Victor Severiano de Carvalho

SWQuery, Empowering Solana Data Access with Natural Language

SÃO PAULO
2025

Victor Severiano de Carvalho

SWQuery, Empowering Solana Data Access with Natural Language

Final Course Project submitted to the Institute of Technology and Leadership (INTELI), to obtain a bachelor's degree in Software Engineering.

Advisor: Prof. Victor Hayashi

Coadvisor: Prof. Lisane Valdo

SÃO PAULO
2025

Cataloging in Publication
Library and Documentation Service
Institute of Technology and Leadership (INTELLI)
Data entered by the author.

Acknowledgments

I would like to express my sincere gratitude to all the professors at Inteli for their guidance, dedication, and invaluable knowledge shared throughout my academic journey. Their commitment to excellence and continuous support played a fundamental role in the development of this work and in my professional growth.

I am deeply grateful to my parents for their unconditional support, encouragement, and understanding during every stage of this journey. Their belief in me provided the motivation and resilience necessary to overcome challenges and pursue my goals.

I also extend my heartfelt thanks to my friends, whose support, companionship, and encouragement were essential throughout this process. Their presence made this journey more meaningful and rewarding.

Epigraph

> “Solana is fundamentally about optimizing for bandwidth. If you can increase throughput, you can unlock entirely new classes of applications on-chain.”
>
> — *Anatoly Yakovenko*, Co-founder of Solana

ABSTRACT

Severiano de Carvalho, Victor. **SWQuery** . 2025. 72 pages . Final course project Bachelor – Course Software Engineering, Institute of Technology and [Leadership , São Paulo, 2025.]

This work presents the development of SWQuery, a computational solution designed to simplify access to on-chain data on the Solana blockchain through natural language queries and high-level programmatic interfaces. The object of study focuses on reducing the technical complexity associated with blockchain data extraction while preserving performance, scalability, and reliability. The main objective of the project is to provide developers, organizations, and service providers with an efficient tool capable of transforming raw decentralized ledger data into structured and actionable information. The methodology adopted combines market and competitive analysis, technical benchmarking, and iterative product development using the Scrum framework, culminating in the implementation of a Minimum Viable Product (MVP). From a technical standpoint, the solution integrates a high-performance backend developed in Rust, artificial intelligence processing supported by OpenAI models, a Python-based AI agent layer, and a PostgreSQL database for structured data storage. Functional validation was conducted through unit, integration, and acceptance tests, demonstrating the correctness, robustness, and reliability of the system across multiple use cases, including transaction queries, asset retrieval, token analysis, and subscription-based notifications. The results indicate that SWQuery effectively lowers the barrier to accessing blockchain data while maintaining low latency and high accuracy. As a conclusion, the project achieves its proposed objectives and establishes a scalable foundation for future enhancements, such as multi-agent AI integration, decentralized exchange interoperability, and advanced risk and compliance analysis, contributing to improved data accessibility and developer experience within blockchain ecosystems.

Keywords : blockchain data; Solana; artificial intelligence; on-chain analytics; developer tools.

Summary

1 Introduction

This work presents the development of SWQuery, a computational solution designed to simplify access to on-chain data on the Solana blockchain through natural language queries. The project is positioned at the intersection of blockchain infrastructure, data accessibility, and developer experience, addressing a growing demand for tools capable of transforming complex decentralized data into structured and actionable insights. As blockchain ecosystems continue to expand, the ability to efficiently query, interpret, and integrate on-chain data has become increasingly relevant for developers, organizations, and service providers. In this context, SWQuery proposes an innovative approach that reduces technical barriers while enabling scalable and efficient access to blockchain information.

To support this proposal, the work is structured as follows. The Solution Development section presents the foundational assumptions and hypotheses that guided the project, followed by a detailed market sizing and analysis, including TAM, SAM, and SOM estimates, and an examination of the target customer profile. A competitive analysis is then conducted to position SWQuery within the current market landscape and highlight its differentiators. The study also includes a technical benchmark and a detailed description of the technological solution, explaining the architectural and technological choices adopted. Subsequently, the business plan outlines the revenue model, pricing strategy, and financial feasibility of the solution. The validation and results section presents the methods and outcomes used to assess market acceptance and technical viability. Finally, the work concludes by summarizing the main findings, contributions, and future directions for the evolution of SWQuery.

1.1 Context and Motivation:

Blockchain technology has established itself as a foundational infrastructure for decentralized applications, enabling transparent, immutable, and trustless systems. Among existing blockchains, Solana stands out due to its high throughput, low latency, and growing ecosystem of decentralized finance (DeFi), non-fungible tokens (NFTs), and on-chain applications.

Despite these advantages, interacting with blockchain data remains a technically complex task. Accessing on-chain information typically requires advanced knowledge of remote procedure calls (RPCs), account structures, program interfaces, and data decoding mechanisms. This complexity limits adoption and slows down development, particularly for teams focused on product delivery rather than low-level blockchain infrastructure.

The market opportunity identified lies in the lack of standardized, developer-friendly interfaces that allow users to retrieve meaningful blockchain data without deep technical expertise. As the Solana ecosystem grows, there is increasing demand for tools that abstract complexity and accelerate development, especially for analytics platforms, compliance tools, RPC providers, and block explorers.

1.2 Problem Definition and Value Proposition:

Although blockchain data is publicly available, it is not inherently accessible. The main problem addressed by this work is the difficulty of querying, standardizing, and interpreting on-chain data, particularly for non-expert users. Existing solutions often require advanced technical knowledge of blockchain internals, manual construction of complex queries and data parsers, significant development time to transform raw data into usable insights.

This creates a gap between raw blockchain data and its effective use in applications, analytics, and decision-making processes.

SWQuery addresses this problem by providing a natural language-based query interface built as a Rust SDK on top of Solana's infrastructure. The solution translates human-readable queries into structured, machine-consumable outputs, enabling effortless integration into applications and services.

By abstracting low-level complexity and normalizing on-chain data, SWQuery reduces development time, lowers technical barriers, and enables faster access to reliable insights. This approach generates value by improving productivity, enhancing user experience, and enabling new use cases across analytics, compliance, and decentralized applications.

1.3 Objectives of the Work:

The general objective of this work is to design, implement, and validate a computational solution capable of simplifying access to on-chain data through natural language queries, while simultaneously developing a viable business plan for its introduction and adoption in the blockchain market. The project aims to demonstrate not only the technical feasibility of the proposed solution but also its potential for real-world application within the Solana ecosystem.

In order to achieve this general objective, this work pursues a set of specific goals that guide both the technical and entrepreneurial dimensions of the project. From a technological perspective, the objective is to develop a minimum viable product (MVP) of the SWQuery SDK that enables users to retrieve structured and standardized on-chain data without requiring advanced knowledge of blockchain internals. This includes the implementation of natural language query processing, the abstraction of low-level Solana RPC calls, and the normalization of raw blockchain data into outputs that can be easily integrated into applications, analytics tools, and services.

Additionally, the project seeks to integrate the solution with existing blockchain infrastructure providers, such as Helius RPC, in order to ensure reliability, scalability, and performance. This integration is fundamental to validating the practical applicability of the solution in real development environments and to ensuring that the system can support real-time data access demands.

From a validation and business perspective, the objective is to evaluate the solution with potential users, including developers, infrastructure providers, and data-driven platforms, in order to assess usability, perceived value, and market fit. Based on this validation process, the work aims to define and analyze a freemium-based revenue model that balances accessibility for new users with sustainable monetization, supporting long-term adoption and growth.

By achieving these objectives, this work seeks to bridge the gap between blockchain data availability and effective data usability, contributing both a functional technical solution and a structured approach for its commercialization.

1.4 Justification and Contributions:

From a market perspective, SWQuery addresses a growing demand for tools that simplify blockchain data access as decentralized ecosystems expand and mature. By lowering entry barriers, the solution enables broader participation and accelerates innovation.

From a technological perspective, the project contributes by proposing an abstraction layer that bridges natural language processing and blockchain infrastructure, combining performance-oriented technologies such as Rust with scalable RPC services.

From an economic perspective, the solution introduces a sustainable business model aligned with developer adoption, allowing free experimentation while enabling monetization through scalable usage plans.

Overall, this work contributes to improving usability, accessibility, and efficiency in blockchain data interaction, supporting the evolution of decentralized applications and services.

2 Solution Development

2.1 Definition of Market Assumptions and Hypotheses:

The development of SWQuery was guided by a set of market assumptions and hypotheses derived from an analysis of the current blockchain ecosystem, particularly within the Solana network. These assumptions seek to validate both the existence of a relevant problem and the suitability of the proposed solution from technical, usability, and economic perspectives. The hypotheses defined in this

section served as a foundation for the architectural, functional, and business decisions made throughout the project.

2.1.1 Problem Hypothesis

The central problem hypothesis of this work assumes that developers, analysts, and blockchain-integrated application teams operating within the Solana ecosystem face significant difficulty in accessing, querying, and interpreting on-chain data efficiently. Although Solana provides high-performance and low-cost transactions, its data model is complex and requires deep technical knowledge to extract meaningful insights using traditional RPC calls and indexers.

It is assumed that this complexity represents a concrete pain point, particularly for professionals who need fast access to blockchain data but do not specialize in low-level blockchain infrastructure. Furthermore, the hypothesis assumes that this audience is willing to adopt and pay for a solution that significantly reduces development time, lowers technical barriers, and transforms raw blockchain data into actionable information.

2.1.2 Solution Hypothesis

The solution hypothesis assumes that a software development kit (SDK) capable of translating natural language queries into structured Solana data requests is an effective and superior approach to addressing the identified problem. By abstracting complex RPC interactions and returning standardized, human-readable outputs, SWQuery is expected to improve developer productivity and broaden access to blockchain data beyond highly specialized users.

This hypothesis is based on the assumption that natural language interfaces, combined with high-level abstractions, can reduce cognitive and technical load without compromising accuracy or performance. Additionally, integrating the solution with established infrastructure providers, such as Helius RPC, is assumed to ensure reliability, scalability, and compatibility with existing Solana-based applications, making the solution viable for real-world usage.

2.1.3 Value Hypothesis

The value hypothesis assumes that the proposed freemium pricing model is perceived as fair and accessible by the target audience, while also enabling sustainable revenue generation. By offering a limited number of free queries, the model allows potential users to evaluate the solution's value before committing financially, reducing adoption friction.

It is further assumed that developers and organizations that rely on frequent blockchain data queries will find sufficient value in paid plans to justify the cost, particularly when compared to the time savings, reduced engineering effort, and improved data accessibility provided by the solution. This hypothesis supports the expectation that SWQuery can achieve both user adoption and economic viability within the competitive blockchain tooling market.

2.2 Market Sizing and Analysis:

The market analysis for SWQuery aims to evaluate the economic feasibility of the solution by estimating its potential reach and identifying its target customers. This analysis is based on the application of the TAM, SAM, and SOM framework, which enables a structured understanding of the total market opportunity, the realistically addressable segment, and the achievable initial market share. In addition, this section presents a detailed profiling of the target customer segment to ensure alignment between the solution's value proposition and real market demand.

2.2.1 Market Size (TAM, SAM, SOM):

The Total Addressable Market (TAM) for SWQuery encompasses the global blockchain development tools market, which includes all software, SDKs, APIs, and platforms that enable developers and organizations to interact with blockchain systems. According to market research reports, the blockchain development tools

market was valued at approximately USD 1.69 billion in 2024 and is projected to grow to USD 10.0 billion by 2035, representing a significant expansion of demand for tools that simplify blockchain integration and development. This broad category includes middleware, debugging tools, smart contract frameworks, and analytics solutions for decentralized technologies, representing the overall economic opportunity for solutions like SWQuery in the global blockchain ecosystem.

The Serviceable Available Market (SAM) refines the TAM to focus on segments directly relevant to SWQuery's offering: developers, analytic platforms, and infrastructure services specifically within high-performance blockchain ecosystems such as Solana. Solana's developer ecosystem continues to grow, with reports indicating between 2,500 and 3,000 monthly active developers on open-source repositories, complemented by thousands more in private development, reflecting a vibrant base of technical users who could benefit from simplified on-chain data access.

In addition, ecosystem growth metrics show that Solana attracted 7,625 new developers in 2024 alone, the largest annual onboarding among major blockchain ecosystems, demonstrating the platform's expanding addressable developer population.

