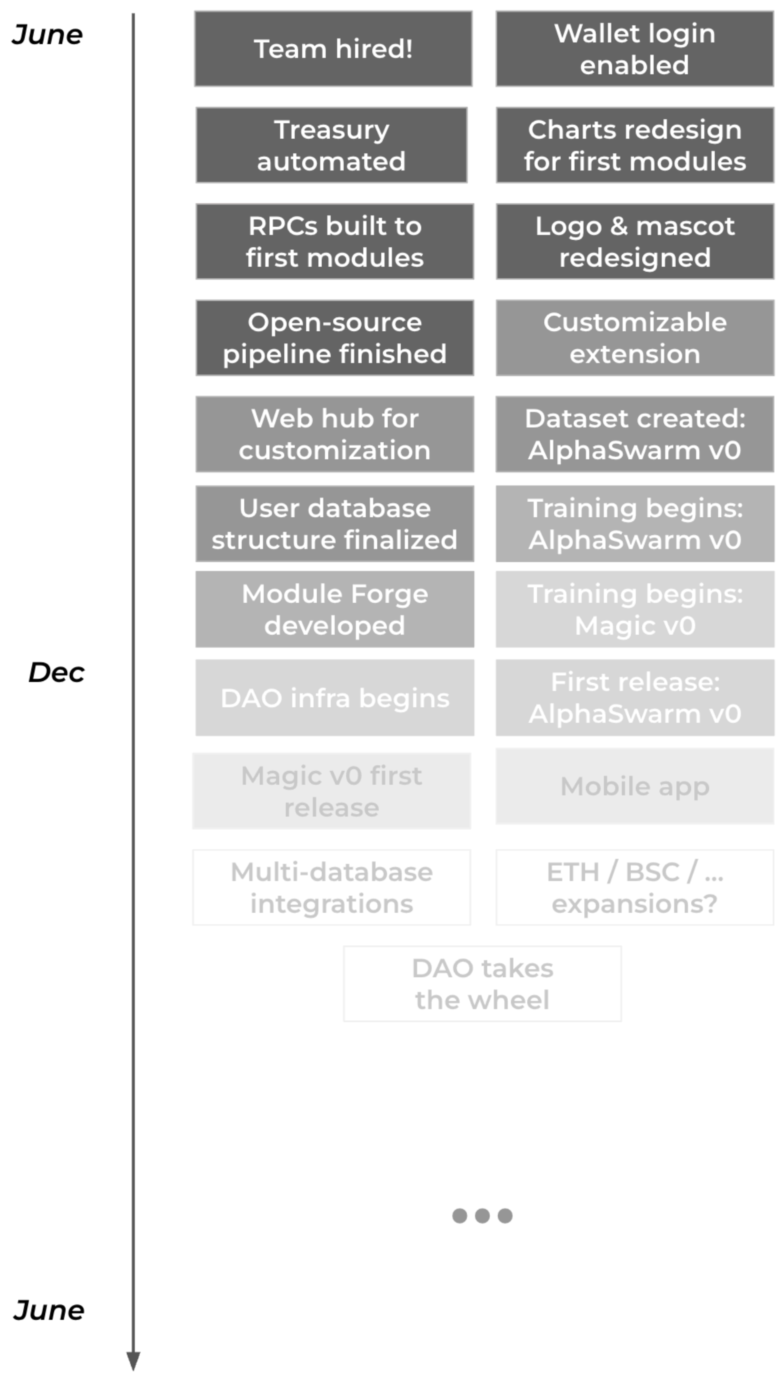
V. When ready, the smart-contract is handed over to a DAO of $DRAGON token holders to decentralize control of the treasury and the project's development

VI. The DAO and core team keeps building together, fully decentralized

The governance rules and logistics of the DAO will be decided in time, with both the token holders and the core team's combined input.

