



Ishup Ali Khan

# Blockchain technology and its implementation with a staking application in an Ethereum Network

Metropolia University of Applied Sciences

Bachelor of Engineering

Information Technology

Bachelor's Thesis

18 November 2021

## Abstract

Author:	Ishup Ali Khan
Title:	Blockchain technology and its implementation with a staking application in an Ethereum network.
Number of Pages:	57 pages + 5 appendices pages
Date:	18 November 2021
Degree:	Bachelor of Engineering
Degree Programme:	Information Technology
Professional Major:	Mobile Solutions
Instructors:	Peter Hjort, Senior Lecturer (Project Supervisor)

---

The main purpose of the thesis is to explore different applications of blockchain technology and provide an overview of the concept. The thesis examines the existing saving accounts system in terms of its low interest return and integrates the blockchain technology into a cryptocurrency staking application and presents it as a viable alternative to the conventional saving accounts. Finally, the thesis aims to provide a decentralized application where a user can track the progress of the transactions in the blockchain network to ensure the complete transparency.

The Ethereum blockchain was used to develop and deploy the cryptocurrency staking application. Various technologies like Solidity, React.js, SCSS, and JavaScript were used during the development process. All of the project requirements were defined, and the project was developed with a test-driven development approach.

As a result, a staking pool application was developed and deployed in the Ethereum Network for the user to stake a cryptocurrency to earn reward. The reward earned can be significantly more than traditional saving accounts' interest return. Furthermore, functionality to unstake the staked amount was also implemented successfully.

Keywords: Blockchain, Bitcoin, Ethereum, Staking, Cryptocurrency, Hash

## **Acknowledgements**

First and foremost, I am extremely grateful to my supervisor Peter Hjort for his invaluable advice, direction, and patience during my thesis.

I would also like to thank Petri Vesikivi for his earlier guidance. I would like to express my gratitude to my Mum, Dad, family members, brothers, and sisters for all the support and encouragement I had received.

Lastly, I would like to thank my wife Narmin for her loving presence, unwavering support, and belief in me during this work.

# Contents

Abstract

Acknowledgements

List of Figures

List of Tables

List of Abbreviations

1	Introduction	1
2	Blockchain Technology	2
2.1	Distributed System	5
2.1.1	Consensus	6
2.1.2	Byzantine Generals problem	8
2.1.3	CAP theorem	8
2.2	Decentralization	9
2.3	Cryptography	11
2.3.1	Symmetric Cryptography	12
2.3.2	Asymmetric Cryptography	13
2.3.3	Public keys, Private keys, and Digital Signatures.	13
2.3.4	Hash functions	14
2.3.5	Merkle tree	15
2.4	A brief History of Blockchain	16
2.5	The Structure of Blockchain	17
2.6	Types of Blockchain	19
3	Applications of Blockchain technology	21
3.1	Hyperledger Fabric	21
3.2	Bitcoin	22
3.2.1	Mining	23
3.2.2	UTXO	24
3.2.3	Blockchain structure	25
3.2.4	Bitcoin limitations	27

3.3	Ethereum	28
3.3.1	Messages and Transactions	29
3.3.2	Ethereum Accounts	30
3.3.3	Ethereum state and transitions	31
3.3.4	Ethereum blockchain	32
3.3.5	Ethereum stacks	32
3.4	Smart Contracts	33
3.5	Alternative Blockchain and Alternative Coins	34
4	Development Environment	36
4.1	Environment and tools	36
4.2	Ethereum Development	37
4.2.1	Frameworks and tools	37
4.2.2	Solidity	38
4.2.3	Web3	39
5	Application design and development	39
5.1	Minimum Viable Product Requirement	40
5.2	Dropped features	40
5.3	Application design skeleton	41
5.4	Metamask wallet setup	42
5.5	Creating and testing smart contracts in Remix	43
5.6	Installing development environment	45
5.7	Creating smart contracts	47
5.8	Compiling and migrating smart contracts	48
5.9	Running the application	50
5.10	Testing the application	54
6	Results and Discussions	55
6.1	Outcome of project	55

6.2 Development challenges and real-life limitations	55
7 Conclusion	57
References	1
Appendices	4
Appendix 1: Contract migration Output	4

## List of Figures

Figure 1. A network view of a blockchain

Figure 2. Structure of different networks as appeared in Paul Baran, Introduction to Distributed communication(Baran, 1979)

Figure 3. Structure of a Blockchain

Figure 4. Process in an Ethereum environment

Figure 5. Application flow diagram

Figure 6. Metamask wallet account and Ethereum Test Networks

Figure 7. Smart contract deployment via Remix IDE

Figure 8. Transaction list from Rinkeby Etherscan

Figure 9. Transaction details for a contract creation

Figure 10. Ganache User Interface

Figure 11. Blocks in the Ganache UI.

Figure 12. Staking Metro coins to earn Yak coins

Figure 13. Two transactions for the (left) approve and staking (right)

Figure 14. Screenshot of the staking application, with four different tokens pool for metro token staking with staked balance, rewards, and remaining balance.

## List of tables

Table 1 Structure of a block

Table 2. Structure of a block header

Table 3. Transaction in an Ethereum environment

Table 4. Message in an Ethereum environment

## List of Abbreviations

APR	Annual Percentage Rate
APY	Annual Percentage Yield
SMTP	Simple Mail Transfer Protocol
HTTP	Hypertext Transfer Protocol
FTP	File Transfer Protocol
TCP	Transmission Control Protocol
IP	Internet Protocol
CAP	Consistency Availability Partition
PODC	Principles of Distributed Computing
PoW	Proof of Work
PoS	Proof of Stake
PBFT	Practical or Distributed Byzantine Fault Tolerant
DPOS	Delegate Proof of Stake
PoA	Proof of Authority
PoH	Proof of History
PoS <sub>t</sub>	Proof of Storage
IPFS	Inter Planetary File System



DB	Data Base
ISP	Internet Service Provider
DAO	Decentralized Autonomous Organization
DAC	Decentralized Autonomous Corporation
DEFI	Decentralized Finance
DAPP	Decentralized Application
DEXES	Decentralized Exchanges
MAC	Message Authentication Codes
RSA	Rivest Shamir Adleman
DSS	Digital Signature Standard
DSA	Digital Signature Algorithm
ECC	Elliptical Curve Cryptography
ECDSA	Elliptic Curve Digital Signature Algorithm
MD	Message Digest
SHAS	Secure Hash Algorithms
RIPEMD	Race Integrity Primitives Evaluation Message Digest
GPUs	Graphic Processing Units
ASICs	Application Specific Integrated Circuits

UTXO	Unspent Transaction Output
EVM	Ethereum Virtual Machine
ERC-20	Ethereum Request for Comment
ECMA	European Computer Manufacturers Association
NVM	Node Version Manager
NPM	Node Package Manager
NPX	Node Package Execute
JS	Java Script
IDE	Integrated Development Environment
ABI	Application Binary Interface
OS	Operating System
KYC	Know Your Customer
UI	User Interface
JSON	JavaScript Object Notation
DOM	Data Object Model
TDD	Test Driven Development
App	Application

## 1 Introduction

Humans have learned to use technology to address issues in a multitude of fields in the current era of developing technology. The adoption and evolution of these technologies has pushed current systems and processes to become more efficient and productive. Despite these progress in the technology, some practices in the field of banking and finance sector are still traditional. One of the forms in this sector where technological progress can be utilized for high yield is standard savings account.

Saving accounts are one of the common options for consumers to earn from their investments. Most banks and financial institutions offer these sorts of accounts, which pay a consistent interest rate on savings. However, depending on the location and the accessibility of the alternatives one can get very less percentage return as annual percentage rate (APR) even with the most generous saving account. These interest rates are low as banks effectively exploit customers' savings for their own profit. Banks may, for example use corporate tactics like arranging loans or make investments with customers' money and subsequently pay out some money as interest to the investors.

This thesis primarily focuses on the staking of digital assets as an alternative to the traditional saving accounts system. Staking in brief is similar to saving but has higher yields to the investments in comparison to the saving accounts' interest returns. The goal of this thesis is to create a decentralized application for staking cryptocurrency. The application is built upon an Ethereum blockchain and provides a concept of staking using some mock cryptocurrencies with the aim to stake real cryptocurrencies in real life backed with real money. Although, the premise for this project is based on staking to maximize interest gain for the investment, the concept can also be utilized to collect funds source for new companies before they list their cryptocurrencies or shares in the market via Initial coin offerings or Initial public offerings. Additionally, in the application

staking participants may pick from a variety of digital assets to stake and withdraw, as well as track their transactions.

Although, staking process gives higher yields than the saving accounts, it comes with some pitfalls. Some staking pools, unlike savings accounts, may be deceptive. Some staking pool campaigns have been known to gather payments and then disappear with them. These projects or campaigns promise extra high returns on the staked amount and later take away all those collected funds. These are also known as rug pools. To prevent being a victim of these scams, users should always double-check the legitimacy of campaigns and see if the staking pool is either centrally managed or entirely based on smart contracts on the blockchain.

The key motivation for this project is to utilize the concept of blockchain technology into an application that can actually bring some variations in traditional saving practices. Having invested into a traditional saving account for years with not much significant returns, is one of the triggering factors for the staking approach taken in this project. An underlying interest with a new fringe of technology, blockchain however is the spurring factor for the project.

This thesis can be viewed in three main parts. The first part begins with an overview of blockchain technology and its historical context. It goes on to explain some of the terms and concepts that make up this technology. The mid-section contains some description on some of its applications. Later parts of the thesis discuss several development tools and methodologies for developing a staking application.

## **2 Blockchain Technology**

In an overly simplistic way, Blockchain is just a computer science term for how to structure, store and share data. Blockchain technology a revolution akin to the internet. However, it is not a passing fad since it offers a solution to a long-standing financial conundrum of relying on a trust free environment without the

dependency onto a trusted authority. Blockchain proponents claim that within a few decades, the technology has the potential to change a wide range of industries. This promise stems mostly from its capacity to provide users with a secure means of transferring value or real assets. Specifically, it is a chronologically developing, append-only database that uses simple cryptographic measures to protect stored transaction from tampering (Bellaj Badr, 2018). To put it differently, each transaction in the database is time stamped, cryptographically protected and any of these transactions data cannot be altered or deleted, however only possible to update through peers' consensus and agreement. In various technical terms Blockchain can be defined as decentralized consensus mechanism, distributed shared ledger of transactions, a data structure, a linked list using hash pointers.

Blockchain is a distributed database, incorporating existing technologies in new ways to drive innovation. Blockchain can be seen as decentralized database controlled by a group of people who store and share data. Blockchain allows a network of independent parties to build a digital ledger of data and share it. From a commercial perspective, a blockchain removes the requirement for a central trusted arbitrator and provides a platform for peers to trade values through transactions. This huge potential of blockchain technology allows it to function as a decentralized consensus process in which the database is managed by no single authority.

As an analogy, Blockchain is Simple Mail Transfer Protocol (SMTP), Hypertext Transfer Protocol (HTTP), or File Transfer Protocol (FTP) which runs on the top of Transmission Control Protocol (TCP)/ Internet Protocol (IP). With the internet as the foundation layer, blockchain may be viewed as a layer of a distributed peer-to-peer network that runs on top of it. This is depicted in Figure 1.

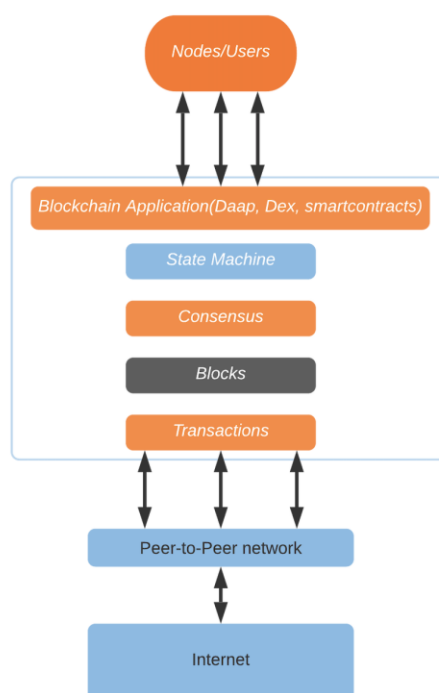


Figure 1. A network view of a blockchain

Blockchain is a new technology that has the potential to transform a wide range of industries. This potential stems mostly from its capacity to provide users with a secure means of transferring value or real assets via the internet. As a result, technology is moving from the internet of information to the internet of value, potentially creating major disruptions to existing financial systems. When any valuable item or transaction was previously documented in a database, people relied on a third party such as a bank, government, or public relations firm to do so. People believe that because banks are regulated by the government, they will not steal their money. People believe that if a bank fails, the government will ensure that their money is safe. Additionally, people trusted the credit card companies and banks to keep their personal details and other information private and safe in their database. Another example of trust in another institution can be people relying on a library to maintain a database with all the information of borrowed books, dates, user information and any overdue books. These institutions maintain their own records and system to store all these information hence, creating a centralized database. In both cases, all this information is maintained by a centralized institution based on the trust in the

system, laws, regulations, and government. Despite these factors being trustworthy, sometimes this trust is still betrayed resulting in the loss of money and possessions. A blockchain-based decentralized database eliminates the need for centralized institutions and database. Transparency and trust are created since everyone on the blockchain can view and validate the transactions.

The concept of blockchain was introduced in 2008 with the invention of bitcoin. Although this technology has the potential to likely revolutionize the whole of society, some describe it as an evolutionary technology and any real benefits from blockchain will take many years to materialize. However, current demand and the increasing popularity of blockchain in various fields point towards the recognition of its disruptive potential by many big organizations. The impact of blockchain technology on present technologies is growing by the day, and it has the potential to fundamentally change them.

## 2.1 Distributed System

Blockchain as a core is a distributed system without any dependency on a central system, hence understanding fundamentals of distributed systems is critical. Distributed systems are a pattern of systems in which two or more computing systems or nodes are correlated and collaborate to work toward a common objective. For an end user of the system, it will be perceived as a platform with a single logic.

In a distributed system where, multiple nodes are operating towards achieving a specific goal, there should be agreement between nodes in the state of distrust. Hence, various algorithms can be used to come into a single agreement or consensus among these nodes, the concept is known as distributed consensus. These algorithms are often called consensus mechanism or consensus protocols.

