

# **BUILDING AND DEPLOYING MODERN WEB 3.0 BLOCKCHAIN APPLICATION**

Project report submitted in partial fulfilment of the  
requirement for the degree of Bachelor of Technology

in

**Computer Science and Engineering/Information  
Technology**

By

Shourya Singh (191212)

Under the supervision of

Dr. Monika Bharti

&

Dr. Udayabanu, M.

To



Department of Computer Science & Engineering and  
Information Technology

**Jaypee University of Information Technology  
Waknaghat, Solan-173234, Himachal Pradesh**

## Candidate's Declaration

I hereby declare that the work presented in this report entitled "**BUILDING AND DEPLOYING MODERN WEB 3.0 BLOCKCHAIN APPLICATION**" in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering/Information Technology** submitted in the department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from July 2022 to May 2023 under the supervision of **Dr. Monika Bharti** (Assistant Professor (SG), Department of Computer Science and Engineering) & **Dr. Udayabanu, M.** (Associate Professor, Department of Biotechnology & Bioinformatics).

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

(Student Signature)

Shourya Singh, 191212

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

(Supervisor Signature)(Supervisor Signature)

Dr. Monika Bharti Dr. Udayabanu, M.

Assistant Professor(SG) Associate Professor

Department of Computer Department of Biotechnology

Science & Engineering & Bioinformatics

Dated: Dated:

**JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT**  
**PLAGIARISM VERIFICATION REPORT**

Date: .....

Type of Document (Tick):  PhD Thesis  M.Tech Dissertation/ Report  B.Tech Project Report  Paper

Name: \_\_\_\_\_ Department: \_\_\_\_\_ Enrolment No \_\_\_\_\_

Contact No. \_\_\_\_\_ E-mail. \_\_\_\_\_

Name of the Supervisor: \_\_\_\_\_

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): \_\_\_\_\_

\_\_\_\_\_

**UNDERTAKING**

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

**Complete Thesis/Report Pages Detail:**

- Total No. of Pages =
- Total No. of Preliminary pages =
- Total No. of pages accommodate bibliography/references =

(Signature of Student)

**FOR DEPARTMENT USE**

We have checked the thesis/report as per norms and found **Similarity Index** at .....(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor)

Signature of HOD

**FOR LRC USE**

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	Excluded	Similarity Index (%)	Generated Plagiarism Report Details (Title, Abstract & Chapters)	
	<ul style="list-style-type: none"> <li>• All Preliminary Pages</li> <li>• Bibliography/Images/Quotes</li> <li>• 14 Words String</li> </ul>		Word Counts	
<b>Report Generated on</b>			Character Counts	
		<b>Submission ID</b>	Total Pages Scanned	
			File Size	

Checked by  
Name & Signature

Librarian

**Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at [plagcheck.juit@gmail.com](mailto:plagcheck.juit@gmail.com)**

## ACKNOWLEDGEMENT

Firstly, I express my heartiest thanks and gratefulness to almighty God for His divine blessing makes it possible to complete the project work successfully.

I am really grateful and wish my profound indebtedness to the Supervisor **Dr. Monika Bharti, Assistant Professor(SG), Department of Computer Science & Engineering& Dr. Udayabanu, M., Associate Professor, Department of Biotechnology & Bioinformatics.** Deep Knowledge & keen interest of my supervisors in the field of “**Blockchain**” to carry out this project. Their endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, and reading many inferior drafts and correcting them at all stages have made it possible to complete this project.

I would like to express my heartiest gratitude to **Dr. Monika Bharti, Department of Computer Science & Engineering& Dr. Udayabanu, M., Department of Biotechnology & Bioinformatics**for their kind help to finish my project.

I would also generously welcome each one of those individuals who have helped me straightforwardly or in a roundabout way in making this project a win. In this unique situation, I might want to thank the various staff individuals, both educating and non-instructing, which have developed their convenient help and facilitated my undertaking.

Finally, I must acknowledge with due respect the constant support and patients of my parents.