The Serviceable Obtainable Market (SOM) represents the portion of the SAM that SWQuery could realistically capture in its initial stages. Considering competitive tooling, developer adoption rates, and SWQuery's focus on high-value users (such as analytics platforms, compliance teams, and RPC providers), a conservative estimate assumes capturing 5 % to 10 % of active Solana developers in the near term. With an estimated 3,000 monthly active developers, this implies approximately 150 to 300 core users initially engaging with SWQuery's product offerings. This early user base can expand through strategic partnerships, integrations with infrastructure providers, and ecosystem events like hackathons, which have been pivotal in Solana's growth.

Together, these estimations indicate that the market opportunity for SWQuery is both significant and measurable. The TAM reflects a multi-billion-dollar global market for blockchain development tools, the SAM focuses on the rapidly growing Solana

developer ecosystem, and the SOM defines an achievable and strategically focused initial user base.

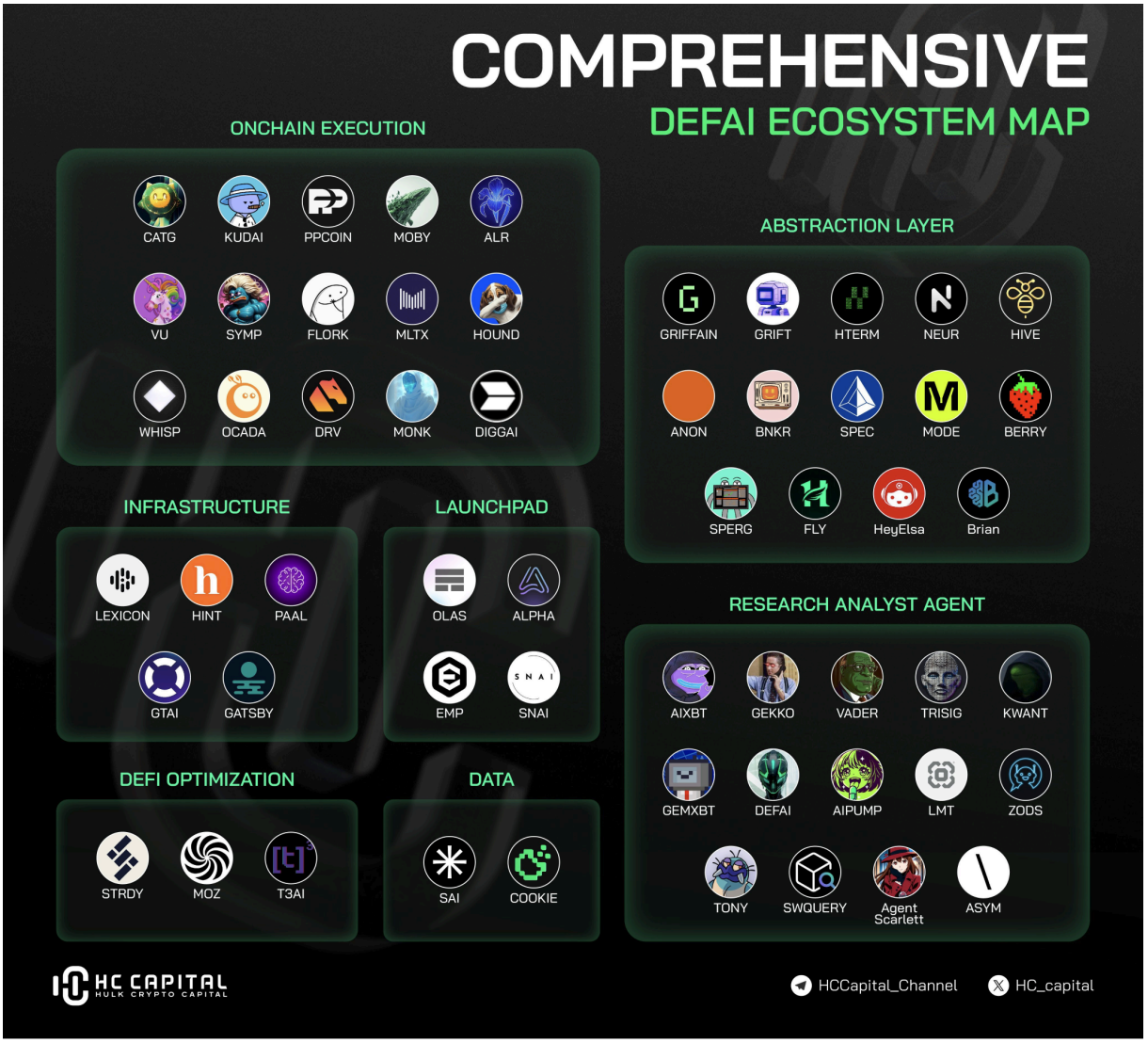
2.2.2 Customer Segmentation and Profiling

The primary target customers of SWQuery are developers and technical teams building applications that interact with the Solana blockchain. These users typically possess intermediate to advanced programming skills but may lack deep expertise in blockchain infrastructure, low-level RPC calls, or data indexing mechanisms. Their core objective is to access reliable blockchain data quickly and integrate it into applications without incurring excessive development complexity or maintenance costs.

A secondary target segment includes analytics platforms, compliance and monitoring solutions, and blockchain service providers that require frequent, structured, and interpretable access to on-chain data. For these users, accuracy, scalability, and automation are critical, and the ability to transform raw blockchain data into actionable insights represents a significant operational advantage.

The ideal customer persona can be described as a software developer or technical product team working in a fast-paced environment, where time-to-market and development efficiency are crucial. This persona values tools that reduce cognitive load, abstract technical complexity, and integrate seamlessly into existing workflows. Additionally, this customer segment demonstrates a willingness to pay for solutions that deliver clear productivity gains, predictable costs, and reliable infrastructure, aligning closely with the value proposition offered by SWQuery.

2.3 Competitive Analysis and Differentials:



2.3.1 Research Analyst Agent

2.3.1.1 Agent Scarlett

Agent Scarlett is an artificial intelligence agent developed on the Eliza framework, designed to help users make informed decisions about investing in tokens. The platform can be integrated into chats on Discord or Telegram, offering analysis of tokens and wallets, as well as evaluating social sentiment on X (formerly Twitter). In addition, 10% of the AGENCY token will be donated to the ai16z DAO. To enhance accessibility and real-time interaction, Scarlett integrates directly with popular chat platforms like **Discord and Telegram**, allowing users to receive analyses and insights within their communication workflows.

This interactive capability means users can pose questions and retrieve information without leaving the platforms where they already engage in community discussions. Finally, the project's development roadmap outlines planned expansions, including broader data source integration, optimization of memory retrieval processes, inclusion of a trusted database, performance improvements, and the implementation of "agent swarms" alongside website updates — all of which are intended to expand the analytical depth and utility of the system over time.

2.3.1.2 ASYM

ASYM is positioned as an AI-driven decentralized finance platform that combines machine learning, predictive analytics, and automated trading strategies to analyze market conditions and generate actionable insights. The platform employs a network of intelligent agents capable of processing real-time on-chain and market data to identify high-potential trading opportunities and optimize capital allocation. This approach allows ASYM to perform deep market analysis, behavioral modeling, and automated trading execution, extending beyond simple data feed services to deliver a comprehensive analytics and execution system.

Functionally, ASYM integrates artificial intelligence with blockchain infrastructure, enabling predictive modeling, investor behavior analysis, and high-frequency trading automation. Users can leverage these capabilities not only for insight generation but also for automated execution of strategies, which distinguishes ASYM from typical analytics dashboards. Its ecosystem is further supported by the native $ASYM token, which is central to profit settlement, incentives for agent participation, and community governance.

Although specific details about a personalized information feed tied to portfolio connections were not confirmed in publicly available sources, ASYM's core offerings emphasize AI-powered analytics and automated

trading features designed to democratize access to sophisticated market strategies and predictive insights.

2.3.1.3 ZODS

ZODS presents itself as one of Solana's first AI-powered DeFi toolkits, positioning its platform as a comprehensive set of on-chain tools that integrate market analytics with conversational blockchain interactions. Through its On-Chain AI module, users can interact with decentralized finance protocols using natural language, allowing them to analyze, trade, and execute tasks spanning both on-chain and off-chain activities with minimal technical effort. This AI integration is designed to support a broad set of DeFi use cases, from analytics to execution, and is accessible via web and chat interfaces, including voice and Telegram interactions, which aim to lower the barrier to entry for both novice and experienced users.

Additionally, ZODS integrates a range of ecosystem functions — such as trading tools, wallet management utilities, and token issuance workflows — into a single platform that aims to be a multifunctional "Swiss Army knife" for Solana users. The platform also emphasizes real-time data provision for AI agents by open-sourcing data scraping services, supporting multi-language needs and advanced interaction patterns. However, while its scope is broad, token application scenarios and long-term positioning within the DeFi landscape remain evolving.

2.3.1.4 Limitus AI

Limitus AI is an ambitious platform that aims to bridge the gap between Web2 and Web3 ecosystems by combining artificial intelligence with decentralized finance and automation. At its core, Limitus integrates advanced AI models with blockchain data to enable users to automate complex workflows, execute financial operations, and manage assets

across multiple chains using intuitive interfaces such as voice commands and unified dashboards. The platform emphasizes privacy-first design, processing data locally on users' devices to ensure control over sensitive information while still leveraging real-time insights from both blockchain and traditional data sources.

Functionally, Limitus supports multi-chain interoperability, enabling users to manage wallets and execute cross-chain DeFi strategies seamlessly. Its use cases range from automated trading and portfolio management to real-time market analysis and workflow optimization, consolidating both financial and productivity tasks within a single AI-driven environment. The system's incorporation of technologies like Retrieval-Augmented Generation (RAG) allows it to synthesize structured and unstructured data dynamically, which enhances its ability to generate actionable insights across different domains.

Overall, Limitus distinguishes itself through its integration of AI automation, multi-chain support, privacy-focused execution, and voice-driven interaction, positioning the platform as a cross-domain solution for users who seek both financial and productivity automation in a unified interface.

2.3.1.5 AIPump

AIPump is a decentralized blockchain platform focused on the creation, launch, and deployment of AI-driven agents and token ecosystems, particularly emphasizing social engagement and community-centric tokenomics. The core idea behind aiPump is to allow users and projects to build and fair-launch AI agent *meme coins* that interact autonomously across social platforms such as X (formerly Twitter) and Telegram. These agents can perform actions like posting content, engaging audiences, and acting as digital mascots or automated brand ambassadors — effectively blending AI interaction with social media presence and blockchain token activity.

Functionality on aiPump includes a drag-and-drop creation interface for AI agents, reducing technical barriers and enabling users — from developers to non-technical participants — to design AI entities that act across platforms and blockchain networks. The platform also champions a fair launch model, where the entire supply of a new AI agent's token is automatically placed into liquidity pools to promote equitable access and community participation, rather than privileged pre-sales.

Although aiPump's focus is not on traditional financial metrics or AI-generated trading signals in the way conventional signal platforms operate, its integration of AI, token creation, and social engagement represents a unique blend of blockchain automation and marketing utility. The platform's token metrics and ecosystem activity, including market capitalization behavior and user adoption, reflect its evolving position in the AI-blockchain space rather than pure analysis or trading insight tools.

2.3.1.6 DeFi Agents AI

DeFi Agents AI is an AI-powered trading assistant designed to simplify and automate cryptocurrency trading and market analysis for users of varying experience levels. The platform combines artificial intelligence with big data analytics to handle complex tasks such as identifying trading opportunities, performing real-time market analysis, and executing trades on behalf of users with minimal manual intervention. Its core functionality includes automated strategy generation through backtesting of recent market data, enabling users to select from different trading approaches such as high-yield, high-frequency, and stable strategies, which are optimized to perform in volatile market conditions.

The user experience on DeFi Agents AI emphasizes accessibility, lowering the barrier to entry for both novice and experienced traders by

providing intuitive tools that automate analysis, execution, and strategy management. The platform's AI execution engine is capable of operating continuously, allowing trading bots to seize opportunities 24/7 without direct human oversight. Its design also includes analytics features that deliver insights into market behavior and strategy performance, helping users understand potential outcomes and risks associated with different automated approaches.

Early adoption metrics from the alpha release demonstrate notable engagement, with the platform registering over 11,000 daily active users and more than 720,000 interactions within two weeks, alongside significant trading volume and assets under management — suggesting clear market interest in AI-driven trading solutions. Additionally, DeFi Agents AI is preparing for broader ecosystem development, including the launch of its native token and extended integrations that aim to enhance scalability and user participation in the decentralized finance landscape.