### 2.1.1 Consensus

Various consensus methods may be used in a distributed system to agree on a proposed value or state, as long as it meets the requirements for integrity, fault tolerance, validity, agreement, and termination. In recent years, new consensus mechanisms have been invented to agree on a state of network some of them are Proof of Work (PoW), Proof of Stake (PoS), Practical or distributed Byzantine Fault tolerant (PBFT) network approach, Delegate Proof of Stake (DPOS), Proof of Authority (PoA), Proof of History (PoH), Proof of Storage (PoSt) to name a few. Some of the most popular consensus mechanisms are discussed below:

PoW: Originally created as a solution of for email spamming problem this type of consensus mechanism requires users in the nodes to expend enough computational resources to solve arbitrary mathematical puzzle to avoid system abuse. As more miners or node users join the network more resources should be spent to solve the puzzle for the transaction to be included in the blockchain, hence making it relatively difficult for hacking and malicious attacks. This consensus mechanism is widely used in cryptocurrency for validating transactions and mining new tokens. Bitcoin runs on this mechanism.

PoS: This consensus mechanism requires a node owner to stake certain number of values or assets example cryptocurrencies to validate transactions. Any malicious activities from the validator will results in dissolving the staked assets. In this mechanism computational power is determined by the number of assets a node owner has staked. It was created as an alternative to the PoW, as PoW requires huge amount of resources and consumption of fuel to provide more computational power. PoS is seen less risky as the miners would be in disadvantage if he aims to attack the network. Cardano, Algorand, Cosmos, Tron, Tezos are some blockchain that uses this consensus mechanism(*Cardano*, n.d.)(*Algorand*, n.d.)(Francisco, 2018; *Tezos*, n.d.).



DPOS: In this mechanism miners with their stakes can delegate other nodes to validate their transaction by voting. It is an innovation to PoS and used by bitshares' blockchain.

PoH: PoH is not exactly a consensus protocol or mechanism, rather it is Solana blockchain's PoS critical component that uses a recursive verifiable delay function to hash incoming events and transactions. Every event has a unique hash and count along this data structure as a function of real time. This information tells us what event had to come before another almost like a cryptographic time stamp giving us a verifiable ordering of events as a function of time(Yakovenko, 2018). Each node gets a cryptographic clock that helps the network agreeing on time and ordering of events without having to wait to hear from other nodes. Blockchain often gets throttle due to various consensus mechanism PoH provides high throughput without sacrificing network security(Yakovenko, 2018).

The nodes in distributed system communicate with each other transmitting or redistributing data. In a network of nodes, an individual node can have its own attributes or components. Some nodes might act as a means of receiving and storing data, some relay information elsewhere rather than inside the network. Overall, nodes are the major centres to route data traffic. Hence, it is essential that these nodes should not act faulty. A malevolent node, such as one that is malfunctioning or irregular, might disrupt the network's operation. These nodes are known as Byzantine nodes.

However, it is often considered that if a node has a problem, it will cease to function entirely. Such nodes can also show erratic operations which are referred as Byzantine behaviour. Hence, a failure model where a faulty node can do a malicious operation is called Byzantinian model. A distributed system should be modelled in such a way that any unexpected behaviour from these nodes should not obstruct the network's operation. Various proposals are made to solve this issue, and research are ongoing actively in search for a perfect distributed system design.

### 2.1.2 Byzantine Generals problem

In 1982, *Lamport et al.* in their paper “The Byzantine General Problem” (Lamport et al., 1982) mention that a distributed system should be able to handle any malicious component decrementing the operation of the system. Abstractly, this situation was expressed as a group of generals of Byzantine army preparing to attack a city. For all generals to agree upon a single plan to attack into the city simultaneously, a messenger can be a medium of communication. However, if one or more generals do not come into an agreement or is found treacherous or even the messenger is intercepted by the enemy, the communication is broken. Hence, there should still be a viable mechanism for at least two thirds of loyal generals to agree upon the plan for a successful attack operation provided the communication is not fully oral. If the communication was supposed to be done orally, more than two thirds of loyal generals are needed as oral message contents can be easily altered.

In a distributed system, nodes, Byzantine nodes, and channel of communication can be considered in analogous to generals, traitors, and messenger respectively as in the situation expressed in the paper. By the invention of Proof of Work (PoW) consensus mechanism in bitcoin, the practical implementation of solving Byzantine General problem was achieved.

### 2.1.3 CAP theorem

In the past, the common option to meet the demand for more data and processing capacity was to either get more powerful machines or optimize the existing code base. Similar jobs can now be done with many machines assigned to accomplish a certain task concurrently, thanks to the evolution of distributed systems.

In 1998, Eric Brewer introduced the CAP theorem also known as Brewer’s theorem. It was presented as a conjecture at the 2000 Symposium on Principles

of Distributed Computing (PODC). Later, in 2002 Seth Gilbert and Nancy Lynch of MIT proved it as theorem (Gilbert, 2002). According to the theorem, simultaneous achievement of Consistency, Availability and Partition Tolerance in a distributed system is not feasible. Consistency states that all nodes in a distributed system see the same data at the same time. Availability in a distributed system requires a 100% operational and accessible system such that every request gets a response on either success or failure. In a distributed system, Partition tolerance assures that failure of multiple nodes should not affect the operation of the system.

Despite of the proven fact that having all three properties in a distributed system simultaneously is not likely, blockchain has attained all these properties differently. This violation of CAP theorem has been successfully implemented in Bitcoin, meaning Bitcoin is able to achieve all these in a period of time rather abruptly. Precisely, Partition tolerance and Availability is achieved through a process of state replication and updating data to thousands of nodes in a peer-to-peer blockchain network. This assures that even if a single node is faulty, the whole network continues to work. However, Consistency is achieved in a period (Bashir, 2017). In this case multiple nodes validate the data to provide a consistent result over time via a process called mining.

## 2.2 Decentralization

The main concept of decentralization is to transfer power and control from a centralized entity like an individual, organization, or a group to a network of distributed entities. Decentralized network aims for limiting the reliance and trust participants must place in one another. It strives to prevent the ability of a single authority to control the organization. Hence, decentralization extends the delegation to various lower levels, rather than only top management levels. As a result, an organization benefits with higher efficiency, faster settlement of transactions, quick decision making, diversification, reduced burden on top

executives, better motivation and more emphasis can be given to the market and the products development.

The concept of decentralization is not new, and it has been in practice in strategy, management, and governance for quite some time now. Centralized, distributed, and decentralized designs are three basic network designs often examined when developing a technological solution. Some examples of centralized design providers are Google, Amazon, Apple, and other major providers. In the designs, these providers are solely in authority to control the operation of the systems, whereas a distributed system design works by spreading the computation and data processing across multiple nodes in the network. Despite multiple nodes involved in the operation of network there is still a central authority governing the process in distributed network. Decentralized network on the other hand disseminates the power, functions, and responsibilities away from the central authority. As an analogy, decentralized system can be compared to collection of small organizations with their own database server which has no central server to report to. Figure 3 illustrates different structure of network design.

Although, decentralized networks are the core service provided by the blockchain technology, a blockchain application cannot simply be classified as such. Decentralization should rather be extended to every component of the blockchain application. Greater and more equitable service can be achieved by decentralizing resource management and access in an application. The benefits of enhanced stability and service levels provided by decentralization outweigh certain drawbacks such as decreased transaction throughput.

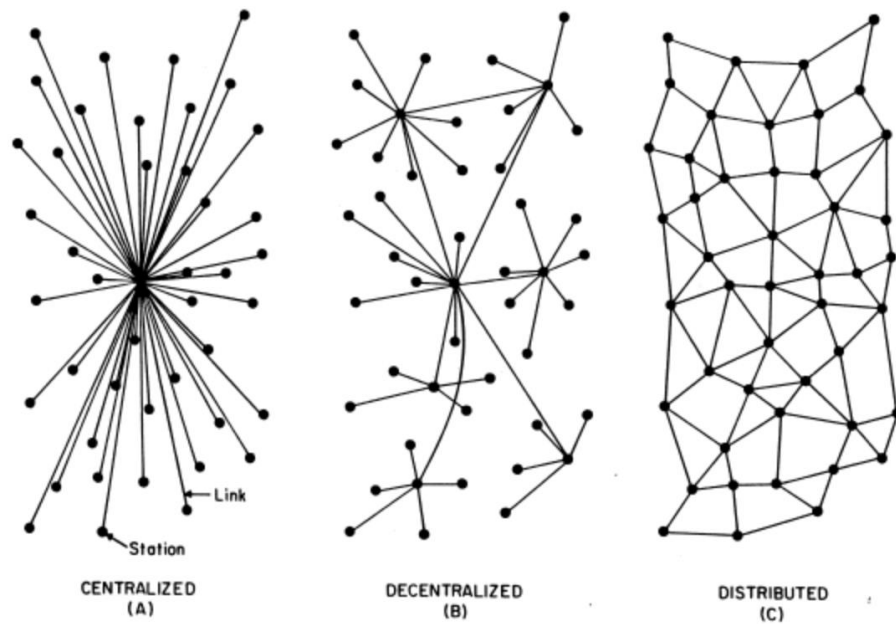


Figure 2. Structure of different networks as appeared in Paul Baran, Introduction to Distributed communication (Baran, 1979).

No matter how distributed or diverse the network system is made, complete decentralization is still unachieved. Blockchain is a distributed ledger that still needs various resources for its operations which are somewhat centralized, such as storage, communication, computations, and identity verifications. The whole environment surrounding the blockchain needs to be decentralized to achieve a complete decentralization. Moreover, wider distributed and decentralized blockchain system solutions are under development and some of them are already in implementation. Examples include Decentralized Autonomous Organization (DAO), Decentralized Finance (DeFi), Decentralized Exchanges (DEXES) and Decentralized Applications (DApp)(Bashir, 2017).

### 2.3 Cryptography

Classically, Cryptography is the art of deciphering codes for secure communications.(Jonathan & Yehuda, 2014). However, modern cryptography encompasses much more than this. Modern cryptography involves using mathematical and scientific techniques to assure integrity, strategies for sharing

secret keys, user authentication protocols, e-voting, digital signature, currencies and more (Jonathan & Yehuda, 2014). Adoption and acceptance of these techniques in various fields is another significant difference between modern and classical cryptography. Military and government entities have always been the primary users of cryptography in the past. However, cryptography is now widely used in almost every field starting with a basic identity verification by inputting a password, online shopping with credit card payment to securing a wide transaction in blockchains.

Cryptography is not a standalone solution for confidentiality but serves as a fundamental unit to build a large security system to remove adversaries. Data privacy (Confidentiality), Data Integrity, Authentication and non-repudiation are some security services provided by the cryptography. Cryptographic evidence can be presented in case of denial to confirm an occurrence of a particular action, this certainty to settle the dispute is called non-repudiation. When it comes to online transactions, digital signatures along with other cryptographic measures can provide non-repudiation to ensure the authenticity of the involved party and settle the dispute if any. Consumers' trust towards the service can be achieved via a network running a non-repudiation protocol. An assurance of accountability can be provided with various audits and logging mechanism to trace the actions of the entity. Additionally, this log file can be further encrypted to provide more security. Some of the cryptographic primitives and processes are discussed below.

### 2.3.1 Symmetric Cryptography

In this type of cryptography, the sender and receiver will have the same cryptographic key which is used to encrypt and decrypt the data. It is also known as shared key cryptography. A secret key is agreed and shared among the entities before the communication or operation begins. Encryption algorithms for this cryptography is comparatively faster as it has fewer complex algorithms and uses less computational power for execution. There are various encryption algorithms and cryptographic checksums also known as ciphers and

Message authentication code (MACs) respectively. These can be used to encrypt and decrypt the data maintaining the integrity (Bashir, 2017).

### 2.3.2 Asymmetric Cryptography

This type of cryptography uses different keys to encrypt and decrypt the data, unlike symmetric cryptography where the user and sender has the same key. It is also known as public key cryptography and uses public keys and private keys to encrypt and decrypt the data, respectively. In this cryptography, an encryption done with a public key can only be decrypted by the private key of the same user. In order to send the data sender needs to encrypt the data using receiver's public key before sending to the network. The receiver on the other hand needs to decrypt the received data with his private key. Hence, each entity has its separate private key and only the public key as the name suggests can be shared in public for communication. Examples of asymmetric encryption include: Rivest Shamir Adleman (RSA), Digital Signature Standard (DSS) which incorporates the Digital Signature Algorithm (DSA), Elliptical Curve Cryptography (ECC), The Diffie-Hellman exchange method and TLS/SSL protocol. Bitcoin uses the Elliptic Curve Digital Signature Algorithm (ECDSA) to digitally sign transaction and validate users (Bashir, 2017).

### 2.3.3 Public keys, Private keys, and Digital Signatures.

Private keys are the secret keys associated to the individual only. Private key needs to be kept safe in such a way that no one other than the user should have access to it. A private key can take any forms for example a 256 character long binary code, 64-digit hexadecimal code, QR code or even a Mnemonic phrase. Generating a public key from private key is possible but it is practically impossible to do the other way. One way to generate private key would be by RSA encryption (Bashir, 2017).

Public key is kept public and easily available and seen by everyone in the network. The public key corresponds to its private key pair. In a communication

between a sender and receiver, sender can encrypt the message using the public key of the receiver. Receiver can then decrypt the message using the private key. The public key that can receive the transaction is usually the address, which is simply the receivers' public key shortened.

Digital signatures are the digital fingerprints which are unique and associated with the specific user. It is a form of encrypted message that securely identifies the signer in the recorded transaction. Digital signatures are used to provide data origin, authentication, and nonrepudiation. When a message needs to be sent via a network, a private key is used by the sender to digitally sign the data. The digital signature is appended to the data and encrypted using the receiver's public key creating an encrypted data envelope. The data can be further timestamped for more security. The receiver can decipher thus received message using the private key.

#### 2.3.4 Hash functions

Typically used to provide data integrity services, hash functions are efficient functions whose length varies from bit string of any length to bit string of fixed length. Its description is assumed to be publicly available. In a typical function example,

$y = H(x)$ ,  $H$  is the hash function where  $y$  is the hash or hash value of  $x$ . Some desired properties of hash function are, it is computationally unfeasible to invert or reverse engineer the value from which the hash was created, and two different input messages should not output the same hash value, property also known as collision resistance (Vojin G. Oklobdzija, 2001). Let's consider a scenario where a hacker is willing to access a large file store in an insecure location whilst the hash of the file is stored in a separate safe location. It is infeasible for the attacker to modify the content of the file without being detected as the modified content of the file will generate a different hash than the one stored (Vojin G. Oklobdzija, 2001). Basically, hash functions deterministically scramble the data, out of fixed length is given with arbitrary input data length and finally irreversible data securing. Some of the hash function categories are



Message Digest (MD), Secure Hash Algorithms (SHAS), RACE Integrity Primitives Evaluation Message Digest (RIPEMD) and Whirlpool.

