The use of Real-World Assets in MakerDAO - A Decentralized Autonomous Organization on Ethereum

Aakash Ranglani,

A Dissertation

Presented to the University of Dublin, Trinity College

in partial fulfilment of the requirements for the degree of

Master of Science in Computer Science (Future Networked Systems)

Supervisor: Professor Donal O'Mahony

August 2022

Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

Aakash Ranglani

August 19, 2022

Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this thesis upon request.

Aakash Ranglani

August 19, 2022

The use of Real-World Assets in MakerDAO - A Decentralized Autonomous Organization on Ethereum

Aakash Ranglani, Master of Science in Computer Science

University of Dublin, Trinity College, 2022

Supervisor: Professor Donal O'Mahony

In the world of traditional finance, individuals or organizations can borrow money from financial institutions such as banks and in return, they have to deposit something of value that acts as a security deposit against the loan amount. Although such systems have existed for a long time, not everyone can access these financial services as they often require credit checks and background verification processes that every individual or a small-to-medium enterprise may not pass.

With the emergence of Blockchain technologies such as Ethereum, which forms a secure peer-to-peer network of nodes, it is now possible for any individual to access similar financial services, thus eliminating the need for any middlemen. Smart contracts are the backbone technology that enables these financial services which are decentralized in nature, unlike traditional financial services that are controlled and managed by a closed group of people.

MakerDAO is a peer-to-peer organization built on the Ethereum network that enables anyone to lend and borrow money in the form of cryptocurrencies. The organizations built on this peer-to-peer network are termed Decentralized Autonomous Organizations. Smart contracts govern the loan and borrowing process. The loan is offered based on an over-collateralized scheme where the borrower has to deposit an asset, the value of which is almost double the amount of amount being borrowed to secure a loan.

MakerDAO previously supported only Ethereum-based cryptocurrencies as a security deposit to take out a loan in the form of Dai. Dai is a cryptocurrency having a value equivalent to one US Dollar. Their efforts are now towards integrating real-world assets such as real estate, credit invoices, and luxury cars into the MakerDAO ecosystem that act as collateral against the loan amount being borrowed. In this work, we will explore the possibility of using equities as collateral in the MakerDAO ecosystem, against which any entity can borrow Dai.

Acknowledgments

Firstly, I would like to thank my supervisor and mentor, Professor Donal O'Mahony. Through his assistance, knowledge, and support, I was able to gain invaluable insights in the field of Blockchain which I've shared through this dissertation.

I would also like to thank all the professors for sharing their knowledge and expertise, also for structuring this Master's course as a practical learning experience.

I owe a tremendous deal of appreciation to my parents as well as everyone else who inspired and motivated me throughout the Master's Degree.

Aakash Ranglani

University of Dublin, Trinity College
August 2022

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Banks accept cash deposits, make interest-bearing loans, and return some of the interest to depositors. That is a perfect example of a bank putting their and your money to work. Typically, they'll repeat this profit-generating process over and over, with you getting a small percentage of it in the form of interest on your savings accounts. For the previous ten years, however, that has been close to nothing: the average interest rate in the United States is just 0.09%.[1]

Third parties who enable money flow between parties in the financial system, with each demanding a charge for their services. Consider the case of using a credit card to purchase a product. The charge is sent from the merchant to an acquiring bank, which then passes the card information to the credit card network. The network clears the charge and asks your bank for payment. Your bank approves the charge and forwards it to the network, who then forwards it to the merchant via the acquiring bank. Because retailers must pay for your ability to use credit and debit cards, each organization in the chain receives paid for its services. All other financial transactions are expensive; loan applications can take days to be accepted; and you may be unable to use a bank's services while abroad.

Traditional finance can be unfair for the end users in many ways, considering the high transaction charges, slow processing of money transfers across borders, and even worse, since it is centralized and owned by a closed group, they could deny services or change policies at their will. We can eliminate these problems with the using of a technology called as Blockchain, which is a peer-to-peer network of independent entities who can join and leave the network at their free will, hence decentralized in nature.

With the use of peer-to-peer financial networks that employ security protocols, network connectivity, software applications and hardware, it is possible to eliminate these middlemen by enabling individuals, merchants, and businesses to execute financial transactions using an emerging technology called Blockchain, and because of the peer-to-peer nature of the applications, they are called Decentralized Finance or DeFi Applications. MakerDAO is a DeFi service that is run on Blockchain as a lending credit system that provides loans at predetermined interest rates that are higher than those in traditional finance, and it also eliminates any uncertainty because it runs entirely on code, i.e Smart Contracts.

1.1 Motivation

In the traditional world of finance, an individual can take out a loan against assets like real estate, luxury cars, credit invoices that act as collateral, or security deposit against the loan being borrowed. Often, these financial services are available only to a few already wealthy individuals or organizations, after the necessary credit checks have been approved by the financial institution. MakerDAO has successfully eliminated the dependency on these financial institutions in order to secure a loan but it has done so mainly for virtual assets like cryptocurrencies. It is moving towards integrating real-world assets, such as real estate, onto the peer-to-peer decentralized finance network to allow any individual to be able to borrow a loan against supported collateral and this would be done without any background verification or credit checks. This would empower small-to-medium Enterprises greatly as well as individuals.

In order to extend MakerDAO's support to provide loans backed by real-world assets, this work explores the possibility of using Equities as an asset against which a user could borrow money. When an equity portfolio is pledged as security, MakerDAO will place the highest value on publicly traded stocks of the company since they are simpler to sell in the event that the borrower is unable to pay back the loan. Until the borrower repays the loan, lenders continue to have authority over the shares. The shares would then be returned to the borrower since they would no longer be required as collateral. This form of borrowing is called as Equity Portfolio loan stock financing.

In this work we will explore the technologies and concepts required to build a prototype

of the use-case of borrowing a loan against Equity Portfolios as Collaterals. We will cover Blockchain, which is a peer-to-peer decentralized network, Ethereum Blockchain that eliminates the dependency on centralized entities through the use of Smart Contracts, and we will be covering multiple other technologies in-depth which will lead to a better understanding of the implementation of the prototype in further sections.

1.2 Structure of the Dissertation

- Chapter 2 - This will consist of the state of the art that enables MakerDAO to function the way it does and to explore the technologies that allow MakerDAO to function with Real-World Assets. This part will cover Blockchain, Ethereum, Decentralized Autonomous Organizations (DAOs), Oracles, ERC20 Tokens, Smart Contracts, and Decentralized Applications. This part will also cover how MakerDAO functions and its components that are relevant to implementing a Real-World Assets as a collateral.
- Chapter 3 - This chapter will explore the design decisions taken while architecting the distributed application prototype tokenizing equity portfolio and use it as a collateral.
- Chapter 4 - This chapter will cover the implementation aspects around technologies and methodologies used for the prototype.
- Chapter 5 - This chapter will evaluate the prototype and assess the newly presented idea.
- Chapter 6 - This last chapter consists of the conclusion and future work.

Chapter 2

State of the Art

2.1 Blockchain

A Blockchain is a shared network run by individuals or organizations which is not centrally owned by any particular organization, thus making it a decentralized network of independent nodes. A "Block" is a data structure that holds information about transactions and each block is cryptographically linked to the previous block, thus maintaining a chain of blocks which is called as Blockchain. This chaining ensures that the information in a block cannot be changed without altering the subsequent blocks of data, which guarantees a tamper-proof data structure. This data structure acts as a ledger of records or blocks. Blockchains are decentralized peer-to-peer networks, immutable, transparent, and they often have a network currency called cryptocurrency.

Back in 2008, the need for financial institutions, or intermediaries, was the key problem that the blockchain attempted to solve. It is for this reason that Satoshi Nakamoto, a pseudo-anonymous identity of the person behind Bitcoin and Blockchain, proposed the notion of a decentralized, peer-to-peer system. With that method, payments would be made more quickly, securely, and with lower rates. The elimination of the middleman was the fundamental objective. Thus, Blockchain is a philosophy as well as a technology. [2]

Blockchains have the following properties because of the way they're designed:

- Decentralized - Decentralization in the context of blockchain describes the transfer of power and decision-making from a centralized entity (an individual, an organization,

or a collection of such entities) to a decentralized network. Decentralized networks aim to limit the amount of trust that participants must place in one another and to prevent them from interfering with one another in ways that would impair the network's performance.

- Transparency & Flexibility - Since the identity on Blockchains are pseudonymous, the transactions happening on chain are publicly verifiable. All the transactions that happen on a Blockchain are visible publicly but the identities of the individuals are not revealed because of the use of cryptographic keys as public addresses.
- Speed & Efficiency - Traditional financial institutions take a day or two to process international transactions across borders, and they often have a very high fee as well to process such transactions. Blockchain-based transactions are comparatively much cheaper and faster as they go through within a few minutes typically.
- Security & Immutability - Hacking a blockchain is a lot more difficult than hacking a centralized entity as there are thousands of blockchain nodes carrying the same copy of blocks, it is fully duplicated and it would be nearly-impossible to hack all the nodes and tamper the block data.
- Removal of counterparty risk - Blockchains, especially smart contracts, make it possible to operate in a trustless environment that is purely driven by code. For example, an insurance company may not approve a claim even though the conditions match the written contract, but if the same rules were coded in a smart contract, the claim would pass if the conditions are met.

These properties of Blockchain make it possible to have a system that everyone has equal access to, a system that is not centrally controlled and cannot be modified by a closed group of people to their benefit. Thus, enabling our use-case of being able to borrow money without relying on any centralized financial institution like banks.

2.1.1 Elements of a Blockchain Network

It is important to understand the elements of a Blockchain network to realize the potential and the impact of the applications being built on the Blockchain Network. It is because of these underlying technical components, we can operate in a trustless environment. Without an understanding of this, the financial applications running on the Blockchain would seem like any other web application.

Blockchain Structure

The structure of the Blockchain is what guarantees Immutability. Along with the transaction data, each block in a blockchain also has the hash of the block before it and also of the block itself. Depending on the type of blockchain, a block may include different types of data. For instance, transaction information such as sender, receiver, and coin count is stored on the Bitcoin blockchain. Additionally, each block contains a hash, which you might think of as a fingerprint. A hash, like a fingerprint, uniquely identifies a block and all of its contents. A block's hash is determined after it has been produced. The hash will change if anything inside the block is modified. In other words, hashes are particularly helpful for tracking block changes. A block is no longer the same block if its hash changes. Lastly, the hash of the preceding block is present in each block. This effectively establishes a chain of blocks, and it is this method that contributes to the security of a blockchain. [3]

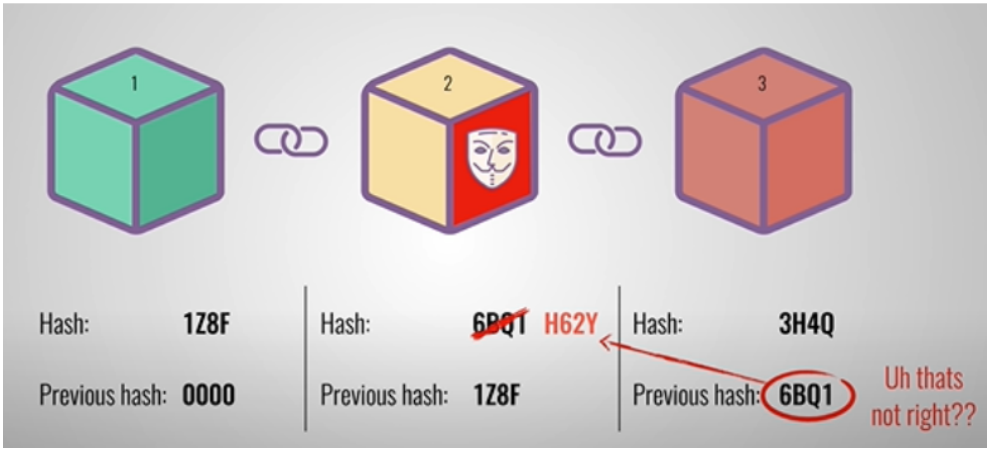

Figure 2.1: Blockchain Structure
Image Source: Designing a Blockchain Architecture (Medium Article)

The Figure 2.1 shows a conceptual structure of a blockchain. We can observe that the first block's hash is stored in the second block and the second block's hash is stored in the third block. But let's assume that a malicious actor manages to change the content of the second block, the hash of the second block changes, and thus the third block's previous hash does not match the hash of the second block. Thus the malicious node would have to re-create all blocks ahead of the second block. This is quite doable with modern computers but blockchain's proof-of-work mechanism ensures that it is not easy to mine a block. This mechanism hinders the generation of additional blocks. In the instance of Bitcoin, calculat-

ing the necessary proof of work and adding a new block to the chain both take roughly ten minutes. By requiring you to recalculate the proof of work for every subsequent block, this approach makes it difficult to manipulate the blocks.