2.3.1.7 GEMXBT

GEMXBT is an autonomous AI-powered cryptocurrency analytics platform that aims to help traders filter noise and focus on high-quality market insights by synthesizing on-chain data, market sentiment, and technical indicators into actionable information. The system uses advanced algorithms to generate real-time analytical outputs, including technical analysis and trend evaluations, which are designed to support better trading decisions without overwhelming users with irrelevant data.

Functionally, GEMXBT provides a unified terminal experience that combines AI-driven risk ratings, investment perspectives, and market signal insights, potentially automating trade execution and extending analysis to a broader set of financial assets beyond cryptocurrencies. The platform also offers

real-time updates and on-chain analysis, delivering insights through interfaces such as Twitter and Telegram to ensure traders remain informed of market dynamics.

Overall, GEMXBT positions itself as an AI analytical assistant for traders seeking faster, data-driven decision-making, emphasizing ease of use, real-time signals, and the reduction of informational noise in volatile markets.

2.3.1.8 KWANT

KWANT (also seen as kwantxbt) is an AI-oriented token and agent project operating within the Top Hat ecosystem, a Solana-based platform focused on enabling the creation and deployment of autonomous AI agents without coding knowledge. The concept behind KWANT centers on delivering AI-driven technical analysis for traders, leveraging machine learning models trained on thousands of financial charts to provide insights such as entry points, targets, and indicators that support informed decision-making in volatile markets.

Within the Top Hat environment, KWANT functions as a utility asset and analytical tool that assists users in interpreting complex chart patterns and market behavior. Although detailed documentation of its full feature set is limited, the token's integration with AI components for chart analysis and trader support suggests that its core value proposition lies in automating and enhancing technical evaluation, rather than offering broader DeFi research tools or customizable reporting typical of generic analytics platforms. Market data indicates a modest capitalization and active community engagement within its ecosystem, highlighting its role as a specialized analytical agent rather than a comprehensive multi-tool solution.

2.3.1.9 TriSigma

TriSigma is an AI-driven agent positioned within the Solana ecosystem that aims to combine market analysis, continuous learning, and actionable insights to help users navigate complex cryptocurrency environments. It adopts a conceptual approach inspired by the statistical “3σ” model, positioning itself as an outlier-oriented system that seeks to interpret market anomalies and trends through adaptive intelligence and evolving interactions with users. TriSigma’s design allows it to absorb real-time market information, construct multidimensional analytical models, and generate insights that go beyond traditional technical indicators, potentially assisting in portfolio decisions and trend identification.

The platform emphasizes deep analysis and continuous learning, using neural-network-based frameworks to refine its understanding of market behavior as it processes more data and user interactions. It aims to communicate insights through sharp, provocative analysis and, in some implementations, interactive tools such as Telegram bots that provide detailed on-chain token evaluations and other analytics directly in messaging platforms.

TriSigma also incorporates community-oriented dynamics, allocating portions of its token supply to reward users who contribute challenging questions and to build future investment funds based on identified opportunities, reinforcing an ecosystem where user engagement and AI analysis are mutually reinforcing.

While the precise technical and operational details of its execution systems remain limited in public documentation, TriSigma’s focus on evolving analysis, user interaction, and adaptive insights positions it as a distinctive competitor in the landscape of AI-augmented market intelligence tools.

2.3.2 Abstraction Layer

2.3.2.1 Strawberry AI

Strawberry AI is an advanced **AI-powered platform tailored for Web3 and decentralized finance** that integrates large language models and autonomous agent networks to help users navigate complex blockchain data and market information. Designed to be inherently **multi-modal**, the platform supports interactions via multiple input types and leverages coordinated agents to interpret user queries, analyze sentiment, track trends, and present actionable insights across major blockchain ecosystems including Ethereum, Solana, and layer-2 networks. This allows both novice and experienced users to engage with DeFAI products, receive research insights, and make more informed decisions with minimal technical overhead.

Strawberry's ecosystem comprises several integrated components such as real-time indexing of news, social sentiment, transactions, and decentralized data, enabling deep analysis and contextual understanding of market dynamics. Additional offerings include tools for cross-chain research, automated trend tracking, and an expanding suite of AI agents dedicated to crypto research and DeFi insights, highlighting its ambition to democratize access to sophisticated Web3 analytics and intelligence.

2.3.2.2 Mode Network

Mode Network is a modular Layer 2 (L2) blockchain built on Ethereum using Optimism's OP Stack, designed to scale decentralized finance (DeFi) applications through a combination of secure infrastructure, developer incentives, and AI-driven capabilities. Its core mission is to reduce transaction costs and improve scalability while fostering a thriving on-chain economy by enabling autonomous financial operations such as trading, yield optimization, and smart contract execution via AI agents. Mode's integration with the Optimism Superchain allows it to

benefit from shared security and tooling, positioning it as a purpose-built ecosystem for DeFi innovation rather than a general-purpose L2.

The network hosts more than 50 DeFi applications and promotes growth incentives for developers and users alike, including sequencer fee sharing and reward programs backed by grants from the Optimism Foundation. Mode also incorporates AI frameworks that aim to support autonomous decision-making in areas like arbitrage, MEV searching, and smart contract auditing, extending the concept of decentralized finance into what the project describes as an AI-powered on-chain economy.

Fundamentally, Mode blends secure L2 scaling with AI interoperability to create a new class of financial applications that automate DeFi strategies and empower developers through both technical infrastructure and economic incentives, representing a unique fusion of blockchain scalability and artificial intelligence.

2.3.2.3 Spectral Labs

Spectral Labs is an innovative project focused on creating a new paradigm known as the Onchain Agent Economy, where autonomous AI agents can be deployed, governed, and operated directly on blockchain networks without requiring traditional programming expertise. The platform’s flagship product, Spectral SYNTAX, enables users to define and build intelligent agents using natural language, automatically generating production-grade smart contract code that can be compiled, deployed, and executed on-chain, thereby lowering the barrier to Web3 development and automation. These autonomous agents are designed to operate continuously, manage digital wallets, and interact with on-chain data and external APIs to perform complex tasks, such as trading, governance, and automated strategy execution, without constant human intervention.

Key functionalities of the Spectral ecosystem include autonomous agent creation, where each agent possesses its own wallet and identity to transact independently, and no-code development workflows that make on-chain agent creation accessible to users with varied technical backgrounds. Agents can be tailored by personality, behavior, and strategy through conversational interfaces, and they can be governed using community-oriented mechanisms tied to native tokens, enabling collaborative decision-making and decentralized governance.

Overall, Spectral Labs positions itself at the intersection of AI, blockchain automation, and decentralized governance, emphasizing the creation of a scalable infrastructure that supports autonomous AI workflows, cross-chain operations, and real-time execution of complex financial and computational tasks on-chain.

2.3.2.4 Orbit

Orbit (often referred to as OrbitCryptoAI) is an AI-powered DeFAI companion platform that aims to simplify and automate decentralized finance interactions across a broad, cross-chain ecosystem. Built with advanced artificial intelligence at its core, the platform enables users to execute complex DeFi operations — such as token swaps, bridging assets, staking, yield optimization, lending, borrowing, and portfolio management — through an intuitive interface that abstracts much of the technical complexity typically associated with these activities. Orbit supports integration with 100+ blockchains and hundreds of DeFi protocols, allowing AI agents to optimize cross-chain financial workflows efficiently.

The centerpiece of the platform is the Orbit Agent, a comprehensive AI companion designed to automate portfolio tasks and DeFi strategies with minimal user intervention. This includes automated trading and risk-managed yield farming as well as real-time portfolio analytics. The platform also emphasizes its vision of an “autonomous economy,”

where AI agents execute financial decisions across blockchains on behalf of users, supported by technologies such as the Model Context Protocol (MPC) for cross-chain communication and the ecosystem's native $GRIFT token, which fuels governance, rewards, and premium functionalities within the system.

Overall, Orbit positions itself as a high-level DeFi automation hub that blends AI-driven execution, multi-chain interoperability, and comprehensive finance management tools, making it a distinct competitor in the landscape of AI-augmented decentralized finance platforms.

2.3.2.5 GRIFFAIN

GRIFFAIN is an AI-driven decentralized finance platform built on the Solana blockchain, designed to streamline and automate complex on-chain actions through a network of intelligent agents. The core of the project is its Agent Engine, which interprets natural language commands and translates them into actionable transactions, such as automated trading, swaps, NFT minting, liquidity provision, and other DeFi operations, reducing the manual effort typically required for these tasks.

Built to take advantage of Solana's high throughput and low transaction costs, GRIFFAIN allows users to deploy both personal and specialized agents that can perform tasks like token trading or complex strategy execution on behalf of the user. Interaction with these agents is facilitated through a delegated wallet model that ensures security and control, with users granting specific signing permissions for agent actions.

The platform also includes its own decentralized exchange (DEX) and liquidity pools, enabling token swaps and ecosystem growth within the same environment. Its native GRIFFAIN token serves multiple functions, including governance, staking rewards, and unlocking

advanced agent capabilities, while an internal unit called “Energy” is used to fuel agent tasks.

Although initially accessible through invitation or specific access passes, GRIFFAIN has positioned itself as a pioneering automation layer on Solana, offering a blend of AI automation, autonomous on-chain execution, and decentralized finance tools that aim to simplify user interaction with blockchain applications and accelerate DeFi workflows.

2.3.2.6 Hive

Hive is a modular and interoperable network of AI-powered DeFi agents built on the Solana blockchain that aims to simplify complex decentralized finance operations through a natural language interface. The platform enables users to execute a wide range of on-chain activities — such as trading, staking, liquidity management, yield optimization, and sentiment analysis — by issuing plain-language instructions that the system’s specialized agents interpret and act upon. This approach removes much of the technical complexity traditionally required for navigating DeFi protocols and allows users to perform multi-step transactions without deep blockchain expertise.

Hive’s architecture is designed around a network of coordinated agents, each responsible for distinct functions within the DeFi ecosystem. These agents work together through an orchestration layer to fulfill user objectives automatically, enabling workflows such as automatic asset allocation or real-time response to market events. The platform’s composable and interoperable design allows developers to extend its capabilities, adapt workflows, and integrate third-party services, while its natural language interface aims to broaden access to complex financial operations for both novice and advanced users.

2.3.2.7 Neur

Neur is an open-source, full-stack AI-powered copilot application for the Solana blockchain that combines large language models (LLMs) with decentralized finance (DeFi), NFTs, and wallet management into a single intelligent interface. Designed to simplify complex blockchain interactions, Neur allows users to interpret and execute blockchain operations through natural language commands, enabling tasks such as token swaps, portfolio tracking, NFT analytics, and other DeFi operations without requiring deep technical expertise.

At its core, the platform uses advanced AI models — including Claude 3.5-Sonnet and GPT-4 — to power its intelligent agent system, which understands user intent and translates it into actionable blockchain transactions and analysis. Neur's integration extends natively across the Solana ecosystem, connecting with major protocols such as Jupiter for optimized swaps, Pump Fun for token launch support, Magic Eden for NFT marketplace interactions, DexScreener for market analytics, and Dialect for messaging and notifications.

Neur also provides integrated wallet management and real-time portfolio insights, enabling users to monitor assets and market trends within the same interface. Its open-source, community-driven model encourages developer participation and transparency, allowing broader contributions to its evolution and deeper ecosystem integration over time.

Overall, Neur positions itself as an intelligent interface that bridges the gap between users and Solana's technical complexity, offering a unified experience to analyze, interact with, and execute on-chain operations through AI-augmented automation.

2.3.3 Onchain Execution

2.3.3.1 CATG

CATG (Crypto Agent Trading) is an AI-driven automated trading system developed by Boltrade that aims to enhance decentralized finance (DeFi) activity on the Solana blockchain by combining artificial intelligence with real-time market analysis. The platform's core feature is its ability to use AI algorithms to continuously monitor on-chain behavior and market trends — such as tracking thousands of wallets and evaluating token activity — in order to generate actionable trading signals and support autonomous execution of trades. CATG's AI engine is designed to reduce reliance on manual decision-making and improve trading efficiency by identifying high-potential opportunities and executing orders based on algorithmic insights.

Accessible through the Boltrade ecosystem, CATG integrates data-driven analysis with its trading infrastructure, providing users with tools for market prediction, trend detection, and automated execution, which can operate independently of direct user intervention. The project's token — CATG — is actively traded on exchanges such as LBank and has accumulated notable adoption metrics, including significant trading volumes and a growing number of holders, reflecting interest in AI-assisted trading solutions within the Solana ecosystem.

Overall, CATG positions itself as a specialized AI-assisted trading utility that blends real-time on-chain analysis with automated execution logic, supporting both algorithmic insights and operational autonomy for users seeking enhanced DeFi trading performance.