(Student Signature)

Project Group Number:24

Shourya Singh, 191212

## TABLE OF CONTENT

<b>Title</b>	<b>Page No.</b>
<b>Chapter-1 INTRODUCTION</b>	
1.1 Introduction	01
1.2 Problem Statement	05
1.3 Objectives	07
1.4 Methodology	07
<b>Chapter-2 LITERATURE SURVEY</b>	16
<b>Chapter-3 SYSTEM DEVELOPMENT</b>	28
<b>Chapter-4 PERFORMANCE AND RESULT ANALYSIS</b>	46
<b>Chapter-5 CONCLUSION</b>	
5.1 Conclusions	54
5.2 Future Scope	54
5.3 Applications Contributions	55
<b>REFERENCES</b>	56
<b>APPENDICES</b>	58

## LIST OF FIGURES

<b>Sr. No.</b>	<b>TITLE</b>	<b>Page No.</b>
1.	Blockchain Representation	03
2.	dApp Flowchart	04
3.	System Architecture	08
4.	Flowchart of EVM	14
5.	Working of Blockchain protocol	29
6.	Operational mechanism of Smart Contract	34
7.	Taxonomy of Blockchain-enabled Smart Contract	39
8.	Basic research framework of Smart Contracts	44
9.	Transaction done using GoerliTestnet Faucet	48
10.	Transaction done using SepoliaTestnet Faucet	49
11.	On clicking “Connect Wallet”, our Metamask wallet will be connected	50
12.	Our Metamask Wallet is now connected	50
13.	Making a transaction with sender’s address, ethers, gif and message	51
14.	RecentTransaction	52
15.	Details of transaction on Etherscan	53
16.	Contract Structure	58
17.	Function to add transaction to blockchain	58

18.	Function to create Ethereum Contract	58
19.	Function to get all the transactions	59
20.	Function to check if the wallet is connected	59
21.	Function to send transaction	60

## LIST OF TABLES

<b>Sr. No.</b>	<b>Title</b>	<b>Page No.</b>
1.	Comparative Analysis of Blockchain Methodologies and their Limitations	27
2.	Vulnerability Analysis of Consensus Mechanisms against Common Blockchain Attacks	47



## LIST OF GRAPHS

<b>Sr. No.</b>	<b>Title</b>	<b>Page No.</b>
1.	Gas Consumption in comparison to Transaction Time	46

## LIST OF ABBREVIATIONS

<b>Sr. No.</b>	<b>Abbreviations</b>	<b>Full Form</b>
1.	PoS	Proof of Stake
2.	API	Application Programming Interface
3.	AI	Artificial Intelligence
4.	ML	Machine Learning
5.	dApps	Decentralised Application
6.	EVM	Ethereum Virtual Machine
7.	NFT	Non-Fungible Token
8.	ETH	Ether
9.	SHA	Secure Hashing Algorithm
10.	PoA	Proof of Authority
11.	UX	User Experience
12.	UI	User Interface

## **ABSTRACT**

Today, most people use digital payment methods to pay for most of their daily transactions and growth of digital payment is booming every year. However, drawbacks to these traditional transaction platforms includes lack of transparency and traceability, leading to increased costs and delays, susceptibility to cyber-attacks. In this project, I proposed an application that remedies all of these drawbacks through use of the Ethereum Blockchain platform. Blockchain, the technology behind Bitcoin, is changing the internet by allowing secure and transparent sharing of information without central authority. This creates new trustworthy, efficient, and cost-effective online services. What makes it a powerful tool for digitalizing everyday services is the introduction of Smart Contracts, as in the Ethereum platform. Ethereum and its network is one of the most suitable ones, due to its consistency, widespread use, and provision of Smart Contracts logic. A transaction system must be secure, as it should not allow the transactions to be altered and be fully transparent, while protecting the privacy of the user.

The application was developed using the Solidity programming language and the algorithm used is Proof of Stake(PoS). The application's functions were contained within an Ethereum Smart Contract, which was then migrated to the Ethereum network. The user's input was read through a web interface and sent to the Ethereum network via the web3.js API. Statistics about the application were gathered on the Goerli test network. Contract creation times for the application were shown to be less than a second. A transaction time comparison and performance evaluation of algorithm was then conducted. The comparison showed that Proof of Stake(PoS) is susceptible to all of the attacks.

# CHAPTER 01: INTRODUCTION

## 1.1 Introduction

Nakamoto first described the idea, framework, and workings of the blockchain in 2008. Since then, people have started to pay attention to blockchain technology and have discovered a number of advantageous properties as machine trust, traceability, and data immutability. With the rise of Bitcoin in the mainstream consumer market, Blockchain technology's potential to upend established markets has gained attention in recent years. Blockchains can be used as a platform to create completely decentralised apps without a single point of failure and without hierarchical ownership of user data in addition to its coin applications. This has important benefits for data ownership, security, and privacy as well as the ability to significantly lower middle-man expenses. The issue of user trust in decentralized applications that run on a Blockchain platform is one that has been studied and developed on recently, and it is now possible to build application that the user can trust with their money.

Blockchain technology is wide ranging, and it is likely to have a significant impact on economic as compared to that of the Internet in this decade. As Blockchain is the centre of Bitcoin and other virtual monetary forms, Blockchain innovation can, at any rate, be expected to control considerably more huge method for trade from now on. Notwithstanding, Blockchain innovation has a lot a larger number of uses than just virtual monetary standards.

