SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY - INFORMATICS

TECHNICAL UNIVERSITY OF MUNICH

Bachelor's Thesis in Information Systems

A Data Exploration Tool for Novel DeFi-Metrics on the Algorand Blockchain

Nguyen Quy Tung Long

SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY - INFORMATICS

TECHNICAL UNIVERSITY OF MUNICH

Bachelor's Thesis in Information Systems

A Data Exploration Tool for Novel DeFi-Metrics on the Algorand Blockchain

Ein Datenexplorationstool für neuartige DeFi-Metriken auf der Algorand Blockchain

Author:	Nguyen Quy Tung Long
Supervisor:	Prof. Dr. Florian Matthes
Advisor:	Burak Öz
Submission Date:	15.10.2023

I confirm that this bachelor's thesis in information systems is my own work and I have documented all sources and material used.

Munich, 15.10.2023 Nguyen Quy Tung Long

Acknowledgments

Firstly, I'd like to express my greatest appreciation to my advisor, Burak Öz, whose supports and guidance was invaluable at every step of my thesis. My sincere thanks also to Prof. Dr. Florian Matthes for granting me the opportunity to write this thesis at his chair, and for his constructive feedback.

Lastly, heartfelt thanks go to my family, friends, and my girlfriend, for their support and encouragement, particularly during the most stressful moments.

Abstract

Algorand is a blockchain platform that provides a secure and efficient infrastructure for building decentralized applications (DApps). The platform uses a pure-proof-of-stake-based consensus approach that allows for fast and secure transactions with low fees. Algorand has gained popularity among developers and users due to its speed, security, and scalability.

In this thesis, we aim to analyze the DeFi ecosystem on the Algorand blockchain concerning the use of decentralized exchanges (DEXs) and available tokens and develop a data exploration tool for presenting novel DeFi-related metrics based on a few recent studies.

For this purpose, we first attempt to understand Algorand's ecosystem by analyzing DeFi-related metrics of the platform provided by the existing block explorer. This would be helpful for any research that requires a comprehensive overview of Algorand's DeFi space or a comparison between this blockchain platform and other developed ones, such as Ethereum, especially in terms of DeFi protocols. We then explore related research papers to gain a more profound comprehension of the challenges and interest regarding DeFi in general and extract valuable metrics that are applicable to further illuminate the intricacies of Algorand's DeFi ecosystem.

In the next part, we propose the design and implementation of an interactive data tool to facilitate the analysis of these metrics. This tool would be useful for traders or researchers who, for instance, want to have an overview of the MEV landscape or specific DEX activities.

Contents

1. Introduction

1.1. Motivation

In 2008, amidst the global financial crisis, an anonymous entity under the pseudonym Satoshi Nakamoto released a whitepaper titled "Bitcoin: A Peer-to-Peer Electronic Cash System" [1]. This paper introduces Bitcoin, a decentralized digital currency that operates without a central authority or single administrator. Bitcoin's underlying technology, the blockchain, allowed transactions to be verified by network nodes through cryptography and recorded on a publicly distributed ledger. This radical idea proposed a system where money could be transferred peer-to-peer without the need for centralized intermediaries such as banks.

While Bitcoin introduced the concept of decentralized money, Ethereum, proposed in late 2013 by programmer Vitalik Buterin, whose development was crowdfunded in 2014 and went live on July 30, 2015, took the idea a step further by enabling the creation and execution of smart contracts. These are self-executing contracts where the terms of the agreement are directly written into lines of code, allowing for more complex financial operations without human intervention [2]. Ethereum's blockchain operates with a native cryptocurrency called Ether (ETH), but its primary distinction lies in its function as a platform for decentralized applications and smart contracts, making it a fundamental component of the Decentralized Finance (DeFi) movement.

The foundations laid by Bitcoin and Ethereum paved the way for the emergence of decentralized finance projects. One of the earliest DeFi projects was MakerDAO, initiated in 2014. MakerDAO introduced the concept of a decentralized stablecoin, DAI, which is pegged to the U.S. dollar and is generated through the collateralization of assets [3]. Another seminal project is Compound, a decentralized protocol that allows users to lend and borrow assets against collateral, introducing the concept of earning interest in a decentralized manner [4].

These early DeFi projects not only exemplified the potential of decentralized financial systems but also signified a shift from traditional centralized financial systems to a more open, interoperable, and composable financial ecosystem. DeFi fundamentally challenges and innovates upon the norms set by centralized systems. Utilizing blockchain technology as its foundation, DeFi offers financial services through decentralized protocols and automated smart contracts. This not only streamlines operations and cuts costs but also ensures transactional accuracy and transparency [5]. Platforms like Ethereum and, more recently, Algorand serve as the backbone for these decentralized services.

As a young and innovative project, Algorand presents itself as a robust platform for DeFi applications. Renowned for its scalable, secure nature, along with its instant transaction finality, Algorand emerges as a promising ecosystem specifically dedicated to enabling decentralized financial services and a borderless economy [6]. The platform is witnessing a growing number of asset types and financial instruments traded on its network. However, as the ecosystem flourishes, the complexity and volume of data coming from Algorand-based DeFi platforms also surge. Presently, there is a lack of dedicated tools for effectively exploring and analyzing this data.

This thesis aims to analyze DeFi activity on the Algorand blockchain, while concerning the use of the DeFi protocols (e.g., DEXs) and available assets, and develop a web application for showcasing metrics based on the conducted analyses.

1.2. Research Questions

With the motivation stated above, this thesis was conducted following the research questions (RQs) below:

- **RQ1: What information can be inferred about the DeFi ecosystem on Algorand using the existing tools?**

  The first RQ aims to investigate the general DeFi ecosystem on Algorand, emphasizing on utilizing existing tools, such as Block Explorer, to extract valuable information and insights. By addressing this RQ, we are able to present the strengths and shortcomings of current tools for providing insightful data.

- **RQ2: What additional DeFi-related metrics could be computed based on existing blockchain literature?**

  The second RQ looks into the interests of researchers regarding DeFi in general, not only on Algorand. The objective is to understand what metrics are usually computed and discussed in recent studies and for what purposes. By addressing this RQ, we aim to demonstrate the gap between research's interest and available data-provider sources, which lays the motivation for this thesis.

- **RQ3: What is a minimum viable pipeline to compute and store the metrics?**

  This RQ focuses on our attempt to build a data exploration tool, namely *AlgoSight*. It was built following a conceptual data pipeline model that can fetch, transform, and store data for further analytical computation and visualization. Both the design and implementation will be discussed in detail.

- **RQ4: What are the most valuable views for traders and maximal extractable value (MEV) researchers?**

  The fourth RQ aims to justify the practicality of the metrics and charts provided by the developed data tool, particularly for traders' and MEV researchers' use cases.

1.3. Thesis Outline

In the following chapters, we subsequently introduce the background for Algorand blockchain and DeFi in general (Chapter 2), the overview of the DeFi ecosystem on Algorand, and discuss it by analyzing data provided on current block explorers (Chapter 3). We then proceed to give a broad summary of the related work and research interest regarding DeFi in Chapter 4, along with their findings, and present our detailed design and implementation of AlgoSight in Chapter 5. Chapter 6 is an evaluation of the data tool, focusing on the insights it can bring to traders and MEV researchers. We conclude the thesis by summarizing our findings in Chapter 7.

2. Foundation and Background

2.1. Introduction to Algorand Blockchain

Blockchain, at its core, is a distributed and immutable ledger (or database) maintained across multiple computers or nodes, ensuring transparency, security, and resistance to censorship. Traditionally, databases or ledgers are controlled by central authorities, such as banks or governments. However, blockchain shifts this paradigm by decentralizing control, making it virtually impossible for a single entity to alter past records without consensus.

This technology became globally renowned with the inception of Bitcoin in 2009, introduced as a peer-to-peer electronic cash system that operated without the need for a central authority. The decentralized nature of blockchain technology paved the way for a variety of applications beyond cryptocurrency, especially after the launch of Ethereum, the first blockchain network that enables smart contracts which is a set of codes pre-programmed that automatically run as long as the predetermined conditions are fulfilled. With this idea, a blockchain network can become the host for a vast amount of DApps, which are developed based on smart contracts and can be used for many disciplines, such as decentralized finance, voting systems, and insurance.

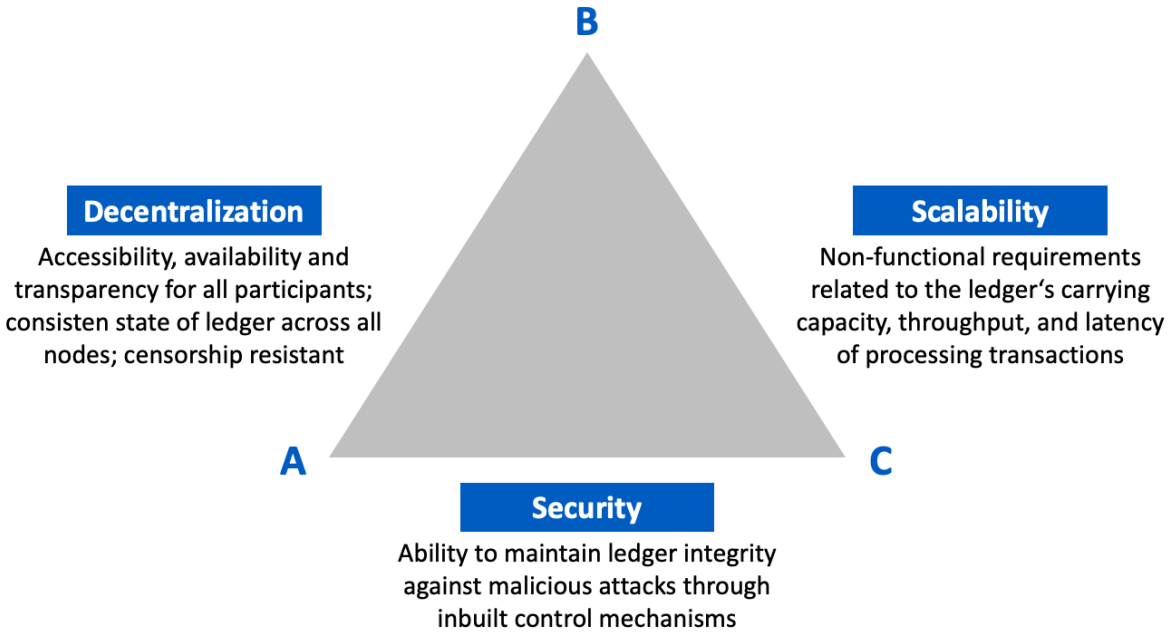

Figure 2.1.: Blockchain Trilemma [7].

Among current blockchain platforms, Algorand stands out as a particularly innovative solution. Founded by Turing Award winner Silvio Micali, Algorand was designed to address three major challenges that plagued earlier blockchains: security, scalability, and decentralization, often referred to as the blockchain trilemma shown in Figure 2.1.

ALGO, the native cryptocurrency on Algorand, functions as a medium of exchange or payment for transactions. Algorand has a maximum throughput of 6,000 transactions which are records or instructions sent by users that denote changes or actions to be taken by the blockchain), with a transaction fee of 0.001 ALGO ($\approx$ 0.0001 USD as of October 2023). Algorand's infrastructure enables blocks to be finalized in less than 4 seconds, making it suitable for large-scale usage, such as financial transactions. In the following subsections, we introduce the fundamental components of the Algorand blockchain.

2.1.1. Network

The network on blockchain is a collective ensemble of computers (referred to as "nodes") that participate, communicate, and operate on a shared ledger. This network is essentially the backbone that enforces the consensus rules, validates transactions, and maintains the integrity and security of the data. On the Algorand blockchain, the network consists of nodes, which are of two types:

- **Relay Nodes** serve as central network hubs and maintain connections to many other nodes, which are other relay nodes and participation nodes. Relay nodes have high-bandwidth network connections, which allow for highly efficient communication paths, ultimately reducing the number of hops in communication [8]. Relay nodes are purely responsible for propagating transactions, blocks, and consensus messages; they do not participate in the consensus process.
- **Participation Nodes** are connected to much fewer nodes, most of which are relay nodes [8]. As the name suggests, participation nodes take part in the consensus process, which fundamentally involves the creation of new blocks.

The cooperation of relay and participation nodes on Algorand contributes to addressing the challenges of decentralization and scalability. Meanwhile, to ensure the security of the network, Algorand emphasizes the diversity and decentralization of its relay nodes. This is achieved by geographically distributing relay nodes across different continents, ensuring they are positioned at crucial internet exchange points near major population and financial hubs. Furthermore, a variety of organizations, ranging from universities to traditional financial entities, operate these nodes. These organizations share a vision of an inclusive, borderless economy, ensuring a robust and distributed network foundation [8].

2.1.2. Consensus Protocol

Consensus, in the context of blockchain and distributed systems, is the mechanism by which all the nodes in a network agree on a single version of the truth or a single state of the system.

The Algorand blockchain uses a decentralized Byzantine agreement protocol based on pure proof-of-stake (PPoS) [9]. This protocol enables Algorand to work in a decentralized fashion while remaining secure.

When a user initializes a transaction, this user-defined transaction - delineating specifics such as the transaction type (e.g., asset transfer or smart contract call), the involved parties, the amount, and other metadata - needs to be signed using the user's private key. This cryptographic signature certifies the transaction's authenticity, ensuring it comes from the claimed source, and it safeguards the transaction's integrity, guaranteeing its contents remain unaltered during propagation.

After the creation, transactions are propagated over the network, utilizing relay nodes and participation nodes. After passing the verification of each node, they are put into the node's mempool - a temporary holding place for transactions that are waiting to be included in a block. Meanwhile, the nodes further propagate those transactions to other nodes.

On Algorand, every account holding ALGOs can participate in the consensus protocol (using a *participation key pair* for security concerns), i.e., become a participation node. The consensus process starts with the **Block Proposal** phase, followed by **Soft Vote** and **Certify Vote**. In each of these phases, Algorand runs a cryptographic sortition algorithm, which uses *Verifiable Random Function (VRF)*, to randomly select a set of participation nodes for performing certain tasks. This algorithm takes a private key and a seed value to generate a random hash value. Additionally, it creates a proof that, with the corresponding public key, allows other nodes on the network to confirm the hash's accuracy in relation to the seed. The seed value is based on the previous block and, thus, cannot be predicted by adversaries or anyone. The chance of a node being selected by VRF is also affected by the number of ALGOs that the node is holding. This characteristic helps prevent Sybil attacks.

The general flow of the consensus protocol can be described as follows:

- In the *Block Proposal*, participation nodes are randomly selected for being block proposers by locally executing VRF. Once selected, the chosen node gathers transactions from its *mempool* to form a block proposal, then broadcasts this proposed block to the network along with VRF proof to prove its eligibility [10].
- After the Block Proposal phase, there is typically more than one node chosen as block proposer; therefore, the purpose of *Soft Vote* is to choose only one from them. For this task, another round of VRF is run to select a committee to vote on the block proposer with the lowest hash value. A quorum of votes is needed to move to the next step and must be a certain percentage of the expected committee size. Once a quorum is reached for the soft vote, the process moves to the certify vote step [10].
- *Certify Vote* is the last phase of the consensus protocol, which aims to check the proposed and chosen block for problems such as overspending or double-spending. A new

committee of accounts is created using VRF random selection for performing this task. Similarly to Soft Vote, a quorum of votes is needed for the block to be certified and written to the ledger (otherwise the network will enter recovery mode). The process then ends and starts over with a new round [10].

The Algorand consensus protocol ensures that every user's influence on the choice of a new block is proportional to their stake (or holdings) in the system, without the need for energy-intensive mining processes seen in platforms like Bitcoin. This protocol ensures immediate transaction finality, eliminating the long confirmation times and the possibility of forks, which is a significant advantage for applications that require quick and definitive transaction settlements, such as financial activities.

Regarding incentives, Algorand differs from platforms like Bitcoin or Ethereum in that it does not distribute block rewards. Instead, it offers governance rewards. Users are empowered to cast votes on ecosystem development proposals, with the weight of their votes being determined by the number of ALGOs they've held over the preceding three months [11].

2.1.3. Smart Contracts Architecture

As described by Silvio Micali in Algorand's Smart Contract Architecture [12], Algorand's innovative approach to smart contracts is manifested through its division between Layer-1 and Layer-2 smart contracts, each tailored to serve unique purposes and requirements.

Layer-1 (On-Chain) Smart Contracts:

- Layer-1 smart contracts are directly embedded in the Algorand blockchain and handle many basic and straightforward transaction types. They streamline processes such as atomic swaps, guaranteeing that transactions approved by multiple parties are either completely executed or not executed at all.
- Algorand natively supports user-defined assets, termed Algorand Standard Assets (ASAs). This built-in support ensures safety against common vulnerabilities like unintended token creation or deletion and facilitates functions like freezing, minting, and burning of tokens.
- These on-chain contracts utilize TEAL (Transaction Execution Approval Language), a language similar to assembly. TEAL is crafted to manage the mentioned transactions and is regularly updated to introduce more features, including the ability to directly store states in Layer-1 [12].

Though Layer-1 handles many typical blockchain transactions, the wide range of blockchain applications often needs more tailored solutions. Some contracts might be too extensive, handle very sensitive data, be too computationally demanding, or be too intricate, meaning they need a modular structure. This is where Layer-2 or off-chain contracts are useful.

- **Execution mechanism:** When activated, an off-chain contract is executed by an entity called the contract execution committee, not by the primary consensus committee of the blockchain. This committee (selected in the same way that the main consensus committee is chosen) is responsible for executing and validating the contract call and producing its effects. These effects are then bundled into a Layer-1 atomic batch of transactions. The committee also monitors the dependencies of each contract call, ensuring that the effects of a call are consistent with the current state of the blockchain [12].
- **Virtual Machine Execution:** Layer-2 contracts are written in high-level languages and run on a virtual machine. The contracts maintain a long-lived state known as contract storage, which remains off the blockchain for privacy concerns. Nonetheless, every contract call commits to the most recent contract storage for security assurance [12].

Algorand's smart contract architecture uniquely integrates on-chain and off-chain solutions, providing a versatile platform that accommodates both standard and specialized blockchain transactions. It sets the foundation for creating complex decentralized applications operating on Algorand that offer multiple innovative services or products, especially in the financial sector. These kinds of DApps are the building blocks of the DeFi ecosystem on Algorand.

2.2. Introduction to Decentralized Finance

Decentralized finance represents a transformative shift in the financial paradigm, moving away from the centralized systems that have historically dominated the global economic landscape. As an innovative offspring of blockchain technology, DeFi challenges the conventional roles of banks, intermediaries, and regulatory bodies by democratizing access to financial instruments and services and offering transparency to the underlying processes. This section aims to provide a comprehensive understanding of the origins, principles, and fundamental components of DeFi.

2.2.1. Centralized Finance and its Limitations

Centralized finance (CeFi) has traditionally underpinned the global economy, anchored by a complex network of banking institutions, regulatory authorities, and numerous intermediaries that govern the storage, management, and flow of capital worldwide. Its purpose extends from basic operations like deposits or withdrawals to more complex ones like loans, foreign exchange, and asset management. However, centralization, while offering some structure and security, is fraught with inefficiencies. Transactions, especially cross-border ones, navigate through multiple intermediaries, each introducing their own procedures, risk checks, and associated fees. Consequently, these bureaucratic layers complicate and protract the process, cumulatively leading to elevated costs and time delays even for basic services. Additionally, this centralized framework often lacks adequate transparency, making it challenging for individuals to discern the path and the full array of costs associated with their transactions.

A system relying on trust necessitates intermediaries: individuals place faith in banks to manage and safeguard their funds, count on regulatory bodies to enforce relevant financial rules, and rely on ledgers to accurately record every transaction.