SHA-256: In this hashing, an input data is shortened by a hashing algorithm in an output that cannot be cracked using modular additions, compression function or bitwise operation. It has block and word size of 512-bits and 32-bits respectively. With an input message of size  $< 2^{64}$ -bits, it gives the output of 256-bit digest. Bitcoin uses SHA-256 hashing alongside with RIPEMD160 (Bashir, 2017).

Keccak-256: A random permutation model with the approach of sponge and squeeze construction is used in this hashing. A fixed length output hash function and fixed length input stream ciphers, in general forms a sponge function, which operates by iteratively applying the inner permutation to itself with every input entry or output retrieval. Analogous to sponge, the input data is first padded then absorbed into the sponge function. An XOR operation changes the output into a subset of permutation state and finally the output is squeezed out from the sponge function representing the transformed state. This hashing method is used by Ethereum blockchain platform.

### 2.3.5 Merkle tree

Ralph Merkle proposed the Merkle tree in his paper (R. Merkle, 1988). A Merkle tree is a data structure that is based on hashes and is an extension of the hash list. Also referred as binary hash tree, it is a tree structure where a hash of a data block is represented by each leaf node and each leaf node further represents the hash of its children nodes. Precisely, each hash of the leaf nodes is concatenated and hashed to give another branch node. These branch nodes are further hashed to give a root hash or root node. The root hash contains the hashes of the entire block. In distributed systems, Merkle trees are used to efficiently verify specific data as they work with hashes rather than whole files. They are mostly used in peer-to-peer networks like Git, Dynamo Database and Blockchain.

## 2.4 A brief History of Blockchain

Blockchain technology was first outlined by the Stuart Haber and W. Scott Stornetta in 1991, proposing a computationally practical procedure to timestamp a digital document which cannot be tampered with (Haber & Scott Stornetta, 1991). But the real-world implementation of the proposal was done only after the invention of bitcoin in 2008 and its launch in 2009. The protocol of bitcoin was built on blockchain. A pseudonymous creator of Bitcoin, Satoshi Nakamoto referred bitcoin as:

“a new electronic cash system that’s fully peer-to-peer, with no trusted third party”(Nakamoto, 2008)

Hence, the idea of electronic cash system has been implemented over the successful application of blockchain, bitcoin. The protocols of e-cash have existed since 1980s which are based on model proposed by David Chaum, an American computer scientist and cryptographer. In e-cash system there were two fundamental issues accountability and anonymity(Bashir, 2017). David Chaum introduces cryptographic operations to solve these issues hence minimising the double spending problem. When a single unit of currency is spent or transacted more than once simultaneously, creating a disparity between the transaction record and available balance, the phenomenon is a Double spending problem. Following years more cryptographically secured algorithms were emerged hence improving the e-cash system.

In 1992 Cynthia Dwork and Moni Naor, introduced an idea of solving computational problems or pricing function to stop email spam. Later, in 1997 Adam Back introduces the usage of hash function with the invention of hashcash(Bashir, 2017). Both approaches required user to show a proof that they have spent reasonable computing resources to compute a hash before sending an e-mail. Spamming thousands of emails requires more processing resources to generate a hashcash, which makes the spamming effort more expensive thus reducing the email spamming problem. B-money was introduced later in the year by Wei Dai, who proposed the idea of solving

computational puzzles to generate money. Integrating the principle of hashcash and b-money, the concept of centralized cryptographic currency was introduced in the year 2005, by Hal Finney. Conclusively, it can be seen how the present blockchain technology pioneer bitcoin was invented combining the principle of distributed system and the idea of e-cash schemes.

## 2.5 The Structure of Blockchain

As the name implies blockchain consists of blocks linked together by a chain, forming a network. Some of the generic elements of blockchain are mentioned below (Bashir, 2017):

- **Addresses:** They are unique identifiers representing senders or recipients' public keys or its derivatives used for a transaction in a blockchain.
- **Transaction:** A fundamental unit of blockchain that represents the transfer of value from different addresses.
- **Block:** A data structure to keep a set of transactions, distributed to all nodes in a network.
- **Peer-to-peer network:** P2P employs the network to put individuals in direct contact with each other without the need of requiring a separate server.
- **A Turing complete programming language:** A virtually universal language that can compute any solution which is done in any other language. In other word, the language used should be able to solve any complications if arise, by itself. Almost all of the programming language at this date is Turing completed.
- **Smart contracts:** Available in limited blockchain these are the programs written in Turing complete programming language containing the business logic to be executed in a blockchain provided a certain condition is met.
- **Virtual machine:** Available in limited blockchain to run a smart contract.

- Nodes: Functional unit of a blockchain that performs mining, validate transaction, and secure the blockchain.

Blockchains are composed of three core parts:

**Block:** The size of the block depends on the design and type of blockchain used. A logically organized selected transactions bundled together made up a block. A whole structure of a single block comprises of block header and block body. Block header additional contains software version number, hash of the block, previous block hash, time stamp, nonce, root hash of Merkle tree and other attributes. Whereas block body contains the transactions that are confirmed with the block. The first block of the blockchain which does not contain hash of the previous block and was created when the blockchain was started is a genesis block.

**Chain:** A cryptographic hash that connects a block to the previous block. A hash is created from the data in the preceding block and locks individual block in order and time.

**Network:** A way for nodes to communicate with each other by agreeing upon a state of a transaction. Each nodes maintain their own copy of transactions ever recorded in the blockchain. These nodes can be operated by anyone and are located all over the world. However, operating a full node is resource heavy and time consuming. Hence, node operators are incentivized in the form of example cryptocurrencies for their operation.

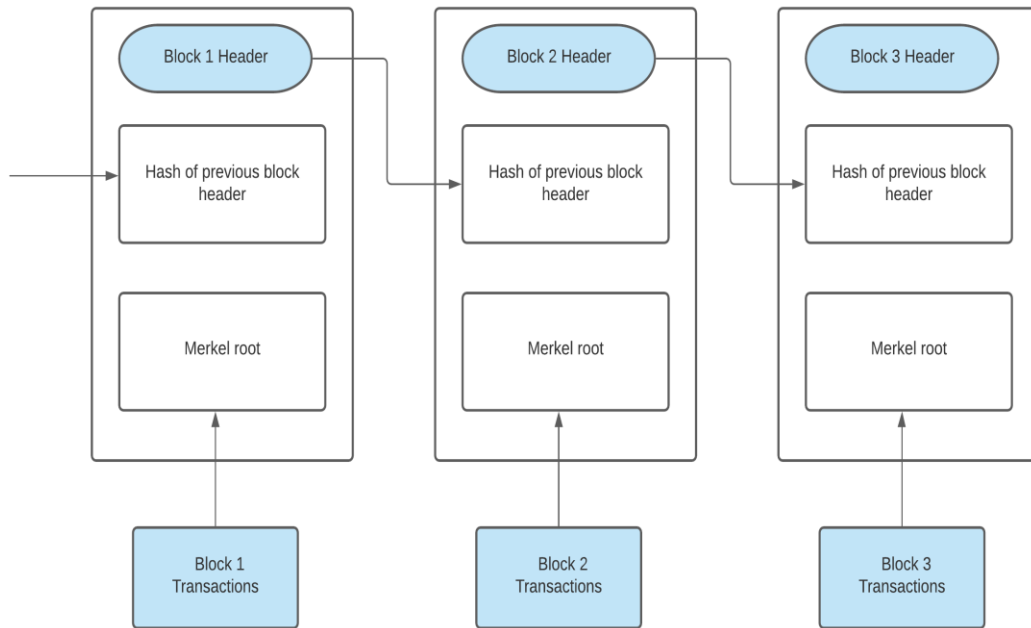


Figure 3. Structure of a Blockchain

Figure 3 illustrates a structure of blockchain as a connected list of records. From data structure perspective blockchain has pointers that keep the records of another variable location and linked lists which is a sequence of block that links to following block with pointers and contains specific data.

## 2.6 Types of Blockchain

Different features of blockchain like distributed ledger, immutability, faster settlements, consensus mechanisms and platform for smart contracts provide much to blockchain technology than just a network to store data. All these features have been evolved continuously over the years resulting in various blockchains where each one of them serves its purpose and solve a particular set of problems.

**Public Blockchains:** These blockchains are publicly available and anyone can participate in the process of decision making. The ledgers in the blockchain are decentralized, meaning they are not controlled by a single authority and can be

accessed by any user. Every user on the blockchain has a local node copy of the ledger and decision are made using a distributed consensus mechanism. Transaction in this blockchain is transparent but anonymous to everyone. These types of blockchain lack transaction speed and suffers scalability issue with addition of more nodes. These kind of blockchain are good for fundraising and voting. Example bitcoin. However, recent development in the lightning network takes the transaction off-chain hence making the bitcoin network faster and scalable(Poon & Dryja, 2016).

**Private blockchains:** These blockchains are private and a single organization has authority over the network. These blockchains are power efficient, provide faster output and provides privacy. Data handling is simple but is not accessible to everyone. As the centralized nodes make the final call, the blockchain is not trust less. If a single or multiple nodes go rogue, it compromises the consensus mechanism thus utilized by the network. These blockchain can be used in supply chain management, asset ownership and internal voting fields. Multichain, Hyperledger fabric, Hyperledger Sawtooth, Corda are some examples of private blockchains (Bashir, 2017).(Iredale Gwyneth, n.d.)

**Federated blockchain:** The operation of this blockchain is influenced by multiple organization. These blockchains are highly scalable and extremely fast. The privacy of the blockchain is preserved by the network regulations. Also known as consortium blockchain, some aspects of the organization are made public while some are kept private. The pre-set nodes in a consortium blockchain regulate the consensus operations. Furthermore, despite the fact that it is not open to the public, it retains a decentralized nature. Since, this blockchain is administered by multiple organizations, there is not a single concentrated force at work here. It has validator node that can validate transaction as well as initiate or receive transactions. The member node, on the other hand can accept and initiate transactions. In essence, it has all the benefits of a private blockchain, such as transparency, privacy, and efficiency, but without the power concentration to a single party. These blockchains can be used in banking and payment, research, and food tracking sectors. Some of the examples of this

blockchain are Marco Polo, Energy Web Foundation, and IBM Food trust (Iredale Gwyneth, n.d.).

Hybrid blockchains: These blockchains have some similarities with the consortium blockchain. When an organization neither wants to deploy a private blockchain nor is it interested in public blockchain, but simply want the best blockchain. Users will not get any incentives for participating and contributing to the network. These blockchain are not quite transparent, however rules can be changed as according to the needs. These blockchains can be used for real state, retail, and highly regulated market. Dragonchain, XinFin's hybrid blockchain are some examples of hybrid blockchain (Iredale Gwyneth, n.d.).

Apart of these four different major types of blockchains there are some other blockchain worth mentioning. Sidechains blockchains are pegged from a main blockchain to move the cryptocurrencies from main chain to pegged chain and vice versa. One type of sidechain blockchain is used by a recent Ethereum upgrade EIP-1559 which burns the coin from the base fees during a transaction. Shared ledger, Permissioned ledger, tokenized blockchains are some other blockchains.

### **3 Applications of Blockchain technology**

#### **3.1 Hyperledger Fabric**

In the year 2015, blockchain was coming into rise and people had started accepting it as a new type of shared database or distributed ledger, different institutions interested in blockchain technology came together with the aim of achieving more utilizing this technology. Utilizing the combine pool of resources, these organizations intend to make the blockchain technology open source for everyone to use with a vision to make blockchain a modular industry standard technology (Aggarwal & Kumar, 2021). Hyperledger fabric is a private blockchain that can be customized due to its modular design, so that different business can utilize different functionalities suitable to their design and needs.

Due to this reason, fabric is most primarily used to make seamless and efficient transaction between businesses in enterprise settings. Fabric keeps the state change record of the transactions distinguishing them as a collection of key value pairs, known as assets. Hyperledger fabric provides a software called chaincode. With this entity involve in the transactions can define and modify the assets. In other word, chaincode is a smart contract containing the business logic that can be deployed in the fabric.

Within the Hyperledger, there are two node types to increase efficiency and scalability. They are peer nodes and ordering nodes. Peer nodes are the validating nodes to verify the transactions whereas ordering nodes are responsible for transaction operation and maintaining the correct history of events in the network. A single true record of transaction is created simultaneously while order nodes work on transaction execution and peer nodes validates the network consensus protocols. There are two components of Hyperledger, blockchain log and a state database. State database maintains the current state of the blockchain whilst immutable sequence record of transactions is stored by blockchain log. A member in the network can query the stored log to track the assets status, origin and more. However, Hyperledger fabric also provides transaction privacy and confidentiality to a specific subset of its network members for keeping the transaction invisible and inaccessible from unauthorized parties. Hyperledger fabric had been adopted by major cloud service providers like Amazon, Ali Baba, Microsoft azure, Tencent, and Oracle (Iredale Gwyneth, n.d.).

### 3.2 Bitcoin

In 2008, a paper was written in a peer-to-peer electronic cash system to transfer payments between peers without the involvement of any third party or bank. The name of the paper was Bitcoin: A Peer-to-Peer Electronic Cash System written by a mysterious and pseudonymous Satoshi Nakamoto (Nakamoto, 2008). After decades of research on cryptography, electronic cash system and hash functions and exercising variety of ideas like b-money, hashcash, BitGold



and many more, a decentralized digital currency, Bitcoin was created in the year 2009. Bitcoin offers a promise to replace the traditional online payment system, with lower transactions cost and without the need of financial intermediaries operating in a decentralized manner. Bitcoin addresses the key issue that existed in traditional transaction systems like Byzantine Generals problem and provide a practical solution to the double spending problem. Bitcoin is a form of digital currencies which uses cryptography hence it is a cryptocurrency. Since bitcoin is digital, every bitcoin transaction is recorded in the public ledger and kept in the blockchain where everyone in the network has access to. Every such transactions are verified by node operators or validators by a process called mining. Bitcoin is not issued by a bank nor is there any central authority controlling it, hence there exist some controversy regarding regulating the digital currency. At the time of writing this content, Bitcoin has been accepted as a legal tender in EL Salvador and has reached its all-time high price of 66930 US dollar. The nodes in the bitcoin network communicate with each other using the Bitcoin P2P protocol.