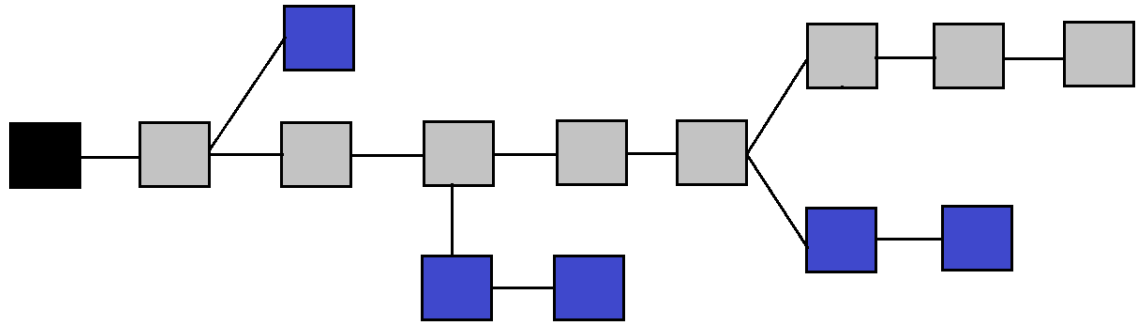
An open and distributed ledger is the Blockchain. It makes use of an append-only data structure, which allows for the addition of new transactions and data to a Blockchain but prevents the deletion of previous data. As a consequence, a permanent and verifiable record of the information and transactions between two or more parties is produced. Our social and economic processes might be improved and increased openness and accountability as a result.

Running programming and interfacing various hubs together make a Blockchain. There are various Blockchains, thus a Blockchain isn't one single worldwide element. Consider an organization of interconnected PCs that are associated with one another yet not to the web, which are all situated in an exceptionally protected office. Like this, a Blockchain can contain many associated hubs while remaining completely unmistakable from other Blockchains. Organisations and banks can create internal Blockchains with their own characteristics for a variety of hierarchical goals.

For a Blockchain to be reliable and practical, it needs both a reward scheme and a contracting mechanism. In the Bitcoin Blockchain, consensus is obtained through "mining," and the reward structure is a custom that awards an excavator with some Bitcoin after they successfully mine a block. Mining is done by powerful PCs testing numerical puzzles.

Miners begin work on the following block as soon as a transaction has been verified and confirmed as true by the whole network. A Blockchain thus continues to expand (linking each new block to the one before it).

When a transaction is entered into the Blockchain, all nodes instantly record, verify, and settle the transaction's data, including the price, asset, and owner. A confirmed change that is recorded on one ledger is also recorded simultaneously on all other copies of that ledger. There is no need for third-party verification because every transaction is openly and permanently recorded across all ledgers.



**Figure 1. Blockchain Representation**

Web3 is the third era of the web that is as of now being created. Using advancements like man-made brainpower (computer based intelligence), AI (ML), enormous information, decentralized record innovation (DLT), and others, sites and applications will actually want to handle data in a shrewd, human-like way.

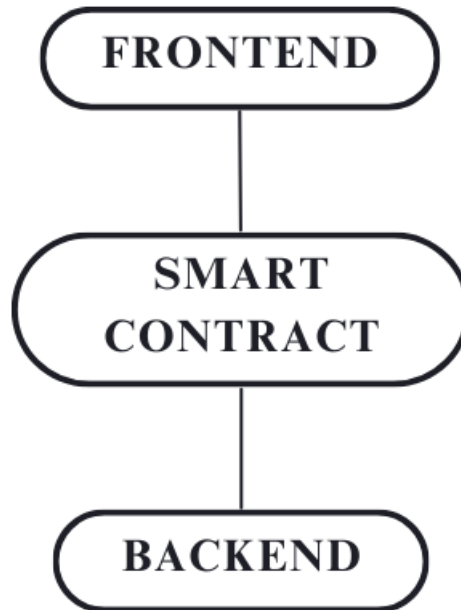
If we think about the transition from Web2 to Web3, Web3 is still a somewhat nebulous concept and might take 5 to 10 years to develop. In fact, what we may initially observe is a protracted Web2.5 phase during which key Web3 protocols are gradually adopted by Web2 systems.

Experts generally concur that, in order to successfully accomplish decentralisation, Blockchain-powered applications will be crucial, and that AI and ML technologies will assist automate and grow it as needed in order for it to become a semantic web.