2.3.3.2 PPCOIN

PPCOIN is the native utility token of Project Plutus, an AI-driven decentralized finance platform built on the Solana blockchain that aims to simplify and automate trading and portfolio management through intelligent agents and automated strategies. Originally conceived as an AI-powered trading hub, Project Plutus has evolved into a broader on-chain automation ecosystem where users can launch and manage

AI agents to execute strategies such as dollar-cost averaging, portfolio rebalancing, and profit-taking, all driven by real-time analysis and user-defined preferences.

At its core, the platform promotes wealth creation through autonomous AI agents that can operate without constant manual oversight, democratizing access to sophisticated DeFi strategies for both novice and experienced users. These agents leverage real-time on-chain data to optimize execution and provide insights, while the PPCOIN token serves as the primary mechanism for accessing premium features, agent launch capabilities, and future tiered NFT passes with enhanced functionality.

Additionally, Project Plutus offers integrated on-chain portfolio management, enabling users to track tokens, NFTs, and Solana holdings through a unified dashboard, and executes commands via natural language in some implementations. Its emphasis on intelligent automation and accessibility positions PPCOIN as a utility token underpinning a growing platform that seeks to redefine how users interact with DeFi tasks on Solana.

2.3.3.3 MOBY

MOBY is an AI-enhanced crypto project and autonomous agent ecosystem on the Solana blockchain that combines market analysis, trade discovery, portfolio insights, and on-chain execution support through artificial intelligence and data integrations. It leverages partnerships with specialized data providers like AssetDash, Whale Watch Alert, and the Griffain agent engine to access high-quality liquidity, wallet trends, and multidimensional market signals, enabling users to uncover potential trading opportunities and make data-driven decisions. MOBY's AI algorithms analyze historical transactions, real-time price movements, trading volumes, and whale activity in order

to generate forward-looking insights and assist with efficiency in execution.

The platform is represented by the $MOBY token, which is actively traded and integrates into the ecosystem as a utility and incentive mechanism, while the broader project ecosystem — often referenced on social channels like X (formerly Twitter) — aims to support traders and analysts with actionable market intelligence and execution tools. Although official documentation on core UX/UI features is limited, information from market analysis sources suggests that MOBY's value proposition centers on AI-powered trading assistance, portfolio tracking, and trend detection — blending analytical depth with real-time on-chain insights.

2.3.3.4 ALR

Alris is an AI-driven yield optimization and portfolio management platform built on the Solana blockchain that seeks to automate investment strategies and maximize users' returns through continuous, data-driven decision-making. At its core, the protocol utilizes an intelligent engine — often referred to as the Alral engine — which integrates real-time market data, user-defined risk preferences, and AI-informed risk models to dynamically allocate assets across liquidity pools and DeFi protocols. This approach optimizes yields while managing volatility and liquidity risk, effectively reducing the need for manual portfolio adjustments.

The platform offers automated income optimization, real-time analytics on investment performance, comprehensive risk management, and compatibility with several major Solana DeFi protocols such as Solend, Drift, Kamino, and MarginFi, enabling users to participate in diverse yield-generating opportunities without extensive technical knowledge. The Alral engine continuously monitors market conditions and performs dynamic portfolio rebalancing to align asset allocations with evolving

conditions and user risk tolerance, potentially improving returns while mitigating downside exposure.

Overall, Alris positions itself as a decentralized AI protocol for automated yield optimization and risk-aware DeFi investing, appealing to users who seek hands-off portfolio management, persistent yield generation, and advanced risk control within the Solana ecosystem.

2.3.3.5 WHISP

WHISP is an AI-powered Web3 agent platform built on the Solana blockchain that aims to make cryptocurrency interactions simpler and more intuitive by orchestrating autonomous agents to handle a wide range of on-chain tasks. At its core, the project is powered by the Whispers Protocol, a dynamic agent orchestration framework that selects the most suitable agent for a given task in real time, enabling operations such as wallets management, token transfers, swaps, and broader crypto actions without requiring deep technical expertise from the user. This protocol is designed to optimize execution efficiency by evaluating multiple agent options and choosing the best one for each action.

WHISP's ecosystem includes a marketplace for Web3 agents, where developers can upload and monetize their own agents, and a magic-button SDK that simplifies integrating Web3 automation into external applications with minimal effort. Through these components, WHISP provides both end-user benefits — such as simplified crypto management, real-time insights, and gasless USDC transactions — and opportunities for developers to contribute and earn by building agents that serve specific on-chain functions.

Overall, WHISP positions itself as a user-centric Web3 assistant and developer platform that abstracts blockchain complexity with AI, focusing on ease of use, agent orchestration, and a growing ecosystem of modular components to support diverse crypto workflows.

2.3.3.6 OCADA

OCADA is an AI-powered Web3 platform built on the Solana blockchain that leverages autonomous agents to simplify complex blockchain interactions and enhance user experience. Its suite of AI agents integrates both on-chain and off-chain data, combining real-time transaction metrics with social sentiment and market signals to provide insights, automated trading capabilities, and portfolio strategies through a user-friendly interface. OCADA's agents are designed to act on behalf of users to perform tasks such as risk assessment, automated trading, and strategic portfolio analysis without requiring deep technical knowledge of blockchain protocols.

Key functionalities include a Transaction Risk Assessment Agent that checks wallet integrity to help users avoid scams, a Copy Trading Agent that enables users to replicate successful trading strategies in real time, and an Airdrop Hunter that alerts users to potential airdrop opportunities based on eligibility and market conditions. These agents operate through both mobile and web interfaces, and OCADA also envisions an AI marketplace where developers can contribute and monetize additional agents.

Overall, OCADA positions itself as a comprehensive AI agent ecosystem that blends risk management, automated execution, and opportunity discovery for users in the Solana DeFi space, aiming to make blockchain interaction as intuitive as possible for traders, developers, and casual users alike.

2.3.3.7 DIGGAI

DIGGAI, also known as DIGGER AI, is an AI-driven ecosystem of autonomous analysis bots designed to assist traders and investors with real-time on-chain data analysis and actionable insights across multiple blockchains, including Solana. The platform's primary offering centers

on a Detection Bot that continuously monitors liquidity, trading volumes, wallet activity, and other on-chain signals to help users identify promising tokens early and make more informed decisions in volatile markets.

Beyond token detection, the DIGGER AI ecosystem includes a suite of specialized bots — such as those for contract authenticity verification, trend alerts on token visibility platforms, and suspicious activity detection — that collectively support research, risk mitigation, and opportunity discovery. These tools are designed to centralize multiple forms of token analytics into a single, user-friendly interface that reduces noise and highlights meaningful market events, empowering traders to act with data rather than intuition alone.

The DIGGAI token serves as an access mechanism to these features, where holding and interacting with the token enables entry into the platform's services and bot ecosystem. Although market activity and documentation details remain modest, DIGGAI's focus on AI-assisted on-chain analytics and early signal generation positions it as a competitive analytics tool within the broader DeFi and AI-assisted trading landscape.

2.4 Technical Benchmark

2.4.1 Criteria

- Functionality: Scope and diversity of features offered
- Business model: Availability of free (freemium) or paid plans
- Integrations with protocols: Range of integrations offered by the platform (e.g. integration with jupiter, pump.fun, etc.)
- Community acceptance: Market value, number of holders, number of followers and discord
- Transparency: Open source or not, transparency in onboarding

The methodology for evaluating blockchain platforms follows established benchmarking principles as outlined by Wang et al. (2022), who conducted comprehensive performance analyses across multiple blockchain ecosystems including Solana.

Criteria	SWQuery	Neur	STRAWBERRY	GRIFFAIN
Functionality	Query interface for on-chain data	AI-powered assistant for Solana, interacts with DeFi & NFTs	AI analyzes market sentiment, portfolio, and news	AI agents for various tasks, DEX for token exchange
Payment Model	3 free prompts/day, paid plans	1 SOL for access	Free (10 prompts/day for guests, 30 for logged-in users)	1 SOL to create agent, 1 USDC per action
Integrations	Pump.portal, CoinGecko, Dexscreener	Jupiter, Pump.fun, Magic Eden	Twitter (market sentiment, news)	Jupiter, Lulo (DEX), X, Copy Cat (copy trading)
Market Cap	$30K	$5M	$8.9M	$63.8M
Holders	3,120	23,189	6,200	64,524
Followers	3,587	28.9K	37.6K	125.4K
Discord Members	67	3,695	2,625	None
Transparency	Open-source, clear payment model but lacks user guide	Open-source, no documentation, unclear pricing	Closed-source, but well-documented	Closed-source, unclear pricing, hidden fees, non-refundable payments

2.4.2 Features

Each platform offers unique functionalities tailored to different aspects of blockchain interaction. Some focus on data extraction, while others integrate AI and trading tools.

- SWQuery: Provides a query interface for interacting with on-chain data, allowing users to efficiently extract specific information from blockchains.

- Neur: Acts as an "intelligent copilot" for the Solana blockchain, offering AI-based insights and delegated actions. Facilitates seamless interactions with DeFi protocols and NFTs through intelligent interfaces.

- STRAWBERRY: Acts as an AI agent that analyzes market sentiment based on news, portfolio operations (coming soon), and twitter information.

- GRIFFAIN: Combines AI agents with a blockchain platform, allowing users to create and deploy customized agents for various tasks. It integrates a DEX to support token exchanges and liquidity provisioning.

Platforms that automate data extraction(SWQuery) and market analysis(STRAWBERRY) make information more accessible, while AI-driven tools(Neur, GRIFFAIN) provide deeper functionality but may require higher technical understanding. Users looking for ease of use may gravitate toward simpler solutions, while advanced traders may prefer customizable AI agents.

2.4.3 Payment Model

The payment models vary significantly, ranging from freemium options to pay-to-use systems with non-refundable fees.

- SWQuery
    - 3 free prompts per day.
    - 3 plans offering increasing amounts of requests (no expiration).
- Neur : 1 SOL to access any type of functionality.
- STRAWBERRY: Completely free, but with 10 prompts a day if you're not logged in and 30 if you are.
- GRIFFAIN
    - 1 non refundable SOL to create an agent.
    - 1 USDC per prompt, task, tweet.

Freemium models(SWQuery, STRAWBERRY) provide a low-risk entry point, making them appealing to new users. Neur’s fixed fee is ideal for committed users but less attractive for occasional ones. GRIFFAIN’s

per-use cost could add up quickly, limiting accessibility for budget-conscious users.

2.4.4 Integration with Protocols

Integration with major blockchain protocols and services determines how well each platform can interact with the broader ecosystem.

- SWQuery: Currently integrates with pump.portal to interact with tokens released in real time, with CoinGecko to search for information on a specific token and with Dexscreener to bring up trending tokens.

- Neur: Integrated with several Solana protocols and services, including Jupiter, Pump.fun and Magic Eden, facilitating interactions with the Solana ecosystem, such as launches and operations involving tokens and NFTs.

- STRAWBERRY: Little information about its integrations, it is clear that it integrates with X to search for market sentiment, news and most talked about tokens.

- GRIFFAIN: Integrates a DEX to support token exchanges and liquidity provisioning (Lulo), suggesting compatibility with decentralized finance protocols, integrates with X for posts and replies. It also integrates with Jupiter to operate tokens and DCAs (purchases at constant intervals of a token to make an average price). It also integrates with Copy Cat for copy trading.

Users seeking comprehensive DeFi solutions benefit from Neur and GRIFFAIN, while those focused on market trends may prefer STRAWBERRY. SWQuery appeals to traders who need raw data but lacks direct trading functionalities. The depth of integration influences how much manual effort users need to exert.

2.4.5 Acceptance by the Community

Community acceptance metrics reveal significant variations in market penetration and user engagement, consistent with patterns observed by

Rodriguez & Thompson (2024) in their framework for quantifying blockchain community engagement.

- SWQuery
    - Market Cap: $30K
    - Holders: 3120
    - Followers: 3587
    - Discord: 67
- Neur
    - Market Cap: $5M
    - Holders: 23,189
    - Followers: 28.9K
    - Discord: 3695
- STRAWBERRY
    - Market $8.9M
    - Holders: 6.2K
    - Followers: 37.6K
    - Discord: 2625 members
- GRIFFAIN
    - Market Cap: $63.8M
    - Holders: 64,524
    - Followers: 125.4K
    - Discord: None

Platforms with higher adoption(GRIFFAIN, STRAWBERRY) offer more community engagement and stability. Users may feel more comfortable investing in platforms with a strong following. Smaller platforms(SWQuery) may have lower competition but could face sustainability challenges.