A blockchain's security stems from its innovative hashing and proof of work mechanisms. Blockchains also secure themselves by being decentralized and distributed. A P2P network that anybody is free to join is used by a blockchain to govern the chain rather than a single company (assuming the blockchain is public). One becomes a node and receives a complete copy of the blockchain when they join this blockchain network. The copy of the blockchain can then be used by this node to confirm that everything is in order.

With this understanding, a user can build trust in the technology, to know that their data and transactions would remain secure, instead of relying on a centralized entity's infrastructure.

2.1.2 Cryptographic Hash

In the previous section we've covered the structural aspects of the Blockchain and observed that hashing is an important aspect of the Blockchain, hence in this section we will explore it in more depth and also cover its properties.

Hashing is mainly a way to produce fixed-sized data using a hash algorithm, and the generated code is represented using a string of characters that acts as a fingerprint of that data. Hashing is similar to fingerprints in the sense that even though the fingerprints are small, they contain a lot of important information. One of the examples of a hash function is the 256-bit Secure Hashing Algorithm, also known as SHA-256, it is quite common and also used in Blockchains. As we can observe from Figure 2.2, any length of input results in the same 64-character hash value, also called as a hash digest. The hash functions have a property called the avalanche effect that causes the hash value to change completely even if the data is modified with the smallest of changes like changing the letter capitalization of one letter.

A cryptographic hash function is used for security purposes and it differs from the non-cryptographic one in a number of ways. Although they're much slower than non-cryptographic hash functions, they're much more secure and difficult to break. Cryptographic hash functions need to have the following properties:

Message	Hash of the message
Hi	3639EFCD08ABB273B1619E82E78C29A7DF02C1051B1820E99FC395DCAA3326B8
Welcome to TechJury	B7FFC27E0A6F559E1F63A927B6132A42124CC790F9EFD0FCFF4C3B653AA0DF4B
(The content of this entire article)	9247D0E6C7A2A45976DCC3D2393A52658B6DDFC846d8AF5743400EA12FB9DE74

Figure 2.2: Hashing Example
Image Source: TechJury - Blog

- Speed - The hash of the data should be computed within a fraction of a second using the hash function.
- Avalanche Effect - The smallest change in the data should result in a major change in the hash digest or the hash value.
- Deterministic - For the same input, the hash function should always result in the same output.
- Pre-image Resistance - You cannot get the input data from the hash value. It has to be a one-way function.
- Collision Resistance - Two different messages should not be able to produce the same hash value. [4]

Blockchain Nodes

Blockchain Nodes are what make up the Blockchain Network, to put it simply, Blockchain is a network of independent nodes. A blockchain node is a computer that runs software that is responsible for verifying all the transactions in each block, thus maintaining the accuracy of the data and keeping the network secure. They validate the blocks that are generated by the miners. When a block is added to the blockchain, all the nodes receive an update to their ledger. Following are the types of nodes in a blockchain network:

- Full Node - Stores the entire blockchain data, although it is periodically pruned so the node does not have all the data ranging back to the genesis block. This node is responsible for verifying all the blocks and state of the blockchain, it also validates the block. These nodes also serve network requests and data.

- Light Node - Light nodes download block headers rather than the entire block data. These headers merely provide an overview of the contents of the blocks. Any additional data that the light node needs is obtained from a full node. Users can join in the Ethereum network using light nodes without needing the robust hardware or high bandwidth needed to run full nodes. The light nodes can access the blockchain with the same capability as a full node, but they do not take part in consensus (they cannot be miners or validators).

- Archive Node - These nodes generate a historical state archive and stores everything stored in the full node. Even though this data takes up terabytes of storage, applications like block explorers, wallet providers, and chain analytics may find archive nodes useful.

Blockchain Transactions

Having covered the types of nodes in a Blockchain network, we will now explore how transactions like sending money from one account to another affect the Blockchain. Transactions are instructions from accounts that have been cryptographically signed. The Blockchain network's state gets updated when a transaction is started by an account. Transferring cryptocurrency between accounts is one of the most basic transactions you can do on a Blockchain, there are other types of transactions we will explore further. For example, Alice wants to send a unit of cryptocurrency to Bob, this transaction updates the state of the blockchain and all the nodes in the network are notified about this transaction and they keep a record of it. In the Figure 2.3 we can observe that a Block consists of a set of transactions T1, T2,

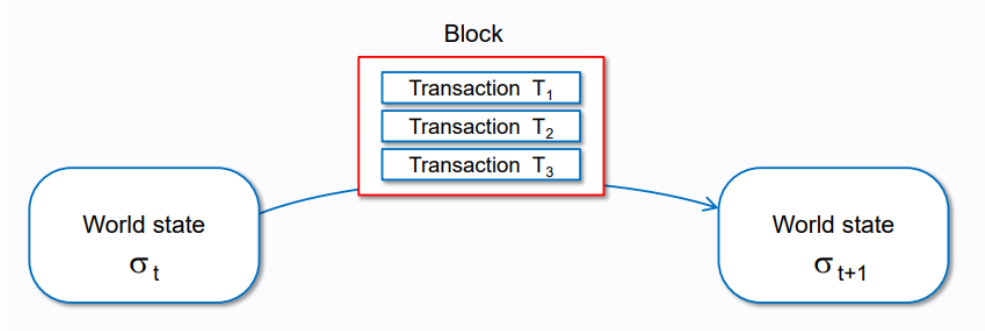

Figure 2.3: Blocks and Transactions
Image Source: Ethereum EVM illustrated

and T3, and once it is processed, all the nodes receive an update of this block and the state of the entire Blockchain network changes as all nodes have this new block. Compared to

the traditional finance applications, the transaction records are maintained privately on the company's infrastructure, in case of blockchain, every node will maintain a copy.

Blockchain Miner

In case of Bitcoin, the first cryptocurrency, miners are high-powered CPUs that solve a complex mathematical problem to win the next block of transactions and they're rewarded in Bitcoins to solve the complex problem. The process of bitcoin mining consists of verifying transactions and it results in minting new Bitcoins. Miners, in case of Bitcoin, typically take about 10 minutes to figure out the solution to that problem. This process of solving a complex mathematical problem is called as proof-of-work, explained in detail in the next section, which involves finding a value called as "nonce" that will fit into the mathematical puzzle. Once the problem is solved, the miners work on the next block, and once the block is verified, it is stored in the nodes permanently, and becomes immutable. By leveraging computing resources, you "prove" that you have completed the "job" by solving the puzzle. Adding a block successfully is rewarded in the network's cryptocurrency, for example Bitcoin or Ether, although mining is often a process of brute force trial and error. The network's nodes check and verify new blocks once they are broadcast to them, updating everyone's knowledge of the blockchain's current state. [5]

Blockchain Consensus

Consensus algorithms are crucial for confirming the legitimacy of distributed blockchain platforms and are the method of establishing consensus among a group of individuals operating in a trustless environment. All nodes in the network must reach consensus in order for the blockchain's state to advance and continue to grow. This is how a decentralized network's nodes are able to maintain synchronization with one another. There is no way to guarantee that the state that one node considers to be true will be shared by the other nodes in a blockchain without consensus for the decentralized network of nodes. While participants each have their own subjective perceptions of the network, consensus seeks to present the objective perspective of the state. It is the method through which these nodes converse, reach consensus, and are able to create new blocks.

Proof-of-Work

The decentralized blockchain network reaches consensus, or agreement on things like account balances and the chronological order of transactions, because of the proof-of-work process. This stops users from "double spending" their currencies and makes it extremely impossible to attack or manipulate the blockchain. It controls the rules and complexity of the job that miners must complete. The "work" itself is mining. It involves adding legitimate blocks to the chain. This is significant because the length of the chain enables the network to understand blockchain's present state and follow the right chain. The network can be more confident in the condition of the world as long as there is more "work" being done, a longer chain, and a higher block number. The longest chain is more likely to be the correct one because it has undergone the most computational work.

The nonce is a value for a block that must be discovered by miners through a rigorous process of trial and error under the proof-of-work protocol. A block can only be added to the chain if it has a valid nonce. It is nearly impossible to add new blocks to a Proof-of-Work network that delete transactions, add fictitious ones, or keep up a second chain. This is due to the fact that a malevolent miner would have to consistently solve the block nonce faster than everyone else. Proof-of-work is also in charge of minting currency for the system and providing incentives for miners to work, in the form of Bitcoin, a cryptocurrency.

Compared to the traditional centralized services that have their own infrastructure for performing computations and providing security, mining can be thought of as an activity achieving the same but by a network of independent nodes contributing their computation resources and they are rewarded for the same. For our application which we will explore in later sections, it is important to understand that a network of independent nodes are performing computations for which we pay the network as well, it is similar to traditional finance, except that we only have to pay the network in some form of currency for every transaction that we perform, not multiple intermediaries.

2.1.3 Oracles

The Blockchain Oracle Problem

Blockchains are unable to pull data from or push data out to any external systems as part of their built-in functionality, they cannot interact with the open internet through APIs. A blockchain's isolated network of independent nodes is precisely what makes it so secure and dependable because the network just needs to come to consensus on a very simple set of binary (true/false) questions using information that has already been written in the ledger. However, smart contracts need to be connected to the outside world in order to actualize their potential use cases. Trade finance contracts require trade documents and digital signatures to know when to release payments, financial smart contracts require market data to determine settlements, insurance smart contracts require IoT and web data to decide on policy payouts, and many smart contracts desire to settle in fiat currency on a conventional payment network. Neither of these traditional services nor the information they provide are directly available through the blockchain. An additional and distinct piece of infrastructure called as an Oracle is needed to connect the blockchain's on-chain and off-chain components. [6] Because they increase the range of possibilities for smart contracts to work, oracles are essential components of the blockchain ecosystem. Smart contracts wouldn't be particularly useful without blockchain oracles since they would only have access to data from their own networks. It's crucial to understand that a blockchain oracle is merely the layer that queries, authenticates, and checks external data sources before relaying that information.

Decentralized oracles are required to effectively combat the oracle problem in order to avoid data manipulation, inaccuracy, and downtime. A Decentralized Oracle Network, or DON for short, creates end-to-end decentralization by bringing together a number of independent oracle node operators and a number of trustworthy data sources.