### 3.2.1 Mining

In a simple way, mining is similar to lottery process to create a new block in the Bitcoin blockchain and get incentives for doing so. Bitcoin mining is the process of using sophisticated computers to verify the legitimacy of Bitcoin transactions and to enter new bitcoins into circulation. Till date 18.78 million bitcoin has been mined. There are two conditions a miner should meet to earn bitcoin through mining. Firstly, they need to verify 1 megabyte worth of Bitcoin transactions also known as a block. This is done by using a complex computational operation to decipher out a proof response for the challenge created by bitcoin software and come up with a 64-digit hexadecimal code or hash. The difficulty level of the challenge increase with more blocks added to the blockchain and more miners into the network. Secondly, a miner must be the first one to match the hash or at least come closest to it in order to get paid (Tovanich et al., 2021).

At this point PoW protocol is used to verify; the user has spent significant number of computational resources solving the challenge to deter fraudulent Bitcoin sales or transactions. The process uses Graphic Processing Units (GPUs) and Application Specific Integrated Circuits (ASICs) for such intensive computation. As a compensation for the efforts from the miners solving the challenge, they are paid in the form of block reward with bitcoin. However, the block reward is halved every 210,000 blocks or roughly four years which reduces the rate of new coins creation. This will lower the supply and hence increasing the demand and raising the price of the bitcoin. Additionally, the difficulty level of the block is also adjusted every 2106 blocks with the goal to keep the mining rate constant. Since, there will be more need of resources with increasing difficulty level some miners forms groups to mine together and share the incentives. The group is known as mining pools (Tovanich et al., 2021).

According to Bitcoin protocol, the number of bitcoins will be capped at 21 million but with the mining rate being reduced overtime due to the complexity of problem and requirement of heavy resources it is likely the final Bitcoin will not be circulated for some time (Antonopoulos, 2017).

### 3.2.2 UTXO

Unspent Transaction output (UTXO) is an output of an unspent transaction, that can be spent into a new transaction as a new input. In a bitcoin network, a transaction is formed when a sender sends some bitcoins from a 26-35 characters long letters and numbers combined address to a receiver's address. Numerous processes are involved during a transaction lifecycle. When a transaction is sent, wallet or software that holds the coin signs it with sender's private key. This transaction is then broadcasted into the Bitcoin network, where it is included into the next block to be mined. After solving the PoW problem the block goes through the verification process to get six confirmations before adding to the blockchain. UTXO can be referred to as a change one gets after a cash transaction. Any changes from the bitcoin transactions are stored in a UTXO database. The starting state of this database or ledger is zero. As the

number of transactions grows, the database fills up with change entries from numerous transactions. Any remaining output after the transaction is completed are deposited back to the database and are used as the input for the next transaction. Bitcoin transactions are unique and can be conducted in fractions of cryptocurrencies. Which means spending a cryptocurrency can take place using a single data byte or in multiple of fractions. However, due to the very reason certain transactions are uneconomical, due to abundance of small coins within bitcoin's network. This is since bitcoin transaction sometimes cost more than actual cost of the thing being purchased with bitcoin. These fees are called transaction fees used as an incentive for miners to include the transaction into the block they are mining (Tovanich et al., 2021).

### 3.2.3 Blockchain structure

A distributed public ledger on the bitcoin network chronologically ordered containing an immutable list of all transactions is the bitcoin blockchain. It contains blocks of transaction linked with previous blocks with its reference hash. Each bitcoin block is limited to one megabyte size and generated every ten minutes. Following tables show the structure and describe the content of a block.

Table 1. Structure of a block

Bytes	Name	Description
80	Block Header	Contains information about the block metadata
variable	Transaction counter	Contains the total number of transactions in the block.
variable	Transactions	Contains all the transactions in the block.

Structure of the block header and its content descriptions are shown in Table 2.

Table 2. Structure of a block header

Bytes	Name	Description
4	Version	Version number of the block to follow block validation rule.
32	Hash of previous block header	Double SHA256 hash of the previous block's header.
32	Merkle root hash	Double SHA256 hash of the Merkle tree of the transactions in the current block.
4	Timestamp	Contains the approximate block creation time in Unix epoch time format. Mining starting time of the header from miner's perspective.
4	Difficulty target	Difficulty target to solve the current block.
4	Nonce	Arbitrary numbers changed by miners to append with the Merkle root hash, producing a hash which then fulfils with the difficulty target.

For the most part, miners try to get a hash equal to or less than the difficulty target hash with different value of nonce in combination with Merkle root hash. If the hash matches or is less than the difficulty target, the miners solve the block. The block with its own header hash is then broadcasted into the network for validation and therefore added to the blockchain. On the other hand, if the hash does not match the difficulty target hash, then another combination of nonce and Merkle has is tried and the process is repeated until succeeded.

### 3.2.4 Bitcoin limitations

Although bitcoin is considered as the baseline protocol of innovation in the blockchain technology, it has some limitations:

**Built in limitations:** There are a lot of hard coded constants that are implemented into the Bitcoin protocol which were chosen when Bitcoin was proposed in 2009. The main aspect of Bitcoin that is a limitation is throughput of the bitcoin system. This restraint comes from the hardcoded limit on the size of blocks. Each block is limited to a million bytes and each transaction must be at least 250 bytes, if block is generated every 10 minutes, then the total number of transactions is about 7 transactions per second which is all that the Bitcoin network can handle (Antonopoulos, 2017).

In a long term, fixed hash algorithms used in bitcoin can be broken by attackers. **Mining harms the environment:** Bitcoin mining demands a lot of recourses in the form of gas, electricity, and oil. The power required to mine bitcoin is more than the electricity consumed by some of the country. The estimated total yearly power consumption by bitcoin network for 2021 is 116 Terawatt Hour (TWh) (<https://ccaf.io/cbeci/index>) which surpasses the annual power requirement of countries like the UAE, the Netherlands, the Philippines, Belgium, Austria or Israel. Continuous mining has already created energy shortage and crisis in most part of the world. On 24<sup>th</sup> of September 2021 China banned the mining of bitcoin in the verse of energy crisis it created. According to statistics from the Cambridge Bitcoin Electricity Consumption Index ([https://ccaf.io/cbeci/mining\\_map](https://ccaf.io/cbeci/mining_map)), the United States currently has the largest percentage of worldwide bitcoin network hashrate (35.4%), followed by Kazakhstan (18.1%) and Russia (11%) as of the end of August. **Privacy and anonymity:** Even though, the main idea of bitcoin was to provide pseudonymity, provided enough time and transaction, the identity of the individual can be tracked eventually. Some recent cases where government officials were able to track the virtual wallet of a criminal group, highlights on these possibilities.

Not a currency replacement: Cryptocurrencies are considered as speculative assets. Transactions in bitcoin network are comparatively slower than existing payment system like VISA and PayPal. Additionally, transaction fees are also high when there is congestion in the network. Due to these reasons, bitcoin has not able to provide efficient medium for exchange.

Despite all the limitations, Bitcoin network has been leading the market for over a decade now proving its stability. Due to its dedicated amount of computing power for security and its underlying technological strength, bitcoin to this date still holds the top position in the cryptocurrency market. More miners are joining the bitcoin network making it much difficult to mine and hack.

### 3.3 Ethereum

In November 2013, a Canadian-Russian programmer name Vitalik Buterin put forward the concept of Ethereum, and later in the year 2015 Ethereum was launched. Basically, the proposal was to add the concept of smart contracts written in a Turing-complete language for the decentralized application in existing blockchain technology. Ethereum was invented with the vision to create an alternative architecture that offers even more simplicity of development and stronger light client qualities, meanwhile providing blockchain security to all applications sharing a common economic environment (Bellaj Badr, 2018).

Ethereum provides a platform for decentralized applications to efficiently communicate in a secure and rapid developing environment. Ethereum provides a build-in Turing complete programming language called Solidity, allowing user to write smart contracts with user defined transaction functionalities for decentralized applications with the full ownership using the Ethereum network associated cryptocurrency Ether.

### 3.3.1 Messages and Transactions

An externally owned account signed a data package containing a message with the private keys to send to another external account or to store in the Ethereum network. This data package is referred as the transactions.

Messages are the objects existing only in the Ethereum execution environment and are send from one contract to another. Unlike transactions they are not send by external accounts rather produced and executed during an operation of a contract call. Tables 3 and 4 show different fields of a transaction and message respectively:

Table 3. Transaction in an Ethereum environment

sender address:	Maximum of 42-character identifier of the sender
receiver address:	Identifier of the receiver can be external receiver or a smart contract
value	Amount of ether transferred, a unit wei is used $1 \text{ ETH} = 10^{18} \text{ wei}$ .
data (optional)	Contains contract related activities for execution or deployment of contract. Empty data field represents a payment transaction.
nonce	Sequence number of transactions incremented by one for over each transaction.
gasLimit	Limit of amount in ether, a sender is willing to pay for the transaction.
gasPrice	It is the amount of gwei user pay for each gas. Specified by a unit gwei, $1 \text{ gwei} = 10^{-9} \text{ ETH}$ .
signature	Signed hash with the sender's private key

Table 4. Message in an Ethereum environment

sender (implicit)
receiver
data (optional)
value
gasLimit

### 3.3.2 Ethereum Accounts

The state in Bitcoin network can be compared to as the distributed ledger, and a new transaction can be a state transition function. When the state is combined with the new the function, a new state is generated which can be compared as the new updated ledger.

In Ethereum, however states are called “accounts”. These are the objects with 20-byte address and transaction between these accounts are called state transitions. There are four fields in an ether account: storage, nonce or account transaction counter, current ether balance and business logic or contract code (Buterin, 2015) if there is any. An external account managed by a private key and a code-controlled contract account are two different Ethereum accounts. Since, there is no code controlling the external account, it can simply create and sign a transaction and send it to another external account or to the contract account. The contract account when receives the transaction or message, triggers the code to either read and write to the internal storage or transfer the transaction to another external account depending upon the business logic of the contract code. The contract code is an autonomous agent and activates when conditions are met. Figure 4 shows the schematic representation of the process.



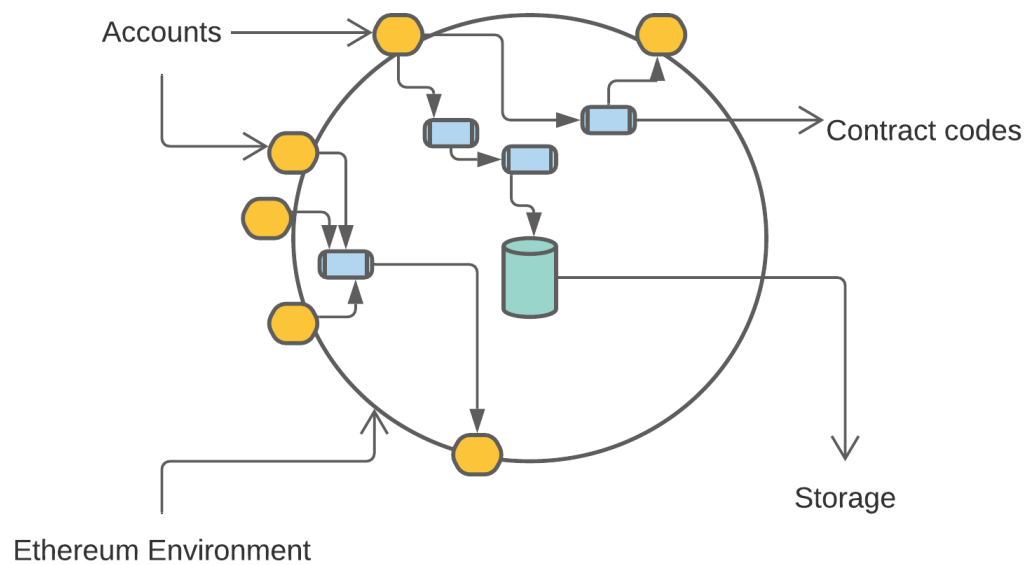


Figure 4. Process in an Ethereum environment

### 3.3.3 Ethereum state and transitions

The process of state transitions in Ethereum can be defined as follows(Buterin, 2015):

- Firstly, checks whether the transaction has right value then the authenticity of the signature is checked and compare the nonce with the nonce of the sender's account. Failure in any of these validation process will result in an error.
- Transaction fee is calculated as  $\text{gasLimit} * \text{gasPrice}$
- The sender's address is determined from the signature.
- Sender's account balance is deducted with the fee and sender's nonce is increased by one. An error is returned if the sender does not have enough balance.
- The transaction value is transferred to receivers' account from sender's account after gasLimit is initialized to deduct certain amount of gas per byte for payment to the byte in the transaction.

- A receiver account is created if there is none. If the receiving address is the contract account, the contract code is executed till its completed or the operation has no gas left.
- All the state changes will be reverted, if the execution failed or if there is no gas left for execution. However, the payment of the fees will go to the miner's account. The fee for all remaining gas is refunded to the sender.

### 3.3.4 Ethereum blockchain

Structurally, Ethereum and Bitcoin have quite resemblance in their blockchain architecture. However, Ethereum differs in some respects. Unlike bitcoin, an Ethereum block also contains the copy of recent transaction state, list of the transactions, block number and difficulty. Basic block validation process in Ethereum includes, checking if a valid reference of previous block exists, checking whether the current block should have greater timestamp than previous branch and 15 minutes less into the future, verifying the validity of block number, difficulty, transaction root, uncle root (two blocks verified at the same time) and gas limit and checking the validity of PoW on the block. After these processes, the state of the block is calculated based on the previous block end state and transaction list. If there is enough gas to pay the transaction fee for the miners, it is added too. And finally, the Merkle tree root hash of the final state is compared to the block header final state root. If it is equal, the block is considered as valid or else rejected (Buterin, 2015).

### 3.3.5 Ethereum stacks

There are some core components of Ethereum that offers a conceptual model for the interaction of a software application with the Ethereum blockchain.

Ethereum can be integrated into a software application implementing some of these layers:

Ethereum virtual machine (EVM): These are the runtime environment to run the smart contracts. All the transactions processing in the Ethereum network is handled by EVM.

**Smart Contracts:** These are programs written in Solidity programming language that compiles to EVM byte code. Custom functionality can be added into the smart contracts to be added into Ethereum blockchain. A smart contract can create various tokens by spending some Ethers. For the interoperability of various tokens created by different smart contracts a standard need to be met. ERC-20 (Ethereum Request for Comment 20) is a guideline or standard to create a token (Gates, 2017).

**Ethereum nodes:** It is a unit of Ethereum network that runs a piece of Ethereum client. These nodes help to verify the transactions in each block, keep the network secure and provide accurate data.

**Ethereum client:** It is an application necessary to run a copy of Ethereum node. It allows an application to connect to Ethereum network, synchronizes with the current blockchain state, create, and manage new addresses and submit new transaction to the blockchain. Geth (Golang ETH), Parity and Besu are some Ethereum clients.

**End user application:** These are the primary web and mobile applications for user-interaction build on the top layer of the Ethereum stack.