6

Development Goalposts

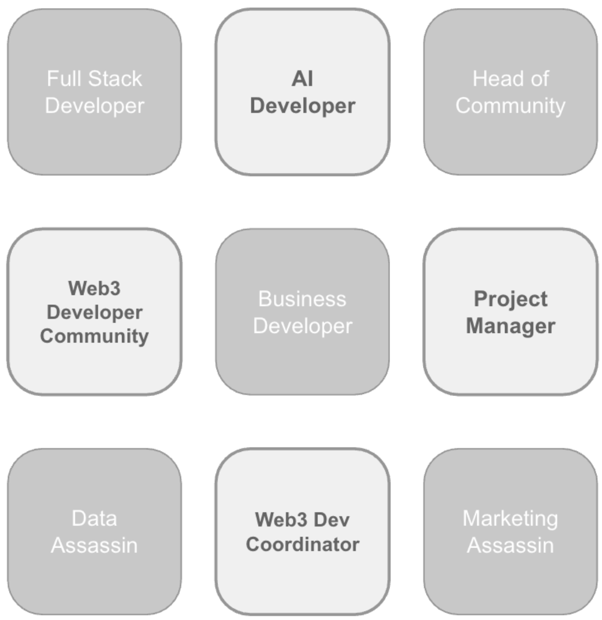


*Dependent on labor & resources. Could go faster if we go higher and slower if we go lower.

7

Founders Note

This project has been launched by Harshan (Harshit) and Frog (Andrew), an AI researcher and a project manager by background. They consider the third founding member to be a Web3 community of developers. This wasn't really the original plan, but it became a happy consequence of neither of us knowing how to build on the blockchain.



If you are interested in joining the team, please send an email to baddragonmygoodness@gmail.com with an introduction and your relevant experience!

Our goal is to lean into this accidental design strength and build a core team that supports this truly decentralized community, including a Web3 developer to coordinate the module integrations. A tenet of the project is its direct relationship to market demand. If it turns out that modular and open-sourced token data is popular among retail traders, we are prepared for the window of opportunity to publish research on user data as it relates to transactions, volume, and effective token analysis modules. The insights gained from this research promises value for blockchains, exchanges, and firms invested in better token design and the intelligent expansion of Web3.

Our advantage as founders lies in our daily exposure to institutional-grade AI and research. Harshit and Andrew met in a Masters of Data Science program at a time when Harshit had just completed research at IBM co-developing their in-house generative AI systems. Andrew approached Harshit for tutoring in the program, but ended up trading memecoins during sessions instead of learning the material. He previously co-founded the blockchain startup Cent, which successfully raised seed funding after their Valuables project gained attention for tokenizing tweets as NFTs on Ethereum.

Harshit currently works at Amazon, building scalable infrastructure for Bedrock AI and developing the latest implementations of MCP methodology. Andrew is busy designing and directing this project's early stages, in what he sees as a necessary initiative at the financial frontier. We both see ourselves as outsiders jumping in the deep end of Web3—retail memecoin traders that lost money to sniper bots and crypto-insiders. We look forward to stewarding and being stewarded by a community that shares in the vision for *everyday traders to discover alpha as composable data sourced by open intelligence*.

How to Contribute

If you like the idea of open-sourcing alpha as composable data, we invite you to co-design this project with us from the ground floor. Here's how:

Build a module
Submit your idea to GitHub for an on- or off-chain analysis that surfaces real-time token insights. This can be written in Python, Rust, or JavaScript and eventually compiles to WebAssembly via our Module Forge (coming soon). Bounties in $DRAGON are awarded for modules accepted to the platform.

Join the community
Connect with us and other contributors in the Telegram to share ideas, plan development, and discuss upcoming features and module priorities.

Collaborate on research
We're looking to collaborate with researchers on topics like token behavior modeling, retail trading patterns, and design ideas for high-impact analyses. Reach out to Andrew on Telegram with ideas.

Join the core team
We're currently looking for a few experienced assassins, including a Web3 dev to lead module coordination and infrastructure development. Send us an email or send Andrew a message on Telegram if you know someone. More positions will be posted on X soon!

Project Links

Website

https://alpha-dragon.ai/

Extension Prototype

https://chromewebstore.google.com/detail/dragon/ncbgllgbplhnbekllhogabdefjidbkoe

GitHub

https://github.com/alpha-dragon-org

X / Twitter

https://x.com/AlphaDragonAI

Telegram

https://t.me/+OU0SLVfcpEZhZWQx

Demo

https://vimeo.com/1062123553