Digital software known as "dApps," which function on a Blockchain network of computers rather than relying on a single computer, is known as decentralised applications. dApps are not managed or affected by a single authority because they are decentralised. The preservation of user privacy, the absence of restrictions, and the adaptability of production are advantages of dApps.

A dApps is like any other piece of software you use. It might be a mobile app or a website. A dApps is an application that is built on a decentralized

organization, like Ethereum, rather than a traditional application. Building own Ethereum Smart Contracts is similar to writing a section of dApp's backend code. Additionally, even if dApps has a user interface similar to a conventional app, Ethereum is used for all or part of the backend.



**Figure 2..dApp flowchart**

A Blockchain-based decentralised application (DApp) has the potential to lower platform usage fees while giving users more control over their data. Blockchain offers a perfect framework for developing a more secure transaction platform, in contrast to centralised platforms that may compromise and sell user data. The requirement for a system without a single point of failure is made clear by previous data breaches, such the 2014 eBay hack that exposed the data of 145 million customers. However, people can protect their own data in a decentralised system.

Due to its design as a platform for building decentralised apps, the Ethereum Blockchain is the best choice for creating such a DApp . It is simple to build a secure transaction platform thanks to Ethereum's user-friendly infrastructure and its own currency, ether, which can be used within applications. By market capitalization, Ethereum is the second-largest cryptocurrency. It features rich documentation and a thriving developer community. On the Ethereum network, a number of DApps have already been developed with great success, with applications like CryptoKitties illustrating their viability.

One of Ethereum's key characteristics is its ability to use "Smart Contracts," which are pieces of code that execute when specific conditions are met. These events are interpreted by the Ethereum web3.js API and can be used to change web pages. Users interact with the Ethereum network through Metamask, a Chrome extension that links to an Ethereum wallet. With the help of the Metamask plugin and a bitcoin wallet, users may interact with the DApp and complete transactions. This makes blockchain programming feasible. It is projected that browsers would support DApps more frequently and by default as they become more popular, boosting user awareness of them.

## **1.2 Problem Statement**

There are several drawbacks of using traditional transaction platforms over Blockchain-enabled transaction platforms, including:

- • Lack of transparency: Due to their reliance on centralised systems managed by a single business or middleman, traditional transaction platforms sometimes lack transparency. Due to the potential for restricted transaction process visibility, it may be challenging to verify or audit transactions.
- • Lack of security: Because they are centralised, traditional transaction systems are vulnerable to fraud, cyberattacks, and data leaks. A single database's data can be manipulated or subject to unauthorised access, perhaps resulting in the loss of private data or money.



- Why Slow and expensive transactions: Complex processes, numerous middlemen, and manual reconciliation may be used in traditional transaction platforms, which causes delays and higher transaction costs. Additionally, due to the involvement of numerous parties and the necessity of regulatory compliance, cross-border transactions can be time-consuming and expensive.
- Reliance on intermediaries: To support transactions, traditional transaction platforms frequently need intermediaries like banks, clearinghouses, or other third-party companies. This could lead to higher costs, longer wait times, and a greater reliance on middlemen—things that a Blockchain-enabled transaction platform might not require.
- Limited traceability: Because traditional transaction systems may not provide end-to-end tracking, it can be challenging to follow the movement of assets or items throughout a transaction. As a result, there may be less accountability and a higher chance of fraud or disagreements.
- Lack of decentralisation: Conventional transaction platforms are centralised, which means that one entity or a small group of entities controls them. This centralization may not be appropriate for applications that call for decentralised decision-making, trust, and ownership because it can result in a concentration of power and control.
- Inefficient reconciliation and settlement procedures: Complex reconciliation and settlement procedures are frequently needed on traditional transaction platforms, which can be time-consuming, costly, and error-prone. On the other side, blockchain-enabled transaction systems can automate these procedures, lessening the chance of mistakes and delays.
- Limited Smart Contract implementation capabilities: Smart Contracts are self-executing contracts with established conditions that can be automatically enforced on a Blockchain. Traditional transaction

platforms may not be able to execute them. With the use of smart contracts, transactions may be automated and transparent, which decreases the need for middlemen and boosts productivity.

### **1.3 Objectives**

The Objective of our project is to build a decentralised application that would ensure secure transaction, provide an easy to use interface that allows users to send transaction through Blockchain and to reduce the third party involvement.

### **1.4 Methodology**

Each node in an Ethereum network holds a copy of the Blockchain, which contains a history of all transactions on the network, and each node also runs an Ethereum client. The network's distributed structure deters fraud and preserves an auditable record of every transaction. On the Ethereum network, user pseudonymity is preserved because each user is identifiable by a public key, enabling them to carry out operations like money transfers.