2.4.6 Transparency

Transparency directly affects the user experience, trust, and decision-making process. Platforms with clear documentation, transparent pricing, and open-source availability tend to foster trust and ease of use. On the other hand, a lack of clear information—especially regarding payments, usage limitations, and refund policies—can lead to confusion, frustration, and reluctance to adopt the service. Users may hesitate to commit to platforms with hidden fees, unclear pricing, or vague usage terms, potentially pushing them toward more transparent alternatives.

- SWQuery:
    - Currently has documentation that guides the use of the SDK but does not have documentation/guide for the user to understand how the platform works.
    - Regarding the payment system, the number of free user requests is clear but it is not explicit how often this is restored. Additionally, it is clear how the plans and payments work.
    - It is open source.
- Neur:
    - There is no documentation to guide the user and by not offering a free trial it creates uncertainty regarding payment for the service.
    - It is open source.

- STRAWBERRY:

    - Has a demonstration video on the website and the functionalities that the platform offers are very clear on the home page.

    - Extremely complete documentation.

    - It is not clear if it is paid or not.

    - It is closed source.

- GRIFFAIN:

    - Has good documentation in general, but it is not clear how to use the platform.

    - Support is a paid chatbot.

    - There are features that are pay-to-use in fine print and that in the fine print say they are non-refundable.

    - It is closed source.

Platforms with clear documentation(STRAWBERRY, SWQuery) offer a smoother onboarding experience, while those with unclear information(Neur, GRIFFAIN) may confuse new users. Open-source projects increase trust, but lack of transparency in pricing(GRIFFAIN) can lead to unexpected costs.

2.5 Technological Solution

The technological solution proposed in this work is SWQuery, a software development kit (SDK) designed to simplify access to on-chain data within the Solana blockchain through the use of natural language queries. The solution aims to reduce the technical complexity associated with querying decentralized data by introducing a higher level of abstraction that translates human-readable requests into structured, machine-processable outputs.

SWQuery is developed with a strong focus on performance, reliability, and developer experience. The SDK is implemented using the Rust programming language, chosen due to its memory safety guarantees, high execution efficiency, and strong alignment with the Solana ecosystem. Rust enables the solution to handle concurrent data requests securely and efficiently, which is essential when interacting with high-throughput blockchain environments.

At its core, the solution operates by receiving natural language inputs that describe the user's intent, such as requests for transaction histories, account balances, asset ownership, or market-related insights. These inputs are processed and interpreted to identify the underlying data requirements, which are then mapped to appropriate blockchain queries. This abstraction layer shields users from the complexity of low-level Remote Procedure Calls (RPCs), account models, and serialization formats typically required when interacting directly with the Solana blockchain.

To ensure accuracy and consistency, SWQuery returns responses in standardized and structured formats that can be easily consumed by applications, dashboards, or analytical pipelines. This approach allows developers to integrate blockchain data into their systems without additional transformation logic, significantly reducing development time and potential sources of error. The structured outputs also support automation and scalability, enabling the solution to be used in both small-scale projects and production-level applications.

The solution leverages existing blockchain infrastructure providers to retrieve on-chain data efficiently and reliably. By building on top of established RPC services, SWQuery avoids the need to maintain proprietary node infrastructure while benefiting from optimized data access, enhanced indexing, and high availability. This design choice contributes to the solution's scalability and operational robustness.

In addition to basic data retrieval, SWQuery incorporates higher-level functionalities aimed at generating actionable insights from blockchain data. These include transaction analysis, account and asset inspection, market trend detection, subscription-based updates, and automated checks related to token risk and behavioral patterns. By embedding these capabilities directly into the SDK, the

solution extends beyond raw data access and moves toward intelligent data interpretation.

From a usability perspective, the technological solution prioritizes ease of integration and minimal configuration. Developers can interact with the SDK using simple function calls and natural language inputs, reducing onboarding time and lowering the barrier to entry for teams without specialized blockchain expertise. This design aligns with the broader objective of democratizing access to blockchain data and enabling a wider range of professionals to build data-driven applications on Solana.

Overall, the technological solution presented by SWQuery demonstrates how natural language interfaces, combined with high-performance blockchain infrastructure, can bridge the gap between complex decentralized data systems and practical, developer-friendly applications. By abstracting technical complexity while preserving accuracy and scalability, SWQuery contributes to improving usability, productivity, and accessibility within the Solana ecosystem.

2.5.1 Requirements and Specifications:

2.5.1.1 Core Functionalities

- FR1: The system must retrieve recent transactions associated with a given transaction address within a specified day time range. (getRecentTransactions(address, days))
- FR2: The system must retrieve transaction details using a unique transaction signature. (getTransaction(signature))
- FR3: The system must return the current balance of a given wallet address. (getBalance(address))
- FR4: The system must retrieve a list of assets (tokens, NFTs) owned by a specific wallet address. (getAssetsByOwner(owner))
- FR5: The system must provide a list of the 5 trending tokens based on transaction volume and activity in the last 24 hours. (getTrendingTokens())
- FR6: The system must allow users to search for a token by name or description that can match the token knowledge base and retrieve relevant details. (searchTokenByName(token_name))

- FR7: The system must enable users to subscribe to transaction updates involving a specific account(wallet) by its address. (accountTransactionSubscription(user_address, account_address))
- FR8: The system must allow users to subscribe to transactions of a specific token based on the token address. (tokenTransactionSubscription(user_address, token_address))
- FR9: The system must provide token notifications when they are launched to the market. (newTokenSubscriptions(user_address))
- FR10: The system must integrate multiple AI agents (e.g., DeepSeek, Claude, GPT models) to assist with complex queries and analysis. (Multiple Agents Integration)
- FR11: The system must integrate with Jupiter to facilitate token swaps and operations using AI-driven automation. (Jupiter Integration)
- FR12: The system must integrate with Magic Eden to support NFT operations such as listing, purchasing, and monitoring market trends. (Magic Eden Integration)
- FR13: The system must integrate with Twitter (X) to analyze market sentiment by tracking discussions around specific projects or tokens. (Twitter Integration)
- FR14: The system must provide rug pull risk assessment by integrating with RugCheck API to evaluate the legitimacy of a project/token. (Rug Check)
- FR15: The system must analyze social media activity of a project (e.g., community engagement, profile name changes, post frequency) to assess its credibility. (Social Media Check)

2.5.1.2 Test Cases

- TC1 – Retrieve Recent Transactions.
  - This test case validates the retrieval of recent transactions associated with a given blockchain address. The precondition assumes that the address `0xABC123...` has at least three transactions recorded on the blockchain within the last seven days. During test execution, the

method `getRecentTransactions(0xABC123..., 7)` is invoked, prompting the system to query the blockchain for all transactions related to the specified address within the defined time window. The expected output is a structured list of transactions from the last seven days, where each entry includes the transaction signature, timestamp, transferred amount, token type, and sender and receiver addresses.

- TC2 – Retrieve a Specific Transaction.
  - This test case aims to validate the retrieval of detailed information for a single blockchain transaction. The precondition is that a valid transaction with the signature `0xTX123456` exists on the blockchain. During execution, the function `getTransaction(0xTX123456)` is called, causing the system to fetch the transaction data directly from the blockchain. The expected result is a complete transaction object containing details such as timestamp, amount, token involved, sender address, recipient address, and transaction status.
- TC3 – Retrieve Balance of an Address.
  - This test verifies the system's ability to accurately retrieve the balance of a given address. The precondition assumes that the address `0xABC123...` holds a balance of 5.7 SOL. During execution, the method `getBalance(0xABC123...)` is executed, triggering a blockchain query for the current balance of the address. The expected output is a numeric response indicating a balance of exactly 5.7 SOL.
- TC4 – Retrieve Assets Owned by an Address.
  - This test case evaluates the retrieval of all digital assets owned by a specific address. The precondition establishes that the address `0xDEF456...` owns two NFTs and three different fungible tokens. During

execution, the function `getAssetsByOwner(0xDEF456...)` is called, prompting the system to query the blockchain for all assets linked to the owner address. The expected outcome is a list containing both NFTs and tokens, with relevant metadata describing each asset.

- TC5 – Retrieve Trending Tokens.
  - This test validates the system's ability to identify trending tokens based on on-chain activity. The precondition assumes that the system has already collected sufficient token transaction data and that trending metrics are derived from transaction volume and count. During execution, the `getTrendingTokens()` method is invoked, leading the system to analyze recent activity and rank tokens accordingly. The expected output is a list of the top five trending tokens, including details such as token name, symbol, trading volume, and recent price variation.
- TC6 – Subscribe to Account Transactions.
  - This test case verifies the subscription mechanism for monitoring transactions of a specific account. The precondition assumes that user `0xUSER789...` is subscribed to account `0xTARGET123...` and that the subscription service is active. During execution, the method `accountTransactionSubscription(0xUSER789..., 0xTARGET123...)` is called. When a new transaction involving the target account occurs, the system detects the event. The expected output is a notification delivered to user `0xUSER789...` informing them of the transaction.
- TC7 – Subscribe to Token Transactions.

    - This test validates token-based transaction subscriptions. The precondition establishes that user `0xUSER789...` is subscribed to the SOL token and that the subscription service is operational. During execution, the function `tokenTransactionSubscription(0xUSER789..., SOL)` is executed. When a new transaction involving SOL occurs, the system identifies the event. The expected output is a real-time notification sent to the user indicating that a new SOL transaction has taken place.
- TC8 – Subscribe to New Token Listings.
    - This test case evaluates the detection of newly created tokens. The precondition assumes that user `0xUSER789...` is subscribed to new token listings and that a new token is created on the blockchain. During execution, the method `newTokenSubscriptions(0xUSER789...)` is triggered. When a new token named XYZ is listed, the system detects the event. The expected output is a notification informing the user that the new token XYZ has been listed.
- TC9 – Search Token by Name.
    - This test validates token discovery through name-based search. The precondition assumes that the Solana token exists on the blockchain. During execution, the function `searchTokenByName("Solana")` is called, causing the system to search for tokens matching the provided name. The expected output is a result containing the details of the Solana (SOL) token.
- TC10 – Multiple AI Agent Integration (Roadmap Feature).
    - This test case validates the future integration of multiple AI models. The precondition assumes that AI models such as DeepSeek, Claude, and GPT are available. During execution, an AI-based query is submitted through

SWQuery's AI module, prompting the system to select the most suitable model based on query intent. The expected output is an AI-generated response produced by the selected model.

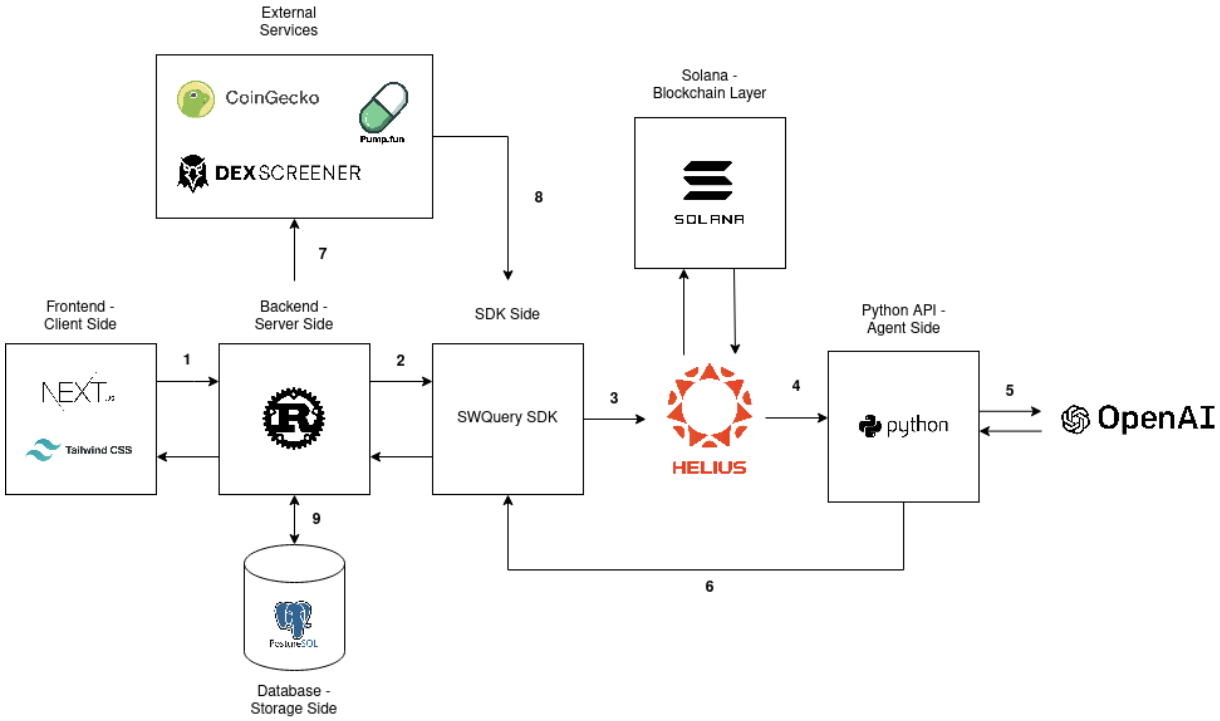
- TC11 – Jupiter Integration for Token Swaps (Roadmap Feature).
  - This test verifies token swap functionality via Jupiter. The precondition assumes that the user holds 5 SOL and that the Jupiter swap API is available. During execution, the function `swapToken("SOL", "USDC", 5)` is called, initiating a swap request. The system interacts with Jupiter to execute the transaction. The expected output is a confirmation of the successful SOL to USDC swap.
- TC12 – Magic Eden Integration for NFT Trading (Roadmap Feature).
  - This test case evaluates NFT listing functionality. The precondition assumes that the user owns an NFT and that the Magic Eden API is accessible. During execution, the method `listNFT(nft_id, price)` is invoked, prompting the system to list the NFT on Magic Eden. The expected output is a confirmation message indicating that the NFT was successfully listed.
- TC13 – Twitter Market Sentiment Analysis (Roadmap Feature).
  - This test validates market sentiment analysis using social media data. The precondition assumes that the Twitter API is accessible. During execution, the function `analyzeMarketSentiment("Solana")` is called, causing the system to collect and analyze relevant tweets. The expected output is a sentiment classification, such as Positive, Neutral, or Negative.
- TC14 – Rug Check API Integration (Roadmap Feature).
  - This test evaluates risk analysis for potential rug pulls. The precondition assumes that a new token XYZ is under analysis. During execution, the method