### 3.4 Smart Contracts

Although, the concept of smart contracts is not new, the advent of blockchain technology has given a rise to smart contracts and various blockchain project utilizing it. Implementation of smart contracts will provide cost effective transaction and simplified agreement terms among the business and its users. A weak concept of smart contract was implemented in the bitcoin blockchain as well in the form of stack-based script programming language, where certain data must be provided satisfying the script for a transaction spending a UTXO. A smart contract is a computer program written in a certain programming language containing business logic representing some agreements between parties. When the condition mentioned in the agreements are met these

contracts are automatically executed. A true smart contract is self-enforced without the involvement of the third party (Gates, 2017).

These contracts can be found across the distributed, decentralized blockchain network and regulates the execution of trackable and permanent transactions. Smart contracts eliminate the requirement for a centralized organization, law, or any third-party enforcement to carry out trustworthy transactions and agreements between distant, anonymous parties. For the satisfaction of the parties involved in the transactions, many stipulations can be added in the smart contracts as per the need. These smart contracts are created with an approach to make it secure and executable even in the presence of adversaries. The internal state of the smart contracts is managed by state machine architecture which enables for the creation of a frameworks for writing smart contracts, where the state of the contract advanced depending on a set of predefined criteria and circumstances. A smart contract produces the same output despite of any node on a network it is deployed into, hence creating a distributed consensus in a blockchain. Various alternative blockchain supports smart contracts as it has multiple real world use cases like real-state deal, lending and burrowing, insurance, supply chain management, ensuring authenticity of copyrighted products, and providing digital identity. Some of the benefits of smart contracts are they are autonomous, secured, cost-effective, trustless, fast performing and compatible.

### 3.5 Alternative Blockchain and Alternative Coins

Various alternative blockchains have emerged along the time leveraging the underlying technology and concept of the blockchain. These new alternative blockchains addresses existing limitations and restrictions in the current blockchain technology and add improvement upon it either by adding new tools to it or adding another layer on top of it for different operations.

Alternative coins or alt coins are the term given to define any other cryptocurrencies from different blockchain platforms other than bitcoin. They are simply digital tokens representing tangible or intangible assets to be used on a

blockchain application. Some of the alternative blockchains and coins are mentioned below (Antonopoulos, 2017).

- Cardano (ADA): It is created using evidence-based methodologies and peer reviewed research and works on PoS protocol. ADA is the alternative coin for Cardano blockchain (*Cardano*, n.d.).
- Ripple (XRP): Ripple net, the global network platform for ripple cryptocurrency is the decentralized network that brings together the payment organization for digital payment network. XRP(*Cbdc-Whitepaper-2020*, n.d.)
- NEO: It is a public community funded blockchain network which aims to automate the digital assets management (*NEO*, n.d.).
- Ethereum Classic (ETC): It is an open source Ethereum blockchain based cryptocurrency, which was formed after the main Ethereum blockchain was hacked in June 2016 (*Ethereum Classic*, n.d.). After the hack this blockchain continued working on the same versions however, Ethereum roll back their version to the timeline before it was hacked.
- Polkadot (DOT): This is a multi-block chain application environment for cross-chain communication between blockchains. It connects various blockchain forming a parachain network (*Polkadot*, n.d.).
- Stellar (XLM): It is an open blockchain network which facilitates large transactions between financial institutions (*Stellar*, n.d.).
- Chainlink (LINK): They are the decentralized data feeding network which provide the communication between real world and block chain network(*ChainLink*, n.d.). Chainlink uses a middleware software or oracle for this purpose.
- Tether (USDT): These are the coins representing the actual value of the real-world currency (fiat), US dollar(*Tethar*, n.d.).

Solana (SOL), MATIC, Binance (BNB), Litecoin (LTC) and Dogecoin (DOGE).

## 4 Development Environment

This section discusses on the tools and environmental setup requirements for both front-end and back-end development of the staking applications to be deployed in the Ethereum blockchain.

### 4.1 Environment and tools

This subsection describes about environment and tools used for the application development of this project.

- **JavaScript (JS):** JavaScript is a first-class programming language that is lightweight, interpreted, or just-in-time compiled. In addition to best known as a Webpage scripting language, it is also used in variety of non-browser settings(*JavaScript*, n.d.). European Computer Manufacturers Association (ECMA)scripts version six (ES6) JavaScript is used for the project development.
- **Node.js:** It is a cross-platform and open-source JavaScript runtime environment. Outside of the browser, Node.js operates the V8 JavaScript engine, which is at the core of Google Chrome. Some of the many benefits of Node.js are, as a developer one can write both client-side code and server-side code using JavaScript, with a single server Node.js handles multiple concurrency connections removing the need of managing thread concurrency by a developer, with Node.js a user is free to choose between different ECMAScripts for the development environment(*Nodejs.Dev*, n.d.) and much more. Node version 10.16.3 was used during the development of this project.
- **Node Version Manager (NVM):** It is a tool that can be used to install and use different versions of Node.js(*Nvm*, n.d.). NVM version v0.35.1 was used in the project.
- **Node Package Manager (NPM):** it is a command line tool to manage different Node.js packages(*Npmjs*, n.d.). NPM version 6.9.0 was used during the project development.

- Node Package Execute (NPX): It is a command line tool to run and execute node packages without needing to install them in the system (*Npx*, n.d.).
- Babel: It is a toolchain used to convert the ES6 version code into backward compatible versions, so as to run it in different versions and types of browsers(*Babeljs.io*, n.d.).
- React.js: It is a JavaScript library for creating User Interfaces (UI)(*Reactjs.Org*, n.d.).
- Mocha: It is a JavaScript test framework that runs on Node.js and can be run in the browser as well. This test framework was used to test a smart contract during development(*Mochajs.Org*, n.d.).
- Chai: It is an assertion library used in pair with Mocha(*Chaijs.Com*, n.d.).
- Visual Studio Code: VS code editor was used for all development and testing purpose of the project.

## 4.2 Ethereum Development

This section discusses on different tools, frameworks and software that was used for the development of the staking decentralized Ethereum application.

### 4.2.1 Frameworks and tools

Frameworks are software or applications that assist in the development of other software applications. There are already existing out of the box functionalities in the Ethereum frameworks. These frameworks provide various functionalities and features like local testing and execution of blockchain instance, tools for compiling and testing smart contracts, add-ons for client development, settings for local Ethereum environment or for deploying into the Ethereum mainnet, and integrations with storage solutions such as IPFS for decentralized app delivery. Some of the frameworks and tools used in this project are (Infante, 2019):

- Truffle: This framework uses EVM to provide a development environment for testing, asset pipeline for Ethereum blockchain and other tools(*Truffle Suite*, n.d.).

- Remix: It is an Integrated Development Environment (IDE) for developing and deploying smart contracts in the Ethereum blockchain(*Remix-IDE*, n.d.). For auditing and analysing smart contracts' code an online remix IDE was used during smart contract manual testing phase.
- Test network: Various test networks are available in the Ethereum network to deploy and test the smart contract before its actual deployment in the Ethereum main net. Rinkeby Test Network, Ropsten Test Network and Kovan Test Network are some examples of test networks. For testing with Remix IDE both remix inbuilt JavaScript VM test Network and External Rinkeby Test Network was used. And later in the development phase local Ganache test Network was used.
- Ganache: A blockchain for Ethereum development that can be installed locally and use to deploy contracts, develop, and test applications. Both Ganache cli version as well as Ganache UI versions were using do to the compile and migrate the smart contracts using different ports value(*Ganache-UI*, n.d.).
- Wallet: In this project a digital wallet, Metamask is used for storing and transacting Ether for testing(*Metamask*, n.d.).
- Etherscan: It is an Ethereum network blockchain explorer which allows user to search through various on-chain data like transactions, wallet address, blocks, and smart contracts.
- Web3.js: It is a set of JavaScript libraries to use HTTP, IPC or WebSocket to communicate with Ethereum network.

#### 4.2.2 Solidity

Solidity programming language was used to develop smart contracts used in this project. Solidity is a high-level object-oriented scripting language for implementing smart contracts(*Solidity*, n.d.). Python, JavaScript, and C++ have influenced this language and it was created with the EVM in mind. It is statically typed and does not verify and enforces constraints at run-time, rather it does that during compilation. Originally, this language was invented by Dr. Gavin Wood, former Chief Technical Officer (CTO) of Ethereum, author of Ethereum



Yellow paper and founder of Polkadot, a next generation blockchain protocol. Later, Dr. Christian Reitwiessner and his team in Ethereum actually created it (*Solidity*, n.d.).

In general, a contract definition is written using solidity programming language, this contract is then feed into a solidity compiler. This compiler gives two different files, a file containing bytecode and an ABI (Application Binary Interface). The bytecode is the actual data that gets deployed into the Ethereum network on the other hand ABI is the interface to encode and decode data between two program modules(*Application Binary Interface (ABI)*, n.d.). In simpler term ABI, translate the bytecode to be understandable by JavaScript code. Solidity version v0.5.1 was used for the development of the project.

#### 4.2.3 Web3

With the introduction of blockchain technology, communications within this technology among individuals has become decentralized without the involvement of any third-party organizations. Compare to the previous generations of internet services like read-only (Web 1.0) and centre authority controlled (Web 2.0), Web 3.0 focuses on providing a data-driven and Semantic Web employing an understanding that is machine based. Web 3.0 aims to provide an intelligent, autonomous, connected, and open internet. Decentralized protocols provided by blockchain technology underlie a platform for Web 3.0. In the context of Ethereum, these services are provided by decentralized applications, where the participation of users are voluntary, permissionless and non-monetized. However, accessibility of Web3 through integration into existing technology, scalability and faster operation and not cost effective are some limitations it holds(Mayukh Mukhopadhyay, 2018).

## 5 Application design and development

This section gives some details on the how the concept of decentralization can be achieved into an application by implementing a simple smart contract for a

staking application into an Ethereum network. The setup and development processes were all done in MacOS hence the dependencies of various tools may vary according to different OS.

### 5.1 Minimum Viable Product Requirement

The application to be built aims to provide a user friendly, secure and trustless application which can provide a good reward percentage to the user on the basis of their staked amount. As according to some users' feedback during the project's grooming phase, these are some expected minimum requirements for the application to be viable:

- The user should be able to connect their digital wallet to the applications.
- As a user one can stake the amount of token as per their wish.
- All the information of the staked amount, remaining balance and rewards should be clearly visible.
- The user should be able to see different options available for their staking.
- All the transactions performed by the user and any contracts invoked due to the transaction should be clearly visible.
- Transaction costs for any kinds of transfer should be clearly visible, so that user can decide either to complete the transaction or not.
- The user should be able to unstake tokens.

### 5.2 Dropped features

Some of the following features were dropped in the applications giving priority to the must have features. Some of the features not included in the current application however is planned to be release with future development are:

- Fully feasible production ready web based mobile application.

- Detail calculation of Annual Percentage Yield, as solidity does not support decimal, JavaScript side implementation is most viable in future releases.
- Although decentralized applications are not controlled by any central authority, one should abide by regulation of a region. Hence, future implementation should include Know Your Customer (KYC), to avoid fraudulent behaviours if any.
- Deployment in the Ethereum main net.

### 5.3 Application design skeleton

The basic flow of the application starts with the owner starting the process of compilation and deploying all the contracts into the local test network, Ganache. A user with a meta mask wallet opens the application in the web browser and interact with the elements. As soon as the application is opened, metamask will inject web3 instances into the browser to interact with the contracts in the blockchain. User will be prompted to interact with contracts and connect the wallet. For this project and testing in local test network, the user is provided with one hundred metro tokens to stake. User can select between different available token pools to stake metro coins. After entering a staking pool, user can stake desired amount into the input field. This will open the metamask wallet to pay some gas fees in the form of test ether. The amount of metro tokens will be staked into the stake pool address in the blockchain. These transactions can be monitored from the Ganache UI. Figure 5 shows the simple application flow diagram from client side to the back end through the Ethereum network.

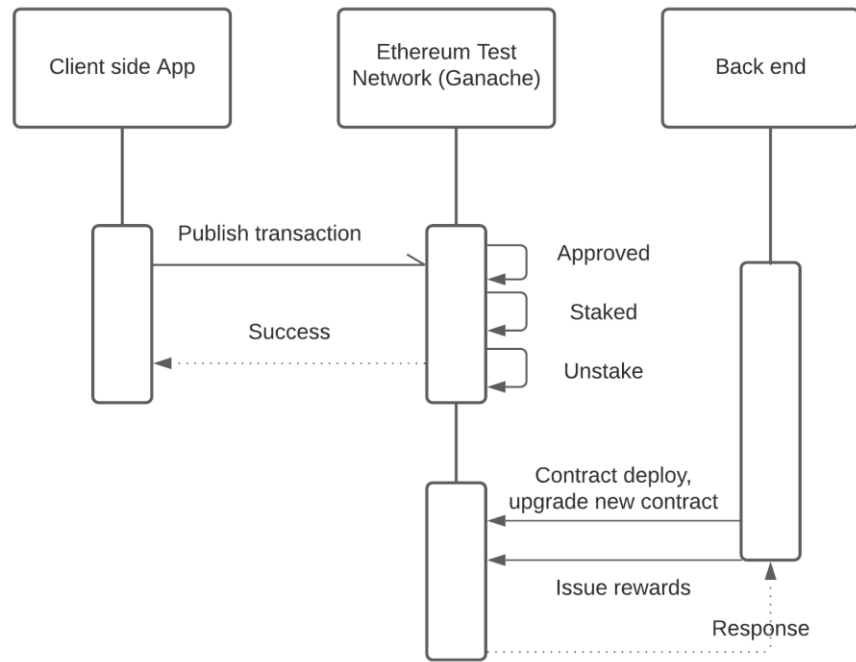


Figure 5. Application flow diagram

For this thesis project a simple reward calculation is done, however automatic complex yield calculation can be implemented in the future releases. The owner issues the reward by running a script. These rewards can be seen in the client-side application. The user can unstake the tokens whenever required.

#### 5.4 Metamask wallet setup

A Metamask chrome browser extension was installed and pinned to the browser. An Ethereum account was created, and 12 words mnemonics were saved safe for future use. When inside the wallet multiple accounts were created for the purpose of creating transaction between them. Figure 6 shows different Metamask accounts and Ethereum test network inside the wallet.