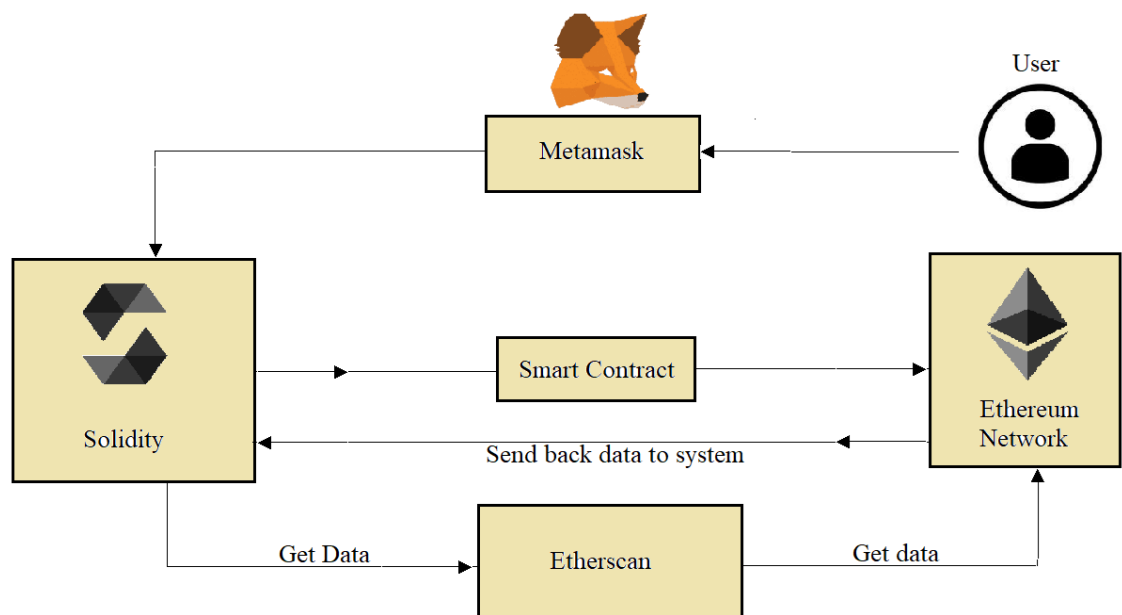
The web3.js API lets programmers communicate with the Ethereum network and gives web apps a way to read events and send transactions. Consumers can access the Ethereum network through Blockchain-enabled extensions and browsers like Metamask for Google Chrome and the Mist browser, giving them control over their ether wallets and allowing them to perform transactions.

To complete a transaction, a fixed amount of ether called "gas" is transferred to the node processing the transaction as an incentive for its completion. The time it takes to create a contract or process a transaction depends on the complexity of the contract or application logic, as well as the system load and number of concurrent transactions occurring on the network.

Solidity is statically typed and supports complicated user-defined types, libraries, and inheritance. Solidity is statically typed, so each variable must be specified by the user[4]. The compiler can verify that variables are being used correctly thanks to data types. Information types for robustness are ordinarily isolated into esteem types and reference types.

We didn't utilize the principal Ethereum network since this framework is just a model; all things considered, we utilized testnet, which performs in much the same way to the primary Ethereum organization. To support exchanges done by clients in this framework, we utilize a proof-of-stake Blockchain called the Goerli test network for this task.

We can't mine Ether because we're on the Goerli test network; instead, we'll have to request it from the Goerli Test Faucet. The Etherscan API may be used to view the specifics of user transactions, whether they were successful or not. The diagram of system architecture is mentioned below.



**Figure3. System Architecture**

The internal organisation of the decentralised application is as follows:

1) Frontend: HTML, CSS, and JavaScript combined with extensive use of web3js. Through this frontend application, users can communicate with the NodeJS server and the Blockchain.

2) Blockchain: All of the application's code and transactions are kept in this section.

### **1.4.1 Software Requirements**

#### **1. Solidity**

All contract code is written in Solidity because it is the language that is used to install contracts on the Blockchain platform. A programming language called robustness is used to create clever decisions that are subsequently carried out on a Blockchain network[11]. These Smart Contracts serve as a representation of all business transactions. Blockchain transactions are automated by smart contracts, which are written in the object-oriented, high-level programming language Solidity. Contributors to the Ethereum project developed the language after it was first suggested in 2014. This language is supported by the Ethereum Blockchain as well as other Blockchains for the development of Smart Contracts.

With Solidity, there is no need to type code in ones and zeros because it is a high-level language. Humans can write programmes much more easily by combining letters and numbers in a way that makes sense to them.