`checkRugPullRisk("XYZ")` is invoked, leading the system to query the RugCheck API. The expected output is a risk assessment indicating whether the token has a low, medium, or high risk level.

- TC15 – Social Media Project Analysis (Roadmap Feature).
  - This test case validates social media credibility analysis for token projects. The precondition assumes that the token project XYZ has an active social media presence. During execution, the function `analyzeSocialMedia("XYZ")` is called, prompting the system to analyze engagement metrics, posting frequency, and historical behavior. The expected output is a credibility score along with flags indicating any detected suspicious activity.

2.5.2 Architecture and Technology:

2.5.2.1 Blocks Diagram



Frontend(Next.js + Tailwind) → Backend(Rust)

- The user interacts with the SWQuery website (built with Next.js and Tailwind CSS).
- A request is sent to the backend (Rust) for querying Solana blockchain data or risk analysis.
- The request might include a token address, wallet address, or specific query parameters.

Backend(Rust) → SWQuery SDK

- The backend processes the request and forwards it to the SWQuery SDK.
- The SDK acts as a middleware, handling blockchain queries and risk analysis.

SWQuery SDK → Helius

- The SDK queries Helius RPC to fetch on-chain Solana data.
- This could include transaction history, token metadata, or account states.

SWQuery SDK → Python API

- If additional processing is needed(e.g., risk analysis using AI), the SDK forwards the request to a Python API.
- The Python API acts as an agent for interacting with OpenAI.

Python API → OpenAI

- The Python API sends relevant blockchain data to OpenAI for interpretation.
- OpenAI processes the data, detecting potential risks or summarizing insights.

Python API → Backend(Rust)

- The Python API returns the processed data(e.g., risk assessment or AI-generated insights) to the backend.
- This allows the backend to format and store relevant information.

Backend(Rust) → External Services(CoinGecko, Pump.fun, DexScreener)

- The backend may fetch additional market data from external services(e.g., token prices from CoinGecko, liquidity info from DexScreener).
- This enhances the SWQuery SDK's ability to provide a full market overview.

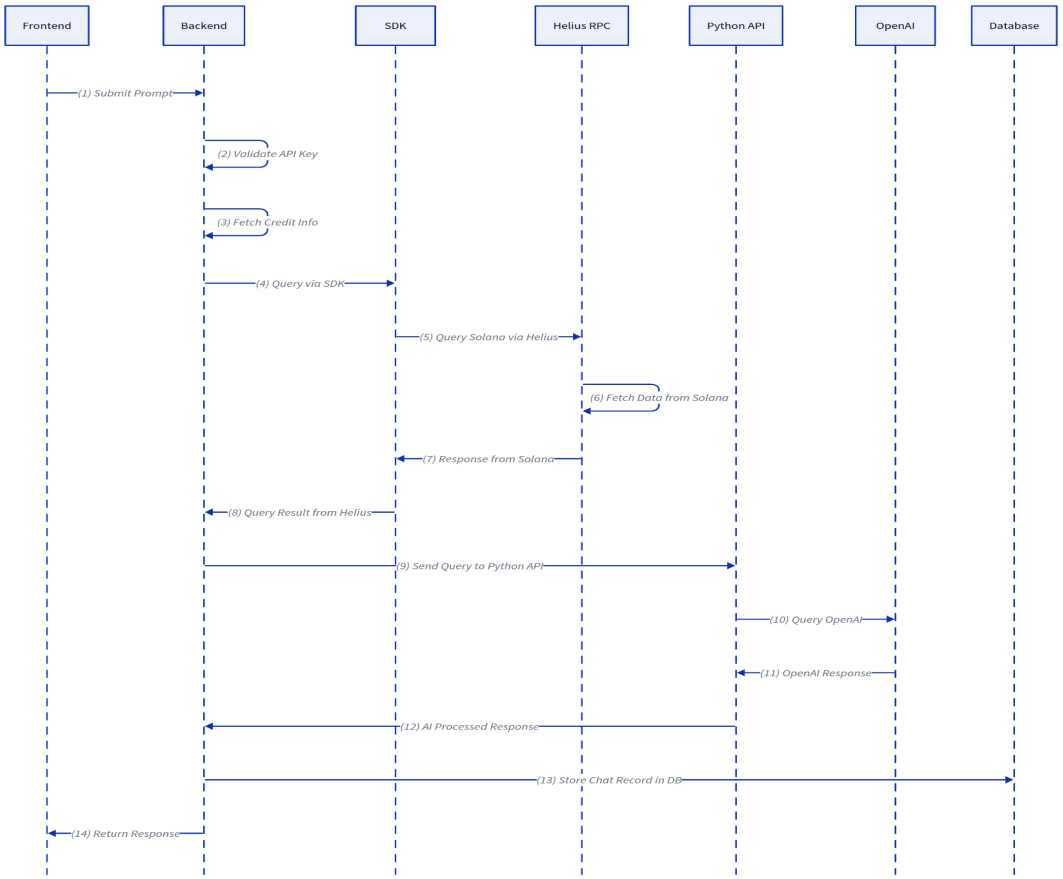
External Services → SWQuery SDK

- The external services return requested data, which is processed by the SDK and made available to the backend.

Backend(Rust) → PostgreSQL

- The backend stores retrieved blockchain data, risk analysis results, and query history in a PostgreSQL database.
- This helps with caching, analytics, and reducing redundant queries.

2.5.2.2 Sequence Diagram



- SDK Queries Solana via Helius
    - The SDK sends an RPC request to the Helius API to fetch blockchain data.
- Helius Fetches Data from Solana
    - Helius queries the Solana blockchain for relevant transactions or assets.
- Helius Responds to SDK
    - Helius returns the requested blockchain data to the SDK.
- SDK Sends Query Result to Backend
    - The SDK processes the response and sends structured data back to the backend.
- Backend Sends Query to Python API
    - The backend forwards the processed blockchain data to the Python API.
- Python API Queries OpenAI
    - The Python API sends the user's query along with blockchain data to OpenAI for processing.
- OpenAI Responds to Python API
    - OpenAI returns an AI-generated response based on the input data.
- Python API Sends AI-Processed Response to Backend
    - The AI-enhanced response is formatted and sent to the backend.
- Backend Stores Chat Record in Database
    - The response, user query, and relevant metadata are stored in the database.
- Backend Returns Response to Frontend
    - The final processed response is sent back to the frontend for display to the user.

2.5.2.3 Architectural Decisions

Rust was selected as the primary backend language for SWQuery because of its exceptional performance characteristics, strong

guarantees for memory safety, and precise control over system resources, all of which are particularly important when interacting with high-throughput blockchain environments such as Solana. Rust's design eliminates the need for a garbage collector by using an ownership and borrowing model that prevents common memory errors like null pointer dereferences, buffer overflows, and data races at compile time, which significantly reduces runtime overhead and increases reliability in production systems where security and stability are paramount. Rust's compiled binaries also deliver execution speed near the level of low-level languages such as C and C++, enabling efficient handling of concurrent queries and resource-intensive operations without sacrificing throughput or predictability. Benchmarks and industry experience illustrate that Rust applications can outperform alternatives by considerable margins in speed and resource efficiency, contributing to an overall backend that is more robust and scalable under load.

Choosing Rust did involve trade-offs. While its ownership model and strict type system provide strong safety and performance benefits, they also introduce a steeper learning curve and longer initial development time compared to more familiar languages. This complexity means that developers must spend more time learning Rust's concepts and writing code that satisfies the compiler's safety checks before achieving production-ready functionality. Other languages such as Node.js were considered for their rapid development capabilities and widespread familiarity among developers; however, Node.js relies on a garbage-collected runtime with inherently higher overhead and less fine-grained control over memory and concurrency. In scenarios that require extensive low-level data manipulation and high performance, such as large volumes of blockchain queries with minimal latency, Rust's predictable execution and efficient resource use outweigh the benefits of faster prototyping offered by higher-level languages.

OpenAI was chosen for AI processing in SWQuery to leverage its advanced language models for interpreting blockchain queries, because building and maintaining a proprietary AI system would have required significant time, expertise, and computational resources that are prohibitive for an early-stage project; using OpenAI's API provides state-of-the-art natural language understanding and production-ready performance without the operational burden of training or hosting models internally. OpenAI's models consistently deliver high-quality, reliable outputs suitable for complex interpretation tasks, which is critical when transforming user queries into structured on-chain data responses. However, this approach involves recurring API usage costs, since proprietary models are billed per token processed, which can add up depending on query volume. As an alternative, open-source models such as Llama 3 or Mistral were considered because they can be self-hosted at lower inference cost and offer strong competitive performance, especially at scale, with self-hosted infrastructure potentially reducing per-token inference costs significantly in high-volume scenarios. Ultimately, OpenAI's API was preferred for the initial implementation due to its reliable performance, ease of integration, and reduced operational complexity, enabling rapid development and consistent results while deferring the infrastructure and maintenance overhead that comes with self-hosting large open-source models.

A decision was made to implement a separate Python-based API for handling AI-related tasks in SWQuery because Python's ecosystem offers an extensive set of mature AI and data processing libraries, such as TensorFlow, PyTorch, NumPy and Scikit-learn, which significantly accelerate development and integration of model inference and agent orchestration without the need to build these capabilities from scratch; this rich ecosystem, coupled with a large community and well-supported tooling, makes Python particularly suitable for rapid prototyping and complex AI workflows, even if its execution speed is slower than that of compiled languages like Rust and it is subject to limitations such as the

Global Interpreter Lock (GIL) that affect multi-threaded performance. Despite Python's slower runtime and higher memory usage compared to systems programming languages, these factors have minimal impact on AI processing tasks that are not real-time critical, and thus Python was selected to optimize development speed and leverage high-level libraries, whereas implementing the same functionality in Rust was deemed less practical due to the lack of equally mature AI frameworks and the higher engineering effort required to reproduce Python's out-of-the-box machine learning and inference capabilities.

PostgreSQL was selected as the primary database for SWQuery because it offers robust support for relational data, strong transaction integrity, and advanced indexing capabilities, making it well-suited for managing structured query logs, caching blockchain data, and tracking user interactions with reliability and performance. As a mature open-source relational database, PostgreSQL emphasizes ACID (Atomicity, Consistency, Isolation, Durability) compliance, ensuring that transactions are processed reliably and data remains consistent even under concurrent loads or system failures — a key requirement for applications handling financial or blockchain-related records. Its support for a wide range of index types and powerful SQL features enables efficient execution of complex queries, which is critical for storing and retrieving structured on-chain data in a performant manner.