Therefore, the centralized nature of CeFi brings with it systemic vulnerabilities, as failures within the network can have broad effects. The 2008 financial crisis served as a grim reminder of how the failure of a major institution within this centralized framework can trigger domino effects and lead to critical consequences, threatening the stability of the global financial ecosystem [13]. Furthermore, another limitation of this centralized system is its inability to be universally inclusive. Despite its extensive reach and evolution, approximately 1.4 billion adults remain alienated from the conventional banking network [14]. This exclusion stems from a range of factors, from strict documentation requirements to geographical or infrastructural constraints to sociopolitical barriers or even perceived credit risks. The vast section of the population, hence, finds itself financially disenfranchised, unable to access secure savings, reasonable borrowing rates, or even basic participation in the modern economy.

2.2.2. Decentralized Finance and its Characteristics

One of DeFi platforms' most notable aspects is the democratization of financial services. They bring about openness and accessibility, allowing anyone, irrespective of their financial standing or geographic location, to partake. Every transaction on DeFi is recorded on a public blockchain ledger, underscoring a commitment to complete transparency. Furthermore, being decentralized, DeFi is inherently resistant to censorship, be it from governmental interventions or other centralized entities. Another feature worth mentioning is its interoperability, as applications within the DeFi ecosystem can interact and collaborate seamlessly, amplifying utility and innovation [2].

The ecosystem of DeFi has evolved continuously with the introduction of novel financial instruments and mechanisms, such as flash loans, which allow users to borrow assets without collateral, or yield farming, also often termed liquidity mining, which enables asset holders to earn rewards by staking their cryptocurrency in DeFi liquidity pools. The pace and scope of such innovations could arguably be unsurpassable in the history of finance. Of course, this evolution of DeFi is not without challenges. For instance, users bear full responsibility for their security in DeFi, and regulatory frameworks for DeFi are still in nascent stages in many regions. Additionally, fraud cases and smart contract hacks pose significant risks, with substantial amounts stolen from cryptocurrency platforms via DeFi protocols, raising security concerns amongst clients [15]. Despite these hurdles, DeFi stands as a revolutionary force in finance, aiming to reshape, redefine, and democratize financial services globally [5].

2.2.3. Overview of DeFi vs. CeFi

DeFi and CeFi represent contrasting financial systems with unique strengths, challenges, and functionalities. The decision tree proposed by Qin et al. [16] in Figure 2.2 is an intriguing method that brings clarity to the distinction between these two services. At its core, the tree

emphasizes user autonomy and control over financial assets as the defining criterion. The starting point is intuitive: if users lack control over their assets, the service is deemed CeFi. Additional layers of inquiries introduce granularity. The ability (or inability) for an entity to unilaterally influence transactions or govern the protocol's functions further differentiates the spectrum between centralization and decentralization. By considering not only asset control but also potential intervention capabilities, the decision tree delivers a more specific breakdown. What's particularly notable about this model is its recognition of hybrid forms. By not forcing a binary categorization, it acknowledges the evolving landscape of financial services, where some platforms may incorporate elements from both CeFi and DeFi. This nuanced approach mirrors the fluidity of the contemporary financial ecosystem and offers a structured lens to navigate it. It might very well become a reference point for future classifications and discussions in the financial sector.

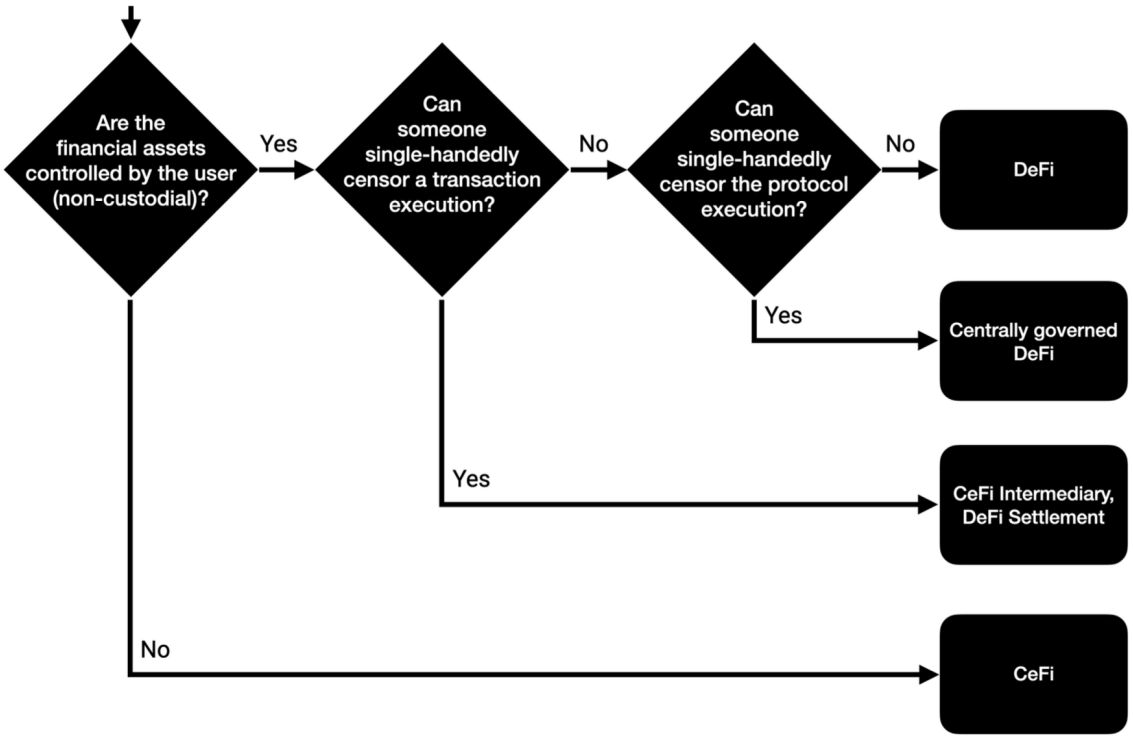

Figure 2.2.: Decision tree to differentiate among DeFi and CeFi [16].

The core distinguishing properties between them are the foundational technology and control mechanisms. DeFi, based on blockchain, removes centralized intermediaries, enabling transparent transactions and full asset control by users. It offers trust through cryptographic assurances and the open source of the code and protocols, diverging from CeFi's institutional guarantees that are often backed by governmental bodies. From a legal standpoint, DeFi typically bypasses the Know Your Customer (KYC) and Anti-Money Laundering (AML) pro-

cesses, with transactions tied to digital addresses, meaning users can interact without directly revealing personal information. Despite this pseudo-anonymity feature, transaction details remain visible on public ledgers. Another aspect lies in the user's autonomy. Centralized systems can impose restrictions like account freezes and capital controls, wielding surveillance powers over user activities. DeFi, in stark contrast, provides users with unmatched autonomy.

Service-wise, both offer standard financial operations, but DeFi introduces unique features like flash loans or automated market makers (AMMs). Meanwhile, the economic models differ significantly. DeFi operates outside the purview of central bank regulations, leaving it free from controls over money supply and inflation, though it introduces its own set of manipulations. Security in CeFi employs more conventional methods like insurance coverage against potential hacks or system failures, layered security architectures, and robust firewalls, whereas DeFi depends solely on smart contracts, which are not immune to flaws and can be vulnerable if inadequately audited. Therefore, DeFi protocols require intricate protective measures against various types of attacks, ranging from network layer, consensus layer, and smart contract code [16].

Both systems present their own sets of risks. CeFi operates under established but sometimes opaque processes and is susceptible to institutional failures. DeFi, in its quest for transparency and user autonomy, confronts technological issues, particularly in smart contract codes, and an uncertain regulatory environment [17]. To provide a comprehensive yet concise overview of the fundamental contrasts between DeFi and CeFi, it is beneficial to condense these differences into a comparative format, as shown in the subsequent table.

Property	DeFi	CeFi
Control Mechanism	Decentralized: Operates without a central authority.	Centralized: Governed by a single entity or a group of related entities.
Trust Mechanism	Trust is established through code and cryptographic assurance. Users trust the protocol itself.	Trust relies on the reputation of centralized entities, often backed by regulatory frameworks.
Asset Custody	Users have full ownership of their assets using cryptographic wallets.	Users entrust their assets to institutions. Funds are stored in company-controlled wallets or accounts.
Operational Transparency	Every transaction is typically visible on a public ledger.	Operations might be opaque, with only the end results being shown to the users.

Property	DeFi	CeFi
User Privacy	Pseudonymous: Transactions are transparent, but user identities remain hidden unless linked.	Requires user identification: Due to KYC/AML procedures, user identities are known to the institution.
User Autonomy	High: Users have complete control, limited only by protocol rules.	Limited: Institutions have power over user accounts, including freezes, capital controls, and surveillance.
Operational Hours	Typically operates 24/7 without downtime.	Has operational hours; can experience downtimes for maintenance.
Security Protocols	Relies on smart contracts, which can have vulnerabilities if not properly audited. Often exposed to various types of DeFi attacks.	Utilizes traditional security measures and often insured against hacks and failures.
Legal Framework	Often operates outside established regulatory frameworks. Lack of KYC/AML processes.	Adheres to national and international regulations. KYC/AML processes are standard.
Financial Instruments	Offers novel instruments like flash loans, automated market makers, yield farming, etc.	Traditional financial instruments, like loans, savings accounts, and trading platforms.
Economic Control	Lacks central bank controls; operates outside traditional monetary policies.	Under the purview of central banks and monetary authorities. Subject to national economic policies.

Table 2.1.: DeFi vs. CeFi Comparison.

As mentioned above, while DeFi brings forth unprecedented levels of transparency and user autonomy, CeFi offers robust regulatory oversight and often more mature risk management processes. This synergy can lead to DeFi platforms utilizing CeFi's infrastructure for certain components, such as fiat on/off ramps or oracle services, and CeFi benefiting from DeFi's innovative financial products. Even with its decentralized nature, DeFi remains interconnected with CeFi. While blockchain-based stablecoins attempt to reduce this reliance, the connection is still undeniable (CeFi vs. DeFi paper). Hence, it is feasible for both to coexist harmoniously within the financial ecosystem, as each offers distinct advantages that can potentially complement the other, paving the way for a more diverse and resilient future of finance.

2.2.4. DeFi Services & Protocols

A conceptual overview of the different constructs within the DeFi ecosystem below was introduced by Werner et al. in their paper [18], presenting DeFi protocols categorized by the type of operation they provide.

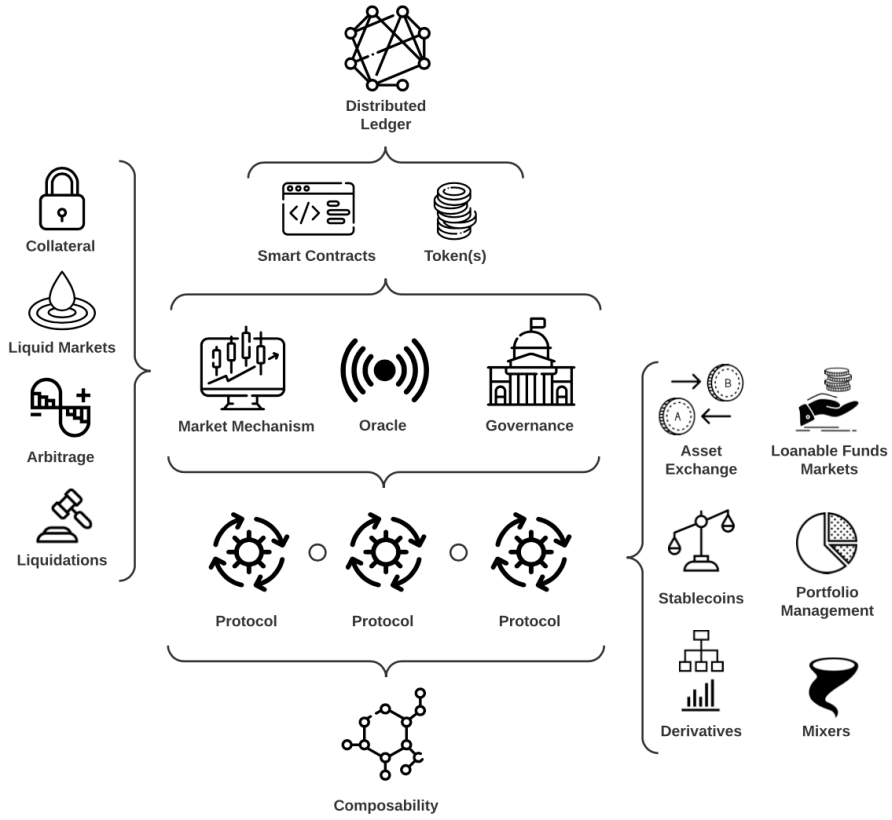

Figure 2.3.: Conceptual Overview within DeFi Ecosystem [18].

Major DeFi services include lending and borrowing platforms like Aave and Compound, decentralized exchanges such as Uniswap or SushiSwap, and asset management platforms like Yearn.finance. Additionally, insurance protocols like Nexus Mutual, synthetic asset platforms like Synthetix, and stablecoin projects such as MakerDAO or Tether are key components in the DeFi landscape. These services contribute to a multifaceted financial ecosystem on the blockchain, offering a decentralized alternative to traditional financial systems, thereby promoting financial inclusion and innovation. In the following, we go through each of these services and protocols to gain a more comprehensive understanding.

Decentralized Exchanges

Exchanges are the fundamental component of any financial system: they are the marketplaces for trading assets. Schär [5] and Pourpouneh et al. [19] noted that, while centralized

exchanges may be simpler to set up, they come with several issues: traders relinquish asset control, must trust the exchange to avoid asset seizure, are vulnerable to security risks due to a single attack point, and face minimal regulation.

Conversely, as On-chain Asset Exchange services, DEXs allow for the non-custodial trading of digital assets, meaning users maintain control of their funds at all times. Orders are created and paired entirely via unchangeable blockchain smart contracts, enabling censorship resistance, where orders remain unalterable before and after execution [20]. DEXs can be implemented through various methods, such as order book DEXs, AMMs, or DEX aggregators.

- **Order Book DEXs** aim to mimic traditional exchange order matching and price discovery but in a decentralized way. Buy and sell orders are matched based on price and time priority from a limit order book. However, settlement occurs on-chain via smart contracts rather than through a centralized intermediary. There are two main approaches to maintaining the order book: on-chain order books and off-chain order books.

  Off-chain order books maintain the order book off-chain through third-party relayers, while settlement is done on-chain. This approach reduces blockchain load, enhancing efficiency, but introduces some centralization as it relies on relayer nodes to facilitate order matching and trade initiations [21]. Projects like 0x [22] pioneered the off-chain order book model. In 0x, orders are signed and broadcast off-chain to relayers. Takers can query relayers to discover prices, select an order, and execute it on-chain. The smart contract settles the trade. Relayers facilitate order placement and discovery but do not ever hold funds.

  Meanwhile, on-chain order books not only process and finalize the transactions on the blockchain but also store all trading details directly there. This provides full decentralization but is inefficient as every order update requires an on-chain transaction. These include projects like Loopring [23] or Algodex [24]. This avoids relayers but faces scalability challenges due to on-chain data storage and order updates. In general, order book DEXs face issues around liquidity and efficiency but can facilitate more sophisticated trading compared to AMMs, which we shall analyze after this. Projects are exploring optimizations like order batching to improve scalability [5].

- **Automated Market Makers** utilize algorithmic pricing based on liquidity pools rather than order books. Liquidity providers deposit tokens into a pool to facilitate trading. Trades then occur directly against this liquidity pool, according to a pricing algorithm. The most common function is the constant product formula, pioneered by Vitalik Buterin [25] and later popularized by Constant Product Market Makers (CPMMs) like Bancor [26] or Uniswap [27]. If a pool trades assets X and Y, where x and y are the reserves, then x * y = k for some constant k. Trades move along the curve x * y = k to determine the asset price based on the pool reserves. Besides CPMM, there are also some further AMM frameworks and each carries its own mechanics and presents trade-offs in terms of capital efficiency, slippage, and other factors [28].

Xu et al. highlight in their work on DEX with AMM Protocols [29] that AMMs provide constant liquidity without requiring a counterparty, streamlining trading. They offer liquidity providers passive income through trading fees without active trading, enhancing capital utilization. AMMs also support less popular token pairs, reducing spreads for less-traded assets. While they bypass issues like book manipulation seen in order books, AMMs may have inefficient pricing, high slippage for large trades, and can come with high gas costs due to on-chain settlements. There's also the risk of manipulation, such as flash loan attacks, and impermanent loss, where liquidity providers can incur losses if token prices in the pool change significantly. However, new protocols are evolving to tackle these challenges, with many AMMs sharing a common foundational structure but differing in specific mechanisms and parameters.

Overall, the pooled liquidity and algorithmic pricing of AMMs unlock advantages in capital efficiency, enabling accessibility to liquidity for long-tail assets and avoiding order book challenges. This has made them hugely popular in DeFi, as for instance Uniswap constantly dominates as the largest decentralized exchange protocol with more than 50% market share of DEX trading volume [30]. Many DEX platforms on the Algorand blockchain, such as Humble Swap [31], Tinyman [32], or Pact [33], implement this AMM mechanism.

- **DEX Aggregators** combine liquidity and pricing across multiple DEX protocols to facilitate trading. This allows users to access the best pricing and deepest liquidity across different sources. Aggregators query various DEX pools and AMMs to find the optimal pricing and trade routing for a given trade. Alammex is a leading aggregator on Algorand that searches across multiple platforms like Tinyman [32], Pact [33], or Humble [31] to split trades across pools, aiming to minimize slippage and provide better trading rates for users [34]. Aggregators aim to maximize the efficiency of decentralized trading by using existing liquidity more effectively. They highlight the composability of DeFi protocols and show how value can be created by combining services.

DeFi Loanable Funds Markets

Protocols for Loanable Funds (PLFs) establish decentralized lending and borrowing markets for crypto assets by pooling deposited funds into a smart contract. Users can then directly borrow against the reserves in the smart contract, assuming sufficient liquidity. There are markets for each supported crypto asset, with available deposits making up the liquidity [18]. DeFi loans typically come in two forms: Overcollateralized Loans and Flash Loans.

- **Overcollateralized Loans** A major type of decentralized lending platform involves overcollateralized loans, where borrowers must lock up collateral worth more than the borrowed amount in smart contracts to take out loans. Collateral factors determine the minimum required collateral rates, which typically range between 120% and 150% [35]. For example, in MakerDAO, users can bring in ETH as collateral to borrow DAI stablecoins. If the minimum collateral ratio of 150% is required, it's possible to borrow only $100 worth of DAI for every $150 of ETH locked up. This over-collateralization is a

measure to ensure the stability of the system and to protect against market volatility. If the collateral value drops below a liquidation threshold, liquidators can seize collateral at a discount [35]. Liquidation penalties compensate liquidators for the service while disincentivizing under collateralization. This protects lenders against default risk. Other lending protocols like Compound or Aave work similarly, allowing borrowing from pooled funds against crypto collateral [5]. To sum up, overcollateralized loans have advantages in credit risk management, facilitate lending without credit checks while ensuring repayment, and provide pseudonymity and permissionless access, but are capital inefficient for borrowers.

- **Flash Loans** are a new concept in decentralized lending that permit borrowers to access uncollateralized loans within a single blockchain transaction, ensuring instant settlement and zero default risk for lenders [36]. Borrowers access a pool's full reserves, use the funds, and must repay the amount plus fees by the transaction's end; otherwise, the entire transaction reverses as if no loan occurred. These loans exhibit unique features, including permissionless access, no collateral requirement, interaction with multiple DeFi protocols, transaction atomicity, and transparent on-chain data verification for repayment, as illustrated by Aave flash loans, where smart contracts ascertain repayment before fund disbursement [37].