Solidity is statically typed and supports complicated user-defined types, libraries, and inheritance. Solidity is statically typed, so each variable must be specified by the user[4]. The compiler can verify that variables are being used correctly thanks to data types. Information types for robustness are ordinarily isolated into esteem types and reference types.

The EVM is one of the essential elements that allows Solidity code to be executed. The EVM is referred to as a virtual computer on the Blockchain that converts user ideas into code that powers Blockchain applications.

Solidity generates machine-level code that is executed on the EVM in the background. High-level, legible code is broken down by a compiler into instructions that the processor can interpret[7]. Free Solidity compilation is available on a variety of platforms, including the Remix online compiler and a PC-downloaded command-like compiler.

Both fungible and non-fungible token Smart Contracts are made with Solidity. In the Ethereum ecosystem, fungible and non-fungible tokens are created using various standards.

These enable the development of many use cases for those who use the Blockchain. On Ethereum, tokens and non-fungible tokens can be used thanks to Solidity[8]. Different kinds of uses for tokens are made feasible by Ethereum, from creating non-fungible tokens to include them in yield farming pools for added interest.

## **2. Remix tool**

Smart Contracts are tested while they are being produced using a tool. Remix can be used to test on a Goerli test organisation or a virtual organisation. The Campaign Factory is created, which includes all of the source code needed to launch new campaigns across the network. New campaigns can be designed with the help of campaign factory. A one-time gas fee is required whenever a campaign factory is deployed, and it is a relatively tiny sum. The new Campaign is initially developed by providing the project's Idea, the project's Minimum Contribution, and a full description of the initiative. A block is created and added to the Blockchain when a new campaign is created.

## **3. MetaMask**

With the assistance of the cryptographic money wallet MetaMask, clients can keep their Ether and other ERC-20 tokens safe. Moreover, clients of the wallet can speak with decentralized applications, or dApps.

Similarly as with other program modules, MetaMask should be introduced before it very well may be utilized as an Ethereum wallet[15]. Once stacked, clients can utilize any Ethereum address to execute by putting away Ether and other ERC-20 tokens.

Customers can link MetaMask to Ethereum-based dApps (DEXs) to utilise their money in games, stake tokens in betting apps, and exchange them on decentralised transactions. Additionally, it provides users with a way to access emerging decentralised financial (DeFi) services like Compound and PoolTogether.

Digital currencies and decentralized applications act as the foundation of the decentralized web, or Web3 (dapps). Be that as it may, you really want a UI to utilize them. One of the most well known digital money wallets, MetaMask, centers around program incorporation and alluring plan to go about as a key passage point into the domain of Web3, decentralized finance (DeFi), and NFTs.

The wallet doesn't expect clients to enter any by and by recognizable data to utilize it, dissimilar to a bitcoin trade[10]. The cryptocurrency that users gather and sell through their wallets, including their private seed phrase, are entirely under their control. Users' secret phrases will never be requested by technical support, and MetaMask does not keep user information.

While using their MetaMask wallet, clients are allowed to carry on with work and trade coins or NFTs, however the item alerts that Blockchain exercises are not altogether mysterious. They mark connections as "pseudonymous," all things considered. Since each exchange is recorded openly on the Blockchain, the wallet ID capabilities as a pen name the client. The client's real personality might be compromised on the off chance that they interface their wallet to their character by using a NFT as their Twitter profile picture or an Ethereum Name Administration space.

#### **4. Web 3.0**

Web 3.0 or Web3 is the term used to describe the third Internet age. It is a concept for a more valued, decentralised, and open Web that is still under development.

The World Wide Web (WWW), the primary information search engine on the internet, is referenced by the word "web." When looking for a specific resource online, the WWW initialism was one of the first letters entered into a web browser, and it still regularly comes before a web URL. Internet pioneer Tim Berners-Lee is credited with coining the phrase "World Wide Web," which refers to the massive network of information and services connected by hypertext links.

Web 2.0 and Web 3.0 are the subsequent, more modern iterations of the original Web 1.0 from the 1990s and the middle of the 2000s[20]. The next phase of the web, which we are currently building, is called Web 3.0, and it will be more decentralised, open, and useful than Web 2.0.

Thanks to technological advancements like social networks, mobile internet access, and cellphones, Web 2.0 has grown quickly. Web 2.0 has disrupted industries that have not adjusted to the new web-based economic paradigm.

Beyond the online gaming, streaming, and shopping that make up the majority of Web 2.0 applications used by users, Web 3.0 may be able to provide users with much more valuable utility[13]. At the centre of Web 3.0 is the Semantic Web, which has the potential to expand applications into new geographies and improve client collaboration.

Decentralization, trustlessness and permissionlessness, man-made reasoning (simulated intelligence) and AI, connectedness, and omnipresence are attributes that best portray Web 3.0.