The entire goal of a decentralized blockchain application is defeated by blockchain oracle techniques that use a centralized entity to transmit data to a smart contract. The smart contract won't have access to the data needed for execution if the sole oracle goes offline, or it will execute incorrectly based on outdated data. Even worse, if the sole oracle is corrupted, the data given on-chain may be seriously flawed, which might result in smart contracts executing with disastrous results. The "garbage in, garbage out" dilemma refers to this situation where poor inputs result in incorrect outcomes. Furthermore, since blockchain transactions

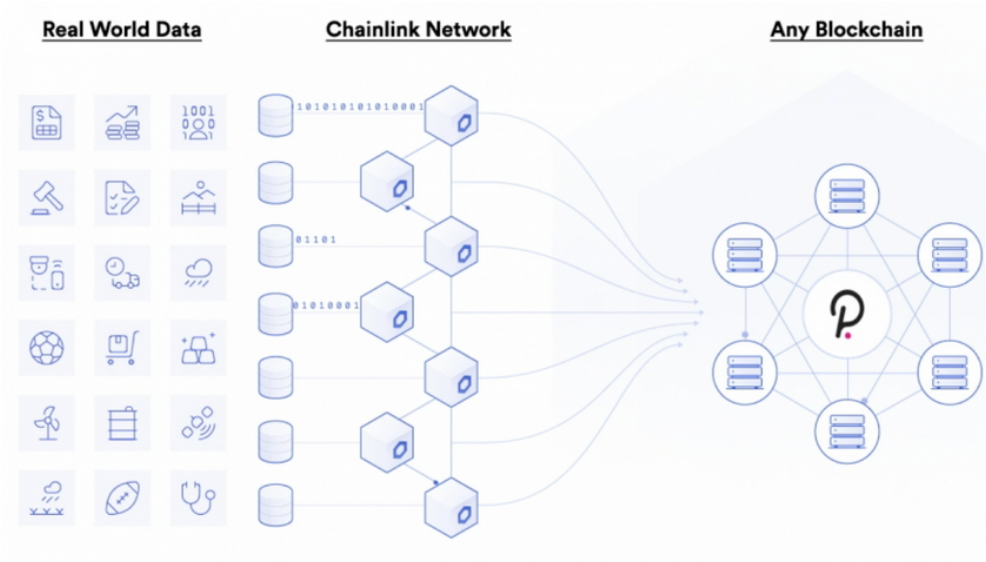

Figure 2.4: An example of Decentralized Oracle Network
Image Source: Blocklink Article

are automatic and unchangeable, a smart contract decision based on inaccurate data cannot be undone, which means user cash may be lost forever. As a result, applications for smart contracts cannot use centralized oracles. [7]

As we will explore in the sections ahead, for real-time market data on the pricing of equities, we will have to rely on Oracles. Also, the existing oracles of MakerDAO get information from a variety of separate feeds made up of people and organizations. A single asset and its reference price are associated with each oracle.

2.1.4 Public Key Cryptography - Elliptic Curve

Since there is no centralized entity with a secured infrastructure responsible for the safety of the user's account, it is important to understand how Blockchain handles security aspects from the user's perspective. In our prototype, we will be using the concepts of public key, private key, and addresses explained in this section, and these are applicable to any Blockchain application in general.

A security technique called Elliptic Curve public key cryptography makes sure that the data we exchange during a transaction on a blockchain network is secure. In a point-to-point

network like blockchain, the security component is essential because nodes in such a network do not know and trust each other personally. A pair of keys are used in public key cryptography, an asymmetric kind of encryption (public key and private key) that are employed to encrypt/decrypt data and validate users. It is a way of providing digital identity to the user using which one can make secure transactions on the blockchain network. These keys are generated using unique algorithms in public key cryptography that are unidirectional, meaning they first generate a private key, from it a public key, and then a public address. We are unable to compute the private key from a public key or a wallet address from a public key, i.e., we cannot reverse the sequence of generation. This data is all safely stored in Wallet which is a piece of software. It keeps track of a blockchain node's address, private key, public key, and transaction balance, among other crucial pieces of data.

This process guarantees two things:

- At the sender's end, the information is encrypted using the public key of the receiver so that nobody outside the network can access or decrypt the encrypted data. Using its own private key, only the intended recipient may decode and read the message.
- Utilizing the sender's private key to sign the data which verifies the sender's identity and establishes his legitimacy as a node in the blockchain network. The recipient confirms this by using the sender's public key. Digital signatures are used in this network user verification method.

Every user or node in the blockchain network has a private key, which is a bit string. A private key is similar to a password in that it can reveal our private information if it is shared. We must therefore keep our private key secret from the network. The private key is essentially stored in the digital wallets (software or hardware), as its security is crucial. The key is typically stored in a wallet import format, which has a key length of 51 characters. The length may vary based on the storage format.

A private key's counterpart, a public key, is derived cryptographically from it. Every node in the network has access to a public key. This facilitates a transaction's verification by all of the nodes in a blockchain network.

The digital signature is a crucial component of public-key cryptography in addition to the private key and public key. If a transaction is not digitally signed by the sender's private

key, it is not secure in a blockchain network. The information we transfer to other nodes is encrypted using public and private keys, or cryptography, which assures that no one in the middle can read or alter it. [8]

2.2 Ethereum Blockchain

Having covered the fundamentals of Blockchain that apply to the Ethereum network as well, we will now explore Ethereum in-depth as it is the platform on which our prototype will be built.

Ethereum is a blockchain that can be programmed. Ethereum, like any other blockchain, is built on a peer-to-peer network protocol that connects numerous independent computers throughout the world. Instead of limiting users to a few predefined activities (such as bitcoin transactions), Ethereum allows them to run pretty much any programming they desire. The code, which is referred to as smart contracts, is stored on the blockchain for anyone to engage with.

The blockchain is maintained and updated by the computers (nodes) in the Ethereum network. They also run the Ethereum Virtual Machine in Ethereum (EVM). The Ethereum Virtual Machine, much like the Java Virtual Machine, is responsible for executing the programmed code and it essentially creates a level of abstraction from the executing machine. On the blockchain, this computing power is used to run user-submitted code (smart contracts). In compensation for the processing power utilized by the smart contract, the EVM charges a very modest transaction fee to execute these. This cost is known as 'gas,' and it is paid in Ether, which is why Ether should not be viewed as a cryptocurrency, but rather as the fuel that keeps the network running. Transaction records are immutable, verifiable, and securely distributed over the network, providing participants with complete ownership and visibility over transaction data. User-created Ethereum accounts send and receive transactions.

The Ethereum Virtual Machine (EVM) is a fundamental component of the Ethereum Protocol and the Ethereum system's consensus engine. It enables anyone to run code in a trustless environment where the outcome of execution is guaranteed and completely predictable (i.e., smart contract execution). A system that tracks execution costs allocates a fee in Gas units

to each instruction executed on the EVM.

Similar to traditional financial services that levy multiple charges behind every transaction like transaction fees, the Ethereum network also has a charge associated with every transaction that the user must pay in the form of Gas, the fuel to keep the network running which we will explore in detail in the following section.

2.2.1 Ether & Gas

Ether (ETH) is the native cryptocurrency of Ethereum. The purpose of Ether is to incentivize computation on the Ethereum blockchain by providing an economic benefit for participants to verify and execute transaction requests and provide computational resources to the network. Any member who broadcasts a transaction request is also required to provide a fee to the network in the form of some ether, called Gas. The node which completes the process of verifying the transaction, carrying it out, committing it to the blockchain, and broadcasting it to the network will receive this fee.

The term "gas" refers to the metric used to express the amount of computational power necessary to carry out operations on the Ethereum network. Since each transaction in the Ethereum network relies on computational resources, there are charges associated with it, which is termed as Gas fees. The Gas fee is paid in the native currency of Ethereum which is Ether, and the prices are denoted in gwei, which is a denomination of Ether. Each gwei is equal to 0.000000001 ETH.

The Figure 2.5 shows how the amount of gas required for a particular operation is dependent on the computational complexity or storage requirements for the same. It can be observed that if the data of the smart contract has to be persisted on permanent storage, then the gas required would be much higher. On the other hands, if the data is used only in the memory of the EVM, then the gas required is much less. Every computational operation requires a some amount of gas.

The amount of ether required to perform a computation is directly proportional to the complexity of the computation. This prevents the network from being clogged by malicious actors who might request for infinite computation, but for their request to go through, they would have to pay a fee equivalent to the high computation requested. [5]

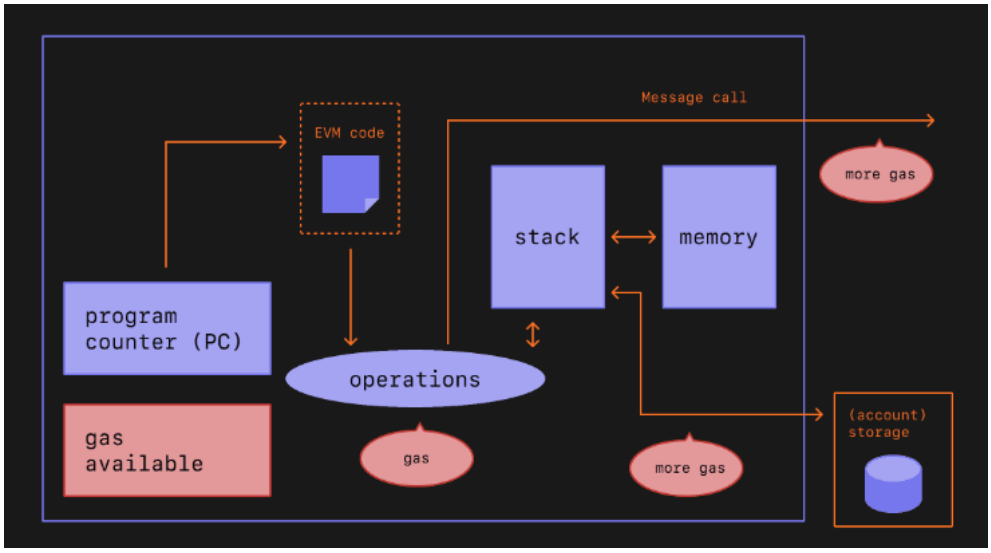

Figure 2.5: Gas fees in Ethereum
Image Source: Ethereum EVM illustrated

2.2.2 Smart Contracts

In this section we will cover Smart Contracts in detail as they are the critical to understand as the applications that run on the Ethereum network are programmed using these Smart Contracts. In our prototype, which we will look at in a later section, we have written multiple smart contracts that are responsible for providing the desired functionality of borrowing money against equities.

A smart contract is a program consisting of code, functions, and data, and it is assigned a specific address on the blockchain when it is deployed. They are often written using Solidity programming language. Smart contracts themselves are a type of Ethereum account and they have their own balance and can even initiate transactions on the network. Once deployed, they can be invoked by anyone as per the programmed functionality. Users can interact with these deployed smart contracts, they can initiate transactions, send ether, or even execute functions written inside the smart contract. It is not possible to delete smart contracts and transactions done on those are irreversible. [9] A computer software that operates on a blockchain network verifies, executes, and enforces smart contracts. The program will start running after the smart contract's terms have been accepted by all stakeholders. As the contract is confirmed and upheld by the blockchain network, a third party is no longer required. Smart contracts eliminate the potential of human mistake and can automate many

operations that would often require human interaction because they are executed by code rather than people. Similar to a typical contract, smart contracts outline the terms and conditions of an agreement and also automatically uphold those responsibilities.

One of the limitations of smart contracts is that they cannot interact with the outside world, the world outside blockchain is unavailable to the smart contracts because they cannot make HTTP requests. Since external information could be unreliable in terms of consensus, the design of the smart contract does not permit any outside interactions, which is essential to maintain it's decentralized nature and security. Although, in further sections, we will explore the concept of Oracles that can be utilized to obtain information from the internet.