  Use cases for flash loans include arbitrage across decentralized exchanges, avoiding upfront capital requirements [38]. While powerful, flash loans present risks. The instant high-volume access has also facilitated "attack" hacks exploiting vulnerabilities before protocols added defenses [17]. Attackers can instantly borrow potentially millions of dollars by just paying network fees. Therefore, despite earning fees, lenders risk major losses from these attacks. Ongoing research is looking into making flash loans more secure while preserving their unique financial benefits. To sum up, flash loans in decentralized finance have transformative potential but come with risks; safeguarding them could unlock new, instantaneous capital access strategies.

Stablecoins

Non-custodial stablecoins aim to maintain price stability relative to a target currency like the US dollar through on-chain collateralization and economic mechanisms, as opposed to relying on a trusted third party. Stability requires agents to perform certain roles, like providing collateral and governance to manage parameters and issuance. Different designs utilize combinations of exogenous collateral like ETH as well as endogenous collateral created by the protocol itself. MakerDAO's DAI is an example of a decentralized stablecoin collateralized by ETH. Stablecoins aim to retain price stability, typically compared to fiat currencies like the US dollar. Their stabilization strategies include:

- **Fiat Collateralization:** Stablecoins, such as USDC and Tether, maintain a 1:1 peg-to-fiat reserve in banks. While centralization is a concern, it guarantees price stability as long as the peg is upheld [5].

- **Crypto Collateralization:** Instead of fiat, some stablecoins are backed by cryptographic assets. DAI by MakerDAO uses over-collateralization and asset liquidation to ensure value. While decentralized, it lacks direct fiat conversion guarantees [39].
- **Algorithmic Stabilization:** These stablecoins auto-adjust supply based on demand without asset backing. They've faced challenges in maintaining pegs due to misaligned incentives [40]. However, algorithmic improvements may enhance stability.
- **Seigniorage Models:** By merging algorithmic stabilization with collateralization, these stablecoins will be balanced out by auctioning surplus coins. Over time, coin staking by holders can stabilize its value [41]. One example of this model is GARD, which is depicted as the first decentralized dollar and algorithmic stablecoin on Algorand, targeting governance participants. Starting with ALGOs as collateral, the protocol aims to include assets like Bitcoin in the future, diversifying its portfolio [42].

Portfolio Management

DeFi portfolio management protocols aim to automate on-chain asset management. Users deposit tokens into a smart contract encoded with yield-generating investment strategies. These strategies transact with other DeFi protocols, like PLFs, to earn interest and rewards, which are tokens distributed to liquidity providers. Algogator.finance is an all-in-one DeFi aggregator on Algorand that can also be considered as a DeFi portfolio management service, as it allows users to manage investments, track assets, and handle liquidity pool positions in real-time [43]. Another one is Yield Monitor which can offer data-driven tools for Algorand DeFi investors, providing insights into leading protocols and facilitating multi-chain management [44].

Derivatives

DeFi derivatives provide on-chain versions of popular derivatives - financial contracts that derive their value from the performance of underlying assets. Common DeFi derivative types include synthetic crypto asset tradings (e.g. Synthetix [45]), futures, perpetual swaps, and options [18]. On November 29th, 2021, the Algorand Foundation had announced its partnership with Thetanuts to lauch Thetanuts.finance [46] - a DeFi derivatives protocol on Algorand that hosts the most extensive "altcoin" option vaults in DeFi and grants users access to high-level strategies via organized product vaults [47].

Privacy-preserving Mixers

Mixers aim to preserve transaction privacy in an otherwise public blockchain setting. They obscure the source of funds either by mixing with other funds or using zero-knowledge proofs (ZKPs) to shield transaction contents. This method can preserve user privacy but could also be used to obscure the source of illicit funds [18].

2.2.5. DeFi Attacks & Market Manipulations

Since the rise of Decentralized Finance, numerous attacks and market manipulations have transpired, underscoring the nascent stage of this financial innovation. Notable attacks

include smart contract exploits such as the DAO attack [48], and economic exploits like the Black Thursday incident on MakerDAO. Market manipulations manifest through MEV, transaction reordering, and oracle attacks, where malicious actors exploit system design flaws for financial gain. In the following, we will briefly introduce some of the major DeFi attacks and market manipulations.

Smart Contract Exploits involve exploiting code flaws in smart contracts. Malicious actors take advantage of bugs or vulnerabilities in the smart contract code to manipulate the contract's behavior for their own benefit.

Flash Loan Attacks occur when attackers exploit DeFi protocols using flash loans. Attackers borrow funds via flash loans, manipulate asset prices or liquidity, and then repay the loan, all within one transaction. A notable example of a flash loan attack is the series of attacks on the bZx protocol in February 2020. Attackers used flash loans to manipulate the price of assets and exploit bZx's flawed liquidation mechanism, causing significant financial loss [49].

Oracle Manipulation: DeFi oracles bridge on-chain and off-chain worlds, providing blockchain with external data, and therefore are crucial for DeFi apps' functionality [50]. However, oracles can be deceived by attackers supplying misleading data, causing harmful deviations in DeFi protocols. Such attacks can misreport asset values or abuse time-specific smart contracts.

Transaction Ordering Attacks: In blockchain networks like Ethereum, gas prices - denominated in units like gwei — determine transaction processing fees. Users can boost gas prices to prioritize their transactions, tempting attackers to manipulate transaction orders for profit. These schemes can drain unsuspecting users and breach blockchain fairness:

- *Front-running:* spotting a pending transaction, attackers pay a higher gas fee to precede it [51].
- *Back-running:* attackers strategically position their transaction post a target's, using slightly less gas to ensure sequential processing [18].
- *Sandwich Attacks:* a combination of front-running and back-running, attackers "sandwich" a victim's transaction. Typically, the attacker creates an imbalance in an AMM before the victim's transaction, and once the victim's transaction is executed, the attacker performs a reverse action to correct the imbalance and profit from the induced price change [18].

Maximal Extractable Value

Miner Extractable Value, or Maximal Extractable Value, refers to the potential revenue miners can extract from reordering and censoring transactions in the block they are mining. This revenue comes at the expense of users and DeFi protocols. The ability for miners to extract MEV arises because transactions on blockchains like Ethereum are executed sequentially in the order miners decide to place them in a block.

By reordering transactions, miners can profit from arbitrage opportunities between protocols. For example, a miner could see a profitable arbitrage opportunity between two DEX pools in the mempool but extract that profit for themselves by placing their arbitrage transaction before the original arbitrage transaction. MEV extraction also includes the ability for miners to censor transactions. For example, miners could censor transactions that would have triggered a liquidation in a lending protocol, letting them later produce a block that does trigger the liquidation so they can profit.

Miners are financially incentivized to maximize MEV profits when creating a block, even if it negatively impacts users. However, MEV extraction is not exclusive to miners. In blockchains with fee-based transaction ordering mechanisms like Ethereum, anyone can offer the miner a high transaction fee and have a block built in the most profitable way for them (therefore, Maximal Extractable Value might be more generic than Miner Extractable Value).

In blockchains with fixed transaction fees, like Algorand with first-come, first-serve transaction ordering mechanics, the MEV strategies can be more sophisticated; for example, MEV searchers might try to congest block space with a large transaction group to force the system to shift to fee-based priority [52]. Quantifying MEV and designing mechanisms to mitigate its risks are active areas of research. The ability for miners to extract value by manipulating transaction order is an inherent challenge facing DeFi applications.

3. Overview of the DeFi Ecosystem on Algorand

Building upon the foundational knowledge of the Algorand blockchain and an introduction to decentralized finance, this chapter offers an overview of the DeFi ecosystem on Algorand. We will introduce key DeFi protocols and notable tokens active on this blockchain, highlighting their functionalities. Additionally, we'll discuss available tools for exploring data about Algorand's DeFi landscape and what information we can infer using them.

3.1. Prominent DeFi Protocols and Assets

Algorand infrastructure enables decentralized applications to operate and facilitate multiple financial services. In the following, we introduce the prominent DeFi protocols and assets on Algorand, based on the Algorand Foundation's list [1].

Folks Finance: As a decentralized borrowing and lending protocol, Folks Finance provides a platform where users can earn interest by lending their assets or borrowing against their holdings. As of October 1st, 2023, the total value locked on Folks Finance has surpassed 60 million dollars, according to the protocol's website [53], making Folks Finance the protocol with the highest TVL on Algorand.

Algodex: Algodex is a highly decentralized marketplace built on the Algorand, enabling peer-to-peer trading of Algorand Standard Assets and non-fungible tokens (NFTs). It is known for its advanced order book functionality. Unlike many DEXs that rely on liquidity pools, Algodex's on-chain order book allows users to set their desired buy or sell prices, offering a higher degree of control over trades [24]. Launched on Mainnet on May 31st, 2022, Algodex also introduced its native token, ALGX, to incentivize platform usage and empower its community with governance capabilities [54].

Tinyman: As one of the pioneers among DEXs on Algorand that utilizes an AMM model similar to Ethereum's Uniswap, Tinyman facilitates seamless and secured token swaps along with liquidity provision and farming. Users can interact with smart contracts via its web-based app and contribute to liquidity pools to earn fees on swaps [55]. Tinyman holds a notable position in the Algorand ecosystem due to its early market entry, which has resulted in deep liquidity pools [56]. Its growing user base can be attributed to its intuitive interface, low transaction fees, and speedy trade executions.

[1] `https://www.algorand.foundation/defi`

Pact: Pact is another vital player in the Algorand DEX space. It allows users to exchange assets in a decentralized manner at low cost and with lightning-fast trading experience. Through continuous innovation, Pact, with its customer-centric design, focuses on enhancing efficiency for both traders and liquidity providers. Pact is also described as a mobile-first dApp, highlighting its focus on easy accessibility for users to interact with the protocol [33].

BRZ Token: Recognized as one of the prominent stablecoins on Algorand, the BRZ Token is pegged to the Brazilian Real. Its growing adoption can be attributed to its ability to provide a bridge between traditional fiat currencies and the digital world, especially catering to the South American market.

Tether USD (USDT): A well-known stablecoin pegged to the US dollar, USDT provides stability and has become a preferred choice for many, given its wide acceptance and reputation as one of the first stablecoins in the industry.

USD Coin (USDC): Another dollar-pegged stablecoin, USD Coin, offers transparency, stability, and wide usability on the Algorand platform. Its increasing popularity stems from its regulatory compliance, transparent reserve holdings, and backing by prominent institutions.

3.2. Understanding Algorand's DeFi Ecosystem

Block explorer is a popular tool to look up general information about transactions and assets on DeFi platforms. On Algorand, AlgoExplorer[2] is the most advanced block explorer, followed by AlgoScan [3] (unfortunately shutdown in August 2023) and Goal Seeker [4]. By conducting a quick comparison of the metrics provided by the three block explorers on Algorand demonstrated in the table below, we conclude that one can get the most information about DeFi's activities by referring to AlgoExplorer.

	Market General Information					Asset Information				Transaction Details					Block Information			Address / Account Information		
Metrics / Block-Explorer	Circulating Supply	Staked Supply	Price	TVL	TX Cost	Price ($)	Chart of Stats	Market Cap	Total Txs	Tx ID	TX Type	Sender	Receiver	Fee	Round	Proposer	Txs	Balances	Assets	Apps
AlgoExplorer	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
AlgoScan	✓	✓	✓	✓	✗	✓	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Goal Seeker	✗	✗	✓	✗	✗	✗	✗	✗	✗	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✗

Figure 3.1.: Block Explorer Comparison [57].

[2] https://algoexplorer.io/
[3] https://algoscan.app/
[4] https://goalseeker.purestake.io/algorand/mainnet/

Based on metrics provided on AlgoExplorer, we want to infer valuable information about the DeFi ecosystem on Algorand.

Total Transaction Counts reflect the overall liveliness within Algorand's DeFi ecosystem. A higher count indicates more interactions, giving the impression of a vibrant and active ecosystem. It demonstrates the level of user participation and the ecosystem's ability to support a variety of transactions. On the other hand, a lower transaction count suggests a lack of activity or engagement, possibly manifesting a nascent or less popular DeFi ecosystem. The time-series chart of transaction counts also gives a sense of the consistency of the DeFi ecosystem.

Total Value Locked (TVL) is a critical metric representing the amount of capital invested in DeFi protocols on Algorand. A large amount of TVL demonstrates capital is being better utilized within the ecosystem, expressing trust and association from users. It shows the ecosystem's capacity to support larger-scale financial activities and secure and manage assets, which is attractive to both developers and investors. Contrarily, a small amount of TVL may hint minus commitment or trust, showing a need for growth or improvement within Algorand's DeFi ecosystem.

Circulating Supply of ALGO, the native token of Algorand, reflects the amount of ALGO available for trading and use within the ecosystem. An intense circulating supply can denote a wider distribution and potentially more involvement in the DeFi ecosystem, providing the necessary liquidity for DeFi applications. In contrast, a less concentrated circulating supply can state deflated liquidity and diminished participation. The circulating supply is a crucial factor for developers and investors to assess the network's health and the achievability of launching or participating in DeFi projects on Algorand.

Asset's Transaction Counts signifies the level of activity and liquidity associated with the asset. A vast number of transactions reflects more active trading or usage, denoting popularity and engagement within the ecosystem. Conversely, a low transaction count may stand for reduced interactions, which could be a sign of shortened demand or interest in the asset. This metric helps investors and users gauge the asset's vibrancy and potential for liquidity in the market.

Asset's On-Chain Volume represents the total value of the asset being transacted over the blockchain within a specific timeframe. An increased on-chain volume reveals additional interaction and liquidity, suggesting the asset is popular and well-utilized within the community. A decreased on-chain volume contrariwise might be evidence of fewer activities, which signifies insufficient demand or interest in the asset. This metric is crucial for investors and traders to measure the asset's market movement and feasibility for returns.

Asset's Number of Users indicates its level of adoption and acceptance within the community. An elevated number of users suggests that the asset is well-received and has a broader base of support, which could likely lead to growed liquidity and stability. Conversely, a

limited number of users might represent a lack of interest or trust in the asset, possibly making it more susceptible to price volatility.

Asset's Daily Unique Senders/Receivers offer insights into an asset's daily activity and user engagement. A substantial count can imply a more active and healthier network with a vibrant community, while a drop in count may signal slighter action or a less engaged community. This metric helps in understanding the level of usage and the asset's relevance in the daily transactions within the ecosystem.

With the metrics mentioned above, we can understand the big picture of the DeFi ecosystem on Algorand, have an overview of the performance of the market, and see how active each asset is individually. These metrics, however, do not give us insights, for instance, about how DeFi protocols on Algorand are used. Using current block explorers, it's also not convenient for users to compare multiple tokens or protocols side by side with respect to particular criteria. This raises the need for a data exploration tool that, as a complement to these current tools, can provide further in-depth, insightful metrics and visualizations.

4. Related Work

In recent years, the emergence of Decentralized Finance as a transformative force within the blockchain and financial sectors has led to a surge in academic, institutional, and independent research. In this chapter, we want to introduce relevant papers and give readers a broad understanding of current research interests in DeFi as general, how these studies are usually conducted and their results. We then contextualize these findings within Algorand DeFi ecosystem, and narrow them down to findings that closely relate to our purpose of creating a data exploration tool for novel DeFi metrics on Algorand.

4.1. Systematization of Knowledge

As DeFi platforms and protocols have been evolving continuously and consistently, there are several systematization of knowledge (SoK) papers that attempt to summarize and distill the collective wisdom of the community and present a structured overview of DeFi's various aspects.

The paper by Werner et al. [18] offers an exhaustive and methodical exploration of decentralized finance (DeFi), breaking it down along three fundamental dimensions: its core components, the protocols facilitating its functions, and its security concerns, which are categorized into technical and economic. A security challenge is defined as technical if an agent can atomically exploit a protocol [18]. Examples of technical security risks are contract vulnerabilities, transaction ordering dependency, or single transaction manipulation. The authors provide an overview of empirical technical security exploits in DeFi protocols from February 2020 to March 2021, in which most attacks are in the form of bug exploitation in smart contracts. Technical security problems are claimed to be addressable with program analysis and formal models to specify protocols, despite the complexity and computational difficulty. On the other hand, economic security risks involve non-atomic manipulations over many transactions and blocks, such as collateralization failures, MEV, governance extractable value, and market manipulation. Solutions for these kinds of risks are new economic models and designs of better incentive structures. The insights from this paper underscore the significance of secure smart contract development, especially as Algorand's DeFi ecosystem continues to grow. Although Algorand's unique consensus algorithm and architecture differ from Ethereum (the main platform where the attacks mentioned above happened), understanding such vulnerabilities and their remedies can be instrumental.

In another SoK paper about DeFi attacks, Zhou et al. [58] undertook a systematic analysis of DeFi incidents, illuminating their nature, frequency, and the resulting financial implications.

From April 2018 to April 2022, the authors documented 181 DeFi incidents occurring on the Ethereum and BNB Smart Chain, aggregating a staggering $3.24 billion in losses. The term "incident" is defined as a sequence of events leading to unintended financial losses for entities such as users, liquidity providers, speculators, or operators. The paper revealed a significant insight that a majority of these incidents stem from issues in smart contracts (42%), followed closely by flaws in protocol design (40%), and operational or oracle challenges (30%). The most recurrent protocol design attack is attributed to on-chain oracle manipulations, constituting 15% of the documented incidents. Notably, 10.5% of these incidents were the result of permissionless interactions between contracts, and in 56% of the attacks, there existed a window of opportunity for a rescue between contract deployment and exploitation. This paper's findings emphasize the importance of rigorous protocol and smart contract testing to a blockchain's DeFi ecosystem, including Algorand's. Furthermore, the prominence of on-chain oracle manipulations highlights the need for Algorand to prioritize secure and reliable oracle solutions.

In summation, while the aforementioned papers primarily delve into DeFi ecosystems beyond Algorand, the lessons contained within them are universally valuable. For instance, both the security issues categorization framework, proposed in [18], and the DeFi system model and threat model, proposed in [58], can serve as references for further studies about any blockchain platform, including Algorand.

4.2. Further Studies

The main focus of DeFi research is the exploration of its security mechanisms. The substantial financial assets locked in DeFi protocols make them lucrative targets for attackers, and the decentralized nature of these platforms, while offering transparency and accessibility, also means vulnerabilities can lead to irreversible financial losses. Among the noteworthy issues, those relevant to transaction reordering, front-running, MEV, or scam detection have gained significant attention. There are currently limited papers on these topics aimed specifically at the DeFi ecosystem on Algorand; however, the following papers' findings and their data-driven methodologies can inspire future studies on Algorand.

Daian et al. [59] explored the strategies of arbitrage bots that exploit inefficiencies to frontrun and profit from regular users' trades. A key revelation from the study is the presence of a significant arbitrage bot economy within DEXes. Such bots capitalize on "pure revenue opportunities," essentially trades that yield profits in every traded asset. The research estimates that over $6 million in profits have been generated from these opportunities. These arbitrage bots, in their race to capture these opportunities, take part in "priority gas auctions" (PGAs), a mechanism wherein they competitively escalate transaction fees to attain preferential transaction ordering and execution within blockchain blocks. By observing the pure revenue transactions, the pure revenue profit, the trend in quantity of gas used per trade, and the relationship between the number of observed PGAs and profit or revenue, the authors were able to comprehensively model and analyze the bot strategies in PGAs. This

paper also mentioned MEV, which can particularly result from the miner's control of the ordering of transactions and can pose multiple threats to the network.