Through decentralized information organizations, clients will actually want to sell the information delivered by different and strong figuring assets, like cell phones, PCs, machines, vehicles, and sensors, while as yet keeping up with proprietorship control.

## **5. Ethereum Virtual Machine**

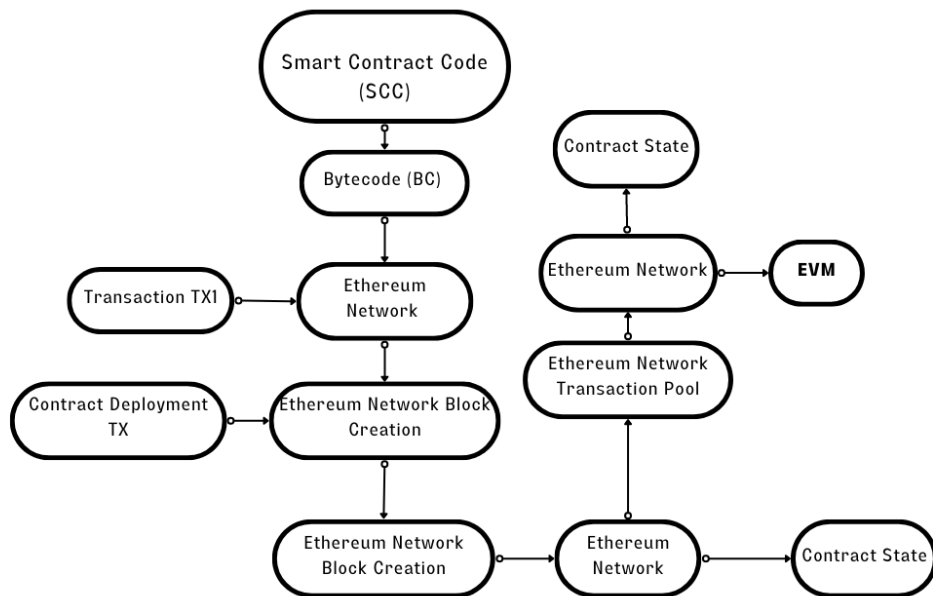
EVM, or Ethereum Virtual Machine, is a runtime climate for ethereum's Smart Contract. It places a lot of emphasis on supplying security and enabling the execution of untrusted programmes across a global network of open nodes. Denial-of-service attacks are specifically prevented by EVM, which also verifies that programmes cannot access one another's state and that communication is created without any potential interference.

The environment in which Ethereum Smart Contracts are run is known as an EVM. There is a necessity to run this code in light of the fact that Ethereum has its own Turing-complete programming language called Robustness[6]. This errand can be completed utilizing a program known as the Ethereum Virtual Machine (EVM).

Since it is based on top of the Ethereum organization, every one of the hubs settle on what code ought to be performed at a specific second. A programme called Ethereum Virtual Machine (EVM) runs scripts that are typically used to implement specific activities in the Ethereum Blockchain.

Making new tokens on the Ethereum Blockchain is simple thanks to the Ethereum Virtual Machine. An algorithm or collection of instructions that teaches the computer what to do in order for something to function properly is referred to in this context as a "script." In order to execute the required commands and easily create new tokens on the Blockchain, the EVM demands that one has access to any network node.





**Figure 4. Flowchart of EVM**

Smart Contract Code (SCC): A developer writes Smart Contract code in a high-level programming language like Solidity. The code is compiled into bytecode that can be executed by the Ethereum Virtual Machine.

Bytecode (BC): The bytecode is the compiled version of the Smart Contract code that can be executed by the EVM.

Transaction TX1: A user creates a transaction to deploy the Smart Contract to the Ethereum network. The transaction contains the bytecode of the Smart Contract.

Ethereum Network: The transaction is broadcasted to the Ethereum network and validated by miners who add it to the Blockchain as a new block.

Contract Deployment TX: Once the transaction is validated, the Smart Contract is deployed to the Ethereum network, and a new address is assigned to it.

Contract State: The Ethereum network maintains the state of all Smart Contracts deployed on the network.

Contract Method Call TX: A user creates a transaction to call a method on the deployed Smart Contract. The transaction contains the method call parameters.

EVM: The Ethereum Virtual Machine executes the bytecode of the method call and updates the state of the Smart Contract.

Contract State: The state of the Smart Contract is updated, and the changes are stored on the Ethereum network.

Transaction TX2: The transaction is complete, and the user receives the result of the method call. The Ethereum network records the transaction on the Blockchain.