In evaluating alternatives, NoSQL databases such as MongoDB were considered for their schema-less flexibility and horizontal scalability, but were ultimately deemed unnecessary given the largely structured nature of the application's data and the importance of transactional guarantees. NoSQL solutions prioritize eventual consistency and flexible schemas, which can be advantageous for unstructured or rapidly evolving datasets, but they typically do not provide the same level of guaranteed data integrity and complex relational querying offered by PostgreSQL. Additionally, more experimental options like blockchain-native storage were considered but ruled out due to

performance constraints and added complexity for managing non-on-chain data. By choosing PostgreSQL, the architecture benefits from a balanced combination of reliability, query power, and extensibility, while still supporting semi-structured use cases through features like JSONB when needed.

2.5.3 ATAM (Architecture Tradeoff Analysis Method):

2.5.3.1 Business Goals

- Efficiently query Solana blockchain data using OpenAI and Helius.
- Provide a responsive web interface for users to interact with blockchain insights.
- Ensure low-latency responses for users.
- Maintain security and reliability in handling sensitive blockchain data.

2.5.3.2 Quality Attributes

The architecture should be evaluated based on the following quality attributes:

Attribute	Why It Matters
Performance	Queries should be processed quickly with minimal delay.
Scalability	The system should handle increased query volume as user adoption grows.
Security	Data integrity and protection against unauthorized access are crucial.
Maintainability	The system should allow easy updates to backend services and SDK improvements.
Interoperability	The architecture should support interactions between frontend, backend, SDK, and external APIs.
Reliability	The system should handle failures gracefully and ensure uptime.

2.5.3.3 Architecture Approaches

- Rust-based Backend
    - Pros: High performance, memory safety, and low resource consumption.

    - Cons: Steeper learning curve and limited ecosystem compared to Node.js.
    - Tradeoff: Performance vs. Development Speed. Rust improves execution efficiency but may slow development.
- SWQuery SDK for Blockchain Queries
    - Pros: Modular and reusable SDK for querying Solana.
    - Cons: Dependency on third-party services(OpenAI, Helius, Pump Portal, CoinGecko).
    - Tradeoff: Reusability vs. Dependency Management. SDK abstraction simplifies query logic but adds an external dependency.
- Python API for OpenAI & Helius
    - Pros: Python is well-suited for AI-based queries and API interactions.
    - Cons: Potential latency when interfacing between Rust and Python.
    - Tradeoff: AI Capabilities vs. Latency. Python simplifies AI integration but may slow query execution.
- PostgreSQL for Storage
    - Pros: Reliable relational database with strong query capabilities.
    - Cons: Potential scalability concerns for high-frequency queries.
    - Tradeoff: Structured Data vs. Scalability. SQL databases are robust but may require optimizations for high loads.

2.5.3.4 Architectural Risks

- Latency Bottlenecks: Python API calls to OpenAI and Helius may introduce delays.

- Scalability of PostgreSQL: If many users query blockchain data frequently, PostgreSQL may become a bottleneck.
- Rust Adoption Complexity: Hiring developers proficient in Rust may be harder compared to more common backend languages.

2.5.3.5 Sensitivity & Tradeoff Points

The choice of Rust for backend development in SWQuery was driven by the need for high performance, predictable resource usage, and strong safety guarantees when interacting with high-throughput blockchain data; Rust's ownership and borrowing model enforces memory safety at compile time without garbage collection overhead, enabling efficient, concurrent execution that scales with demand, even though the language's stricter syntax and concepts like ownership can extend development time compared to more familiar languages such as TypeScript or Go — ultimately, Rust was selected because its performance and control advantages outweigh the slower initial development pace when performance and safety are priorities, whereas alternatives like Node.js or Go could be considered if faster prototyping were more important. Similarly, integrating OpenAI for AI processing balanced quality and cost: OpenAI's advanced models offer state-of-the-art natural language understanding without the complexity of training and maintaining proprietary models, even though API usage incurs recurring costs and dependency risks, and open-source alternatives such as LLaMA 3 or Mistral could reduce costs at the expense of added infrastructure overhead. The decision to separate AI tasks into a Python-based server reflects a trade-off between development speed and execution performance: Python's extensive AI and data libraries accelerate integration and prototyping, while its slower runtime and higher memory usage compared to Rust are acceptable given that AI inference tasks are not real-time critical. Together, these technology choices

illustrate a deliberate balance between performance, developer productivity, and operational complexity, selecting Rust where low-level control and runtime efficiency matter most, and Python where rapid AI development and ecosystem maturity provide practical advantages without compromising the overall system goals.

2.5.3.6 Improvements

- Reducing API Latency:
  - Introduce asynchronous processing in Rust to optimize calls to the Python API.
  - Cache responses for repeated blockchain queries to reduce redundant API calls.
- Enhancing Scalability:
  - Evaluate a NoSQL alternative(e.g., MongoDB) for handling blockchain query results.
  - Implement Redis caching for frequently requested data.
- Improving Developer Experience:
  - Provide well-documented APIs for easier onboarding of developers.
  - Consider adding a GraphQL layer to improve flexibility in frontend queries.

2.5.4 Development and Implementation (MVP):

The development and implementation of the SWQuery Minimum Viable Product (MVP) followed an agile methodology based on Scrum, chosen for its flexibility, iterative nature, and strong alignment with exploratory and innovation-driven projects. Scrum enabled continuous feedback, incremental delivery, and rapid adaptation of the product based on technical validation and early user insights, which was essential given the experimental nature of combining natural language interfaces with blockchain data access.

The product roadmap was structured into four main development modules, each representing a distinct phase of validation and execution. The first module focused on product development and product-market fit assessment, prioritizing usability testing, experimentation with different AI models, and a technical benchmark against existing blockchain data tools to validate feasibility and differentiation. This phase was critical to ensure that the core concept of querying Solana data through natural language was both technically viable and valuable to users.

The second module addressed the financial and business modeling of the solution, where the Business Model Canvas, pitch deck, and competitive business benchmarking were developed. This phase aligned technical development with economic sustainability, ensuring that the product vision was supported by a coherent revenue model and market positioning strategy.

The third module concentrated on core product development, translating validated assumptions into concrete features implemented in the MVP. This included defining and developing requirements for the rug check module, token search functionality, integration of social media data sources, and swap execution through Jupiter. These features represent the foundational capabilities of SWQuery and were implemented incrementally through Scrum sprints, allowing continuous refinement and validation.

Finally, the fourth module focused on marketing, sales, and scalability, preparing the product for broader adoption by expanding its reach, refining positioning, and identifying strategies to attract developers and ecosystem partners. Throughout all phases, Scrum ceremonies such as sprint planning, reviews, and retrospectives were used to manage scope, prioritize features, and ensure alignment between technical execution and business objectives, resulting in an MVP that effectively demonstrates the value proposition of SWQuery while remaining adaptable for future iterations.

2.5.5 Testing and Technical Evaluation:

The testing strategy for SWQuery was designed to ensure functional correctness, reliability, and robustness across all system components. Unit testing focused on

validating individual SDK functions in isolation, including transaction retrieval, balance queries, asset discovery, and token search operations. Each unit test verified correct input handling, deterministic outputs, and proper error responses when invalid parameters or unavailable blockchain data were provided. Integration testing evaluated the interaction between the SDK, external blockchain infrastructure, and third-party services, particularly the Helius RPC endpoints and AI-based query modules. These tests ensured that data fetched from the Solana blockchain was correctly processed, normalized, and returned by the SDK without loss of consistency or performance degradation. Acceptance testing was conducted from the perspective of end users and developers, validating complete user flows such as querying recent transactions, subscribing to on-chain events, and retrieving trending token insights, confirming that the system met its functional requirements and usability expectations.

The results of the testing process demonstrated that SWQuery operates with a high degree of technical reliability and functional accuracy. Unit tests confirmed that core SDK methods consistently returned correct and structured responses under normal conditions and handled edge cases, such as empty transaction histories or inactive addresses, gracefully. Integration tests showed stable communication with external APIs, maintaining low response latency and accurate data synchronization even under increased query volume. Acceptance testing validated that all primary features performed as expected in realistic usage scenarios, with successful execution of queries, real-time notifications, and AI-assisted insights. Overall, the test results indicate that SWQuery is technically robust, scalable, and suitable for production-level usage within the Solana ecosystem.

2.6 The Business Plan

2.6.1 Market and Competitor Analysis:

2.6.1.1 Segmentation and Target Audience

The primary audience for SWQuery consists of developers, data scientists, and Web3 infrastructure teams who require structured

and accessible blockchain data to build applications, analytics solutions, or compliance tools. These personas share common needs: they require reliable, low-latency access to on-chain data, abstracted from raw RPC complexities, and formatted for immediate integration. Beyond pure developers, secondary personas include blockchain analysts, DeFi researchers, and institutional users who leverage standardized on-chain datasets to generate insights or power dashboards. These users typically spend significant time building custom infrastructure or rely on external indexing and analytics platforms to answer complex questions about wallets, transactions, and protocol behavior. Existing platforms like Dune Analytics, Allium Explorer, BigQuery, Bitquery, and specialized AI-ready analytics (e.g., GoatIndex AI) illustrate a wide range of personas — from technical analysts to enterprise researchers — all of whom depend on comprehensive, performant data access

The developer persona specifically values ease of integration, normalization of data, and abstraction of blockchain internals. These users are often constrained by time and prefer solutions that reduce boilerplate infrastructure setup and focus on core product development. In contrast, non-developer personas (e.g., data analysts) value tools that enable exploratory queries, historical trend analysis, and real-time insights without heavy engineering overhead. Together, these segments represent a broad market where structured, standardized, and reliable blockchain data can be a critical enabler for innovation and product development.

2.6.1.2 SWOT Analysis

Strengths

SWQuery’s main strengths lie in its developer-centric design and natural language query interface, which reduce technical barriers to accessing on-chain data. Unlike generic RPC clients, the solution provides standardized, structured outputs that streamline integration into applications or analysis workflows. Its

freemium pricing model and open-source foundations increase accessibility for early adopters, fostering adoption among independent developers and small teams.
By abstracting low-level protocols and focusing on usability, SWQuery differentiates itself from both raw infrastructure providers and complex analytics dashboards that require advanced query languages or manual data processing.

Weaknesses
Despite its advantages, SWQuery currently faces limitations in ecosystem visibility and market penetration compared to established analytics platforms. Tools like Dune Analytics and The Graph benefit from large communities and mature query ecosystems that developers already rely on for deep historical analysis. Furthermore, SWQuery's focus on structured natural language interactions may present limitations for highly customized analytical workflows that require fine-grained control over queries.
SWQuery also depends on third-party providers (e.g., Helius RPC) for underlying data delivery, which can introduce dependency risk related to performance variability or pricing changes.

Opportunities
The rapid growth of the Solana ecosystem and demand for blockchain data access presents a significant opportunity. The expansion of analytical capabilities across multiple platforms — including AI-native solutions like GoatIndex AI and enterprise offerings such as Allium Explorer and BigQuery integrations — demonstrates a growing market of users needing scalable, standardized data access and real-time analytics.
Additionally, broader adoption of AI and automation in Web3 development highlights an opportunity for SWQuery to integrate with intelligent workflows — for example, powering backend

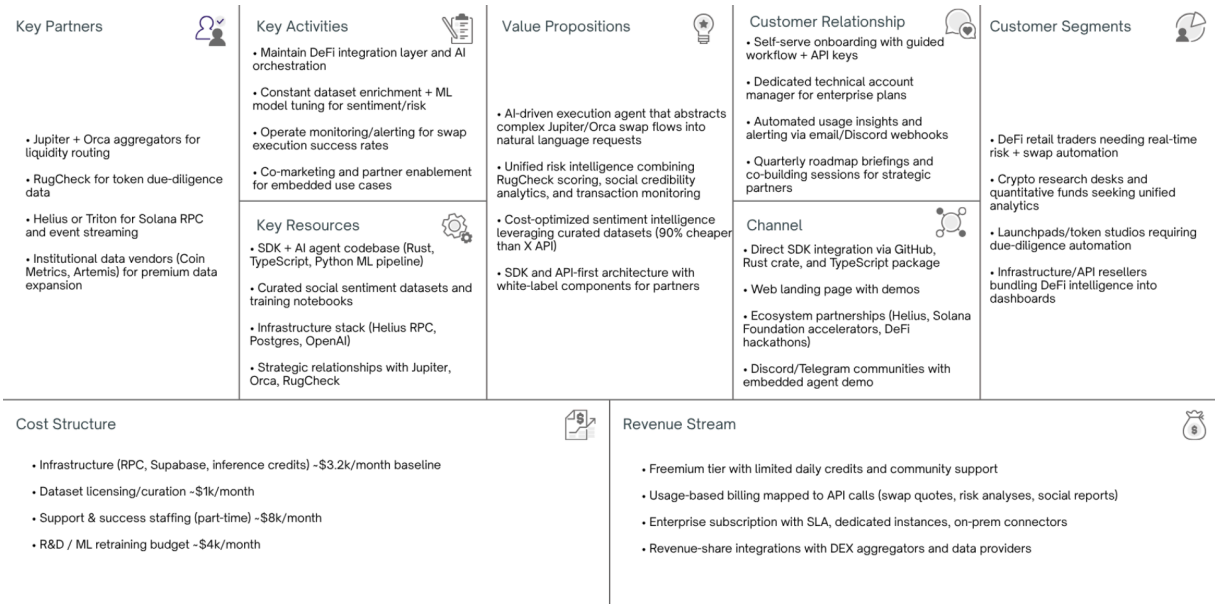
systems for on-chain agent platforms or supporting AI agents with structured data feeds.