To address the prevalent challenges of blockchain frontrunning and MEV activities on Ethereum, a proposed solution is Flashbots - a private pool with architecture designed to eliminate externalities relevant to MEV. Weintraub et al. [60] investigated and evaluated the efficacy of Flashbots as a solution to the MEV crisis through a data-driven methodology in which they gathered data from over 4 million blocks with additional public Flashbots data. They analyzed Flashbots usage by looking at the number of Flashbots bundles (transaction groups) in collected blocks, comparing the usage of Flashbots blocks and non-Flashbots blocks on the whole network, and seeing a greater occurrence of Flashbots blocks. They also attempted to measure the hashing power, miner participation, and especially the usage of MEV transactions. The findings show a rapid growth in Flashbots usage, reaching 99,9% hashpower on Ethereum in 5 months, and that 80% of MEV extraction happens through Flashbots. However, there's no clear evidence to claim any of Flashbots' missions (to increase the transparency of MEV in the mempool, to democratize MEV, and to distribute the benefits of MEV more evenly) were accomplished. Weintraub et al.'s comprehensive assessment can be valuable for designing a better MEV solution, e.g., for Algorand or any blockchain platform.

Beside research that tries to explore the current situation of DeFi's activities by analyzing historical data, there are also other attempts to use that data to develop and verify tools that allow them to investigate the market's issues. For instance, Zhou et al. [36] created two bots, DeFiPOSER-ARB for detecting profitable arbitrage transactions and DeFiPOSER-SMT for discovering more demanding trades than arbitrage, such as bZx attacks or MEV opportunities. The performance of the two programs in a simulated environment and historical block data shows impressive results.

- DEFIPOSER-ARB estimated to generate weekly revenue of 191.48 ETH (76,592 USD) and DEFIPOSER-SMT 72.44 ETH (28,976 USD) based on historical data from Dec 2019 - May 2020.
- Highest single transaction revenue was 81.31 ETH (32,524 USD) for DEFIPOSER-ARB and 22.40 ETH (8,960 USD) for DEFIPOSER-SMT.
- Total estimated revenue over 150 days was 4,103.22 ETH (1,641,288 USD) for DEFIPOSER-ARB and 1,552.32 ETH (620,928 USD) for DEFIPOSER-SMT.
- Majority of strategies require less than 150 ETH initial capital without flash loans, reduced to less than 1 ETH with flash loans.
- Average computation time per block was 6.43 seconds for DEFIPOSER-ARB and 5.39 seconds for DEFIPOSER-SMT, fast enough for real-time operation.

Methodologically, DEFIPOSER-ARB is designed to build a graph of DeFi markets, detect negative cycles using the Bellman-Ford-Moore algorithm, and use local search to find profitable arbitrage parameters. DEFIPOSER-SMT, on the other hand, models DeFi protocols

and translates to logical representation in the Z3 theorem prover. It then applies heuristics to prune search space and uses binary search to find maximum revenue. During this work, the authors also explored interesting insights into the market; for instance, the number of holders and markets of a token can indicate the number of transactions related to that token, or the trading frequency of a token can affect the success rate of trading strategies on it. These findings are applicable for other blockchain platforms, including Algorand, and thus integrated into our data tool.

In their work in 2021, Xia et al. [61] propose an approach for identifying scam tokens and liquidity pools on the Uniswap DEX, leveraging machine learning techniques. The study first collected over 20 million transaction events, involving 21,778 kinds of tokens and 25,131 liquidity pools in total. The dataset is used to train a machine learning model designed to classify tokens based on a set of characteristics. The first aspect is the token's active period, since scam tokens and pools are usually short-lived. The number of transactions is also taken into account since it reflects the popularity and volume of a token. The marks of investors in a token can also be relevant, since victims of scam tokens are, as the authors observed, mostly inexperienced. Therefore, for each token, investor details such as the average number of trading pools they interacted with and the average number of transactions are taken into consideration. Furthermore, the number of liquidity pools a token involves, along with its trading volume and total liquidity on Uniswap DEX, are also the measure for the machine learning classification model. According to the results of this study, more than 10,000 tokens and 11,000 liquidity pools are flagged as scams. Alarmingly, this represents 50% of all tokens and 45% of all pools on Uniswap, responsible for a total trade volume exceeding $365 million. Additionally, tactics involving unrestricted token minting, sale restrictions, and advance-fee tokens were identified. Over 70,000 scam-related addresses were traced, encompassing token creators, pool creators, and collaborating entities. Collectively, these scammers have reaped over $16 million from close to 40,000 unsuspecting victims. These findings inspire us to present insightful yet intuitive metrics regarding tokens' usage frequency or DEX's activities for Algorand on our data exploration tool.

A recent study conducted by Öz et al. [52] aims to specifically investigate the MEV landscape on Algorand and reveals interesting findings. Over nearly two years, from September 2021 to July 2023, they detected more than 1.1 million arbitrage events spanning 13.7 million blocks, with profits surpassing $251,650. A deep dive into the distribution of these arbitrages revealed a dominant player: one particular searcher alone was responsible for executing 653,001 arbitrages, represents 57% of all detected arbitrages, and contributes to 44% of the entire profit pool. By analyzing the MEV activities, the authors observed 265,637 Batch Transaction Issuance (BTI) events. In such occurrences, a single address would dominate, taking up over 80% of a block's capacity. This dominance has the potential to force Algorand first-come, first-serve transaction ordering to a fee-based model, thereby facilitating frontrunning. The findings of this paper are particularly insightful for understanding the MEV landscape on Algorand.

5. Development of the Data Exploration Tool

In Chapter 3, the capabilities of existing block explorers on Algorand were comprehensively examined, while Chapter 4 provided a discourse on the current interest within the research community pertaining to DeFi ecosystems. Through this examination, it becomes evident that there exists a vacuum in the representation of certain metrics related to the DeFi space on the Algorand blockchain across available tools. These metrics can, however, bring valuable insights, which will be discussed further in the next chapter, to different groups of users.

This chapter pivots to a more constructive theme, detailing the development of our data tool. This tool is engineered to be capable of collecting and storing data in a data warehouse, as well as transforming and visualizing data on a user-friendly web interface. We begin by introducing the architecture of our data tool, AlgoSight, and its components, along with some constraints it should fulfill. The design is based on proven principles with a few adjustments. We then proceed to describe in detail our implementation of AlgoSight as a "minimum viable product", which is still able to provide the core functionalities, such as data collection, transformation, and visualization.

5.1. Architecture, Components and Constraints

A typical data exploration application primarily seeks to provide users with intuitive tools to observe, comprehend, and analyze data effectively through its front-end features, such as search, filter, and visualize data, while maintaining the quality and integrity of the data through pipelines in the back-end.

In practice, the specific implementations of data pipelines vary for different use cases, depending on the complexity of the data set and analytical needs; hence, there is no standard design that would fit every scenario. However, there are certain efforts in research papers to conceptualize the building of data pipelines, for instance in [62], where the authors propose a conceptual model (Figure 5.1) that was validated through a case study with companies from different domains. We adopt this idea and tailor it to suit our use cases.

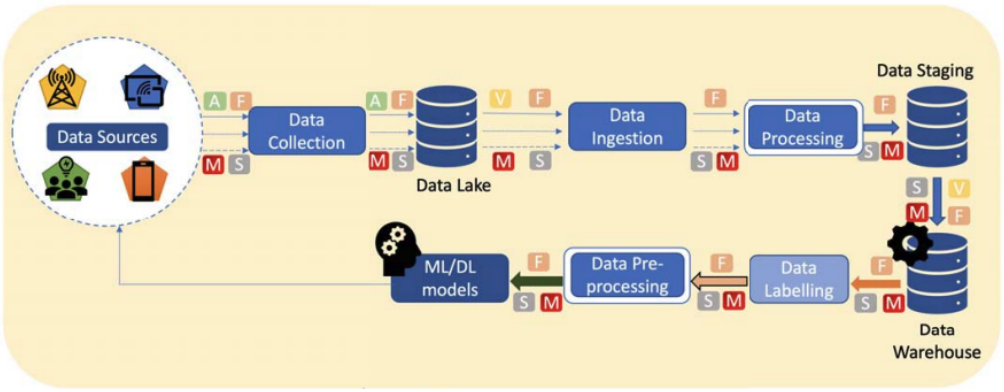

Figure 5.1.: Conceptual Model Data Pipeline [62].

AlgoSight's Architecture and Components

Below is our design of AlgoSight's architecture, which can be divided into two parts: data pipelines in the back-end and the analytical tools providing users with metrics and visualizations in the front-end.

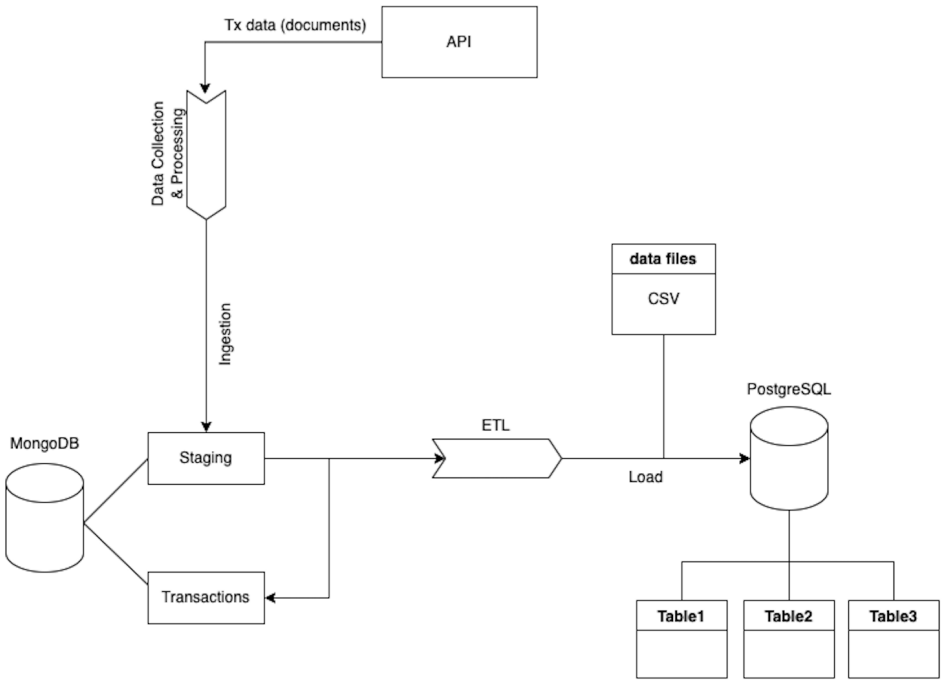

Figure 5.2.: AlgoSight Data Pipeline Architecture.

We first discuss the design and components of the data pipelines. In Figure 5.2 the arrows illustrate the flows of data. The first component of the pipeline is the **Data Source**, which is where the data is generated. This typically comes from internal systems, software, log files,

or external APIs. In our case, we will use the public API provided by *algonode*. The Data Collection and Processing pipeline takes responsibility for triggering the API call to fetch data. The data retrieved from the API is transaction data and is in JSON format. However, not all the data fields in the response are relevant for later analytical purposes. Therefore, we first implement a processing step for each batch of data fetched and then load it into the data warehouse.

Data Warehouse is the first destination for data flowing into our system. The choice of data warehouse for our tool is MongoDB. It is a popular NoSQL database specifically designed to manage large volumes of unstructured or semi-structured data. Unlike traditional relational databases (RDB), which utilize fixed-schema tables, MongoDB employs a flexible schema model, i.e., documents (equal to entries in RDB) in a collection (equal to tables in RDB) can possess diverse fields, offering flexibility in data representation. This is particularly suitable for data in JSON-like format. Moreover, its horizontal scalability and performance optimizations assure that it will be able to facilitate the expansion of the application in the future.

Data loaded into MongoDB must first be put in the **staging area**. This consists of the collection(s) responsible for temporarily storing data before it is loaded into the final data warehouse, as illustrated by the "Transactions" collection. The purpose of having a staging area is to implement any further data cleaning or validation steps or to facilitate the loading of data into another database. In our case, this staging area also makes the update process of data into the Postgres database easier. Every time new data is fetched from APIs, the ETL pipeline should then be triggered to perform its operations on the staging area and easily update new data to Postgres without having to distinguish it from old data. The update process inside MongoDB is carried out by a separate component (not illustrated in the Fig.) by simply transferring data from collection(s) in the staging area to the main collection(s).

Extraction, **T**ransformation, and **L**oad (**ETL**) pipeline takes the responsibility for extracting data from MongoDB, more specifically from the staging area, and performing certain transformation processes. They help convert the extracted data to fit the desired structure or format of the destination database. Here in our case, data from the staging area is in JSON format, and it needs to be transformed into data that can be added to tables in a RDB but still retain certain relations between its attributes. After the transformation step, the ETL pipeline will load the transformed data into corresponding tables in the destination database, which is PostgreSQL.

PostgreSQL is our choice for an analytical **database**. It's a popular RDB, renowned for its extensibility, robustness, and strong adherence to SQL standards. When it comes to analytical queries, Postgres is especially suitable due to its capability to efficiently join large datasets and perform aggregations, which are integral to data analysis. In addition to storing data loaded from ETL, Postgres also ingests additional CSV files to enrich the dataset. These files contain rather static information, such as the name, application ID of DEXs, and asset ID of

tokens on Algorand. A notable exception is the price historical data, which we collected from Yahoo Finance [1].

Moving forward, we'll discuss the design of the front-end as well as analytical functionalities. To present the data to users through metrics tables and visualizations, we utilize Streamlit to create an interactive web-based user interface (UI). Streamlit is an open-source Python framework designed to help developers build and share data projects as web applications with minimal effort. The data tool built with Streamlit, after being deployed to the Streamlit Community Cloud, can be accessed by public users.

On the UI of AlgoSight, multiple pages will be implemented. Each of them is meant to display a group of metrics and visualizations. Users interact with the data tool by navigating to the desired metric page (illustrated as "Pages" in Figure 5.3) and performing their actions. On AlgoSight's pages, users should be able to sort, filter, and see data in the form of tables or charts.

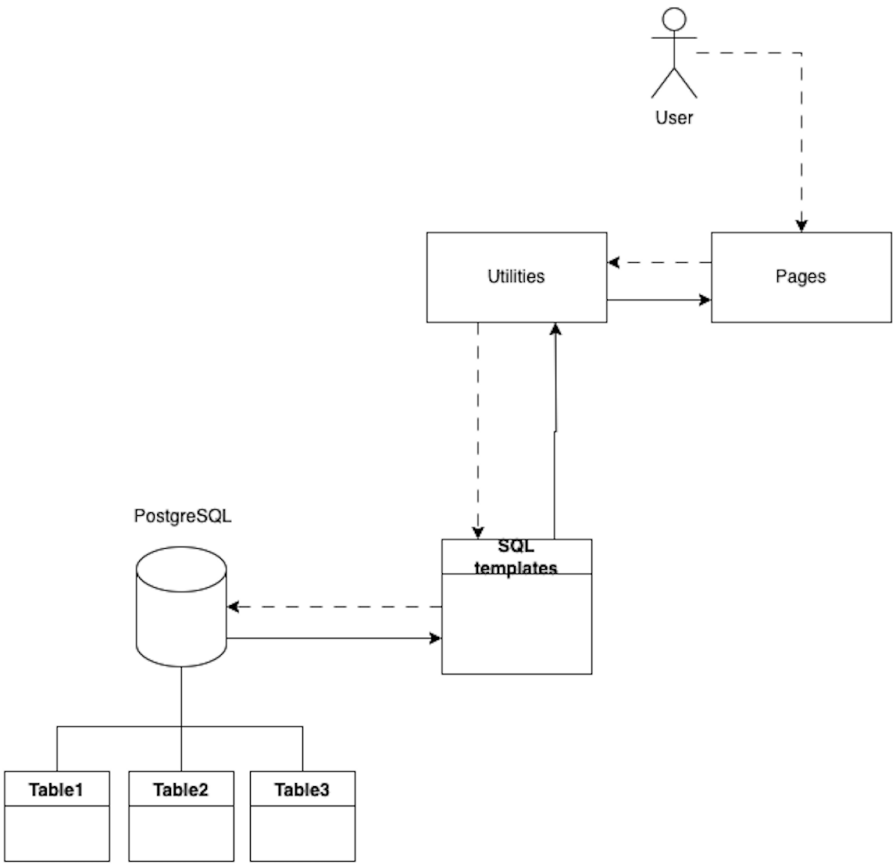

Figure 5.3.: AlgoSight Analytical Pipeline.

[1] `https://finance.yahoo.com/lookup`

"Utilities" is the component containing a variety of useful functions, such as getting desired data through executing queries, creating tables, etc. In the Figure 5.3 the dashed lines are supposed to illustrate the "requesting" interactions, such as user-requesting graphs, function calls, and query executions, whereas the solid lines represent data flows.

The content of each page is different, but the underlying processes are fundamentally the same. Different pages can utilize some common functions, but they typically involve different SQL queries to gather the relevant data. These queries are built by combining templates with the corresponding user's input, such as a date range. Ultimately, the complete query is passed to the PostgreSQL database to retrieve the desired data.

Constraints

Beyond the aforementioned functionalities, our data pipeline, like every software product, must also fulfill certain constraints to operate effectively. These requirements usually focus on the quality of performance, including aspects of usability, speed, and other attributes that determine how efficiently the software functions. Following are some typically crucial criteria for a data pipeline, which we want to apply for the development of AlgoSight.

- **Usability:** Paramount in any software, especially data tools, usability refers to the ease with which users can navigate and operate the tool. An intuitive user interface is vital, ensuring that even those unfamiliar with the tool can quickly become proficient, eliminating the need for extensive training.
- **Performance:** For a data exploration tool, efficiency is crucial. It should guarantee reasonable processing times, even in the case of large or complex datasets, while maintaining a seamless user experience. Excessive lag or delay can impede analytical processes and user satisfaction.
- **Scalability:** As data grows, the tool's capabilities should not wane. Scalability is about the software's ability to gracefully handle increased data volume, ensuring that as datasets expand, performance remains consistent and unhindered.
- **Documentation:** Comprehensive documentation can significantly enhance the tool's usability and maintainability. This includes user manuals guiding non-experts, help sections for troubleshooting, and developer documentation detailing the software's architecture for potential enhancements or integration.
- **Maintainability:** It's a critical aspect for a data tool as well as for any software product. Maintainability pertains to the ease with which updates, enhancements, or additions can be integrated, i.e., they can be easily modified to correct faults, improve performance or other attributes, or adapt to a changed environment. Because of the modular nature of data pipelines, if correctly implemented, they should have low coupling and high cohesion. Furthermore, it's a good practice to have log files for each stage of the whole process for better monitoring the data flow and detecting the root of issues. The data tool should be designed in such a way that new minor features can be added or existing ones modified without causing extensive system overhauls.

5.2. Implementation of the Data Pipelines

After laying out the basic design of our data exploration tool, AlgoSight, in this section we will delve into the detailed implementations, first and foremost, of the data pipelines.

5.2.1. Setup

The first thing that needs to be done is to install and set up the databases, MongoDB and PostgreSQL, and look for relevant APIs. Here, we use those provided by *algonode*. For interactions with databases in Python, we use available libraries like *psycopg2* for PostgreSQL and *pymongo* for MongoDB. They provide a variety of functions for connecting, inserting data into tables, as well as retrieving data.

5.2.2. Data Collection and Processing

The implementation of data collection and processing can be batch processing or stream processing. In batch processing, data should be collected over a specific period and then loaded into the system all at once. The other paradigm, stream or real-time processing, involves ingesting data nearly instantaneously as it's generated. Batch processing is suitable for use cases where there's a large volume of data to be handled and the immediacy of the data is not crucial, such as when making analyses or reports. Whereas stream processing is utilized for time-sensitive data-driven events such as anomaly detection and adjusting product recommendations.

For this situation, we are interested in fetching data about transactions on Algorand to facilitate the computation of metrics and visualization. Therefore, a batch processing approach is chosen, i.e., for collecting (new) data, the tool repeatedly runs API calls over a range of block numbers, using the endpoint `v2/blocks/{round_number}`, then processes and loads data into the target database. In this implementation, we work solely with historical data; hence, we fetch data from the algonode API based on block numbers predetermined.

The chosen data set is transactions within 1 month, from 01.06.2023 to 30.06.2023, which ranges from block number 29467143 to 30161443.