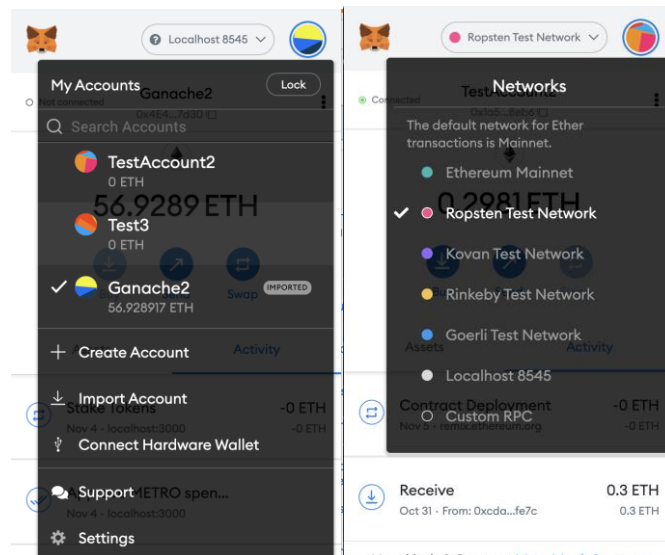


Figure 6. Metamask wallet account and Ethereum Test Networks

### 5.5 Creating and testing smart contracts in Remix

A sample smart contract was first created, deployed, and tested in Remix online IDE, before actual development. The Remix IDE can be used to debug the code and individual function of the smart contract can be tested too. For the purpose of testing a smart contract, firstly an inbuild Remix JavaScript VM test network was used and secondly Rinkeby Test Network was chosen too.

However, the transaction in Rinkeby Network failed due to low Ether balance in the account earlier. Some test ethers can be requested from the Rinkeby test Network faucet. These are the test network pools which can provide test ethers to be used for development and testing in the respective networks. About 18 test ethers were transferred from the faucet to one of the accounts in metamask. Although the test could be done in Ganache Network locally. Figure 7 shows an example of a smart contract deployed and tested in remix editor. Figure 8 shows the list of transactions result from Rinkeby Etherscan.

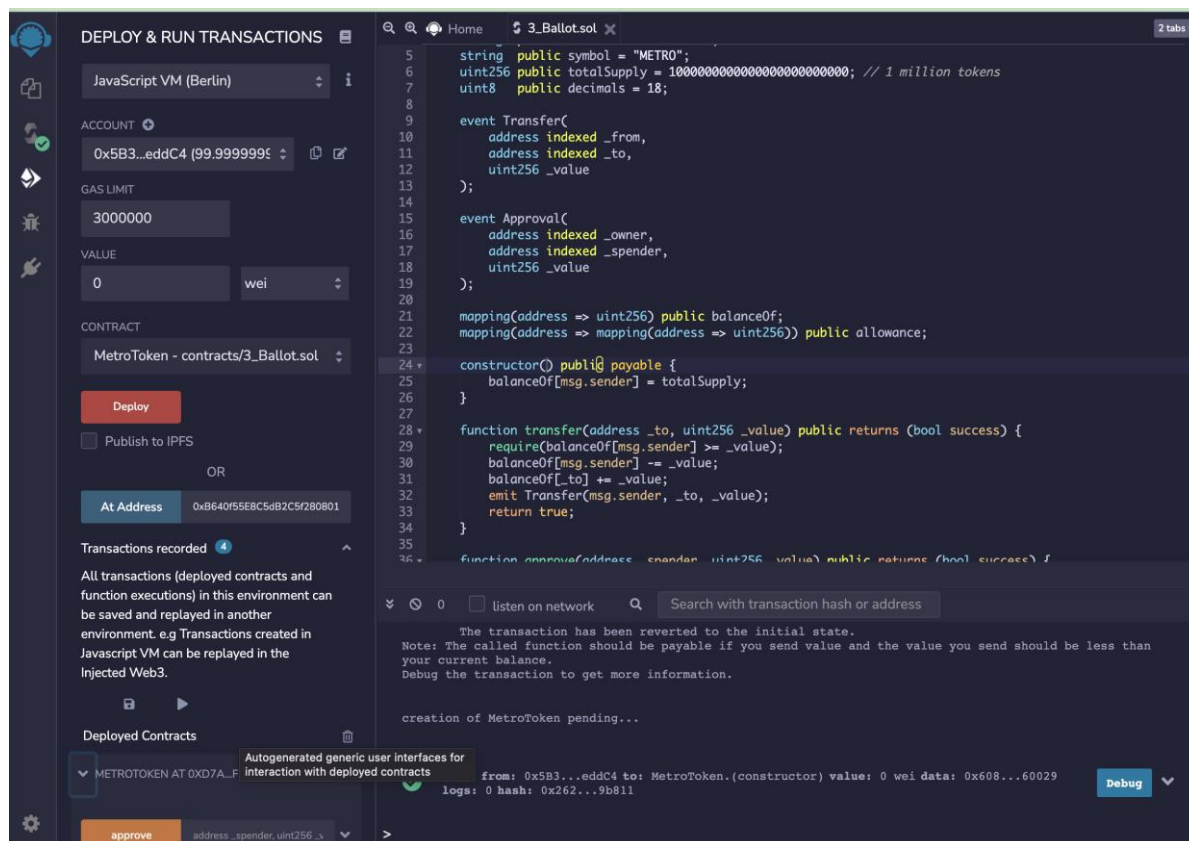


Figure 7. Smart contract deployment via Remix IDE

Etherscan  
Rinkeby Testnet Network

All Filters Search by Address / Txn Hash / Block / Token / Ens

Home Blockchain Tokens Misc Rinkeby

Address 0x1a5a0f48e3e51436B041d5048733aA41e41A8eb6

Overview

Balance: 17.848059579987581312 Ether

More Info

My Name Tag: Not Available

Transactions

Latest 12 from a total of 12 transactions

Txn Hash	Method	Block	Age	From	To	Value	T
0x6cd5e7ad6c7f755ef9b...	Contract Creation	9612216	27 secs ago	0x1a5a0f48e3e51436b0...	OUT	1 Ether	0.
0x15e1a85f2947a3c6e6...	Transfer	9600398	2 days 1 hr ago	0x31b98d14007bdee637...	IN	18.75 Ether	0.
0x9d3d163395535cc501...	Transfer	9563509	8 days 16 hrs ago	0xa500b2427458d12ef7...	IN	0.01 Ether	0.
0x946e3d9a67683f41ae...	Transfer	9563506	8 days 16 hrs ago	0xa500b2427458d12ef7...	IN	0.01 Ether	0.
0x2ac6134fa0425a2544...	Transfer	9563504	8 days 16 hrs ago	0xa500b2427458d12ef7...	IN	0.01 Ether	0.
0x0b03548b3ae3ef8533...	Transfer	9563501	8 days 17 hrs ago	0xa500b2427458d12ef7...	IN	0.01 Ether	0.
0xc63b37d528cbba1ae...	Transfer	9563498	8 days 17 hrs ago	0xa500b2427458d12ef7...	IN	0.01 Ether	0.

Figure 8. Transaction list from Rinkeby Etherscan

Figure 9 shows the detail of one transaction result of the deployment of a contract from Remix into Rinkeby Test Network. These indicates how one can keep track of the transaction through a blockchain network.

Rinkeby Testnet Network

Home Blockchain Tokens Misc Rinkeby

### Transaction Details

Overview Access List State

[ This is a Rinkeby Testnet transaction only ]

Transaction Hash:	0x6cd5e7ad6c71755ef9bdba9d7007b33ba01650a79b75f24aaca70f6f5fbb7720
Status:	Success
Block:	9612216 2 Block Confirmations
Timestamp:	55 secs ago (Nov-09-2021 01:41:26 PM +UTC)
From:	0x1a5a0f48e3e51436b041d5048733aa41e41a8eb6
To:	[Contract 0x3803a1b79adc457ea9afda1202c3ed99ffa564b Created]
Value:	1 Ether (\$0.00)
Transaction Fee:	0.001940420012418688 Ether (\$0.00)
Gas Price:	0.000000002500000016 Ether (2.500000016 Gwei)
Txn Type:	2 (EIP-1559)

[Click to see More](#)

Figure 9. Transaction details for a contract creation transaction

## 5.6 Installing development environment

Various installation procedures were performed for the project setup. In the MacOS terminal following commands were entered for different installations (Infante, 2019).

- NVM installation: Latest version of the NVM was downloaded and install from the website <https://github.com/nvm-sh/nvm#git-install>
- Node.js installation: `nvm install version`
- Creating a React.js project structure: `npx create-react-app stakingpool-metrotoken`

A fully structured directory with the name stakingpool-metrotoken was created.

- Install various node dependencies:

```
npm install --save web3 babel truffle@5.1.39 chai mocha
ganache-cli solc
```

This will install web3, babel and its' dependencies, truffle, chai, mocha ganache-cli and solidity compiler respectively.

- Creating truffle structure inside the project file: `truffle init`

This command created a truffle-config.js file inside the project folder where necessary settings to connect to ganache is provided. There were some issues regarding connecting to the port already inside the config file. The port was directing towards ganache-cli initially, later different port address was given to match with the ganache UI. The path of the truffle-config.js was added in the ganache UI settings.

Furthermore, there were some additional dev dependencies for babel, which was fixed installing each required dependency separately.

- Ganache User Interface (UI): Installed for MacOS from the website:

<https://trufflesuite.com/ganache>

Figure 10 shows the local Ganache UI. The local blockchain network is already provided with ten addresses. The workspace name was changed and the path to the project directory was added from the settings. The user interface shows account addresses with the current balance and transactions counts. Private keys can be accessed from the key symbol to the right.



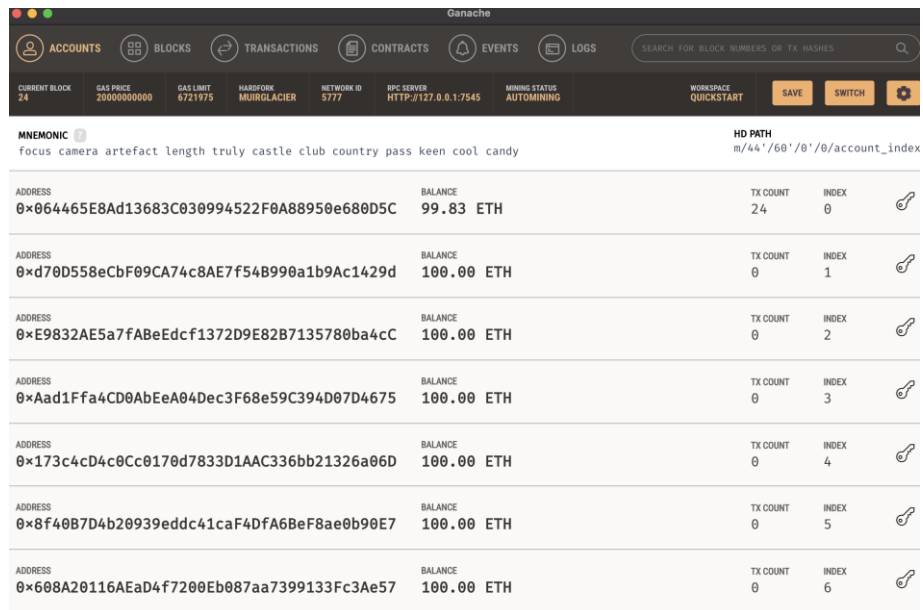


Figure 10. Ganache User Interface

The first account in the Ganache is for the owner or person who deploys the smart contract into the blockchain.

## 5.7 Creating smart contracts

There are mainly three different contracts. One contract is to create a token, other is to stake and transfer coins to different address and finally one is to migrate the contract to the blockchain. A very simple smart contract for token is created for this project.

```
pragma solidity ^0.5.1;

contract MetroToken {
    string public name = "Metro Token";
    string public symbol = "METRO";
    uint256 public totalSupply = 1000000000000000000000000; // One million tokens
    uint8 public decimals = 18;

    constructor() public {
        balanceOf[msg.sender] = totalSupply;
    }
}
```

```
}
```

### Listing 1. MetroToken.sol file content

Listing 1 shows a part of the code of a MetroToken.sol file. Pragma solidity ^0.5.1 line is declaring the version number of the solidity used for the development of the contract. Next is the standard contract naming class like structure inside which variables and methods can be defined. A constructor mentioned is the special method that gets invoked whenever the instance of class is created. In the current contract there are additional three functions: transfer(), approve (), and transferfrom () which are quite self-explanatory as they transfer, approve and transfer tokens across addresses. Other contracts have similar structure but different methods as per the contract.

## 5.8 Compiling and migrating smart contracts

Prior to compiling any contracts, a file with the name “2\_deploy\_contracts.js”, must be created in the migration folder inside the project directory. In case of this project, I have created “1\_initial\_migration.js” file which require the Migration.sol contract and invokes its method of setting up and upgrading of contracts. Listing 2 and Listing 3 show the content of the respective files.

```
module.exports = async function(deployer, network, accounts) {  
  // Deploy Metro Token  
  await deployer.deploy(MetroToken)  
  const metroToken = await MetroToken.deployed()  
}
```

### Listing 2. Contents of 2\_deploy\_contracts.js

With the command `truffle compile` truffle will compile all the contracts. Subsequent use of the command will only compile files that had undergone

changes. Adding `--all` at the end of the command will override changes and compile all contracts. Ganache must be kept running for the migration.

```
const Migrations = artifacts.require("Migrations");

module.exports = function(deployer) {
  deployer.deploy(Migrations);
};
```

Listing 3. Contents of `1_initial_migrations.js`

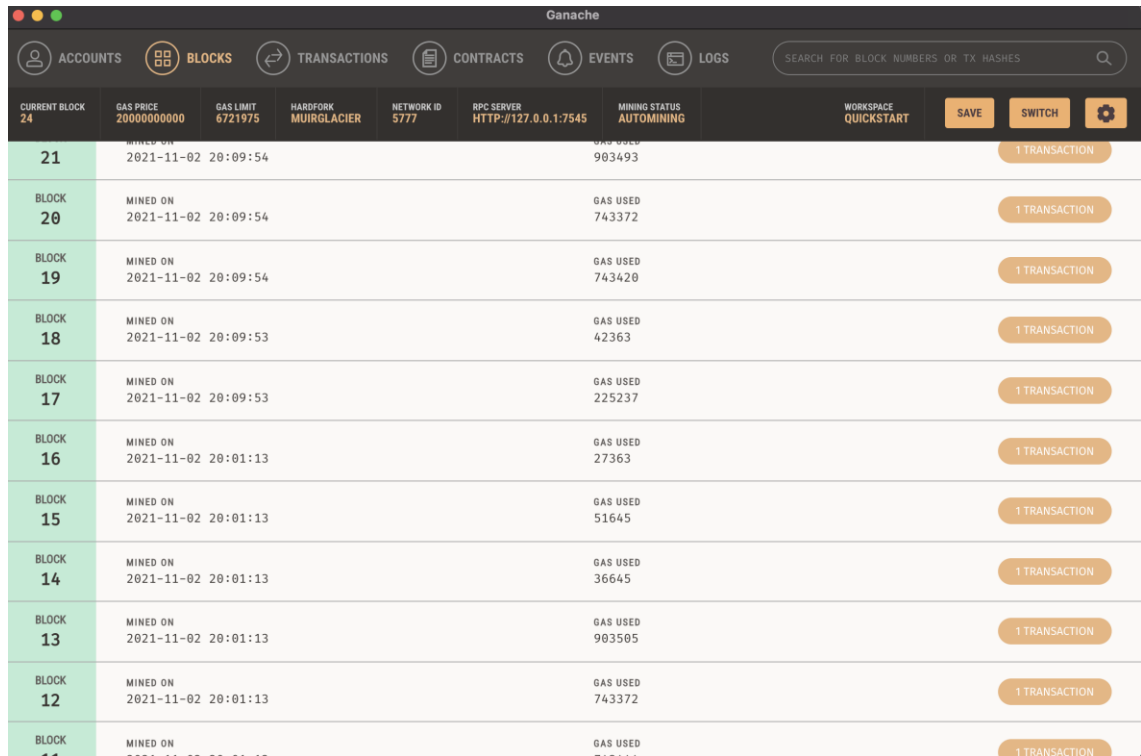
By using the command `truffle migrate` thus compiled contract can be migrated and deployed. Adding tag `--reset` will run all the migrations from beginning.