Smart Contract Anatomy

- Data - All of the contract data either resides in memory or in storage. Saving data in storage is an expensive operation, hence we must clearly evaluate and store only that data which is absolutely needed.
- Storage - State variables are persistent data, which is referred to as storage. On the blockchain, these values are permanently recorded. Declaring the type is necessary so that the contract can monitor how much blockchain storage it requires during compilation.
- Memory - Memory variables are defined as values that are only kept on hand during the execution of a contract function. These are far more affordable to use because they are not permanently kept on the blockchain.
- Environment Variables - These are special global variables that provide information about the blockchain or the current transaction. For example, 'block.timestamp' holds the current block's epoch timestamp, and 'msg.sender' holds the address of the sender of the message (current transaction)
- Functions - These are either used to perform a set of operations or to set/get some information. Internal functions can be accessed from within the contract or its derived contracts, whereas external functions can be called from other contracts or via transactions. Internal functions do not create an EVM call but external functions do create one. Functions can either be public, which can be called from within the contract but

also externally, or they can be private, which would mean that they can only be called from within the contract in which they are defined.

- View Functions - These do not modify the state of the contract, and they can be thought of as "getter" functions which simply access information that resides in the smart contract.

- Constructor functions - These functions are called upon contract creation and they are called only once when the contract is first deployed. These can be used to initialize state variables.

Having covered the programmable aspects of a Smart Contract, it is important to understand how the Ethereum network handles these smart contracts and how our application interacts with them. The next section will cover Smart Contract Bytecode and ABI, which are crucial components of applications being built on the Ethereum Blockchain.

2.2.3 Bytecode & ABI

Since the Ethereum network relies on the EVM (Ethereum Virtual Machine), smart contract code written in high-level languages must be converted into EVM bytecode in order to be executed. The data that our Solidity code is "converted" into is bytecode. It includes binary instructions for the computer. Numeric codes, constants, and other types of information are typically stored in bytecode. Each step of instruction is an operation, or "opcode," which is typically one byte (eight bits) long. They are known as "bytecode" because they are one-byte opcodes. To ensure that the computer understands exactly what to do when running our code, every line of code is divided into opcodes. The bytecode is actually what is deployed on the Ethereum blockchain. [10]

A Contract ABI is an interface to communicate with bytecode deployed on the EVM. They specify the variables and methods that are available in a smart contract and that we can use to communicate with it. We need a way to know what operations and interactions we can initiate with smart contracts because they are converted into bytecode before they are deployed to the blockchain. We also need a standardized way to express those interfaces so that any programming language can be used to interact with smart contracts. For instance, ABI acts as a bridge between your JavaScript code and EVM bytecode so that they may communicate with one another when you want to invoke a function in a smart contract using

your JavaScript code. The architecture of the Contract ABI, EVM bytecode, and external components is depicted in the Figure 2.6. The right side is interacting with the deployed code, while the left side is compiling the code from Solidity to Bytecode. [11]

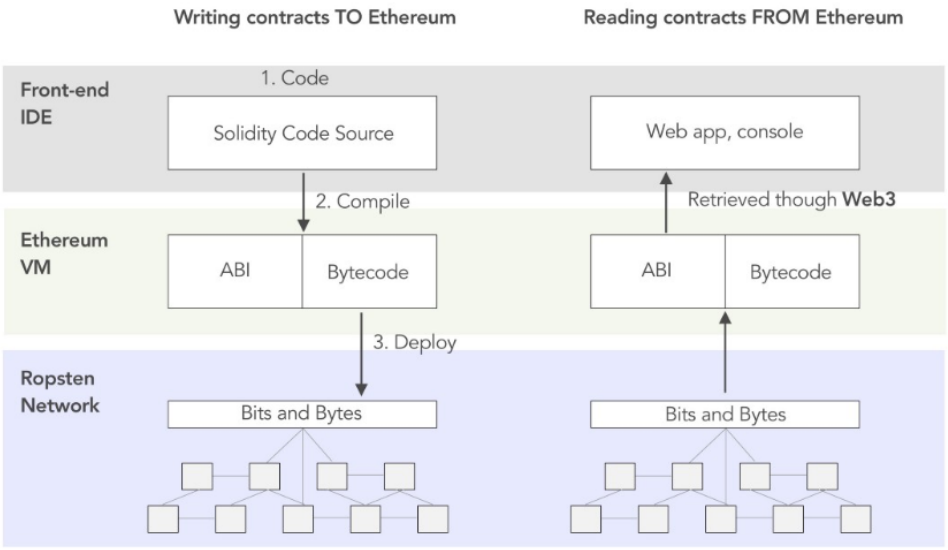

Figure 2.6: Smart Contract Compilation and Interaction
Image Source: Hackernoon

2.2.4 Smart Contract Deployment

In the previous section, we've explored what happens when we compile the smart contract. In this section we will look at the deployment process.

The smart contract needs to be deployed on the Ethereum network for distributed applications to be able to interact with it. To deploy a smart contract, the deployer needs to send a Ethereum transaction that will contain the compiled code of the smart contract and there would be no recipient mentioned for this transaction. Deploying a smart contract costs ether. Once the smart contract has been deployed, it will have an Ethereum address just like any Ethereum account and this address can be used to call contract functions or initiate transactions like sending Ether to the contract.

So far we've covered the building blocks of the Etheruem Blockchain covering the architecture behind running programmable smart contracts on it. In the following sections, we

Figure 2.7: Smart Contract Deployment
Image Source: Medium Article

will focus on the application-level design aspects with respect to the prototype being built as a part of this work.

2.2.5 ERC-20 Token Standard

ERC-20 standard establishes a common set of guidelines for Ethereum tokens to adhere to within the wider Ethereum ecosystem. Developers can program how new tokens will operate in this environment by using this standard. Additionally, it enables developers to foresee how different token interactions would behave. These regulations cover the exchange of tokens between addresses and the access to the data contained in each token. It can be thought of as an interface from an object-oriented point of view, where any class implementing this interface would have to provide implementations for all the functionalities mentioned in that interface. ERC-20 token is a fungible token which means that each token is identical to every other token in the set. This is comparable to how a US dollar is practically identical to every other dollar (at least in the digital realm). Each set of ERC-20 tokens is identified by a ticker symbol, such as DAI (MakerDAO's stable cryptocurrency) or LINK (Chainlink's currency).

Cryptocurrency tokens that carry out these features are known as ERC-20 tokens since they adhere to that specification. The following are the three categories into which these functions fall.

Getter Functions - functions used to retrieve information

- Total supply - this indicates the maximum number of tokens that can be minted
- Balance of - this gives the token balance owned by an account address
- Allowance - with the help of allowances, an account is able to use some tokens that belong to a separate owner. For instance, if address A grants address B a 50 token

allowance, address B is permitted to spend up to 50 tokens on address A's behalf.

Functions - perform an operation or action

- Transfer - exchanges tokens between sender and receiver
- Approve - the owner of the tokens uses this method to allow a spender to spend on behalf of the owners
- Transfer from - this mechanism allows the transfer of tokens from owner to recipient using the allowance functionality

Events - emitted when action occurs

- Transfer - emitted when the specified number of tokens are sent from the sender to the receiving address.
- Approval - emitted when the owner approves the spender to spend their tokens

In our prototype of borrowing a loan against Equities, we will be using this ERC-20 token, named Equity Token with the symbol "EQT", to represent the Equity Portfolio so that it can be integrated with the MakerDAO ecosystem as it supports the ERC-20 token standard. This will be explained in detail in section 2.4.5.

2.3 Decentralized Autonomous Organization (DAO)

A Decentralized Autonomous Organization is any group that is governed by a transparent set of rules founds on a blockchain or smart contract. All the decisions of the organizations are taken transparently in a decentralized manner that allows the users to verify and operate in a trustless environment. In fact, anyone can become a part of the organization at any point of time and participate in the governing and decision-making processes. DAOs are operated exclusively through code and have a completely flat hierarchy. They are a productive and secure method to collaborate with like-minded people all across the world. Imagine them as a group of individuals who collectively own and operate an internet-based company. In order to ensure that everyone in the organization has a voice, proposals and voting are used to make decisions. A DAO's smart contract is its structural foundation. The agreement holds the group's funds and outlines the organization's policies. Once the contract has gone live on Ethereum, only a vote will allow for changes to the terms. Anything that is attempted

that is not permitted by the logic and principles of the code will fail. Furthermore, since the smart contract also establishes the treasury, no one is permitted to utilize the funds without the consent of the group. DAOs are therefore not dependent on a centralized authority. Instead, decisions are made jointly by the group, and when votes are successful, payments are immediately issued. [12]

2.3.1 How do DAOs Work?

The following are the general steps a DAO needs go through to launch sustainably, even though the precise underlying mechanics powering a DAO vary across various blockchain initiatives.

- Smart Contract Setup - The underlying rules must be specified and encoded in a number of smart contracts before a DAO can be initialized and deployed. This stage is arguably the most crucial for building a sustainable and autonomous DAO, as any initial mistakes or overlooked details could potentially destabilize the project down the road. This is because future changes to the DAO's operational workflows, governance system, and incentive structures will need to be approved by the community through a voting mechanism in order to take effect, hence this stage also involves extensive testing of the smart contracts.

- Funding - A DAO requires funds to function after its founders have established the smart contracts that will control it. The production and distribution of some kind of internal property, like as a native currency that the DAO can spend, use in voting processes, or use to reward specific activities, is included in the DAO's smart contracts. From there, anyone or organizations who are interested in supporting the DAO's expansion can buy or otherwise obtain the native token of the DAO, which usually entails gaining voting privileges.

- Deployment or Launch - All of a DAO's choices are made by consensus vote after it has enough funding to be put into use. As a result, all token owners turn into stakeholders with the ability to suggest changes to the DAO's operations and financial management. The DAO's stakeholders work towards the best result for the whole DAO network considering that the token distribution policy and consensus processes specified in its underlying smart contract architecture are well-designed. As a result, the emerging DAO organization is free to function without the guidance of its founders or any other

central authority. DAOs are open source, which generally guarantees complete openness and immutability. All of their rules, transactions and other activities are also recorded on the blockchain and available for anyone to review. [13]

2.3.2 Traditional Organizations VS. DAOs

In traditional organizations, all employees are subject to employment contracts that govern their interactions with the business and one another. Their rights and obligations are governed by written agreements and upheld by a legal system that is governed by the country in which they live. The legal contract will specify who can be sued in court for what if something goes wrong or someone doesn't keep their half of the bargain.

DAOs, on the other hand, entail a group of people interacting with one another in accordance with an open-source protocol that self-enforces its rules using smart contracts. The native network tokens are given in exchange for securing the network and carrying out other network duties and are used to reward individual behavior and encourage it to work toward a group objective. A DAO's members are not formally bound by any legal contracts or by any other kind of external organization. Instead, they are directed by rewards linked to the network tokens and completely open rules that are included in the software. [14]

2.3.3 Decentralized Finance

Decentralized Finance (DeFi) enables financial interactions between participants without the use of middlemen. By doing away with a central authority and enabling peer-to-peer transactions for financial services including banking, loans, mortgages, and more, it employs the blockchain technology, mainly built on Ethereum platform, hence devoid of any centralized control over financial transactions. Participants may quickly carry out peer-to-peer exchanges, manage their assets, and create decentralized applications. DeFi gives everyday people access to the essential components of the activity currently done by banks, exchanges, and insurers, such as lending, borrowing, and trading. Since services are managed by code that anybody can view and scrutinize, they are automatic and safer than they were when they were previously slow and susceptible to human error.