Code 5.1: Code Snippet `fetch_blocks()`.

```
import requests, json, logging, time
BATCH_SIZE = 1000 # number of blocks fetched and processed in an iteration
API_URL = 'https://mainnet-idx.algonode.cloud/v2/blocks/'

def fetch_blocks(start: int, end: int) -> list:
  data = []
  for block_nr in range(start, end+1):
    for attempt in range(3): # Retry up to 3 times
      response = requests.get(API_URL + f"{block_nr}")
```

```
    if response.status_code == 200:
      try:
        block = response.json()
        data.append(block)
        break
      except json.JSONDecodeError:
        logging.error(
          f"Failed to decode JSON response for block {block_nr} "
          f"on attempt {attempt+1}: {response.text}")
        time.sleep(1) # Sleep for 1 second before retrying
    else:
      logging.error(f"Failed to fetch block {block_nr} "
                    f"on attempt {attempt+1}: {response.status_code}")
      time.sleep(1) # Sleep for 1 second before retrying
  else:
      logging.error(f"Failed to fetch block {block_nr} after 3 attempts.")
return data
```

In Code 5.1 is our implementation for the `fetch_blocks()` function, which takes the starting and ending block numbers as arguments and returns a list of transactions. Due to specific memory limitations, the data collection process is implemented so that it divides the whole dataset into smaller batches (`BATCH_SIZE = 1000`) and runs iterations of `fetch_blocks()` on each subset. The fetched data from each iteration is then processed and loaded into MongoDB, freeing up memory space for the next iteration.

For later convenience in the transformation phase of ETL, there are a few minor processing operations performed on the data fetched from the API before it is loaded into MongoDB, such as filtering out irrelevant fields or adding dummy fields to maintain consistency.

5.2.3. ETL Pipeline

The next pipeline that needs to be implemented is the ETL from MongoDB to PostgreSQL. This pipeline, as the name suggests, requires the implementation of three steps: extraction, transformation, and load. The complexity of the first and third steps depends on the data format, data source(s), and destination(s), while the second step relies on the specific analytical needs and purposes.

Code 5.2: Example Transaction Data.

```
{
  "_id": {
    "$oid": "64de478da4bbc43b0e75fd1f"
  },
  "application-transaction": {
```

```
    "accounts": [],
    "application-id": 971335937,
    "foreign-apps": [
      971335616
    ],
    "foreign-assets": []
  },
  "id": "3IXJEORMC7YSRKONZIBN5GNBXZVJKUCMS6YOX7BNQZSGZBTXQTTQ",
  "tx-type": "appl",
  "sender": "TCEIBM7IHGQHX7JCN43AQLYAZVCWCZK4IMTUDB7E4KSV6J377J4XJ3ND3I",
  "fee": 2000,
  "confirmed-round": 29473143,
  "group": "Ty2ZI8tGC5Kql3wP4yjxUY7iZJGdS8WwOIJyJVCSZSA=",
  "inner-txns": [
    {
      "application-transaction": {
        "accounts": [],
        "application-id": 971335616,
        "foreign-apps": [],
        "foreign-assets": []
      },
      "id": "N/A",
      "tx-type": "appl",
      "sender": "MMQQL75R4E62VETQEEGYANZNF2YYFLLT22SDRIRRIYRIVKS3XN2LX63SNE",
      "fee": 0,
      "confirmed-round": 29473143,
      "group": "N/A",
      "inner-txns": "N/A",
      "round-time": 1685600160
    }
  ],
  "round-time": 1685600160
}
```


We need to first look at the structure of transaction data retrieved from API calls (see Code 5.2). The data format is JSON with required fields such as `'id'`, `'tx-type'`, or `'sender'` indicating the id, the type of the transaction, and the account that triggered it, respectively. Some other fields are optional, i.e., they might exist in one transaction but not in the other. For example, `'asset-transfer-transaction'` only exists in transactions of this type. On Algorand, a transaction can also contain a set of inner transactions, listed in the field `'inner-txns'`. These transactions might in turn have their inner ones, too. Inner transactions don't have their own ID but are binding to the "outer" transaction, and all of them must be registered together at the same time and in one block.

Code 5.3: Code Snippet `generate_postgres_data()`.

```
app_txns =[]
axfer_txns = []
def generate_postgres_data(transaction:dict, level:int=0, parent_id:int=None):
  related_tx = transaction['id'] if parent_id==None else parent_id
  match transaction['tx-type']:
    case 'appl':
      app_txns.append([
        transaction['application-transaction']['application-id'],
        convert_datetime(transaction['round-time']),
        related_tx,
        level
      ])
    case 'axfer':
      axfer_txns.append([
        transaction['asset-transfer-transaction']['asset-id'],
        int(transaction['asset-transfer-transaction']['amount']),
        convert_datetime(transaction['round-time']),
        related_tx,
        level
      ])
  if transaction['inner-txns'] != "N/A":
    for tx in transaction['inner-txns']:
      generate_postgres_data(transaction=tx, level=level+1, parent_id=
          related_tx)
```

In the code above, the function `generate_postgres_data()` is implemented as a transformation function to generate data that is both suitable for PostgreSQL and relevant for analytical purposes. For later computations and visualization of metrics, we are particularly interested in application call transactions (`"tx-type": "appl"`), and asset transfer transactions (`"tx-type": "axfer"`). `generate_postgres_data()` takes a transaction (in the form of a dictionary) as the required argument. The transaction is then categorized by its type and put into either of the two corresponding lists. Since RDB doesn't support nested data, two optional arguments of the function are introduced to handle inner transaction fields: `level` indicates the "inner transaction level" and level = 0 means the associated transaction is not among the inner transactions of any other transactions, and therefore its `parent_id` should be *None*.

In case the transaction has inner transactions, `generate_postgres_data()` will recursively process each inner transaction, incrementing the `level` by 1 and setting the `parent_id` to the ID of the current transaction. By doing so, all transactions will have IDs, and we can still preserve the relationship between transactions. The extraction and load phases of our ETL are rather straightforward since there's only one source and one destination database. Similar

to the data collection and processing pipeline, our ETL pipeline is also implemented to run in multiple iterations, i.e., it extracts, transforms, and loads data batch by batch.

5.3. Implementation of the Analytical Functionalities

After the data is loaded into tables in PostgreSQL, it's ready to be used for computations and visualizations.

5.3.1. Metrics Computation

As mentioned in the previous section, we are particularly interested in data about application calls and asset transfer transactions. They are materialized to `app_txns` and `axfer_txns` tables in PostgreSQL.

	app_id bigint	round_time timestamp without time zone	related_tx character varying (100)	inner_tx_level integer
1	1002541853	2023-06-01 02:00:00	B4KLRP5UJ4FCUGVNVXKRLTY6RCB56TKHVBFH7AHIZQ2RIE5FOOFQ	0
2	971335937	2023-06-01 02:00:00	NO7H4TN4CG5FPJXW2O6BH7P2JWSBX73F7IZCENPIB4CSZ7LJ6CRA	0
3	971335616	2023-06-01 02:00:00	NO7H4TN4CG5FPJXW2O6BH7P2JWSBX73F7IZCENPIB4CSZ7LJ6CRA	1
4	1040271396	2023-06-01 02:00:00	GWIOTPDZWUH4CS25JINENEGQO2VVHY6CMTLWC6KYO3DN57SX56…	0
5	855331006	2023-06-01 02:00:00	LXQYAUIC722VAFFVXXKSWC7U4VYG4DJ7FTKOCEUBC7GKSB7TJUTQ	0
6	0	2023-06-01 02:00:00	LXQYAUIC722VAFFVXXKSWC7U4VYG4DJ7FTKOCEUBC7GKSB7TJUTQ	1
7	649588853	2023-06-01 02:00:00	A5NIWCM3UQPUR3LDWGNN4U2HFH57TFJXGRFNEV3GUKLBHN5FVT…	0
8	1104858577	2023-06-01 02:00:00	HW7RC7NHHYIVBQB6KLH7WB7R7TA54EIU5KVJKVXEUDN56FSZBYKQ	0
9	1114264527	2023-06-01 02:00:00	EANXIXGYTNUK2AZWPJSKIDDVCSFYKEQ2WHWNWWIG4BBS4ISQ4Q…	0
10	1009583905	2023-06-01 02:00:00	U4CJQ5BCXA7FZPDLKDWI6TARNRNLRMSIISFDSN7P3R6UYVGC4K2Q	0

Figure 5.4.: `app_txns` Table

	asset_id bigint	amount numeric	round_time timestamp without time zone	related_tx character varying (100)	inner_tx_level integer
1	27165954	0	2023-06-01 02:00:00	7BD4AVLD4DTWGQY7UM6ABRGPZL2CHS3VGLL6COVXPOEDHHX4O…	0
2	1114264136	0	2023-06-01 02:00:00	ME4KSALHUG37WJ3VIUVGBY4HTVP66HIXZLAKUXZ4FZMIOTNGZPYQ	0
3	744665252	33700	2023-06-01 02:00:00	KCYDKS26ZSL75V54GTI7V6FLR7EWWAIEVZ5J7EPWR2YTC3VPIRXA	0
4	312769	9098288	2023-06-01 02:00:00	YHDSAZKNYXPC4CP7R44ZGLMT3E2SVO6HSAZI6WIGPVNKUPDQVX…	0
5	1114264136	552725	2023-06-01 02:00:00	B4KLRP5UJ4FCUGVNVXKRLTY6RCB56TKHVBFH7AHIZQ2RIE5FOOFQ	1
6	27165954	0	2023-06-01 02:00:00	MEZHBD5S5QW2AGE4F2ECQU6CL4B5QJ7GODGXFOZ3IECNH62TVL…	0
7	1104858578	1	2023-06-01 02:00:00	HW7RC7NHHYIVBQB6KLH7WB7R7TA54EIU5KVJKVXEUDN56FSZBYKQ	1
8	1104858578	0	2023-06-01 02:00:00	5AMPI65HHVIZWD6FCNTD7OCWWXV2TQ43UB5YMESY4FMG5EYJS…	0
9	312769	0	2023-06-01 02:00:00	I3HIR7ID7QHPAUYW2E7ZEFG4BS5QCZ7TQGQGNAXZTKXWU7KMMG…	0
10	287867876	0	2023-06-01 02:00:00	NWEPNMW4EIMQCCXASO77P7EAJ57RHVNG4GSHXELNCLX2JGGQI…	0

Figure 5.5.: `axfer_txns` Table

From these two base tables, we create views (virtual table which is the result set of a stored query) that will be repeatedly used for actual queries later.

`txns_on_dexs` contains all application calls on any of the DEXs we are interested in (see Code A.8 for SQL code).

	app_id bigint	platform character varying (100)	tx_date date	related_tx character varying (100)
1	605929989	AlgoFi	2023-06-01	2GYP3YPF2HMWSJEEJ4UWKIO5N3A3WP62I3RAEWF2DYMKZRUIS...
2	605929989	AlgoFi	2023-06-01	2HZLL5OHZML7BBQZ4M35RUAGQU6QD7ZL63AQBWA5OEDJV7S5...
3	605929989	AlgoFi	2023-06-01	2J6YR5BEI3ZKZPPZTXKJRSTFCMQPONCMPCXTPNMQ26CJH5V2...
4	605929989	AlgoFi	2023-06-01	2LB572OCZGWFBJIVDFKK6L7RKIZXA3QYWHIDVQOZLQVKPRLWCP...
5	605929989	AlgoFi	2023-06-01	2NOJUF67SJCF475AX553T55LS4BOCKHARD6VERTFGYKQIKHUSP...
6	605929989	AlgoFi	2023-06-01	2PKGESYFHLZDHX2ZNSQOUMJNAUYZHGEXXQPKUJDGCX3WDID3...
7	605929989	AlgoFi	2023-06-01	2PNLQAU5OLBJKPD7GYZN3RT6LJR2ONUJHGJ5XFC5LTVOZ3NLW...
8	605929989	AlgoFi	2023-06-01	2PXDK4KXD7QGM47Q5QE46V6M5V5X74DAIZ5WNHHQLTPD7QOQ...
9	605929989	AlgoFi	2023-06-01	2PYUKSBPP2LNDPNFFA3QVATDXLRZ55WBW4KLKG7MNHPX2KLU...
10	605929989	AlgoFi	2023-06-01	2QR2BYSWJMVM6RKU2K2AJIZN5XFIFCISOCL33HZKWWBHVQ4KS...

Figure 5.6.: `txns_on_dexs` View

`txns_of_hot_tokens` consists of asset transfer transactions using any of the tokens under our concern (see Code A.7 for the SQL code).

	asset_id bigint	token_name character varying (100)	amount numeric	decimals integer	tx_date date	related_tx character varying (100)
1	312769	USDT	0	6	2023-06-01	22EZ5ZODL3BUGTZWVTZ5ZUDIC4RKMG5OIODVOM2ON7NILBY5N5UQ
2	312769	USDT	0	6	2023-06-01	22GAILTW42DSHPBZ6YCGQMOH4XKWU4TQI2G4CIG4ZITCZXFB4KLQ
3	312769	USDT	0	6	2023-06-01	2476ODDGBCMMS3TPKXAJ2CJ2NOP2YSMMJE54TAPVYM66H7TKM...
4	312769	USDT	0	6	2023-06-01	24LUNNWCF5E7OIMZHUC526WV4MMYRR467LM3PQYXPX5ZGKRML...
5	312769	USDT	0	6	2023-06-01	25ZNHV4ZJ44QRK3V6PZ5F3VYEOOGKL25IP6EH7XXNG73DVJQKGJQ
6	312769	USDT	0	6	2023-06-01	26CNZ56WRFWG2GNIQFRNAA7AFMETE57PL5PLSIGRKW75VZFXJNLA
7	312769	USDT	0	6	2023-06-01	275MECPT32NABRV3IA6YQB7EB5EH2ZMWHX4SNIDJWLJ3JYN3JZNA
8	312769	USDT	0	6	2023-06-01	27N6T22OST6BZ2IPAL3WF4WABZSKBRJF5JAL72EXN23F4EO4B23A
9	312769	USDT	0	6	2023-06-01	2BBJ6YRAABJJ43GOTWBK6WBUPPEBAYV4FDXWTRFXXK7LFE2XN3JA
10	312769	USDT	0	6	2023-06-01	2E6MCAH5AKQWWSHB6I3OBNU2T4BOQ3NBU577FUPVU23B57RVNI...

Figure 5.7.: `txns_of_hot_tokens` View

As previously described, all user's requests on the front-end boil down to executions of SQL queries. It's more flexible to create templates, in form of Python string, that can be turned into a complete query when combined with user input. Code 5.4 is an example of such a query template: `start_date end_date` are parameters inside the string `MOST_ACTIVE_DEXS`, whose values are passed in by dates selected by user.

Code 5.4: Code Snippet `MOST_ACTIVE_DEXS` in Python.

```
MOST_ACTIVE_DEXS = """
SELECT
    APP_ID,
```

```
    PLATFORM,
    COUNT(APP_ID) AS TOTAL_TX_COUNTS
  FROM TXNS_ON_DEXS
  WHERE TX_DATE BETWEEN '{start_date}' AND '{end_date}'
       GROUP BY APP_ID, PLATFORM
ORDER BY TOTAL_TX_COUNTS DESC LIMIT 10
"""
```

AlgoSight is implemented to provide users with following metrics from the data fetched from algonode API.

DEX & Token Metrics	**Details**	**Computation Logic**
Most Active DEXs	Requires start date and end date as user's input. Shows the most used DEXs (by transaction counts) during the chosen time period.	Count all entries in `txns_on_dexs` having `tx_date` between specified time frame, group by DEX `app_id` and `platform` (see Code A.1).
Most Active Tokens	Requires start date and end date as user's input. Shows the most traded tokens (by transaction counts) during the chosen time period.	Count all entries in `txns_of_hot_tokens` having `tx_date` between specified time frame, group by token (see Code A.2).
DEXs Activity Track	Requires start date and end date as user's input. Shows DEXs' activities of every day over the chosen time period.	Count all entries in `txns_on_dexs` having `tx_date` between specified time frame, group by DEX and date (see Code A.3).
Tokens Activity Track	Requires start date and end date as user's input. Shows tokens' activities of every day over the chosen time period.	Count all entries in `txns_of_hot_tokens` having `tx_date` between specified time frame, group by token and date (see Code A.4).

DEX & Token Metrics	Details	Computation Logic
DEXs Trading Volume	Requires start date and end date as user's input. Shows total trading volume of all transactions related to each DEX during the chosen time period.	Find all transactions related to DEXs and Tokens under concern by joining `txns_on_dexs` with `txns_of_hot_tokens`. Sum up the transaction amount and filter by dates lying between chosen time frame, multiply with corresponding tokens exchange rates against USD. Group by DEX (see Code A.5).
Tokens Trading Volume	Requires start date and end date as user's input. Shows total trading volume of all transactions related to each token during the chosen time period.	Find all transactions related to DEXs and Tokens under concern by joining `txns_on_dexs` with `txns_of_hot_tokens`. Sum up the transaction amount and filter by dates lying between chosen time frame, multiply with corresponding tokens exchange rates against USD. Group by token (see Code A.5).
DEX Activity by Tokens	Requires start date, end date and a specific DEX as user's input. Breakdown activities on the selected DEX in transaction counts of different tokens over the chosen time period.	Find all transactions related to any DEXs and any Tokens under concern by joining `txns_on_dexs` with `txns_of_hot_tokens`. Filter by dates between chosen time frame. Filter to one selected DEX, count all entries and group by token (see Code A.6).
Token Activity by DEXs	Requires start date, end date and a specific token as user's input. Depicts activities of the selected token across different DEXs over the chosen time frame.	Find all transactions related to DEXs and Tokens under concern by joining `txns_on_dexs` with `txns_of_hot_tokens`. Filter by dates between chosen time frame. Filter to one selected token, count all entries and group by DEX (see Code A.6).

Table 5.1.: DEX & Token Metrics on AlgoSight.

External Data

Beside the data fetched by our data pipeline in this implementation, we will use a data set that was generously provided by Öz et al. from their study [52] to illustrate MEV activities

on Algorand. The data set covers all blocks between *28 Sep 2021* at *12:55:52* and *03 Jul 2023* at *21:22:22*, it is processed for the their study and consists of:

- *arbitrage data* with details about arbitrageurs, arbitrage transactions, profit token, profit amount.
- *block data* with details about block time and the number of arbitrage transactions on each block.

Following are the metrics to be presented using this data set.

MEV Metrics	**Details**	**Computation Logic**
Arbitrage Activities	Depicts daily arbitrage activities (as transaction counts) throughout the whole duration of the data set.	From block data, count the number of arbitrage transactions and group by each date.
Arbitrage Counts per Token	Shows the total arbitrage transaction counts per token throughout the whole duration of the data set.	From arbitrage data, count all entries and group by token.
Arbitrage Profit per Token	Shows the total arbitrage profit in USD per token throughout the whole duration of the data set.	Join block data with price data to get the ALGO price of each date. Join the result with arbitrage data to calculate profit amount in USD. Sum up and group by token.
Arbitrageur Profit Leaderboard	Ranks and displays the list of arbitrageurs by earned profits.	Similar to Arbitrage Profit per Token, but at the last step we group by arbitrageurs instead of by token.

Table 5.2.: MEV Metrics on AlgoSight.

5.3.2. UI and Visualization

AlgoSight leverages the Streamlit framework to create an user-friendly UI. As mentioned, there are multiple pages with different sets of metrics and charts. To comply with the rule of the framework, pages are essentially Python files and must reside in the same folder named “pages”. The specific computation and visualization of each metrics is implemented in the corresponding page. Code 5.5 is an example how we presented the "Most Active DEXs" as a table of 10 most used DEXs. `user_date_input()` is a help function which we defined to

create a widget for users to enter their desired dates. The parameter `key` is for distinguishing between multiple widgets on the same page. After user clicks on button `Show graph`, a query will be created and executed using user's date input and corresponding template. The result is then displayed as a table.