Output of the compile command is shown in listing 4 below:

```
Compiling your contracts...
=====
> Compiling ./contracts/MetroToken.sol
> Compiling ./contracts/NomToken.sol
> Compiling ./contracts/NomTokenPool.sol
> Compiling ./contracts/YakToken.sol
> Compiling ./contracts/YakTokenPool.sol
> Artifacts written to
/Users/iakai/Desktop/React/stake_metro_coin/src/artifacts_contracts
> Compiled successfully using:
  - solc: 0.5.16+commit.9c3226ce.Emscripten.clang
```

Listing 4. Compile log

These events can be tracked from the Ganache UI in the blocks tab as shown in Figure 11. Compilation of each contracts created a transaction reducing some ether amounts from the senders' ether balance. Each block thus, created is as the result of the transaction occurred. After the compilation of the contracts, ABIs were created inside the artifacts folder of the projects. These JSON format ABIs was later utilized to interact with the JavaScript side of the application.



CURRENT BLOCK	GAS PRICE	GAS LIMIT	HARDFORK	NETWORK ID	RPC SERVER	MINING STATUS	WORKSPACE	SAVE	SWITCH	SETTINGS
24	20000000000	6721975	MUIRGLACIER	5777	HTTP://127.0.0.1:7545	AUTOMINING	QUICKSTART			
BLOCK 21	MINED ON 2021-11-02 20:09:54		GAS USED 903493		1 TRANSACTION					
BLOCK 20	MINED ON 2021-11-02 20:09:54		GAS USED 743372		1 TRANSACTION					
BLOCK 19	MINED ON 2021-11-02 20:09:54		GAS USED 743420		1 TRANSACTION					
BLOCK 18	MINED ON 2021-11-02 20:09:53		GAS USED 42363		1 TRANSACTION					
BLOCK 17	MINED ON 2021-11-02 20:09:53		GAS USED 225237		1 TRANSACTION					
BLOCK 16	MINED ON 2021-11-02 20:01:13		GAS USED 27363		1 TRANSACTION					
BLOCK 15	MINED ON 2021-11-02 20:01:13		GAS USED 51645		1 TRANSACTION					
BLOCK 14	MINED ON 2021-11-02 20:01:13		GAS USED 36645		1 TRANSACTION					
BLOCK 13	MINED ON 2021-11-02 20:01:13		GAS USED 903505		1 TRANSACTION					
BLOCK 12	MINED ON 2021-11-02 20:01:13		GAS USED 743372		1 TRANSACTION					
BLOCK 11	MINED ON 2021-11-02 20:01:13		GAS USED 743372		1 TRANSACTION					

Figure 11. Blocks in the Ganache UI.

## 5.9 Running the application

With the Ganache running on the background and all those contracts compiled and migrated to the blockchain, the command `npm run start` will run the application in the `localhost:3000` web address. A function was written to connect the application with the blockchain utilizing `web3` instance. It is shown in the Listing 5 below.

```
async loadWeb3() {
  if (window.ethereum) {
    window.web3 = new Web3(window.ethereum);
    await window.ethereum.enable();
  }
}
```

```

} else if (window.web3) {
  window.web3 = new Web3(window.web3.currentProvider);
} else {
  window.alert(
    "Non-Ethereum browser detected. You should consider trying MetaMask!"
  );
}

```

Listing 5. Connecting application to the blockchain with web3.

After running the application, it should be connected to the Metamask wallet. The second account's private key from the Ganache blockchain was imported into the Metamask wallet and the test network is configured to connect to the Ganache network. When the app starts after it connects to the blockchain it loads the data in the blockchain with the function: `loadBlockchainData()`. This function contains various states of the individual token data which was fetched from their respective ABIs. A basic application flow is discussed below:

When the value is input in the text field to stake token and "Stake", button is pressed an event is triggered which call for the function `stakeTokens()`, which takes a parameter amount. It changes the loading state to true and calls for additional functions `approve` and `send`, to send the amount from the current address `YakTokenPool` address. Listing 6. Shows the staking function:

```

stakeTokens = (amount) => {
  this.setState({ loading: true })
  this.state.metroToken.methods
    .approve(this.state.yakTokenPool._address, amount)
    .send({ from: this.state.account })
    .on('transactionHash', (hash) => {
      this.state.yakTokenPool.methods.stakeTokens(amount)
        .send({ from: this.state.account })
        .on('transactionHash', (hash) => {
          this.setState({ loading: false })
        })
    })
}

```

Listing 6. Staking amount to the TokenPool

The metamask wallet pops up, keeping the app into loading state. There will be two transactions, first one is to approve the tokens and second one to actually stake the token. After staking the staking balance of the account will increase with the staked amount and thus the balance is deducted from the investor's account. A small transaction fee is charged for every approved transaction which can be analysed from the blockchain transaction in the ganache network. Figure 12 and Figure 13 show the procedure of staking and the two transactions in the metamask wallet respectively.

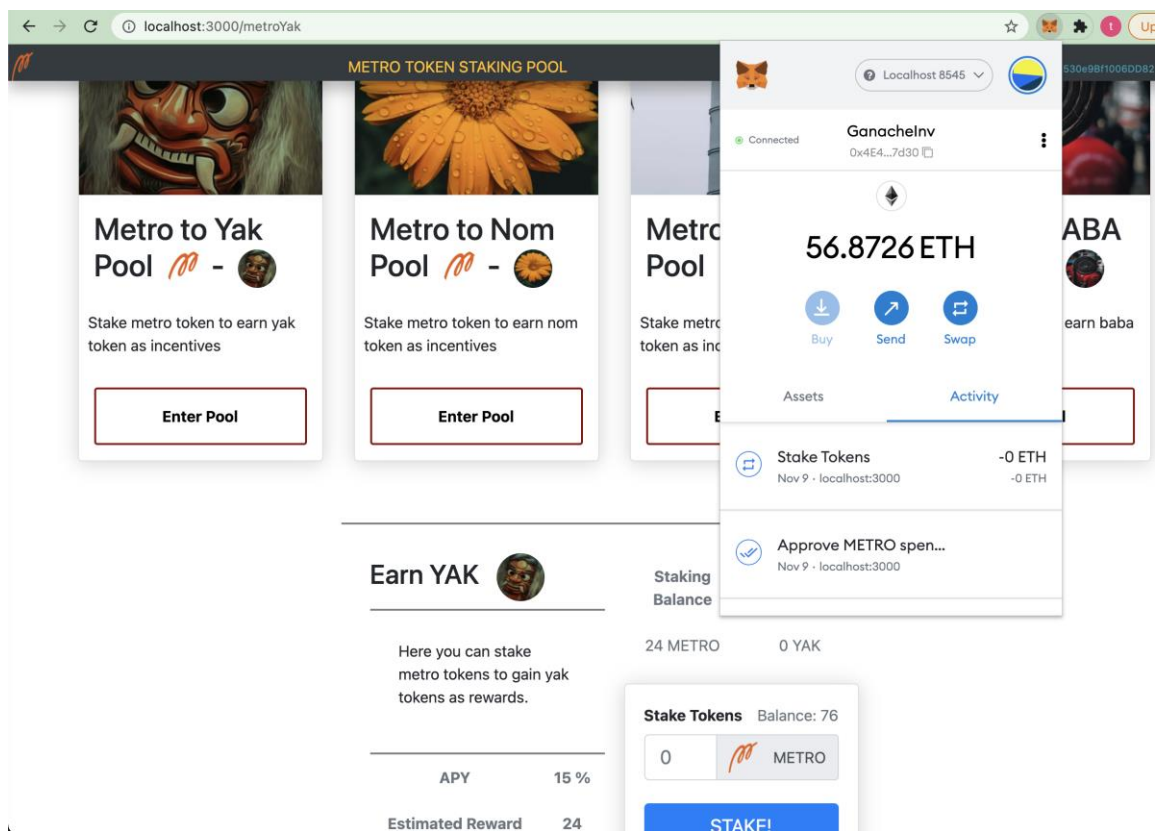


Figure 12. Staking Metro coins to earn Yak coins

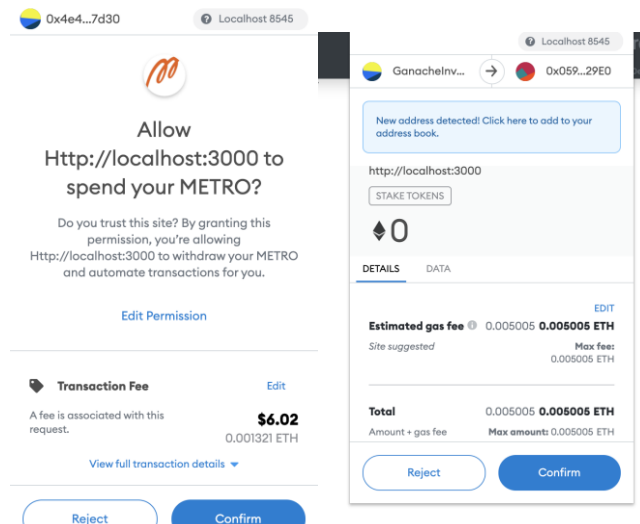


Figure 13. Two transactions for the (left) approve and staking (right)

After staking, the investor, in this case account [1] from the Ganache blockchain can either unstake the staked token or wait for certain period to get rewards in the form another token. For the thesis project, a simple reward return is implemented.

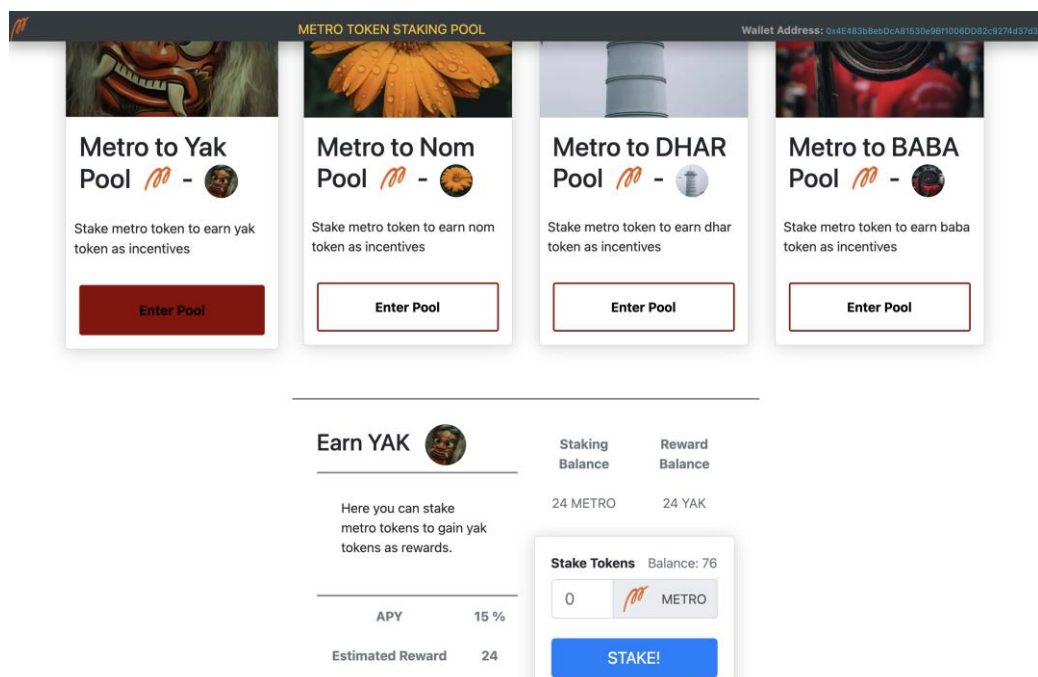


Figure 14. Screenshot of the staking application, with four different tokens pool for metro token staking with staked balance, rewards, and remaining balance.

Only the owner of the pool or the one responsible for deploying the contracts can distribute the reward by running the already existing contract in the blockchain. In this case, by running a script `issue-token.js` which triggers a function `issueToken()` in the smart contract. The command to run the script inside the scripts folder was `truffle exec scripts/issue-token.js`. These reward distribution tasks can be automated in the real production environment with various conditions for distributions like monthly and daily. These rewards can be considered as interest earned on the staking. Figure 14 shows the rewards and staked balance.

```
unstakeTokens = (amount) => {
  this.setState({ loading: true })
  this.state.yakTokenPool.methods.unstakeTokens().send({ from: this.state.account
}).on('transactionHash', (hash) => {
  this.setState({ loading: false })
})
}
```

Listing 7. Unstaking function for metroToken

As an investor one can also withdraw or unstake the whole coin staked. Unstaking the coin will however have no effect in the rewards distributed. Listing 7 shows the unstaking function for the metroToken.

## 5.10 Testing the application

The application smart contract was tested manually in Remix online editor before actual development. Later, a Test-Driven Development (TDD) approach was incorporated to test the `YakTokenPool.sol` contract file's functionalities where every functionality in the contract was written and tested before writing the full contract. The command `truffle test` was used to execute the test and its output is shown in the listing 8.

```
Contract: TokenPool
Metro deployment
  ✓ has a name (39ms)
Yak Token deployment
```



```

    ✓ has a name
Token Pool deployment
    ✓ has a name
    ✓ contract has tokens
Pooling tokens
    ✓ rewards investors for staking metro tokens (496ms)
5 passing (1s)

```

Listing 8. Output from truffle test

## 6 Results and Discussions

This chapter presents and discusses the results and the outcome of the project alongside some of the development challenges encountered. Additionally, it also summarizes about some approaches that can be taken in the future releases.