Decentralized Finanace (DeFi)	Traditional (Centralized) Finance
The individual is has complete responsibility for the security of their assets on a DeFi Blockchain	The individual has to trust the organization for keeps their assets secure
The asset management decisions rely solely upon the individual to which they belong	The organization can utilize and individual's asset as their own investments which could be risky
Transfer of funds and assets happen within minutes	Payments, especially international ones, can take days due to manual processes
DeFi is open to all	Financial services offered by organizations have the right to restrict anyone from their services

Table 2.1: A comparison of Decentralized Finance (DeFi) and Traditional Finance

Decentralized Finance Applications [15]

- Personal Transactions - Ethereum is a blockchain that is made for sending transactions in a safe and international manner. Ethereum enables international money transfers that are as simple as writing an email.
- Borrowing - The decentralized applications built on Ethereum allow you to borrow money by depositing an asset as a collateral.
- Lending - By lending your cryptocurrency, you can get interest on it and watch your money increase in real-time.
- Insurance - Decentralized insurance seeks to reduce costs, increase payout speed, and increase transparency. Increased automation makes insurance coverage more inexpensive and accelerates payout times. The information used to evaluate your claim is openly disclosed.

2.4 MakerDAO

MakerDAO functions fundamentally like a credit institution that grants loans with a set interest rate. It is a potent peer-to-peer organization with the mission of creating technology solutions for cryptocurrency-based borrowing, saving, and lending. On MakerDAO, the entire loan and borrowing process is controlled by smart contracts, which are completely

decentralized and safe. A strong infrastructure layer is provided by MakerDAO for the decentralized Ethereum-based economy, also known as DeFi or Decentralized Finance, which aims at providing financial services at a global level with the use of a stable cryptocurrency called DAI. The project, started in 2015 by Rune Christensen, the current leader of MakerDAO, has succeeded in releasing the potential of DeFi across the globe.

The way that MakerDao operates and is managed sets it apart from other initiatives that lend or manage stablecoins. Stablecoin projects frequently feature a central organization working to maintain the stablecoin's peg against a fiat currency. Lending projects typically involve a corporation that manages the lending process from beginning to end. MakerDao uses Ethereum smart contracts in its governance and automation systems to carry out lending and stabilizing tasks independently of a single party.

2.4.1 DAI Stablecoin

One kind of cryptocurrency that is intended to keep its value constant over time is a stablecoin. A stablecoin's value is often tied to a certain real currency, frequently the US dollar. In this scenario, one cryptocurrency unit often equals one unit of fiat money. The major disadvantage of cryptocurrencies, like Bitcoin and Ethereum, is their extreme price volatility, which is why stablecoins, which have more stable pricing, have a strong attractiveness. Cryptocurrencies have a relatively small market cap compared to conventional assets, which causes their prices to be quite unstable. Even Bitcoin, the most widely used cryptocurrency, has wildly fluctuating prices. Loans, derivatives, prediction markets, and other long-term decentralized finance services built on blockchain depend on price stability and they are hampered by price volatility, hence the need for stablecoins.

Tether is one Stablecoin that is most widely used and it has the highest market cap, i.e the most amount of money (valued in US dollars) currently invested in this cryptocurrency. Tether, having the Token Symbol of "USDT", is a cryptocurrency that aims to always maintain a constant price. Tether Limited developed the cryptocurrency USDT to serve as the internet's Digital Dollar, with each token having a value of $1 USD and being backed by $1 USD in actual assets. Since the USDT does not theoretically appreciate or depreciate, its function is to offer liquidity and act as a hedge against market volatility. The reserves of Tether are the only factor affecting the value of Tether (USDT). The value of Tether will always equal one US dollar as long as it is backed 1:1. Since the company holds the reserves

privately, it is seen as a centralized stablecoin because the users have to trust the company completely to hold the equivalent amount of US dollars worth assets as the number of USDT coins that are there in the market. In contrast to Tether, DAI is decentralized, which implies that no centralized entity controls the supply of new DAIs in circulation.

DAI is a decentralized cryptocurrency that aspires to keep its value relative to the US dollar at 1 to 1. Therefore, 1 DAI must equal 1 USD. DAI may be stored and transferred directly to anybody, wherever in the globe, without going through intermediaries like banks or other centralized organizations because it is a cryptocurrency built on the Ethereum blockchain. There are other additional stable cryptocurrencies that are tied 1:1 to the US dollar.

DAI has the following benefits of use:

- For transactions which require price stability like exchange of goods and services between retailers and individuals, DAI would serve the purpose as its value is kept stable at 1 Dollar. It is difficult to price a product or service using ETH (Ethereum) or BTC (Bitcoin) because their prices fluctuate greatly.
- Because DAI cryptocurrency is built on the blockchain, it can operate without the need for a bank account and the transfer can be made across the globe with a very low transaction fee and will always be quicker than an international fund transfer. They can be more secure than your funds in a bank account if the individual uses correct wallet and security techniques of storing the cryptocurrency.
- DAI is a great technique to add or exit holdings without going off-chain and will assist to balance some of the overall risk in your portfolio.

2.4.2 Collateralized Loans

Lenders want to be sure that you have the means to pay back any loans before they give them to you. Because of this, many of them need security of some kind. Collateral is a type of security that lowers the risk for lenders. It aids in ensuring that the borrower fulfills their financial commitment. If the borrower does fall behind on the loan, the lender has the right to seize the collateral, sell it, and use the proceeds to cover the outstanding balance. Traditionally, collateral may be in the form of real estate, inventory financing, invoices, or any object that holds a certain value higher than the loan amount.

Collateralized loans are the foundation of open lending protocols in decentralized finance. No one has a credit score or any other type of formal identify associated with the loan they are taking out since DeFi supports open, pseudo-anonymous finance. As a result, much like mortgages, the majority of DeFi lending applications will ask for collateral in order to hold borrowers responsible for repaying the amount. However, the main distinction between DeFi collateralization and conventional collateralization (as it currently exists) is that, in order to collateralize a loan on MakerDAO or Compound, the borrower will have to over-collateralize the loan. This indicates that the collateral must be worth more than the loan itself in order to obtain the loan. Borrowers must put up a minimum of 150 percent of the loan value in collateral for MakerDAO loans. To elaborate, if you wanted to borrow 100 Dai on MakerDAO, you would need to put up collateral of at least $150 in ether. In order to determine the liquidation price and the quantity of Dai you will receive in return, you can select your collateralization ratio. The price of ether at which your loan will be worth more than the required minimum level of collateralization is known as the liquidation price. Because the cryptocurrency market is extremely volatile, the risk of liquidation is high. To minimize this risk, investors can either take out less DAI or collateralize their loans with more value to avoid the penalty of liquidation, which gives investors a comfortable cushion in the event of market volatility and to avoid the liquidation penalty. [16]

2.4.3 MakerDAO Collateral Vaults

A crucial element of the Maker Protocol is the Maker Vault, through which it generates Dai against secured Collateral. The use of all the vaults affects the total amount of Dai available. Users generate Dai against their Collateral, which the protocol then destroys when they pay back their generated Dai balance. This procedure takes place on-chain, enabling complete auditability of the Dai that is in circulation and the Collateral that supports it. In order to prevent the liquidation of their positions, vault owners are required to maintain a Liquidation Ratio and to over-collateralize their vaults, based on the principles explained in the previous section. Any user may generate Dai by depositing Collateral into a Vault and, in exchange for the generated Dai balance, paying a Stability Fee. The Stability Fee is an amount that gets added to the Vault owner's Dai balance by the protocol and this fee may vary with time. To handle the inherent risk of generating Dai against collateral in Maker Vaults, and to support the functioning of the Maker Protocol, including the DSR, Risk Teams, and other expenses related to maintaining the protocol, a Stability Fee is charged on the loan amounts.

[17]

Additionally, a global Debt Ceiling is imposed on the Maker Protocol as well as a specific Debt Ceiling for each type of Vault. The maximum amount of DAI that may be created using a certain vault type by all vault users is controlled by the Debt Ceiling parameter. The transaction will fail and no DAI will be minted if a user attempts to mint DAI and the quantity of DAI minted will put the vault type's amount of DAI minted above its Debt Ceiling. The Debt Ceiling parameter's main objective is to give Governance control over how much DAI may be generated utilizing a particular vault type. The risk exposure to the collateral employed within a given vault type is reduced by managing the amount of DAI minted from that vault type.

Users of the Vault are free to add or remove Collateral at any time and generate or repay Dai as per their convenience. Owners of Vaults are permitted to engage with their Vaults as long as they maintain a minimum Collateralization Ratio, which is designated for each type of Vault as the Liquidation Ratio. A position becomes liquidated if a vault's liquidation ratio is exceeded. By closing out Vaults that fall short of the minimum needed Collateralization Ratio, Liquidation helps to ensure that Dai is always backed by an adequate quantity of Collateral, thus maintaining the stability of the protocol and the stablecoin Dai. In case the value of the collateral drops below the liquidation price, the process of liquidating Vaults is automated by the use of "keepers" which will be explained in the following section.

2.4.4 Keepers

Keepers are typically bots that are run to interact with decentralized protocols in order to "keep" them in a healthy state. The most well-known example is liquidation bots on Maker or Aave, which liquidate (close) user loans with a health factor that is too low, to maintain the financial stability of the system. Because blockchains are passive environments where nothing happens unless a user interacts with the network to execute and validate a transaction, keepers are necessary. As a result, an external transaction is necessary to update the status of a protocol on a blockchain and trigger further operations. While some of these transactions can't be completed automatically, they are required for protocols to work as intended. Keepers are directly or indirectly rewarded with the protocol's currency as they bear the cost of providing computation resources.

In MakerDAO, there are five primary types of Keepers as follows:

- Bite Keeper - This keeper is responsible for triggering the liquidation of unsafe vaults, the collateral value of which has dropped below the liquidation price. It then makes the collateral available for sale. The Bite-Keeper can then take part in the collateral sale by buying the collateral at a reduced price and selling it right away on the an exchange to make money.

- Arbitrage Keeper - In order to profit from a price differential, an investor will use the investment method of arbitrage to simultaneously buy and sell an asset in other marketplaces. The returns can be impressive when multiplied by a high volume, despite the fact that pricing variations are often tiny and transient. The Arbitrage-Keeper continuously searches for lucrative arbitrage chances. The advice provided by this Keeper on where to search and how to take advantage of arbitrage opportunities in the Dai system is also quite helpful.

- CDP Keeper - CDP stands for Collateralised Debt Position. This keeper is responsible for monitoring and managing the vault health by either topping up with additional collateral or by repaying the Dai debt.

- Auction Keeper - The auctioneer may take part in surplus, bad debt, and collateral auctions. It primarily focuses on interacting with smart contracts and enables pluggable bidding models, making it simple for users to create their own auction participation tactics without having to deal with complicated Ethereum and low-level smart contract issues.

2.4.5 Real-World Assets

During the intial stage of MakerDAO, only Ether (ETH), a cryptocurrency on the Ethereum platform, was supported as collateral, later it evolved into supporting multiple Ethereum-based cryptocurrencies. The list of supported cryptocurrencies as collaterals can be found on the OASIS App which is used to trade, borrow and lend. The most recent development has been to integrate Real-World Assets into the system to be used as collateral. The market cap of cryptocurrencies is roughly 2 trillion whereas the market cap of Real-World Assets is about 650 trillion [18].