Code 5.5: Code Snippet of Most Active DEXs.

```
import streamlit as st
from utilities import user_date_input, get_data_with_date_input, display_table
from sql_templates import MOST_ACTIVE_DEXS

st.subheader('Most Active DEXs')
# Ask for user input
date_input_top_dexs = user_date_input(key='Most Active DEXs')

# Calculate the data required by user and save to the corresponding session
    state
if st.button('Show graph', key='Most Active DEXs'):
  st.session_state['top_dexs_df'] = get_data_with_date_input(
    date_input=date_input_top_dexs,
    query=MOST_ACTIVE_DEXS
    )

# Visualize data saved in session state.
# Keep the visualization presented
# even when user interacts with other parts of the page or navigates away.
if st.session_state['top_dexs_df'] is not None:
  st.write(f"Most active DEXs between {date_input_top_dexs[0]} and {
      date_input_top_dexs[1]}")
  display_table(st.session_state['top_dexs_df'])
```

The output of visualization functions as well as the UI are presented in the next chapter and Appendix B, respectively. The full source code of AlgoSight can be found at `https://github.com/tung-michael/algosight`

5.4. Performance

In the course of this thesis, the dataset, despite comprising only a single month of Algorand transaction data, yields in excess of 950,000 entries in the `txns_on_dexs` view and over 4.7 million entries in the `txns_of_hot_tokens` view. Such substantial data volumes notably impacted the query performance of AlgoSight. To mitigate this issue, particularly given the hardware constraints (a personal MacBook Pro with an Intel i5 quad-core processor and 8GB RAM), we explored the subsequent approaches:

- **Materialize** views and create **index** for them. Views on PostgreSQL are virtual tables; a materialized view can be seen as a snapshot of a regular view. Instead of running the underlying query every time, it stores the query's result. This means that accessing a materialized view is just like accessing a table with pre-computed results, making data retrieval significantly faster. The integration of an index further reduces the query time. We created indexes on the `tx_date` columns of the materialized versions of `txns_of_hot_tokens` and `txns_on_dexs` views and observed a significant improvement. For instance, when running a complex query, such as compute trading volume (Code A.5) of 20 days, the duration diminished from 72 seconds down to 23 seconds.
- Utilize Streamlit's `@st.cache_data` decorator for caching the result of a computed query so that it can be re-used when the same input data is entered. Concurrently, we implement some SQL queries so that their results are used to calculate different metrics. The desired metrics can be achieved with just a few further computations done by Python pandas. By doing these, we leverage `@st.cache_data` to mitigate the query time in case the input data of these different metrics is the same.

Additionally, the `st.session_state` attribute is used to save the visualized data of the entered input (Code 5.5), so that the table or chart will not disappear after every user's interactions with the page. Although this feature does not directly contribute to reducing the querying time, it helps create a smoother user experience.

6. Evaluation

In the last chapter, we introduce the architecture as well as the implementation of our data exploration tool, which aims to fill the vacuum in the representation of certain metrics related to the DeFi space on the Algorand blockchain across available tools. In this chapter, we discuss the practicality of the data tool for certain target users, namely traders and MEV researchers on Algorand, by showing how the metrics and visualizations provided by AlgoSight are relevant and can be used to extract valuable insights for these target users.

For traders, market metrics are crucial in informing their trading strategies, understanding market conditions, and making investment decisions. Some fundamental metrics are:

- **Price and Price Movement:** Understanding the price trajectory helps traders determine entry and exit points for trades.
- **Token Trading Volume:** High trading volumes can indicate a token or platform's popularity and liquidity. Besides, when combined with price movement and other factors, it can signal a significant price change on the market.
- **Tokens Trading Frequency:** Similar to trading volume, the number of transaction counts can express how liquid a certain asset is. Furthermore, from the findings of the study by Xia et al. [61], a token that is frequently traded with a high trading volume over a decent period of time tends to be less of a scam token.
- **Total Value Locked:** For liquidity pools, a high TVL can indicate trust in the protocol and its potential profitability. It's a measure of the protocol's size and significance in the DeFi ecosystem.
- **DEXs Trading Frequency:** Each DEX platform often consists of multiple liquidity pools, facilitating the trading of different pairs of tokens. From Xia et al. study, the number of transactions executed on a DEX over a decent period of time can be used to assess the safety of the pool, since pools of scam tokens are usually very short-lived [61].

For MEV researchers, the market information above can give a sense of which tokens or DEXs are targets for extraction strategies, such as less frequently traded tokens, which can imply an easier capture of arbitrage opportunities [36]. Besides, they are interested in more sophisticated figures, such as:

- **Arbitrage / MEV profit over Time:** The aggregate value has been extracted through MEV / arbitrage activities. This metric helps researchers quantify the economic incentives and

impacts of MEV on the blockchain. High trading volumes on DEXes can be focal points for MEV opportunities, especially when they involve high-value trades susceptible to front-running.

- **Transaction Fee:** On blockchains with fee-based transaction ordering like Ethereum, MEV researchers would be particularly interested in this metric, as higher fees can hint at MEV strategies being deployed (e.g., priority gas auctions). However, on a first-come, first-served blockchain like Algorand, this seems to be the case only when the searcher first congests the block, potentially forcing the network to shift its transaction ordering mechanics to a fee-based system, as stated in the study by Öz et al. [52].
- **MEV Transaction Counts over Time:** DEXs with high MEV transaction counts over time are the MEV hotspot; focusing on investigating these DEXs might be better to spot trends and strategies and understand the activity of MEV bots.

While TVL, transaction fee, or price information are available on current Algorand block explorers, many of the metrics mentioned above are absent. In our implementation, AlgoSight mostly provides information about how tokens and DEXs are used over a time period. This information is, however, manifested from different angles:

- The **Most Active DEXs** and **Most Active Tokens** outputs give our target users a quick glance at which tokens and DEXs are the most and least popular over a time period. For this purpose, these metrics are displayed in a simple interactive table, allowing sorting by each column.

Most active DEXs between 2023-06-01 and 2023-06-08

	APP ID	PLATFORM	TOTAL TX COUNTS
1	1002541853	Tinyman AMM v.2	250424
2	818179346	AlgoFi	13329
3	818182048	AlgoFi	10567
4	620995314	Pact	7814
5	605929989	AlgoFi	6648
6	1075389128	Pact	5716
7	855716333	AlgoFi	4268
8	607645439	AlgoFi	2952
9	645869114	Pact	2046
10	835609896	HumbleSwap	1572

Figure 6.1.: Most Active DEXs from *01.06.2023* to *08.06.2023*.

Most active Tokens between 2023-06-01 and 2023-06-08

	ASSET ID	TOKEN NAME	TOTAL TX COUNTS
1	27165954	PLANET	720207
2	31566704	USDC	186762
3	796425061	COOP	124216
4	1096015467	PEPE	57085
5	312769	USDT	34955
6	287867876	OPUL	27483
7	818179690	AF-BANK-ALGO	20546
8	226701642	Yieldly	17722
9	470842789	DEFLY	12812
10	386195940	goETH	12727

Figure 6.2.: Most Active Tokens from *01.06.2023* to *08.06.2023*.

- The treemaps of **DEXs Trading Volume** and **Tokens Trading Volume** also inform users of how monetarily active these tokens and DEXs are. Looking at the examples below, we can see a consistent dominance of Tinyman as a DEX platform, while PLANET was the favorite token during the period between *01.06.2023* and *08.06.2023*, but due to the price, its trading volume is much smaller than the popular stable coin USDC or high-value tokens like goETH.

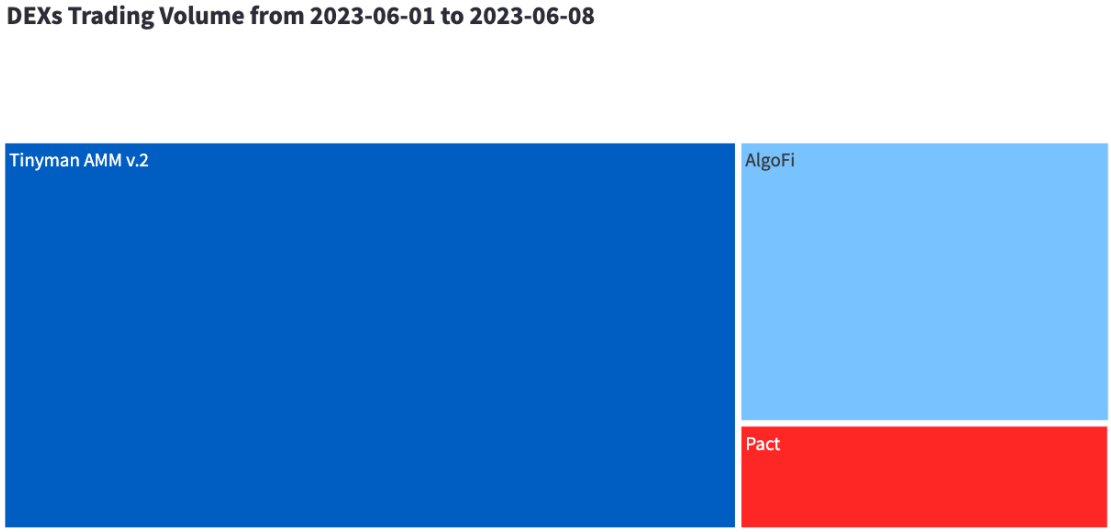

Figure 6.3.: DEXs Trading Volume from *01.06.2023* to *08.06.2023*.

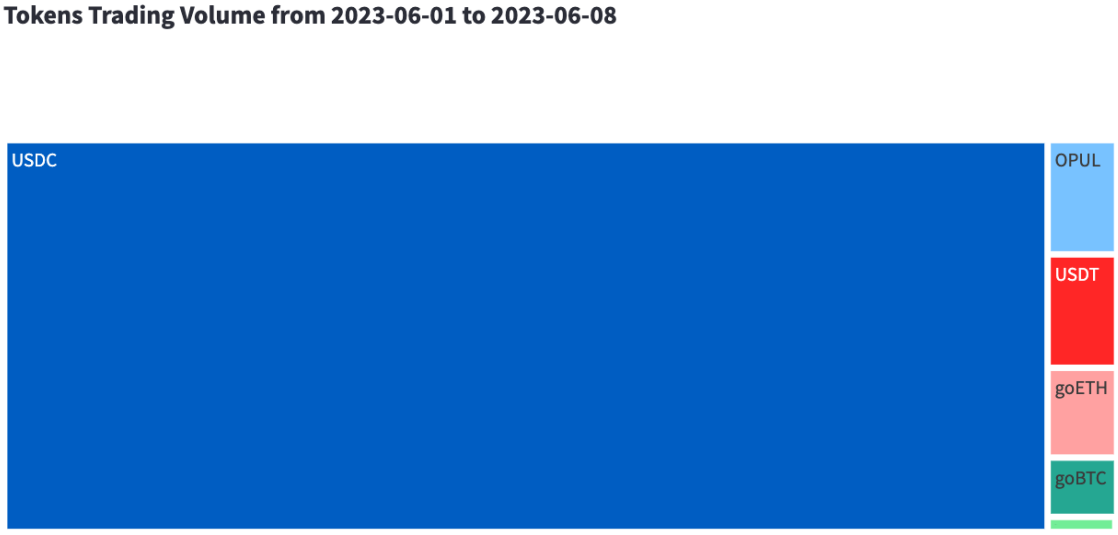

Figure 6.4.: Tokens Trading Volume from *01.06.2023* to *08.06.2023*.

- The daily **DEXs Activity Track** allows users to observe the activities of their interested platforms in terms of transaction counts. This chart would also be particularly useful to check and compare the consistency of each platform. Meanwhile, the **DEX Activity by Token** on each DEX (e.g., AlgoFi in Figure 6.6) illustrates which tokens are more frequently traded on the chosen platform and what their trading trajectories look like.

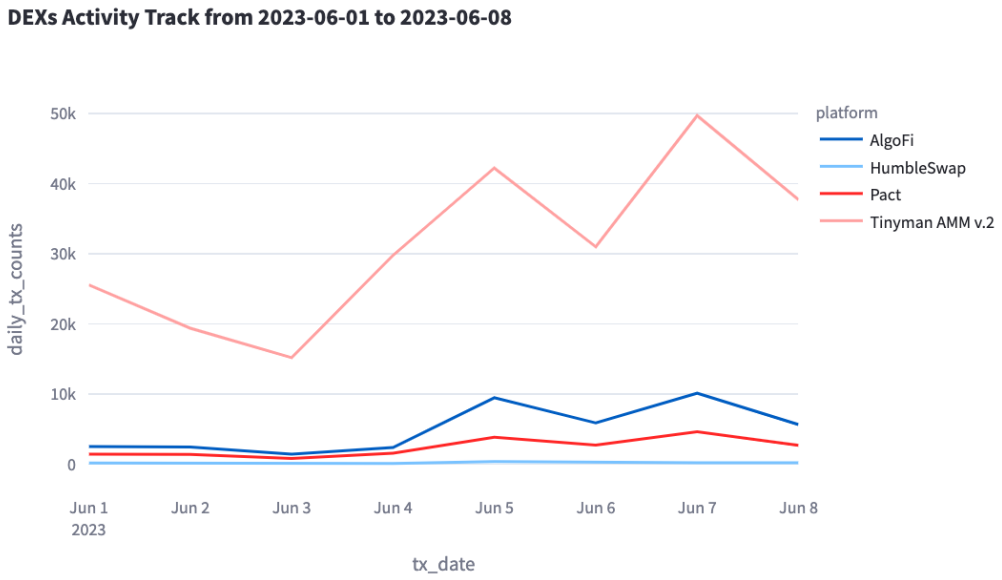

Figure 6.5.: DEXs Activity Track from *01.06.2023* to *08.06.2023*.

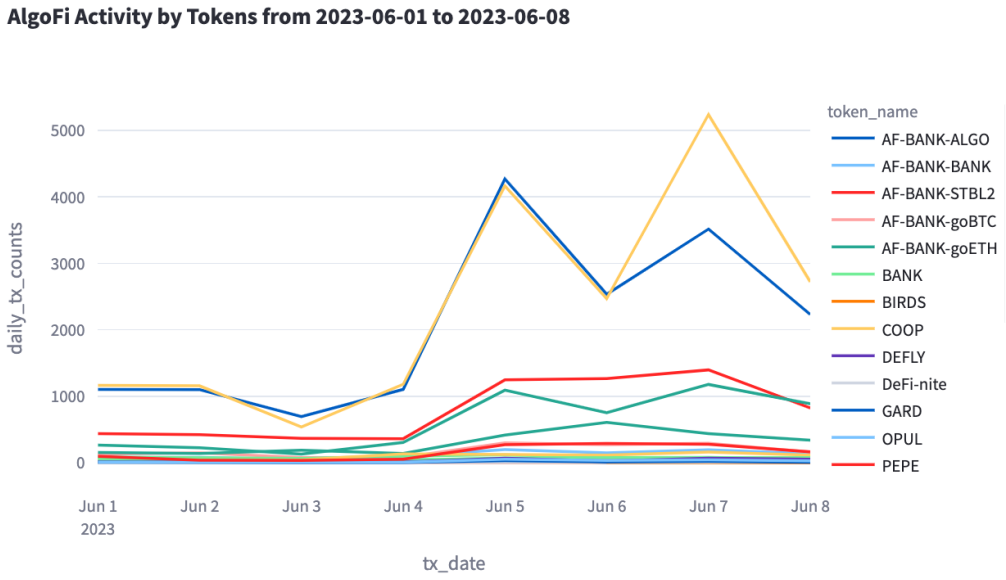

Figure 6.6.: DEX Activity by Tokens from *01.06.2023* to *08.06.2023*.

- Similarly, on the Tokens page of AlgoSight, we also provide a visualization of the transaction counts of all tokens during a time period. This daily **Tokens Activity Track** is useful for understanding and comparing tokens' trading frequency over time, while the detailed chart of **Token Activity by DEXs** illustrates where and how each token is popular.

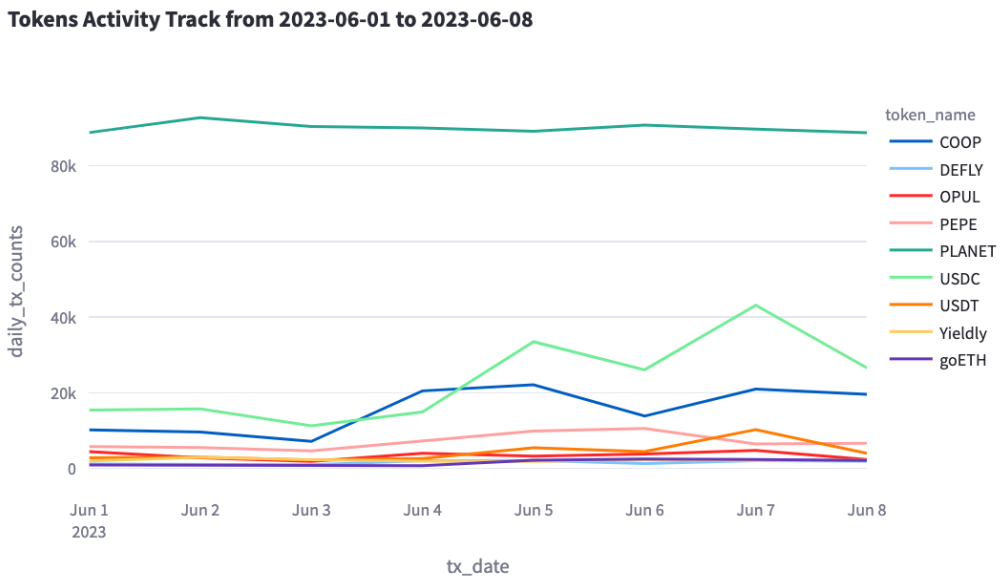

Figure 6.7.: Tokens Activity Track from *01.06.2023* to *08.06.2023*.

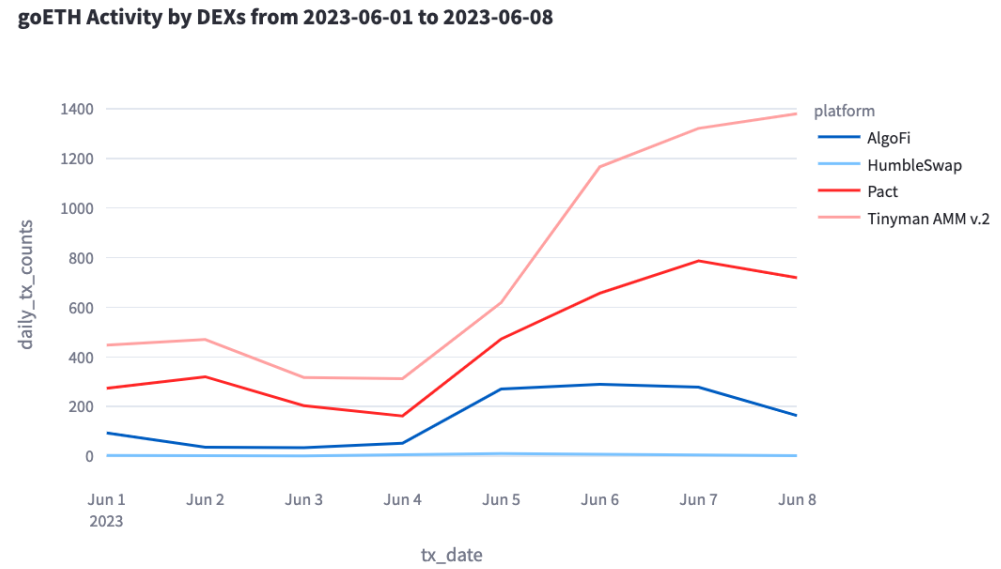

Figure 6.8.: Token Activity by DEXs from *01.06.2023* to *08.06.2023*.