### 6.1 Outcome of project

The result of the project is a functional decentralized staking application platform where a user can stake a metro coin to get other secondary coins. Instead of using traditional saving system, users can utilize the staking of their tokens to get other tokens with comparatively higher returns. The tokens used in the projects are just an example and can be replaced by a real cryptocurrency, when deployed in the main Ethereum network. After the smart contracts are deployed, a user can use the application to choose between various token pair pools to stake tokens and earn rewards. A user can simply unstake these coins as per their will hence, removing the needs of any third-party acceptance or waiting time. Moreover, all of these transactions can be tracked along with the addresses, hence providing a complete transparency.

### 6.2 Development challenges and real-life limitations

During this project, there were a few obstacles to overcome. Solidity programming language is a completely new tool to learn and implement into a

project, which is one of the challenges faced in the earlier phase. Furthermore, the frequency of language version changes is high. Also, with the release of every version there have been some breaking changes, hence adopting to new changes was another hurdle in the development process.

Additionally, the Ethereum test net faucets are extremely unreliable and do not provide test ether as they should. Various faucets were attempted and tested in order to obtain a sufficient amount of test ether so that the application development process could continue. None of them, however, were very successful and took a long time to answer to queries. This necessitated the usage of Ganache, a local test network, for testing and deployments. There were a lot of dependencies between different programs and their versions, which slowed down the development process. Since, the contracts deployed on the blockchain are immutable, there were some initial issues with modifying the deployed contracts. The Ethereum developer support and development forums are expanding by the day, which has been incredibly helpful in the application development process.

Finally, all of the issues were resolved in order to produce a minimal practical, functioning, decentralized staking pool application for users to experience a different system of earn rewards on their investment.

Despite of all of the benefits discussed, Ethereum blockchain development comes with some limitations. As Ethereum platform has different usage like decentralized ledger, a platform for smart contracts and other decentralized applications these multiple fields are prone to different errors, shutdown, and hacks. Ethereum platform hack of 2016 can be a good example of its vulnerability. However, multiple protocol changes and modifications have been done by developers to make it more secure after the event but advancement in technology and cryptography will always put a threat on the security issue of the platform even to the slightest. Furthermore, there are a few scalability challenges with this network, as well as worries about excessive transaction fees. As more people and businesses use the network, congestion occurs,

causing gas prices to climb, sometimes more than the cost of the transaction itself. Nonetheless, the platform is evolving, and future updates such as Ethereum 2.0, which includes a transition from PoW to PoS and a multilayer solution for scalability, promise to address some of the network's biggest flaws.

## **7 Conclusion**

The main goal of the thesis was to provide an overview of blockchain technology and its various aspects, as well as to develop a staking pool application where users can stake their cryptocurrencies to earn substantially higher interest than a regular savings account.

Throughout the thesis, it was concluded that by employing blockchain technology, a decentralized application for staking can be created with complete transparency and user control. Additionally, running the application on the Ethereum blockchain resulted in a trustless and fast-performing solution. To summarize, a functional decentralized cryptocurrency staking application was developed on the Ethereum blockchain providing a better alternative in terms of interest return to the saving accounts used in banks. Furthermore, during the application testing phase, some constructive user feedback was obtained based on the overall user experience with the application's concept and operation, which will be taken into consideration for future development. Nonetheless, following the completion of the development process, a successful and fulfilling practical application was developed with plans to release new versions in the near future.

Aside from being a viable alternative to a savings account, the staking procedure entails a significant amount of risk as compared to savings accounts. Full transparency of the staking pool is necessary in order to avoid multiple rug pools and misleading marketing. Furthermore, KYC and other authenticity verifications may be applied in staking pool apps to give users with safe and dependable investment possibilities. These features can be a better complement to the application's future development.

## References

- Aggarwal, S., & Kumar, N. (2021). Hyperledger☆. *Advances in Computers*, 121, 323–343. <https://doi.org/10.1016/bs.adcom.2020.08.016>
- Algorand. (n.d.). Retrieved November 15, 2021, from <https://www.algorand.com/technology/white-papers>
- Antonopoulos, A. M. (2017). *Mastering Bitcoin: Unlocking Digital Cryptocurrencies*.
- Application Binary Interface (ABI). (n.d.). <https://docs.soliditylang.org/en/v0.8.8/abi-spec.html#:~:text=The Contract Application Binary Interface,contract-to-contract interaction.&text=This specification does not address,known only at run-time>.
- Babeljs.io. (n.d.). <https://babeljs.io/docs/en/>
- Baran, P. (1979). Distributed communications. In *Computer Communications* (Vol. 2, Issue 3, pp. 137–138). [https://doi.org/10.1016/0140-3664\(79\)90214-7](https://doi.org/10.1016/0140-3664(79)90214-7)
- Bashir, I. (2017). *Mastering Blockchain - Master the theoretical and technical foundations of Blockchain technology and explore future of Blockchain technology*.
- Bellaj Badr, R. H. and X. W. (2018). *Blockchain By Example: A Developer's Guide to Creating Decentralized Applications Using Bitcoin, Ethereum, and Hyperledger* (www.packt.com, Ed.).
- Buterin, V. (2015). Ethereum whitepaper. *Ethereum.Org*, 1–32. <https://ethereum.org/en/whitepaper/%0Ahttps://whitepaper.io/document/5/ethereum-whitepaper>
- Cardano. (n.d.). Retrieved November 15, 2021, from [docs.cardano.org/core-concepts/delegation](https://docs.cardano.org/core-concepts/delegation)
- cbdc-whitepaper-2020. (n.d.).
- chaijs.com. (n.d.).
- ChainLink. (n.d.). Retrieved November 15, 2021, from <https://chain.link/>
- Ethereum classic. (n.d.). Retrieved November 15, 2021, from <https://ethereumclassic.org/>

- Francisco, S. (2018). *Advanced Decentralized Blockchain Platform Whitepaper Version: 2.0*.
- Ganache-UI. (n.d.). <https://github.com/trufflesuite/ganache-ui>
- Gates, M. (2017). *Blockchain: Ultimate guide to understanding blockchain, bitcoin, cryptocurrencies, smart contracts and the future of money* (pp. 3–5).
- Gilbert, S. and N. L. (2002). *Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services*. 8.
- Haber, S., & Scott Stornetta, W. (1991). How to time-stamp a digital document. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 537 LNCS, 437–455. [https://doi.org/10.1007/3-540-38424-3\\_32](https://doi.org/10.1007/3-540-38424-3_32)
- Infante, R. (2019). *Building Ethereum Dapps: Decentralized applications on the Ethereum blockchain* (p. 514). <https://www.manning.com/books/building-ethereum-dapps>
- Iredale Gwyneth. (n.d.). *Types-of-blockchain*. Retrieved November 15, 2021, from <https://101blockchains.com/types-of-blockchain/>
- JavaScript. (n.d.). <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
- Jonathan, K., & Yehuda, L. (2014). *Introduction to Modern Cryptography, Second Edition* (Vol. 1).
- Lamport, L., Shostak, R., & Pease, M. (1982). The Byzantine Generals Problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3), 382–401. <https://doi.org/10.1145/357172.357176>
- Mayukh Mukhopadhyay. (2018). *Ethereum smart contract development: build blockchain-based decentralized applications using Solidity*. Packt Publishing.
- Metamask. (n.d.). <https://metamask.io/index.html>
- Mochajs.org. (n.d.).
- Nakamoto, S. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*. <https://doi.org/10.2139/ssrn.3440802>
- NEO. (n.d.). Retrieved November 15, 2021, from <https://neo.org/>
- nodejs.dev. (n.d.). <https://nodejs.dev/learn/introduction-to-nodejs>
- npmjs. (n.d.). <https://docs.npmjs.com/>
- npx. (n.d.). <https://nodejs.dev/learn/the-npx-nodejs-package-runner>

*nvm*. (n.d.). <https://github.com/nvm-sh/nvm>

*Polkadot*. (n.d.). Retrieved November 15, 2021, from <https://polkadot.network/>

Poon, J., & Dryja, T. (2016). *The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments*.

*reactjs.org*. (n.d.).

*Remix-IDE*. (n.d.). <https://remix-ide.readthedocs.io/en/latest/>

*Solidity*. (n.d.). <https://docs.soliditylang.org/en/v0.8.9/>

*Stellar*. (n.d.). Retrieved November 15, 2021, from <https://www.stellar.org/>

*Tethar*. (n.d.). Retrieved November 15, 2021, from <https://tether.to/>

*Tezos*. (n.d.). Retrieved November 15, 2021, from <https://wiki.tezosagora.org/whitepaper#pos>

Tovanich, N., Soulie, N., & Isenberg, P. (2021, April 19). Visual Analytics of Bitcoin Mining Pool Evolution: On the Road Toward Stability? *2021 11th IFIP International Conference on New Technologies, Mobility and Security, NTMS 2021*. <https://doi.org/10.1109/NTMS49979.2021.9432675>

*Truffle Suite*. (n.d.). <https://github.com/trufflesuite/truffle>

Vojin G. Oklobdzija. (2001). *The Computer Engineering Handbook* (1st ed.).

Yakovenko, A. (2018). *Solana: A new architecture for a high performance blockchain*.

## Appendices

### Appendix 1: Contract migration Output

Compiling your contracts...

=====

> Compiling ./contracts/BabaToken.sol

> Compiling ./contracts/BabaTokenPool.sol

> Compiling ./contracts/DharToken.sol

> Compiling ./contracts/DharTokenPool.sol

> Compiling ./contracts/MetroToken.sol

> Compiling ./contracts/Migrations.sol

> Compiling ./contracts/NomToken.sol

> Compiling ./contracts/NomTokenPool.sol

> Compiling ./contracts/YakToken.sol

> Compiling ./contracts/YakTokenPool.sol

> Artifacts written to  
/Users/lakai/Desktop/Materials\_for\_thesis/\_routing\_react\_project/stakingpool-  
metrotoken/src/artifacts\_contracts-

> Compiled successfully using:

- solc: 0.5.16+commit.9c3226ce.Emscripten.clang

Starting migrations...

=====

> Network name: 'development'

> Network id: 1337

> Block gas limit: 6721975 (0x6691b7)

1\_initial\_migration.js

=====

Deploying 'Migrations'

-----

```
> transaction hash:
0xf2374907958e4c6f444b85c2767b2761cc97009951571364ee6e5df57cd440bf

> Blocks: 0          Seconds: 0

> contract address:  0xFb855f7578468743586199A1f2E5C793987cE850

> block number:      111

> block timestamp:    1636462283

> account:           0xf3cefed66f62e03A2C6A9A718c140EF98F81E690

> balance:            111.49249358

> gas used:           197877 (0x304f5)

> gas price:          20 gwei

> value sent:         0 ETH

> total cost:         0.00395754 ETH
```

> Saving migration to chain.

> Saving artifacts

-----

```
> Total cost:         0.00395754 ETH
```

2\_deploy\_contracts.js

=====

Deploying 'MetroToken'

-----

```
> transaction hash:
0x413e2dfd1dda4266fab8962c772a9b283c755032ead279eeb849b66fb07ea3b0

> Blocks: 0          Seconds: 0

> contract address:  0xA55429E5570648aF69a71D3fD16F26eB54cB6dB6

> block number:      113

> block timestamp:    1636462284
```



```

> account:          0xf3cefed66f62e03A2C6A9A718c140EF98F81E690
> balance:          111.47665986
> gas used:         749343 (0xb6f1f)
> gas price:        20 gwei
> value sent:       0 ETH
> total cost:       0.01498686 ETH

```

Deploying 'YakToken'

-----

```

> transaction hash:
0x592fdfffec392ac8e5cbddf93e5888a0708b2be874fca405cc45ede2a1463965

```

```

> Blocks: 0          Seconds: 0
> contract address:  0x953E5595338108bF1b601347248F8d1Aa9b50aFA
> block number:      114
> block timestamp:   1636462284
> account:          0xf3cefed66f62e03A2C6A9A718c140EF98F81E690
> balance:          111.46167372
> gas used:         749307 (0xb6efb)
> gas price:        20 gwei
> value sent:       0 ETH
> total cost:       0.01498614 ETH

```

Deploying 'NomToken'

-----

```

> transaction hash:
0x4a9ef5a5cf2466a3d8a1d780d21a23f9036e68dea47e5f3af2143c2a9265700c

```

```

> Blocks: 0          Seconds: 0
> contract address:  0xC6c439fbe6F652E610C9e566756ceD1027fF6406
> block number:      115
> block timestamp:   1636462284
> account:          0xf3cefed66f62e03A2C6A9A718c140EF98F81E690
> balance:          111.44668694

```

```

> gas used:          749339 (0xb6f1b)

> gas price:         20 gwei

> value sent:        0 ETH

> total cost:        0.01498678 ETH

```

Deploying 'YakTokenPool'

-----

```

> transaction hash:
0x53a9c90165e59d36800125e7119efe5d67e4486cb047712a23364a0e242e1793

```

```

> Blocks: 0          Seconds: 0

> contract address:  0x049574a51482272A46F4A81962DE9f6000e2AAE1

> block number:      116

> block timestamp:   1636462284

> account:           0xf3cefed66f62e03A2C6A9A718c140EF98F81E690

> balance:           111.4281548

> gas used:          926607 (0xe238f)

> gas price:         20 gwei

> value sent:        0 ETH

> total cost:        0.01853214 ETH

```

Deploying 'NomTokenPool'

-----

```

> transaction hash:
0x450e127f35a35ed1d3e470347d5d36ae88eae4d8569ccad17f217d1c62288782

```

```

> Blocks: 0          Seconds: 0

> contract address:  0x6195e3ba0781dAf6C56A295628023B678Ba1C94F

> block number:      117

> block timestamp:   1636462284

> account:           0xf3cefed66f62e03A2C6A9A718c140EF98F81E690

> balance:           111.40962266

> gas used:          926607 (0xe238f)

> gas price:         20 gwei

```

```
> value sent:          0 ETH
> total cost:          0.01853214 ETH
> Saving migration to chain.
> Saving artifacts
```

```
-----
```

```
> Total cost:          0.08202406 ETH
```

Summary

```
=====
```

```
> Total deployments:   6 > Final cost:          0.0859816 ETH
```