Implementation of RWAs creates prospects for a wide range of assets, including artwork,

music royalties, real estate, invoices, trade receivables, as well as any kind of tangible object that can be tokenized, which can essentially be any asset. Additionally, it opens up amazing prospects for people and Small-to-Medium Enterprises who are now unable to get capital through conventional banking channels. Even though they only make up 1% of the entire value locked, Maker's real world assets presently account for 10% of the protocol revenue [19]. This presents huge growth potential for the MakerDAO protocol but it comes with its own set of challenges as well in terms of representing the Real-World Assets on-chain and maintaining the legal and commercial infrastructure around it.

MakerDAO's Dai will become more stable and safe to use if it is backed by various Real-World Assets, the value of which do not fluctuate as much as those of cryptocurrencies, thereby increasing its demand.

Tokenization of Real-World Assets

Asset tokenization is the process of transferring of ownership rights of a physical asset into a digital token in the blockchain, usually Ethereum environment. This can be achieved using an ERC20 token standard explained in Ethereum section, which leads to a bridge between the tangible asset (such as real estate, a car, gold, etc.) and its representative token, that is recognized by the law. A solid legal framework connecting the token and the asset is a must for all tokenization schemes. To give the owner of a tokenized asset a legal right to the actual physical item, sound legal structuring is necessary. The owner of the token has a strong, legally-backed claim to ownership of the asset if it has been fully tokenized. Owning the entire set of tokens (because tokens are frequently divisible) that correspond to an asset means that the owner completely and unrestrictedly owns the asset. Effective tokenization, for instance, would allow the owner of the token (or tokens) to have complete access to the real estate property. When tokenization is done well, holding the token and the title to the property are one and the same.

One of the first blockchain companies to support real-world assets (RWA) is Centrifuge. Anyone can use the integration to fund a Centrifuge pool with assets and obtain a loan in DAI, MakerDAO's stablecoin. Any physical asset that may be digitized, including vehicles, real estate, and other types, may be included in these assets. Tinlake is the first decentralized application built by Centrifuge, which gives the DeFi ecosystem real-world yield. Once tokenized, Tinlake assets can be freely transferred and used across the vast ecosystem of

DeFi products and services. [20]

The Maker (MKR) Token

MakerDAO has an ERC-20 token called Maker (MKR). This serves as a governance token, allowing holders of MKR funds to cast votes inside the MakerDAO ecosystem to influence changes and the direction of the project. The risk management and business logic of the Maker system are voted on by MKR holders. The success and survival of the system depend on effective risk management. Continuous approval voting mechanism is used in the system governance process. As a result, each MKR holder has the ability to vote for as many proposals as they choose, propose new proposals, and cast or withdraw votes whenever they choose. The top proposal is the one that receives the greatest support from all MKR owners and can be used to alter the system's risk thresholds. Prior to the proposal's changes being applied, there is a security pause to give the community time to respond and stop malicious proposals from disrupting the system. [21]

Chapter 3

Design

In the previous sections, we've explored the basics of Blockchain, Ethereum, Smart Contracts, and various technicalities involved in MakerDAO. We've covered the aspects of borrowing against collateral and how MakerDAO uses the principle of over-collateralization to maintain the stability of the financial platform. Using that knowledge, we will further explore the design aspects of an application that would allow an individual to borrow Dai stablecoin against their equity portfolio using a decentralized application that will act as a bridge between MakerDAO and the "Equity Management Service".

The term "Equity Management Service" in this work refers to an entity that manages equities for individuals and institutions as well. The main function of this service is to hold and transact equities in electronic form and facilitates the settlement of trades on stock exchanges

3.1 Overview of the Approach

The design covers the following scenario of borrowing Dai (or Dollars) using equity portfolios as collateral.

An individual borrower approaches their Equity Management Service to take out a loan against their equity portfolio. The Equity Management Service holds the individual's equities in an electronic format at their end. The Equity Management Service, based on the borrower's instructions, can tokenize the borrower's equity portfolio using our Decentralized Application, Equity Tokenization dApp, which acts as a bridge between the Equity Management Service and the MakerDAO ecosystem.

The Equity Tokenization application is responsible for representing the borrower's equity on-chain using the ERC-20 standard. Once tokenized, this Equity Token will be locked in a Maker Vault and the Equity Management Service will hold Dai on behalf of the borrower, and the Equity Management Service will give out a loan in dollars to the end user. This ensures that the end user does not have to deal with any of the blockchain technologies or web portals by themselves, and can easily get a loan based solely on the valuation of their equity portfolio.

In case the value of their equity portfolio drops below the liquidation price, the ownership of the stocks will be transferred to the Equity Management Service, and they can then decide further course of actions, whether to sell it off in the equity markets or to hold. As far as MakerDAO is concerned, it will get the Dai back and it will be burnt, or in other words, destroyed, to maintain a balance between Dai generated and the collateral that it holds. In case of liquidation, MakerDAO will transfer back the Equity Token to the Equity Management Service, thus maintaining financial stability.

The functional aspects of borrowing Dai using MakerDAO will remain the same for equity collateral as well, as they are for other collaterals. The user will be able to repay Dai at any point, and get back their Equity Token, as well as the liquidation use case would remain the same as it is for other collaterals.

Equity Management Service

This is an independent entity that is responsible for holding an individual's equity stocks in an electronic format. The individual would have to approach their Equity Management Service in order to take out a loan using their equity portfolio.

Equity Portfolio

An equity portfolio is a collection of stocks that is owned by an individual. This portfolio is digitally maintained and kept secure by an equity management service or a depository service. Traditionally, these equities were held in the form of physical equity certificates which indicated ownership and rights on those equities. Over time, the depository services or the equity management services digitalized the service and enabled holding and trading of these assets online.

In our application, we will tokenize a user's equity portfolio using the ERC20 token standard explained in section 2.2.5 and make it available on the blockchain. We've chosen ERC20 token standard because MakerDAO's Collateral Vault functionality integrates well with ERC20 token stardard. The tokenization of the user's equity portfolio would involve the equity management service to access the application called as Equity Tokenization dApp (Decentralized Application), where the service acts based on the borrower's instructions and it can enter the name of the equity and the quantity the individual wishes to borrow a loan against. These equities and their respective quantities would then be represented on-chain using the ERC-20 token standard, and this token is called as "Equity Token" and has the symbol "EQT". There will be an Equity Token Manager that will act as a Factory Contract responsible for generating different contracts for different borrowers and maintaining a record of all of those Equity Tokens. A factory contract is basically one that produces other contracts, hence the term "factory".

Once the Equity Token has been created, the ownership of it will remain with the Equity Management Service as the borrower will not be able to access this token directly.

Equity Portfolio Value

Each equity in a user's portfolio will have a value based on its current market price on the exchange where it is listed. For example, at the moment the price of one Apple (AAPL) stock is $170 on the NASDAQ Global Select Market. The equity portfolio's value is the total sum of all the equities' current market price multiplied by individual equity quantities.

Since blockchains cannot access outside data because of the "blockchain oracle problem" as explained in section 2.1.3, we have to make use of the Decentralized Oracle Network offered by Chainlink to fetch the prices of the equities. Chainklink Data Feed at the moment offers data feeds for a number of equities like Apple, Amazon, Meta, Google, and a few more but this could be extended to including all the equities listed on a particular exchange, if enough demand builds up in the future. In our design, instead of building an oracle network on the MakerDAO's end to fetch the prices of equities, we can directly integrate the Chainlink data feed in the ERC20 tokens to calculate the value of the equity portfolio. These ERC-20 Equity Tokens (EQT) will hold the user's equity names and quantities and upon requesting the valuation of this token, the smart contract will query the Chainlink Oracle Network to fetch the current market prices of each of the equities and then multiply with

the user's quantity to get the total value of the portfolio. This value would change based on the market index.

MakerDAO Equity Vault

For real-world assets, at the MakerDAO end, we would have to create a Vault that supports the Equity Token as it will have to access its valuation to calculate the amount of Dai to be minted. Also, each vault type has different risk parameters and they would have to be configured separately for Equity Token vault type as well. The Vaults can call the "getValuation()" method on the ERC-20 Equity Token, which will then query the Chainlink's oracle network to fetch the current prices, and calculate the Equity Token vault and pass it over to the equity token vault. Once the vault has the value of the equity token, then based on the collateralization ratio for this vault type, it will mint Dai on the borrower's address. Similar to the Equity Token Manager, there is a Vault Manager that is responsible for maintaining a record of different vaults.

MakerDAO Vault Keepers

In addition to creating a MakerDAO vault catering to the Equity Tokens, we would also have to create a keeper type, as mentioned in the section 2.4.4, that would be responsible for checking whether the equity token vault is at a healthy collateral level or not. In case the total equity value of the ERC-20 Equity Token falls below the liquidation price, this keeper would trigger the liquidation process that would involve the Equity Management Service to get back the Equity Token and the Dai would be transferred back to MakerDAO and it will be burnt.

3.2 Architecture

The figure 3.2 shows the architecture diagram of borrowing Dai using equity portfolio as collateral. We will explore each of the blocks in this section.

Equity Management Service

This is the infrastructure part of the Equity Management Service that holds and manages equities on behalf of their users. This is an independent pre-existing service that already exist. Even though the company holds and manages equities, the rights on those equities

still belong to the users themselves and no action can be taken without the user's approval.

In order for this entity to interact with the Equity Tokenization dApp, we will need an integration layer that will be responsible for getting approval from the user and also the details of which equities to take a loan against, along with individual equity quantities. The loan request from the user would be a manual one where the user would have to contact their Equity Management Service personally, and the employees of that management service would be responsible for interacting with the tokenization dApp on behalf of the user. This integration layer would be responsible for the following activities:

- Getting the user's approval on which equities and their respective quantities to tokenize through a One-Time Password authentication mechanism that the user would receive once they request the processing of the loan.
- Maintaining a record of Ethereum addresses for each of the user loan accounts, and vault addresses as well, as this equity management service will be responsible for managing the loan process using MakerDAO.

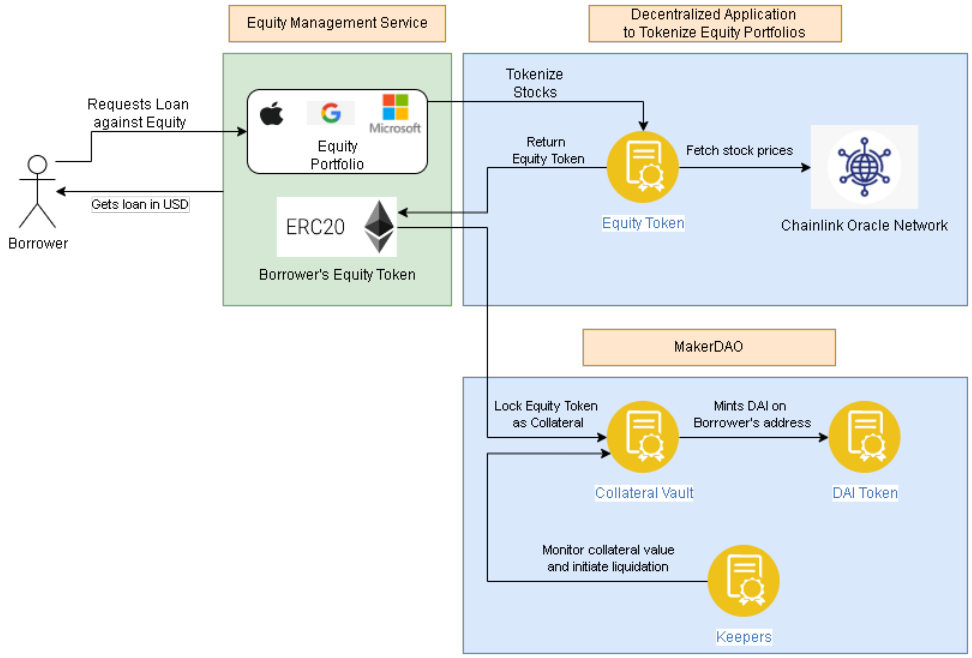

Figure 3.1: Architecture Diagram of Borrowing Dai backed by Equities using MakerDAO

Equity Tokenization dApp (Decentralized Application)

This component is responsible for creating ERC-20 tokens based on the requests coming from the Equity Management Service. This will be a decentralized application that the Equity Management Service will use to tokenize an individual user's equity portfolio. The result of this component would be an Equity Token (EQT) that will be given back to the Equity Management Service. The Equity Token Smart Contract would have the following information:

- Equity Name - The name of the company that the user holds
- Equity Quantity - The quantity indicating the number of stocks of a particular company
- Borrower's Ethereum Address - This will hold the borrower's wallet address, Dai would be minted to this address.
- Owner - This field will indicate the current owner of this equity token which would be the Equity Management Service.
- Price Feeds - This would consist of all the oracle price feeds of equities which would fetch the current market price of each individual equity.
- Valuation Method - This function would be responsible for calculating the current value of this equity token based on the equity prices fetched using oracle price feeds.