Thanks to the data set provided by Öz et al. [52] in their study, we are able to display certain metrics from *28.09.2021* to *03.07.2023* about the MEV landscape on Algorand.

- **Arbitrage Profit per Token** and **Arbitrage Transaction Counts per Tokens** tell us specifically that ALGO is dominantly most often exposed to opportunities, both in terms of total transaction counts and total value in USD, and thus used to realize the profit. This insight can be the start of a discussion, for instance, about whether it is more likely to capture an arbitrage opportunity on the native token of a blockchain than on other tokens. The result of such a discussion can be beneficial for both researchers and traders.

Figure 6.9.: Arbitrage Profit per Token.

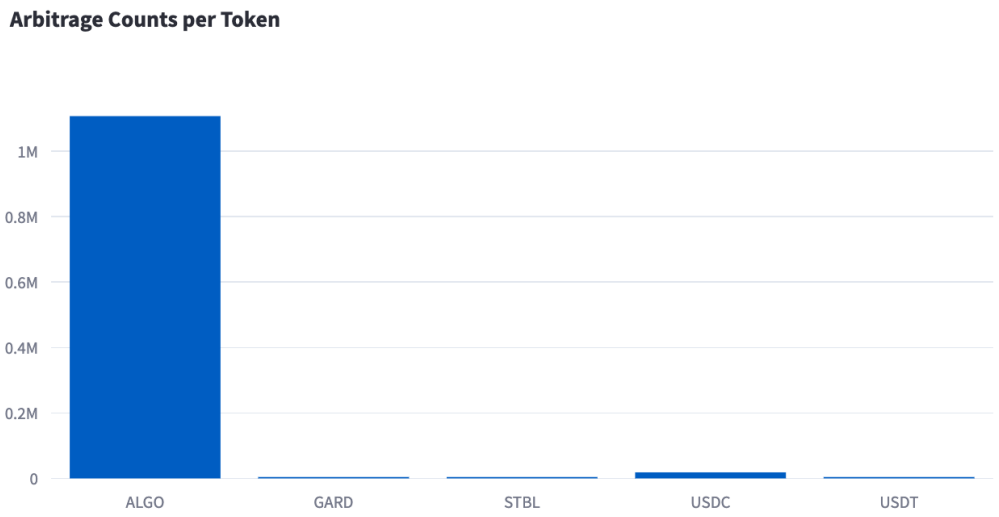

Figure 6.10.: Arbitrage Transaction Counts per Token.

- Furthermore, from the provided data set, we are also able to create an overview of arbitrageurs' profits and arbitrage activities over time. This can give researchers and traders a first impression of MEV-related activities and opportunities on the Algorand, as well as how profitable they are. These insights might inspire a study or help decide whether to develop a trade strategy on Algorand.

Arbitrageur Profit Leaderboard

	ARBER	PROFITS IN USD
1	AACCDJTFPQR5UQJZ337NFR56CC44T776EWBGVJG5NY2QFTQWBWTALTEN4A	110967.67
2	URKF45CZD6JGFIRBH67VQX6OCLUOXGDRRCCB3R2M7UXFY4EPOSQQ6VDQZU	31201.12
3	TEICJUFENMNMZXQTADIAUVJ25FRSYDZI3AYOBQAQQXJ4JZQDCS2RAUI634	25304.40
4	ODKHWTGQUBJ2I62QBLBL3BZP5YUSPJ5OVL7JHUKCJOE3T4YET6RXVT65QY	24897.54
5	EVESCVBC6VDIJAZM3HMUGYVQLKWHH4YJBMDV5EF65RMS67TFS5URZQ5YNY	18241.63
6	TZ3U7KHVHWPM34BJKIWKSSBRNS3LDJGSLHY2DIZVP7EJV5OKOJ5ZJW76FA	14391.43
7	EAFSBZIDWYH4BAR34FZHQXKKQ6IYRVITQPDM2XXY3RBKSRO6EIR6COTCN4	10484.67
8	XEYEWDWEHMIOAXFZN2HSBZJNCROOG7JKJLKFQSGG25JGE5UUZBZCGEWGEA	5935.95
9	MDC5Y5MOYKYRMOLR56ZQKYFQK2IR4LOOXGSHWSIRNJ3CT635FRB37YKVSA	3374.57
10	HS2YUNZNWS4S6YJUEZTHYLBTVOZ6YBBPXMIEF3PQOCSH5DPMUD24O677BQ	2947.39

Figure 6.11.: Arbitrageurs Profit Leaderboard in USD.

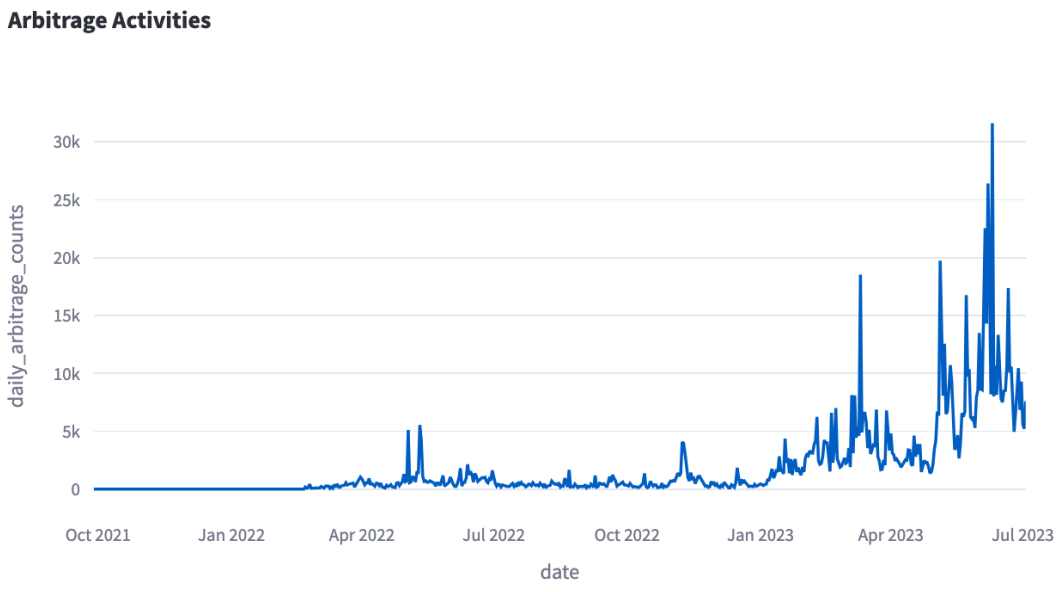

Figure 6.12.: Arbitrage Activities.

7. Conclusion and Future Work

In this thesis, we present a data exploration tool that aims to provide valuable insights to certain types of users, namely traders and MEV researchers. Our work starts with an investigation into existing tools, showcasing their potential but also identifying their limitations. While they offer some insights into the DeFi landscape on Algorand, there's a noticeable absence of depth and specificity in the information they present.

A concise review of related work on the DeFi ecosystem, spanning beyond Algorand, underscores a pronounced inclination toward methodologies that explore and quantify DeFi vulnerabilities and threats. When these study findings are contextualized within Algorand's DeFi landscape, they unveil a set of valuable, insightful metrics. These metrics, while potentially helpful for both traders and researchers, are currently missing from the available data exploration tools.

To propose a simple yet functional solution for that shortcoming, we present the design and implementation of our data exploration tool, AlgoSight, to provide novel and insightful metrics that are absent on even the most advanced block explorer on Algorand. AlgoSight is built following a proven conceptual model of data pipelines and is able to perform the fundamental tasks of a data exploration tool, including data collection, storage, transformation, executing analytical queries, and providing users with intuitive UI and visualizations. Subsequently, we explain the implications of the showcased metrics and their visual representations, aligning them with the interests of traders and researchers. This serves to underscore the pragmatic utility of our data tool.

As future work, we aim to improve the performance and UI of AlgoSight, especially in terms of query time. For a more extensive version of our data tool, we will expand it with more relevant, complex metrics. This could require more data sources and demanding computation resources and urge a deployment of cloud services. However, given AlgoSight's current architecture, it can seamlessly integrate new metrics and ensure that they are readily visualizable for users.

A. SQL Queries

In this appendix, we present the SQL codes for querying the metrics mentioned in Table 5.1.

Code A.1: Query for Most Active DEXs Metric.

```
SELECT
    APP_ID,
    PLATFORM,
    COUNT(APP_ID) AS TOTAL_TX_COUNTS
  FROM TXNS_ON_DEXS
  WHERE TX_DATE BETWEEN '{start_date}' AND '{end_date}'
        GROUP BY APP_ID, PLATFORM
ORDER BY TOTAL_TX_COUNTS DESC LIMIT 10
```

Code A.2: Query for Most Active Tokens Metric.

```
SELECT
    ASSET_ID,
    TOKEN_NAME,
    COUNT(ASSET_ID) AS TOTAL_TX_COUNTS
  FROM TXNS_OF_HOT_TOKENS
  WHERE TX_DATE BETWEEN '{start_date}' AND '{end_date}'
        GROUP BY ASSET_ID, TOKEN_NAME
ORDER BY TOTAL_TX_COUNTS DESC LIMIT 10
```

Code A.3: Query for DEXs Activity Track Metric.

```
SELECT
    APP_ID,
    PLATFORM,
    TX_DATE,
    COUNT(APP_ID) AS DAILY_TX_COUNTS
  FROM TXNS_ON_DEXS
  WHERE TX_DATE BETWEEN '{start_date}' AND '{end_date}'
        GROUP BY APP_ID, PLATFORM, TX_DATE
ORDER BY TX_DATE ASC
```

Code A.4: Query for Tokens Activity Track Metric.

```
SELECT
```

```
            ASSET_ID,
            TOKEN_NAME,
            TX_DATE,
            COUNT(ASSET_ID) AS DAILY_TX_COUNTS
          FROM TXNS_OF_HOT_TOKENS
          WHERE TX_DATE BETWEEN '{start_date}' AND '{end_date}'
               GROUP BY ASSET_ID, TOKEN_NAME, TX_DATE
        ORDER BY TX_DATE ASC
```

For both DEXs Trading Volume and Tokens Trading Volume, we run Code A.5 on the PostgreSQL database to get an intermediary table and implement the specific group by steps on that table using Python pandas. This approach can leverage the caching feature of Streamlit to reduce query time in cases where the input for these two metrics is the same.

Code A.5: Query for Trading Volume Metrics.

```
WITH TXNS_TOKENS_DEXS AS
       (SELECT PLATFORM,
                    TXNS_ON_DEXS.TX_DATE,
                    TOKEN_NAME,
                    SUM(AMOUNT) AS TX_AMOUNT,
                    DECIMALS
             FROM TXNS_OF_HOT_TOKENS
             JOIN TXNS_ON_DEXS ON TXNS_OF_HOT_TOKENS.RELATED_TX =
                TXNS_ON_DEXS.RELATED_TX

         WHERE TXNS_ON_DEXS.TX_DATE BETWEEN '{start_date}' AND '{end_date}'

             GROUP BY PLATFORM,
                    TOKEN_NAME,
                    TXNS_ON_DEXS.TX_DATE,
                    DECIMALS
             ORDER BY TX_DATE ASC, PLATFORM,
                    TOKEN_NAME),
       TRANSACTION_VOLUME AS
       (SELECT PLATFORM,
                    TX_DATE,
                    TOKEN_NAME,
                    TX_AMOUNT / POWER(10,DECIMALS) AS TRADING_AMOUNT
             FROM TXNS_TOKENS_DEXS)
SELECT PLATFORM,
       TX_DATE,
       TOKEN_PRICES.TOKEN_NAME,
       TRADING_AMOUNT,
```

```
        RATES_TO_USD,
        TRADING_AMOUNT * RATES_TO_USD AS TRADING_VOLUME_USD
FROM TRANSACTION_VOLUME
JOIN TOKEN_PRICES ON TRANSACTION_VOLUME.TX_DATE = TOKEN_PRICES.AS_OF_DATE
AND TRANSACTION_VOLUME.TOKEN_NAME = TOKEN_PRICES.TOKEN_NAME
```

For both DEX Activity by Tokens and Token Activity by DEXs, we run Code A.6 on the PostgreSQL database to get an intermediary table and implement the specific filter and group by steps using Python pandas. This approach can leverage the caching feature of Streamlit to reduce query time in cases where the input for these two metrics is the same. Furthermore, it makes the user experience smoother when switching between different DEXs or tokens while keeping the same date input.

Code A.6: Query for Detailed DEX / Token Activity Metrics.

```
SELECT
    TOKEN_NAME,
    PLATFORM,
    TXNS_OF_HOT_TOKENS.TX_DATE,
    COUNT(*) AS DAILY_TX_COUNTS
  FROM TXNS_OF_HOT_TOKENS JOIN TXNS_ON_DEXS
        ON TXNS_OF_HOT_TOKENS.RELATED_TX = TXNS_ON_DEXS.RELATED_TX
  WHERE
    TXNS_OF_HOT_TOKENS.TX_DATE BETWEEN '{start_date}' AND '{
        end_date}'
GROUP BY TOKEN_NAME,PLATFORM, TXNS_OF_HOT_TOKENS.TX_DATE
ORDER BY TX_DATE ASC, TOKEN_NAME, PLATFORM
```

Code A.7: Query for `txns_of_hot_tokens` View.

```
CREATE VIEW AS TXNS_OF_HOT_TOKENS

SELECT DISTINCT
  AXFER_TXNS.ASSET_ID,
        HOT_TOKENS.TOKEN_NAME,
        AXFER_TXNS.AMOUNT,
        HOT_TOKENS.DECIMALS,
        DATE(AXFER_TXNS.ROUND_TIME) AS TX_DATE,
        AXFER_TXNS.RELATED_TX
FROM AXFER_TXNS
JOIN HOT_TOKENS ON AXFER_TXNS.ASSET_ID = HOT_TOKENS.ASSET_ID
```

Code A.8: Query for `txns_on_dexs` View.

```
CREATE VIEW AS TXNS_ON_DEXS
```

```
SELECT DISTINCT
  APP_TXNS.APP_ID,
       DEXS_DISTINCT.PLATFORM,
       DATE(APP_TXNS.ROUND_TIME) AS TX_DATE,
       APP_TXNS.RELATED_TX
FROM APP_TXNS
JOIN
       (SELECT DISTINCT DEXS.APP_ID,
                        DEXS.PLATFORM
              FROM DEXS) DEXS_DISTINCT
       ON APP_TXNS.APP_ID = DEXS_DISTINCT.APP_ID
```

B. AlgoSight UI

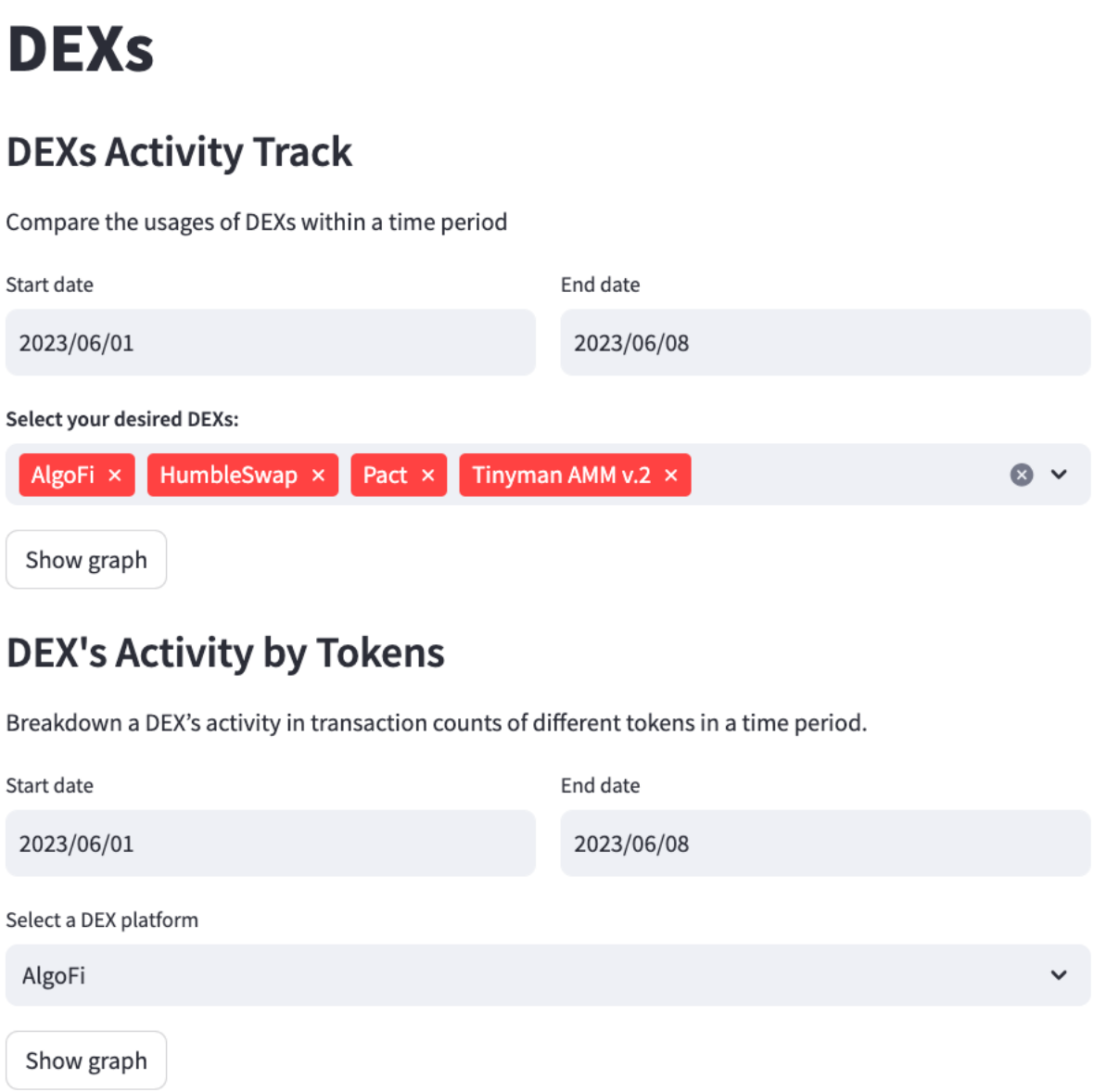

Figure B.2.: AlgoSight DEXs Page.