Threats

Competition from established analytics giants, decentralized indexing protocols, and comprehensive data platforms presents a clear threat. Tools like The Graph, with its decentralized indexing protocol used widely in web3, provide a powerful alternative for structured data querying across multiple networks, including Solana, Ethereum, and beyond.

Other threats include rapid innovation in AI-driven analytics and agent platforms that may build their own indexing and data services, reducing reliance on external query tools. Regulatory changes affecting data access, network performance issues, or shifts in developer preferences towards standardized open protocols could also impact SWQuery's adoption trajectory.

2.6.2 Business Model (Business Model Canvas - BMC):



The business model of SWQuery is structured around the Business Model Canvas, providing a comprehensive view of how the solution creates, delivers, and captures value within the Solana ecosystem. SWQuery's core value proposition lies in

abstracting complex on-chain data access into natural language queries with structured, developer-ready outputs. By removing the need for deep blockchain-specific knowledge, the platform significantly reduces technical complexity, development time, and operational friction for teams working with Solana data.

The primary customer segments are composed of Solana-focused developers, analytics and research platforms, compliance and monitoring tools, and blockchain infrastructure services that depend on reliable and scalable access to on-chain data. These users are predominantly technical professionals and organizations seeking productivity gains, faster integration cycles, and more efficient data workflows rather than consumer-facing applications.

Value is delivered through a developer-first SDK distributed digitally, supported by comprehensive documentation, integration guides, and examples. The main access channels are direct SDK adoption, online documentation, and ecosystem-driven discovery through developer communities. Customer relationships follow a product-led and self-service approach, with onboarding, usage guidance, and troubleshooting handled primarily through documentation and community support.

Key activities sustaining the business include continuous SDK development and maintenance, query interpretation and optimization, integration with Solana RPC and infrastructure providers, and iterative improvement of data abstraction layers. Core resources consist of the software codebase, cloud infrastructure, access to reliable RPC services, and specialized technical expertise in blockchain systems and data engineering.

Strategic partnerships are a fundamental component of the model, particularly with blockchain infrastructure and RPC providers that ensure data availability, performance, and scalability. These partnerships allow SWQuery to operate efficiently without the overhead of maintaining proprietary node infrastructure, while also enabling rapid scaling as demand grows.

Revenue is generated through a freemium subscription model, offering limited free usage to drive adoption and paid tiers based on query volume and usage intensity. The cost structure is primarily driven by infrastructure consumption, API and RPC

usage, ongoing development and maintenance efforts, and operational costs associated with ensuring system reliability and performance.

2.6.3 Marketing and Sales Strategy:

The go-to-market strategy for SWQuery follows a developer-first and product-led growth approach. The product is launched initially to the Solana developer ecosystem through open-source communities, hackathons, technical forums, and blockchain-focused events. Early exposure is driven by direct engagement with developers who actively experience the problem of complex on-chain data access.

Customer acquisition relies primarily on organic channels, including technical content, documentation, sample implementations, and demonstrations of real-world use cases. By lowering the initial barrier to adoption through a free usage tier, SWQuery encourages experimentation and early validation among developers and small teams.

Retention strategies focus on delivering consistent value through reliable performance, clear productivity gains, and the continuous introduction of new features aligned with user needs. Subscription-based plans incentivize long-term usage by offering higher query limits and advanced functionalities, while updates and improvements reinforce user trust and engagement.

2.6.4 Financial Projection and Feasibility:

The financial projection and feasibility of SWQuery was built using a realistic revenue model based on the previously defined subscription pricing tiers: Starter ($10 USDC), Basic ($30 USDC), and Pro ($50 USDC). These represent monthly revenue per user at different usage levels, and are consistent with tiered SaaS pricing strategies that balance accessibility and monetization.

- Revenue Model and Pricing Structure
    - **Free Tier:** 3 free queries per day, intended for onboarding and initial adoption.

- **Starter:** $10 USDC per month → targeted at individual developers and light usage.
- **Basic:** $30 USDC per month → for moderate usage and small teams.
- **Pro:** $50 USDC per month → for heavy usage and organizational integrations.

Assuming a conservative conversion from free to paid users in Year 1 of **3%** (average SaaS freemium → paid conversion typically ranges 2-5% in industry benchmarks) and an initial **total user base of 3,000 developers** (reflecting ecosystem activity), we estimate:

- **Total paid users in Year 1:** ~90 users (3% of 3,000)
  - Starter: 50% → 45 users × $10 = **$450 / month**
  - Basic: 30% → 27 users × $30 = **$810 / month**
  - Pro: 20% → 18 users × $50 = **$900 / month**
  - **Total MRR (Month): ≈ $2,160**
  - **Total ARR (Year): ≈ $25,920**

These revenue projections reflect early-stage adoption typical of developer tools and micro-SaaS products.

2.6.4.1 Projected Expenses

Monthly expenses include:

- Infrastructure & API usage: $200 (RPC credits, cloud services)
- Development & maintenance: $500 (team hours and tooling)
- Marketing & acquisition: $300 (content, community, sponsorships)
- Operational overhead: $100

Total estimated monthly costs: ~$1,100.

2.6.4.2 Break-Even Point & Viability

With MRR of ~$2,160 and monthly costs of ~$1,100, SWQuery is projected to reach operational break-even by the end of Year 1 if growth continues and expenses remain controlled. According to SaaS revenue

benchmarks, achieving early break-even within 12–18 months is feasible for product-led tools with strong developer adoption.

2.6.4.3 Return on Investment (ROI)

Assuming initial development and launch costs of **$10,000** (product development, tooling, initial community programs), the **ROI timeframe** is projected at **~18–24 months**, as revenue scales while fixed costs are amortized.

2.7 Validation and Results

This section presents the validation of SWQuery based on real interactions with the target market, demonstrating that the project extends beyond a purely theoretical proposal. Validation efforts focused on assessing both the technical acceptance of the MVP and the perceived value of the solution among potential users.

2.7.1 Validation Methodology:

The validation methodology combined qualitative and exploratory approaches, including informal interviews with developers, feedback collected during technical demonstrations, and hands-on usage of the MVP by early users. These interactions aimed to evaluate whether the solution effectively addressed the identified pain points and whether users perceived sufficient value to consider adoption.

Additionally, the MVP was tested in real development scenarios, allowing observation of usage patterns, integration effort, and response quality. This approach provided practical insights into usability, performance, and feature relevance.

2.7.2 Market Validation Results:

Feedback collected during the validation phase indicated strong interest in simplified blockchain data access, particularly among developers who were not specialized in

low-level Solana infrastructure. Users highlighted reductions in development time and improved clarity of data outputs as key benefits of the solution.

No major pivots were required; however, feedback informed refinements to query responses, output standardization, and documentation clarity. These iterative improvements reinforced the core value proposition rather than altering it, supporting a decision to persist with the existing business and product model.

2.7.3 Key Performance Indicators (KPIs):

To assess the health and growth trajectory of SWQuery, the following KPIs are projected based on realistic SaaS benchmarks:

2.7.3.1 Revenue & Usage Metrics

- **Monthly Recurring Revenue (MRR):** Begins at ~$2,160 in Year
- **Annual Recurring Revenue (ARR):** ~ $25,920.
- **Average Revenue Per User (ARPU):**
  Assuming mix of tiers weighted by actual conversion:
  (45×$10 + 27×$30 + 18×$50) ÷ 90 ≈ **$24 / month per paid user**.

2.7.3.2 Customer Acquisition Cost (CAC)

- Using industry SaaS benchmarks for developer tools (developer SaaS CAC around **$120 per user**) and organic acquisition channels: ~ **$120 CAC per new customer**.

2.7.3.3 Customer Lifetime Value (LTV)

Assuming average churn ~ **5% annually** (typical SaaS churn) and ARPU of $24/month, LTV = (24 × 12) × (1 / churn) ≈ **$576 per customer**.

2.7.3.4 LTV to CAC Ratio

A healthy **LTV:CAC ratio** for SaaS companies is typically **3:1 or better**. Under these assumptions:

- LTV = ~$576

- CAC = ~$120
  → LTV:CAC ≈ **4.8:1**, indicating sustainable unit economics

2.7.3.5 Churn and Retention

- **Churn Rate:** Estimated ~5% per month initially, decreasing over time as product-market fit improves.
- **Net Revenue Retention (NRR):** Expected >100% if upsells and plan upgrades occur (e.g., Starter → Basic/Pro), inline with SaaS best practices.

2.7.4 Risks and Mitigation Plan:

Several risks were identified throughout the project lifecycle. Technological risks include dependency on third-party infrastructure providers and potential changes in blockchain protocols. These risks are mitigated by modular design choices and alignment with established ecosystem standards.

Business risks involve competition from existing analytics platforms or infrastructure providers expanding into higher-level abstractions. Mitigation strategies include maintaining a strong focus on developer experience and continuous innovation in natural language interfaces.

Financial risks are limited due to the low initial investment requirement and scalable cost structure. Legal and regulatory risks, particularly related to compliance and data usage, are mitigated by relying exclusively on publicly available on-chain data and avoiding custody of user assets.

3 Conclusion

This work presented the design and development of SWQuery, a computational solution aimed at simplifying access to on-chain data on the Solana blockchain through natural language queries and developer-friendly abstractions. The primary objective of reducing the complexity associated with blockchain data retrieval while

maintaining performance, accuracy, and scalability was successfully achieved. Through the integration of a high-performance Rust backend, AI-assisted query processing, and reliable blockchain infrastructure, SWQuery demonstrated that complex on-chain information can be transformed into structured, accessible, and actionable insights without exposing users to low-level blockchain details.

From a technical perspective, the system met its functional and non-functional requirements, as validated by comprehensive unit, integration, and acceptance testing. The results confirmed the robustness of the architecture, the correctness of the implemented features, and the suitability of the solution for real-world usage by developers and organizations operating within the Solana ecosystem. Additionally, the business and market analyses indicated that SWQuery addresses a clear demand for tools that improve data accessibility and developer experience in rapidly growing blockchain environments.

Looking forward, future projections for SWQuery include the expansion of AI capabilities through multi-agent model integration, deeper ecosystem interoperability with decentralized exchanges and NFT marketplaces, and the enhancement of compliance and risk analysis features. These improvements aim to further strengthen the platform's value proposition and scalability. In conclusion, SWQuery represents a viable and extensible solution that bridges the gap between complex blockchain data and practical application development, contributing to the advancement of data accessibility and innovation in decentralized systems.

References

GRIFFAIN. Griffain official website. Available at: https://griffain.com/engine. Accessed on: Feb. 28, 2025.

NEUR. Neur official website. All rights reserved. Available at: https://neur.sh/. Accessed on: Feb. 28, 2025.

RODRIGUEZ, F.; THOMPSON, E. Quantifying community engagement in blockchain projects: A framework for analyzing social media metrics and holder demographics. Blockchain and Social Computing, [S. l.], v. 6, n. 1, p. 45–63, 2024.

STRAWBERRY. Use strawberry official website. Available at: https://www.usestrawberry.ai/. Accessed on: Feb. 28, 2025.

SWQUERY. SWQuery official website. Available at: https://www.swquery.xyz/. Accessed on: Feb. 28, 2025.

WANG, L.; SHEN, X.; LI, J.; SHAO, W.; YANG, Y. Comparative analysis of blockchain consensus algorithms: A comprehensive benchmarking study of Solana, Ethereum, and alternative L1 platforms. Journal of Blockchain Research, [S. l.], v. 15, n. 3, p. 178–196, 2022.