This component would be dependent on the Chainlink's Decentralized Oracle Network which was explained in section 2.1.3, for fetching the prices of equities.

MakerDAO

At the MakerDAO's end, we would need to develop the following two important components to enable borrowing against Equity Tokens.

- Equity Token Collateral Vault - This would be a specialized Vault that would only deal with Equity Tokens as collaterals. This vault would be able to interact with the Equity Token to get its current valuation based on which it can decide how much Dai to generate. The following information would be important to store in the Vault:
  - MakerDAO's Address - This is an Ethereum address representing MakerDAO's account that will be responsible for holding locked collaterals, it will gain the

ownership of the collaterals and the vaults will generate Dai in return of this collateral lock.

  - Borrower's Address - This is the address on which Dai will be minted (created).
  - Collateral Value - This is the total valuation of the equity portfolio
  - Liquidation ratio - The value at which the collateral would be liquidated
  - Equity Token address - This field holds the token address of the ERC-20 collateral

- Equity Token Collateral Keeper - At MakerDAO's end, we would need to create a keeper that would check the health of the Equity Token Vault, much like the keepers explained in section 2.4.4. This keeper would be responsible for monitoring the current equity portfolio value and will trigger liquidation of the asset in case the price drops below the liquidation price.

3.3 Working

Figure 3.2 shows the steps involved in borrowing a Dai loan using Equity Portfolio as collateral in the MakerDAO ecosystem. The steps are described as follows:

- 1. Borrower approaches their Equity Management Service with the intention of borrowing a loan using their equity portfolio. At this stage, the borrower also allows the Equity Management Service to access their equities against which they wish to take a loan

- 2. Equity Management Service then utilizes the Equity Tokenization DApp to tokenize the borrower's equity portfolio. This results in an ERC-20 token called as Equity Token (EQT) which consists of a digital blockchain-enabled representation of the borrower's equity portfolio.

- 3. An Equity Token is generated and the Equity Management Service has the ownership of this token, while the borrower's address is credited with this token. This Equity Token can now interact with the MakerDAO's ecosystem as it follows the ERC-20 token standard.

- 4. The Equity Management Service can now create a Vault that supports Equity Tokens in the MakerDAO ecosystem and can lock this Equity Token as Collateral.

- 5. Once the collateral is locked, the MakerDAO Vault mints (creates) Dai based on the collateralization ratio and transfers this newly generated Dai to the borrower's address that is managed by the Equity Management Service itself.
- 6. Once the Equity Management Service has access to the generated Dai, it then offers a loan in Dollars to the borrower. This design ensures that the borrower remains independent of this mechanism.

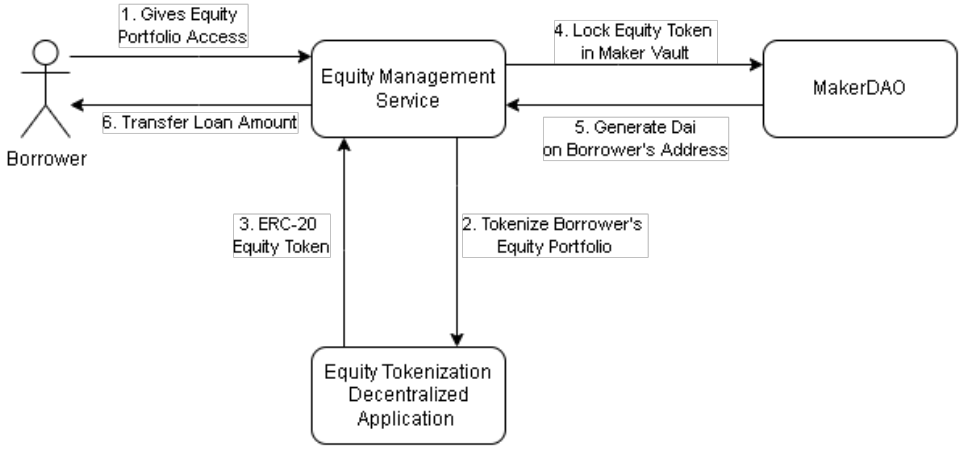

Figure 3.2: Workflow Diagram of Borrowing Dai backed by Equities using MakerDAO

Chapter 4

Implementation

4.1 Implementation Details

In the previous sections we've covered the technologies and the design aspects that would enable the borrowing of Dai using Equity Portfolio as collateral in the MakerDAO ecosystem. In this section we will explore an implementation of a proof-of-concept prototype that highlights the working pieces of this use case. The prototype was implemented and tested locally and the figure 4.1 shows the different components involved in the successful implementation of the prototype. In the following sections, we will explore each of the components.

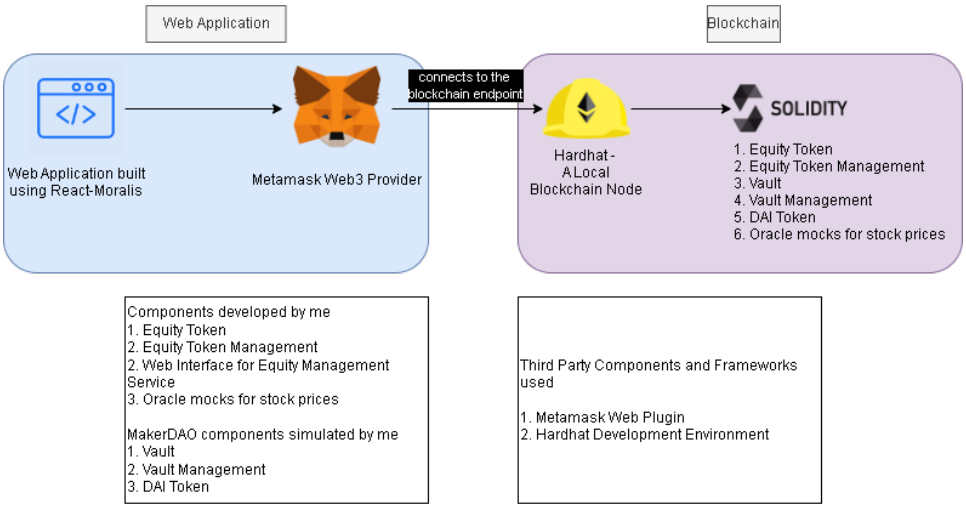

Figure 4.1: Implementation of the Prototype

Frontend

The frontend was built using React Moralis library. React (also known as React.js) is a free and open-source JavaScript UI library for creating interactive user interfaces. React is a collection of pre-written JavaScript code that abstracts away basic UI tasks, allowing developers to focus only on developing efficient UIs. React.js also makes use of JSX, a templating language that compiles your code to plain JavaScript functions. The abbreviation JSX stands for "JavaScript XML." When developing UI elements, you can use JSX to blend HTML syntax and JavaScript function logic. It also requires tag closure, as inspired by the XML language.

React-Moralis is a wrapper around the functionalities provided by the Moralis framework, which makes it easier to call functionalities such as checking whether the browser supports interaction with the blockchain, and we have also used it to make it easier to call the smart contract functions as it provides a function called as "useWeb3Contract", where we just have to specify which function we wish to invoke, along with the required details such as contract ABI, contract address (the address where the smart contract is stored on the Ethereum Blockchain), function name, and parameters.

Figure 4.2: Frontend of the Proof-of-Concept

As it can be seen in figure 4.2, the Equity Management Service can perform the follow-

ing steps on behalf of the borrowers:

- 1. The service can enter the quantity of stocks of Apple, Google, and Microsoft against which the borrower wishes to take a loan. Once the stock quantities are entered, the "Tokenize Borrower's Equity Portfolio" will create an ERC-20 Equity Token on the borrower's address entered in the beginning.

- 2. Once the Equity Token has been created, its address can be fetched using the "Get Borrower's Equity Address", and the "Get Portfolio Value" will fetch the latest value of the Equity Token based on the mocked stock prices shown below on the screen.

- 3. The service can then create a Vault using the Equity Token address and give the ERC-20 spending allowance to MakerDAO using the "Approve Token Exchange between Borrower and MakerDAO" button. This will allow MakerDAO to perform actions on the ERC-20 Equity Token.

- 4. Once the service clicks on "Draw DAI", the Vault will transfer the Equity Token to MakerDAO's address and it will generate Dai based on the collateralization ratio on the borrower's address.

- 5. The "Update Stock Prices" button is used to simulate real-time stock prices updates and it drops the values to nearly half for each of the three stocks.

- 6. The "Liquidate" button is used to simulate liquidation of the collateral in case the liquidation price for this Vault is hit, and the Vault becomes unsafe. The liquidation process involves MakerDAO getting back the Dai minted and the Equity Management Service gets the Equity Token back.

Metamask

MetaMask is a browser plugin that acts as an Ethereum wallet and is installed in the same way that any other browser plugin is. It allows users to store Ether and other ERC-20 tokens once loaded, allowing them to transact with any Ethereum address. Users may interact with websites that host Ethereum-based apps and smart contracts using MetaMask, and it allows users to save Ethereum-related information like public addresses and private keys like any other Ethereum wallet (i.e. it turns your web browser into an Ethereum browser). It can communicate with the website you're now viewing as a browser plugin. It accomplishes this by inserting the JavaScript library web3.js into each website you visit. Once injected, a web3

object will be accessible through window.web3 in the website's JavaScript code.

As the Equity Management Service initiates the tokenization of the user's equity portfolio, the Metamask window opens up and the transaction would have to be approved from the Equity Management's account. Each and every transaction that modifies the state of the blockchain must be approved from the Metamask plugin, and this costs some ether to the account from which the transaction was initiated. Only those transactions that do not modify the state of the blockchain, for example read operations, would not cost any ether to the account from which the "get" transactions are triggered, for example, to fetch the value of the Equity Token, a Metamask transaction would not be triggered.

Hardhat

Since the prototype was developed locally, Hardhat framework seemed like the most appropriate choice as it simplifies the process of smart contract creation and deployment. It even offers deployments of test or mock contracts that are quite helpful to simulate oracles or any external contract dependency.

Hardhat is an Ethereum software development environment which is made up of various components that work together to offer a complete development environment for compiling, debugging, and deploying your smart contracts and dApps. When utilizing Hardhat, Hardhat Runner is the main component developers interact with and it is a versatile and extendable task runner that assists you in managing and automating the recurrent processes associated with developing smart contracts and dApps.