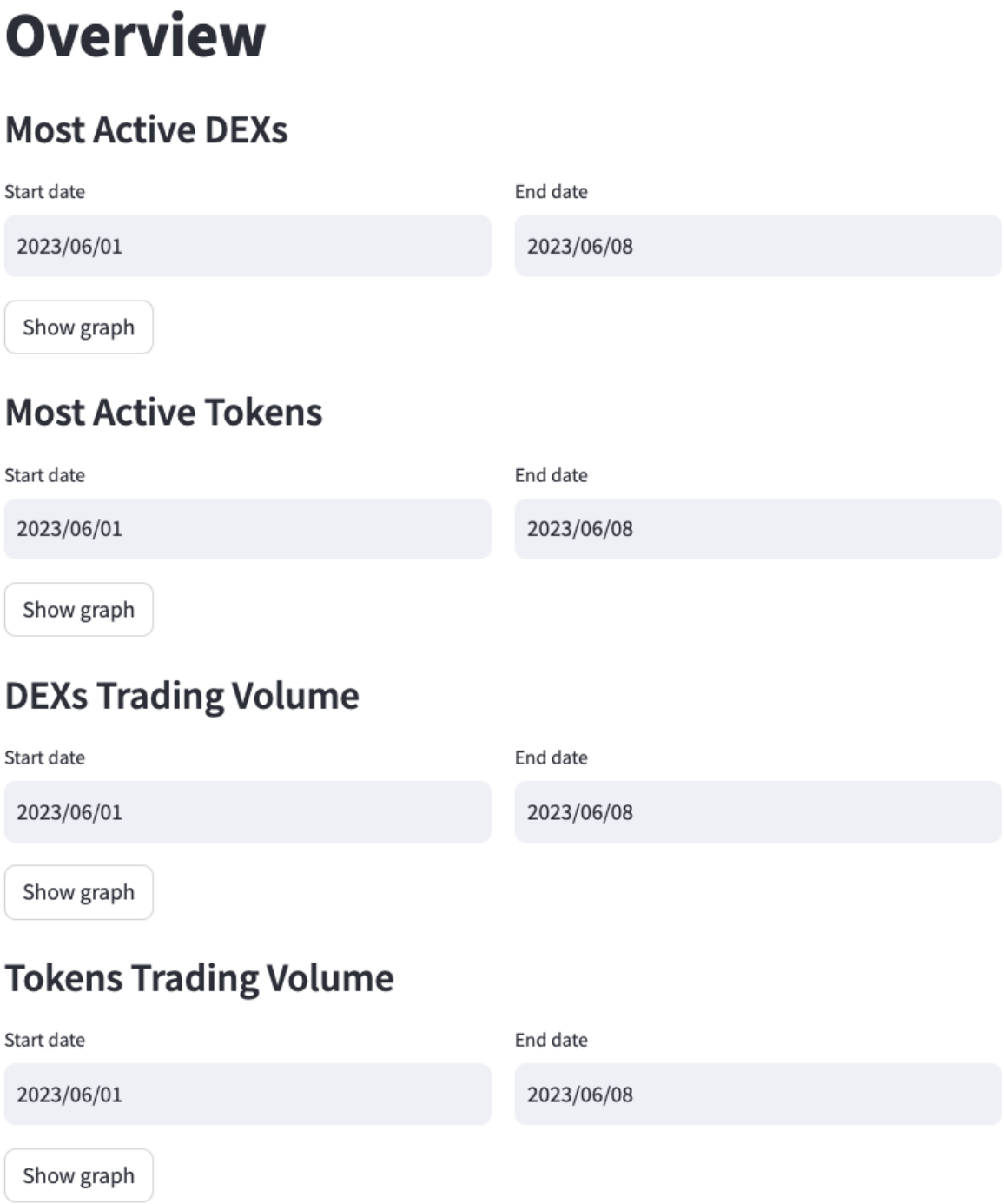

Figure B.1.: AlgoSight Overview Page.

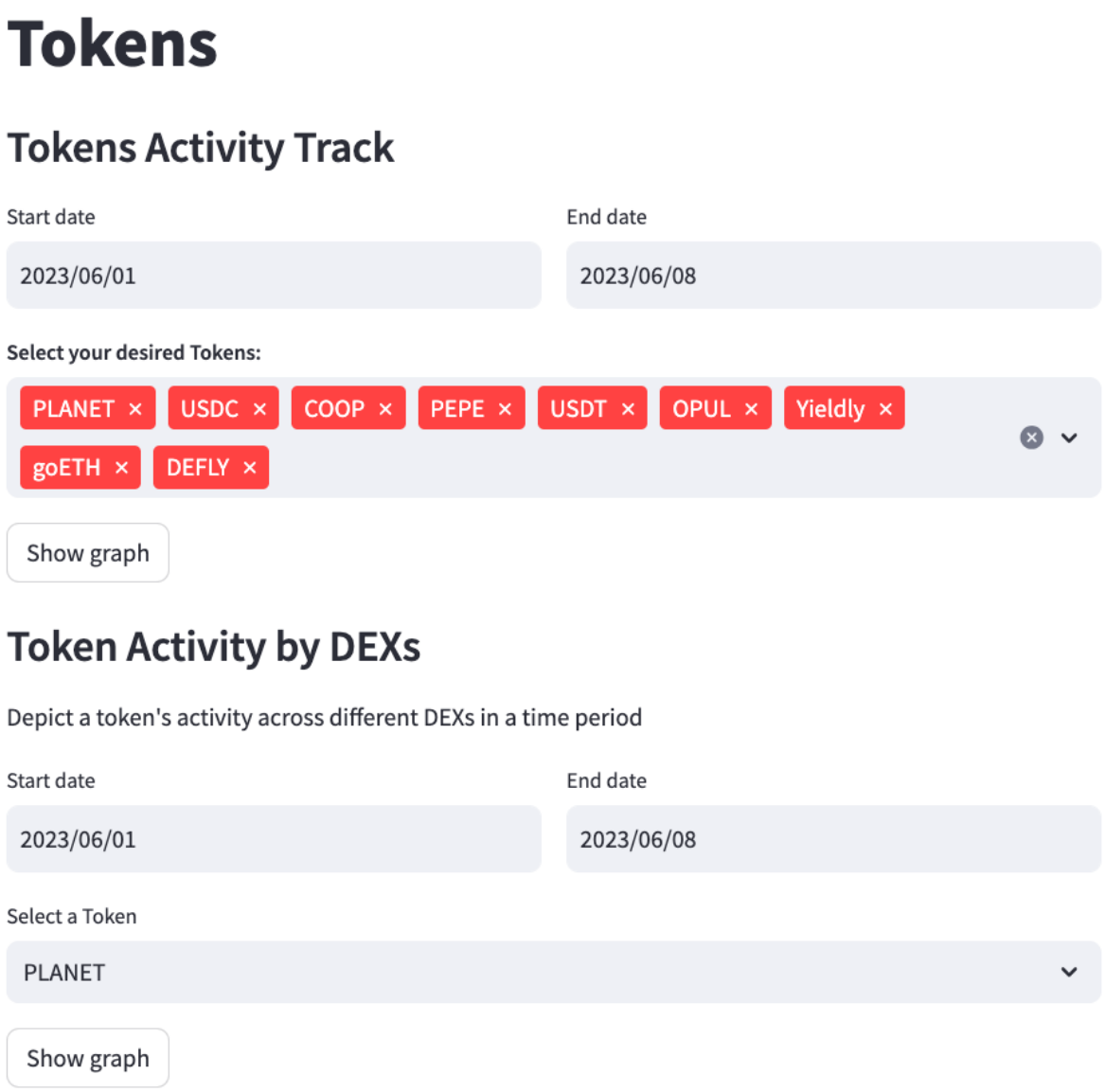

Figure B.3.: AlgoSight Tokens Page.

List of Figures

List of Tables

Bibliography

[1] S. Nakamoto. "Bitcoin: A peer-to-peer electronic cash system". In: *Decentralized business review* (2008). URL: https://assets.pubpub.org/d8wct41f/31611263538139.pdf.

[2] V. Buterin. "A next-generation smart contract and decentralized application platform". In: *Ethereum White Paper* 3.37 (2014), pp. 2–1. URL: https://ethereum.org/669c9e2e2027310b6b3cdce6e1c52962/Ethereum_Whitepaper_-_Buterin_2014.pdf.

[3] T. M. Team. "The Dai Stablecoin System". In: *MakerDAO White Paper* (2017). URL: https://makerdao.com/whitepaper/Dai-Whitepaper-Dec17-en.pdf.

[4] R. Leshner and G. Hayes. "Compound: The money market protocol". In: *Compound Finance White Paper* (2019). URL: https://compound.finance/documents/Compound.Whitepaper.pdf.

[5] F. Schär. "Decentralized finance: On blockchain-and smart contract-based financial markets". In: *FRB of St. Louis Review* (2021). DOI: 10.20955/r.103.153-74.

[6] Algorand. *The Borderless Economy Is Here*. 2019. URL: https://algorand.com/resources/blog/the-borderless-economy-is-here.

[7] reverseacid. *The Scalability Trilemma in Blockchain*. 2018. URL: https://steemit.com/blockchain/@reverseacid/the-scalability-trilemma.

[8] Algorand. *Algorand Network Architecture Overview*. 2023. URL: https://algorand.com/technology/algorand-network-architecture.

[9] Algorand. *Algorand Protocol Overview*. 2023. URL: https://algorand.com/technology/protocol-overview.

[10] Algorand. *Algorand Consensus*. 2023. URL: https://developer.algorand.org/docs/get-details/algorand_consensus/.

[11] *Participate in Algorand Governance*. URL: https://governance.algorand.foundation.

[12] Algorand. *Algorand's Smart Contract Architecture*. 2023. URL: https://algorand.com/resources/blog/algorand-smart-contract-architecture.

[13] T. Walker, E. Nikbakht, and M. Kooli. *The Fintech Disruption: How Financial Innovation Is Transforming the Banking Industry*. Palgrave Macmillan Cham, 2023. DOI: 10.1007/978-3-031-23069-1.

[14] A. Demirgüç-Kunt, L. Klapper, D. Singer, and S. Ansar. *The Global Findex Database 2021: Financial inclusion, digital payments, and resilience in the age of COVID-19*. World Bank Publications, 2022. URL: https://www.worldbank.org/en/publication/globalfindex.

[15] D. Balaban. *The Dark Side Of DeFi: Fraud, Hacks, And Regulatory Uncertainty*. 2023. URL: https://www.forbes.com/sites/davidbalaban/2023/05/11/the-dark-side-of-defi-fraud-hacks-and-regulatory-uncertainty/.

[16] K. Qin, L. Zhou, Y. Afonin, L. Lazzaretti, and A. Gervais. "CeFi vs. DeFi–Comparing Centralized to Decentralized Finance". In: *arXiv preprint arXiv:2106.08157* (2021). DOI: 10.48550/arXiv.2106.08157.

[17] L. Gudgeon, D. Perez, D. Harz, B. Livshits, and A. Gervais. "The decentralized financial crisis". In: *2020 crypto valley conference on blockchain technology (CVCBT)*. IEEE. 2020, pp. 1–15. DOI: 10.1109/CVCBT50464.2020.00005.

[18] S. Werner, D. Perez, L. Gudgeon, A. Klages-Mundt, D. Harz, and W. Knottenbelt. "Sok: Decentralized finance (defi)". In: *Proceedings of the 4th ACM Conference on Advances in Financial Technologies*. 2022, pp. 30–46. DOI: 10.1145/3558535.3559780.

[19] M. Pourpouneh, K. Nielsen, and O. Ross. *Automated market makers*. Tech. rep. IFRO Working Paper, 2020. URL: http://hdl.handle.net/10419/222424.

[20] L. Zhou, K. Qin, C. F. Torres, D. V. Le, and A. Gervais. "High-frequency trading on decentralized on-chain exchanges". In: *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2021, pp. 428–445. DOI: 10.1109/SP40001.2021.00027.

[21] E. E. Community. *Off-chain Order Book*. URL: https://unlock-protocol.github.io/ethhub/built-on-ethereum/decentralized-exchanges/off-chain-orderbook/what-is-off-chain-orderbook/.

[22] W. Warren and A. Bandeali. "0x: An open protocol for decentralized exchange on the Ethereum blockchain". In: *0xProject White Paper* (2017). URL: https://github.com/0xProject/whitepaper/blob/master/0x_white_paper.pdf.

[23] D. Wang, J. Zhou, A. Wang, and M. Finestone. "Loopring: A decentralized token exchange protocol". In: *Loopring White Paper* (2018). URL: https://loopring.org/resources/en_whitepaper.pdf.

[24] A. Mamdani and A. Trefonas. "Algodex Whitepaper v1". In: (2022). URL: https://github.com/algodex/algodex-public-documents/blob/master/Algodex%20Whitepaper%201.0.pdf.

[25] Y. K. Chaturvedi. *What is an Automated Market Maker (AMM)?* 2023. URL: https://coincodecap.com/what-is-an-automated-market-maker-amm-2021#History.

[26] E. Hertzog, G. G. G. Benartzi, and G. G. G. Benartzi. "Bancor Protocol Continuous Liquidity for Cryptographic Tokens through their Smart Contracts". In: *Bancor Protocol White Paper* (2018). URL: https://storage.googleapis.com/website-bancor/2018/04/01ba8253-bancor_protocol_whitepaper_en.pdf.

[27] H. Adams. "Uniswap Whitepaper". In: (2018). URL: https://hackmd.io/C-DvwDSfSxuh-Gd4WKE_ig.

[28] V. Mohan. "Automated market makers and decentralized exchanges: A DeFi primer". In: *Financial Innovation* 8.1 (2022), p. 20. DOI: 10.1186/s40854-021-00314-5.

[29] J. Xu, K. Paruch, S. Cousaert, and Y. Feng. "Sok: Decentralized exchanges (dex) with automated market maker (amm) protocols". In: *ACM Computing Surveys* 55.11 (2023), pp. 1–50. DOI: 10.1145/3570639.

[30] hagaetc. *Market Share DEX by Volume*. 2023. URL: https://dune.com/queries/4319/8411.

[31] *HumbleSwap App*. URL: https://app.humble.sh/swap.

[32] Tinyman. *The Tinyman V2 docs*. 2023. URL: https://docs.tinyman.org/.

[33] Pact. *Pact's vision is to be the Home of Liquidity on Algorand*. 2023. URL: https://docs.pact.fi/pact/introduction/about.

[34] Alammex. *How does Alammex work?* 2022. URL: https://docs.alammex.com/how-does-alammex-work.

[35] S. Aramonte, S. Doerr, W. Huang, and A. Schrimpf. *DeFi lending: intermediation without information?* Tech. rep. Bank for International Settlements, 2022. URL: https://www.bis.org/publ/bisbull57.pdf.

[36] L. Zhou, K. Qin, A. Cully, B. Livshits, and A. Gervais. "On the just-in-time discovery of profit-generating transactions in defi protocols". In: *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2021, pp. 919–936. DOI: 10.1109/SP40001.2021.00113.

[37] Aave. *Flash Loans*. 2023. URL: https://docs.aave.com/developers/guides/flash-loans.

[38] D. Wang, S. Wu, Z. Lin, L. Wu, X. Yuan, Y. Zhou, H. Wang, and K. Ren. "Towards a first step to understand flash loan and its applications in defi ecosystem". In: *Proceedings of the Ninth International Workshop on Security in Blockchain and Cloud Computing*. 2021, pp. 23–28. DOI: 10.1145/3457977.3460301.

[39] M. Mita, K. Ito, S. Ohsawa, and H. Tanaka. "What is stablecoin?: A survey on price stabilization mechanisms for decentralized payment systems". In: *2019 8th International Congress on Advanced Applied Informatics (IIAI-AAI)*. IEEE. 2019, pp. 60–66. DOI: 10.1109/IIAI-AAI.2019.00023.

[40] H. Arslanian. *The Book of Crypto: The Complete Guide to Understanding Bitcoin, Cryptocurrencies and Digital Assets*. Springer Nature, 2022. DOI: 10.1007/978-3-030-97951-5.

[41] A. Klages-Mundt, D. Harz, L. Gudgeon, J.-Y. Liu, and A. Minca. "Stablecoins 2.0: Economic foundations and risk-based models". In: *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*. 2020, pp. 59–79. DOI: 10.1145/3419614.3423261.

[42] R. Rueda, D. McCabe, and R. Soscia. "The GARD protocol". In: (2022). URL: https://algogard.com/white-paper.pdf.

[43] Algogator.Finance. *Algogator.Finance Documentation*. 2023. URL: https://docs.algogator.finance/.

[44] Y. Monitor. *An overview of the Yield Monitor application. (Beta)*. 2023. URL: https://www.yieldmonitor.io/documentation/.

[45] Synthetix. *Synthetix Protocol*. 2023. URL: https://docs.synthetix.io/synthetix-protocol/readme.

[46] Thetanuts.Finance. *What is Thetanuts*. 2023. URL: https://docs.thetanuts.finance/.

[47] A. Foundation. *Algorand Foundation pioneers Defi 2.0 with Algo Option Vaults on ThetaNuts*. 2021. URL: https://www.algorand.foundation/news/algo-option-vaults-on-thetanuts#:~:text=Singapore%2C%20November%2029th%2C%202021%20,strategies%20through%20structured%20product%20vaults.

[48] S. Falkon. *The Story of the DAO — Its History and Consequences*. 2017. URL: https://medium.com/swlh/the-story-of-the-dao-its-history-and-consequences-71e6a8a551ee.

[49] C. Base. *Around the Block 3: Analysis on the bZx attack, DeFi vulnerabilities and the state of debit cards in crypto*. 2020. URL: https://www.coinbase.com/learn/market-updates/around-the-block-issue-3.

[50] S. Eskandari, M. Salehi, W. C. Gu, and J. Clark. "Sok: Oracles from the ground truth to market manipulation". In: *Proceedings of the 3rd ACM Conference on Advances in Financial Technologies*. 2021, pp. 127–141. DOI: 10.1145/3479722.3480994.

[51] S. Eskandari, S. Moosavi, and J. Clark. "Sok: Transparent dishonesty: front-running attacks on blockchain". In: *Financial Cryptography and Data Security: FC 2019 International Workshops, VOTING and WTSC, St. Kitts, St. Kitts and Nevis, February 18–22, 2019, Revised Selected Papers 23*. Springer. 2020, pp. 170–189. DOI: 10.1007/978-3-030-43725-1_13.

[52] B. Öz, J. Gebele, F. Rezabek, F. Hoops, and F. Matthes. "A First Study of MEV on an Up-and-Coming Blockchain: Algorand". In: *arXiv preprint arXiv:2308.06513* (2023). DOI: 10.48550/arXiv.2308.06513.

[53] F. Finance. *Unleash the power of DeFi*. 2023. URL: https://folks.finance/.

[54] algonaut. *AlgoDex: Algorand's Order Book Exchange and Decentralized Marketplace*. 2022. URL: https://algonaut.space/algodex/#:~:text=February%2016%2C%202022%20What%20is,fungible%20tokens%20%28NFTs.

[55] Tinyman. *A basic overview of the Tinyman AMM*. 2022. URL: https://docs.tinyman.org/tinyman-v1/tinyman-amm-basics#overview.

[56] A. G. Club. *The Tinyman Definitive Guide*. 2022. URL: https://algogovclub.substack.com/p/tinyman-definitive-guide#:~:text=Tinyman%20Overview%20Tinyman%20is%20a,market%20advantage.

[57] C. Bassi, J. Paulos, P. Grassano, M. Almanza, S. De Angelis, and C. Kim. *Algorand Virtual Machine (AVM) Architecture*. 2022. URL: https://github.com/cusma/algorand-school.

[58] L. Zhou, X. Xiong, J. Ernstberger, S. Chaliasos, Z. Wang, Y. Wang, K. Qin, R. Wattenhofer, D. Song, and A. Gervais. "Sok: Decentralized finance (defi) attacks". In: *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2023, pp. 2444–2461. DOI: 10.1109/SP46215.2023.10179435.

[59] P. Daian, S. Goldfeder, T. Kell, Y. Li, X. Zhao, I. Bentov, L. Breidenbach, and A. Juels. "Flash boys 2.0: Frontrunning in decentralized exchanges, miner extractable value, and consensus instability". In: *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2020, pp. 910–927. DOI: 10.1109/SP40000.2020.00040.

[60] B. Weintraub, C. F. Torres, C. Nita-Rotaru, and R. State. "A flash (bot) in the pan: measuring maximal extractable value in private pools". In: *Proceedings of the 22nd ACM Internet Measurement Conference*. 2022, pp. 458–471. DOI: 10.1145/3517745.3561448.

[61] P. Xia, H. Wang, B. Gao, W. Su, Z. Yu, X. Luo, C. Zhang, X. Xiao, and G. Xu. "Trade or trick? detecting and characterizing scam tokens on uniswap decentralized exchange". In: *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 5.3 (2021), pp. 1–26. DOI: 10.1145/3491051.

[62] A. Raj, J. Bosch, H. H. Olsson, and T. J. Wang. "Modelling Data Pipelines". In: *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. 2020, pp. 13–20. DOI: 10.1109/SEAA51224.2020.00014.