Tasks and plugins are key to the architecture of Hardhat Runner. A task is a JavaScript async function with certain metadata attached to it. Hardhat uses this metadata to automate some tasks for you. Parsing arguments, validity, and help messages are all handled. In Hardhat, everything you can accomplish is defined as a task. Tasks can call other tasks, allowing for the creation of complicated processes. Plugins as reusable pieces of code that add extra functionality to the base layer. Users and plugins can override current tasks, allowing workflows to be customized and extended. [22]

Hardhat Network

Hardhat includes the Hardhat Network, a private Ethereum network node intended for development. The Ethereum protocol (that is, a client) has several implementations, the most common of which being GETH (written in GO). Others, though, are written in other languages. The crucial thing is that they all adhere to Ethereum's standards. To run your files, Hardhat employs the JS implementation of the EVM. This indicates that Ethereum JS is operating on your PC. That is how Hardhat understands what to do when you send transactions, test your contracts, and deploy them internally.

Solidity and Smart Contracts

The smart contracts mentioned in 4.1 were written using Solidity programming language.

Solidity is an object-oriented programming language developed by the Ethereum Network team primarily for building and developing smart contracts on Blockchain platforms. It is used to establish smart contracts in the blockchain system that apply business logic and generate a chain of transaction records. It serves as a tool for writing machine-level code and compiling it for execution on the Ethereum Virtual Machine (EVM).

The following are the smart contracts written to achieve the functional aspects of the prototype.

- Equity Token - This smart contract represents the equity portfolio of the borrower as it has a list of stocks along with the quantities that the borrower wishes to take a loan against. This Equity Token is collateral that can be used to secure a loan. This basically is the tokenization contract that converts and tracks a real-world asset into an ERC-20 token that can function on the Ethereum Blockchain.
- Equity Token Management - This smart contract acts as a contract factory that generates other contracts. It is responsible for creating new Equity Tokens and keep a track of all of those tokens.
- Vault - This contract will reside on the MakerDAO system but it was simulated locally to demonstrate the functional aspects of it. It will hold the Equity Token as collateral and keep a track of the valuation. This contract is responsible for generating Dai

against the collateral and lock the collateral on MakerDAO's address once the Dai has been generated on the borrower's address.

- Vault Management - Similar to Equity Token Management, this is a contract factory that generates and manages multiple Vaults that hold Equity Token as collateral.
- Dai - This is again a simulation of the MakerDAO Dai contract to show the functional aspects of this prototype. The Vault generates Dai based on the valuation parameters set for the Equity Token collateral.
- Oracle mocks - These contracts are responsible for simulation of live oracles where the Equity Token contract will query an oracle to fetch the price of the stock. The details for this are explained in the following section.

Oracles for Stock Prices

As seen in the screenshot 4.2, the proof-of-concept was built using 3 stocks, Apple, Google, and Microsoft. The current market prices of these equities are displayed at the bottom. These prices were fetched using a mocked oracle contract built by Chainlink, but in the real world application, these would be fetched from the Chainlink oracle service offered on the Ethereum network. It is achieved using an AggregatorV3Interface, the detailed guide of which can be found here. For the prototype, I've used a MockV3Aggregator interface developed by Chainlink to mock the stock prices of Apple, Google, and Microsoft. To mock the stock prices, we're simply required to extend this interface and pass the initialization parameters, like the stock prices, in the constructor. Because it is built using an interface mechanism, the method to access the latest stock prices remains the same irrespective of whether it is running locally or on the blockchain network, so we will not require functional changes.

4.2 Implementation Results

Through this tech stack, we were able to successfully implement the functional aspects of the idea of borrowing a loan against equities by simulating the MakerDao ecosystem and by building the bridge to connect the Equity Management Service with the MakerDAO system.

The code repository for the frontend can be found at this Github link.

Chapter 5

Conclusions & Future Work

Through this work we have learnt about Blockchain, a decentralized network, and Ethereum that makes decentralized applications possible with the use of Smart Contracts. We have covered the important aspects of Decentralized Autonomous Organizations that give rise to Decentralized Finance applications like MakerDAO. DeFi is a major revolution in the world of finance because of the core principles on which it functions backed by technology, which ensures that there's no single entity in control but a group of people that operate the business transparently using a Decentralized Autonomous Organization (DAO). The importance of this technology lies in the fact that no individual has to rely on the trust of a centralized entity, rather they can operate in a trustless environment powered by smart contracts that are immutable and distributed ledgers, used as storage of records, unlike a centrally managed storage.

MakerDAO makes it possible for any user from any corner of the world to borrow a loan without having to go through credit checks. Although taking a loan was possible only against Ethereum-back cryptocurrencies, MakerDAO is moving towards including real-world assets on-chain that can be used as collateral. They have successfully implemented the functionality of borrowing a loan against real estate, which was an inspiration to look into other asset classes that can be brought on-chain as collaterals.

To extend onto the ideas of integrating real-world assets on-chain, we have explored the possibilities of using Equities as collateral in the MakerDAO ecosystem and with the use of a prototype, we have successfully implemented the functional aspects of this use case. The design of the prototype can be scaled-out to function as a real-world application and the

real-world implementation aspects could also be achieved by using the architectural principles of this prototype.

For the first time in history, a large-scale financial system is emerging without the need of middlemen. So far, DeFi applications cannot compete with traditional banking solutions in terms of security, speed, and convenience of use. However, DeFi has created genuine, operational apps that have already received billions of dollars in funding. These funds will be utilized to create more competitive, user-friendly software.

5.1 Future Work

Integrating with MakerDAO's Ecosystem

In this work, we've worked with a local prototype covering the functional aspects of equity as collateral. This work could be extended into the MakerDAO ecosystem by having a equity-backed integrated Vault that the MakerDAO can create and manage. Instead of developing new oracles to fetch the prices of the equities, we could continue to use Chainlink's decentralized oracle network to fetch the prices. Although it contains pricing oracles for only 9 equities, this work could be replicated to cover an enter market index consisting of all the stocks. But for a trial run, an experimental real-world implementation could be built using the existing equities listed on the Chainlink oracle network as they are the largest companies of the US stock market.

Automating Tokenization of Equities

The design of this approach requires the borrower to depend on the equity management service to provide a loan, but it would be interesting to see if the equity management portal can directly be integrated with the "bridge" between the management service and the MakerDAO ecosystem, the bridge that converts borrower's equity portfolio into an ERC-20 token. Instead of relying on the equity management service, the borrower could directly access his equities through an online portal and carry out the loan process by himself, equity management service would still retain the ownership of the equities in case the borrower defaults on their vault, the equity management service will automatically get the access to the equities which they can then sell-off in the equity market. This would remove most of the functional dependency on the equity management service, and they can provide this as

a service to their end users.

Auctioning Equities

In the current design, the Equity Management Service would get ownership of the equities in case the Vault is liquidated, but using the existing design principles of MakerDAO's auctioning mechanism, these equities could be auctioned off to the users of the same Equity Management Service, and the transfer of ownership of these equities could be reflected in the Equity Management Service's records as well.

Bibliography

[1] Zerion. Blogpost - introduction to decentralized finance (defi). https://www.finimize.com/wp/guides/introduction-to-decentralized-finance-defi/, note = Last Accessed: 2022-08,.

[2] Deyan Georgiev. Blogpost - what is blockchain? https://techjury.net/blog/what-is-blockchain/, month = Aug 15, year = 2022, note = Last Accessed: 2022-08,.

[3] Svetlana Cherednichenko. Blogpost - designing a blockchain architecture: Types, use cases, and challenges. https://medium.com/mobindustry/designing-a-blockchain-architecture-types-use-cases-and-challenges-9894fb7b58e, month = Nov 10, year = 2020, note = Last Accessed: 2022-08,.

[4] Deyan Georgiev. Blogpost - what is cryptographic hash? https://techjury.net/blog/what-is-cryptographic-hash/, month = July 06, year = 2022, note = Last Accessed: 2022-08,.

[5] Joshua. Blogpost - intro to ethereum. https://ethereum.org/en/developers/docs/intro-to-ethereum/, month = July 22, year = 2022, note = Last Accessed: 2022-08,.

[6] Chainlink. Blogpost - what is the blockchain oracle problem? https://blog.chain.link/what-is-the-blockchain-oracle-problem/, month = August 27, year = 2020, note = Last Accessed: 2022-08,.

[7] Chainlink. Blogpost - what is a blockchain oracle? https://chain.link/education/blockchain-oracles, month = September 14, year = 2021, note = Last Accessed: 2022-08,.

[8] Data Flair Training. Blogpost - what is public key cryptography in blockchain. https://data-flair.training/blogs/public-key-cryptography/, note = Last Accessed: 2022-08,.

[9] Joshua. Blogpost - introduction to smart contracts. https://ethereum.org/en/developers/docs/smart-contracts/, month = August 1, year = 2022, note = Last Accessed: 2022-08,.

[10] Zubin Pratap. Blogpost - what are abi and bytecode in solidity? https://blog.chain.link/what-are-abi-and-bytecode-in-solidity/, month = August 17, year = 2022, note = Last Accessed: 2022-08,.

[11] Eiki. Blogpost - explaining ethereum contract abi evm bytecode. https://medium.com/@eiki1212/explaining-ethereum-contract-abi-evm-bytecode-6afa6e917c3b, month = July 16, year = 2019, note = Last Accessed: 2022-08,.

[12] Ethereum. Blogpost - what are daos? https://ethereum.org/en/dao/, month = August 18, year = 2022, note = Last Accessed: 2022-08,.

[13] Cryptopedia Staff. Blogpost - what is a decentralized autonomous organization. https://www.gemini.com/cryptopedia/decentralized-autonomous-organization-dao, month = June 28, year = 2022, note = Last Accessed: 2022-07,.

[14] Blockchain Hub. Blogpost - tokenized networks: What is a dao? https://blockchainhub.net/dao-decentralized-autonomous-organization/, month = August 16, year = 2019, note = Last Accessed: 2022-08,.

[15] Ethereum. Blogpost - what's defi? https://ethereum.org/en/defi/, month = August 18, year = 2022, note = Last Accessed: 2022-07,.

[16] DeFiRate.com contributor. Blogpost - collateralized loans in defi. https://defirate.com/borrow/collateralized-loan, month = August 16, year = 2022, note = Last Accessed: 2022-07,.

[17] MakerDAO. Blogpost - stability fee. https://defirate.com/borrow/collateralized-loan, note = Last Accessed: 2022-07,.

[18] MakerDAO. Youtube - makerdao - rwf youtube channel: The case for continuity. https://www.youtube.com/watch?v=H14oL9BM_-U, note = Last Accessed: 2022-07,.

[19] Ally Zach. Blogpost - makerdao's dive into real world assets. https://messari.io/article/makerdao-s-dive-into-real-world-assets, month = July 22, year = 2022, note = Last Accessed: 2022-07,.

[20] Lucas Vogelsang. Blogpost - centrifuge: Where defi meets real world assets. https://www.gemini.com/cryptopedia/centrifuge-crypto-tinlake-tokenization-real-world-assets, month = February 22, year = 2022, note = Last Accessed: 2022-08,.

[21] MakerDAO. Blogpost - what is mkr? https://medium.com/@MakerDAO/what-is-mkr-e6915d5ca1b3, month = Sept 15, year = 2015, note = Last Accessed: 2022-07,.

[22] Hardhat. Blogpost - getting started with hardhat. https://chain.link/education/blockchain-oracles, month = August 14, year = 2022, note = Last Accessed: 2022-08,.