## CHATER 02: LITERATURE SURVEY

The distinctions and parallels between conventional fundraising methods and the peer-to-peer lending industry have been emphasised by Alexander Backmann [5]. The amount raised, the screening procedure, and risk management procedures used in these techniques differ. The outcomes of peer-to-peer lending may be applicable to conventional fundraising, or vice versa, depending on the results of this research. This study also focuses on conventional fundraising strategies, where failure rates were high and returns on investment were often low.

Crowdfunding has many benefits over other options for new businesses and small and medium-sized organisations (SMEs), according to a study on its consequences in India [2]. Even if crowdfunding is still not readily available to the general public, the younger generation's growing awareness of it is a positive first step in the platform's development. It may make it possible for new businesses to attract a larger range of financiers and raise financing from them.

Blockchain technology is still in the exploratory stage and faces technological and regulatory obstacles before it can be made freely available to the public, according to Huasheng Zhu and Zach Zhizhong Zhou [6]. The way that market and corporate influencers collaborate to develop new concepts and utilise Blockchain technology in the market has room for improvement. There has to be a deeper knowledge of Blockchain innovation, its value, potential, and hazards if financial efficiency and societal advantages are to be realised. Blockchain application development may be accelerated by aggressively promoting them in the Chinese crowdsourcing market.

With regard to crowdfunding, Michael Gebert [7] has discussed its uses and importance, especially for small enterprises that are constantly threatened by job uncertainty and shortages. Governments must make it easier for small

businesses to get capital, especially given the hostile conditions in several European countries where crowdfunding has not been successful. The expansion of crowdfunding platforms is essential to the ability of small enterprises to function and raise capital.

In the paper "Crowdsourcing and Crowdfunding Platform using Blockchain and Collective Intelligence" [8], it is analyzed that crowdfunding and crowdsourcing in India are still in the early stages of adoption. Despite being a relatively new concept, crowdfunding has not yet been widely embraced by the Indian population. However, the future of crowdfunding and crowdsourcing in India is expected to be bright. Business capital and human resources are essential requirements for any business, especially for startups and small organizations that often struggle with resource allocation. The use of Blockchain in these platforms can enhance security. Although the potential of such platforms in India is promising, public participation is crucial for their success.

## **Hyperledger**

Giving the fundamental establishment, rules, rules, and devices to build open-source Blockchains and related applications for use across various ventures, Hyperledger is a worldwide business Blockchain project [16]. Projects from Hyperledger include a range of permissioned Blockchain systems that are business-ready, where network users are familiar with one another and have an inherent incentive in taking part in consensus-making.

A company can implement multiple modular Blockchain solutions and services using the parts that are offered under the Hyperledger umbrella to dramatically increase the effectiveness of their operations and business processes.

A gathering called Hyperledger is attempting to make a steady arrangement of structures, devices, and libraries for permissioned, endeavor grade Blockchain organizations.

It is an international partnership that The Linux Foundation is hosting, and its member companies include some of the top names in supply chains, manufacturing, technology, finance, banking, and the Internet of Things[17]. There are several related initiatives, such as Hyperledger Fabric, Sawtooth, Composer, and Cello.

The San Francisco-based Linux Foundation launched the Hyperledger project in December 2015. Today, there are more than 120 member companies, up from 30 when it first started.

To work on the effectiveness, execution, and exchanges of different business tasks, Hyperledger was laid out fully intent on accelerating broad cooperation for the improvement of superior execution and reliable Blockchain and disseminated record based innovation system.

Driving organizations from the monetary, banking, Web of Things (IoT), store network the board, assembling and creation, and innovation areas are important for the worldwide joint effort known as Hyperledger[18]. Alongside various Blockchain-based new businesses like Blockforce and ConsenSys, they incorporate commonly recognized names like Bosch, Daimler, IBM, Samsung, Microsoft, Hitachi, American Express, JP Morgan, and Visa.

Generally, Hyperledger is neither an organization nor an organization of digital forms of money nor a Blockchain framework. In spite of the fact that it doesn't uphold a digital currency like bitcoin, it capabilities by offering the necessary norms and framework for the making of an assortment of Blockchain-based frameworks and applications for use in the modern area[2]. Believe Hyperledger to be a center where various free Blockchain-based tasks and instruments that follow its predetermined plan methods of reasoning work.

The several initiatives comprise the following:

- A technology called Hyperledger Fabric is used to create numerous Blockchain-based goods, services, and business applications. Since then, Hyperledger Composer, a defunct layer, has been combined into Fabric.

-Blockchain might be used with Hyperledger Cello on account of its on-request "as-a-administration" organization procedure (Blockchain-as-a-Administration).

- A dashboard instrument called Hyperledger Wayfarer makes it conceivable to monitor Blockchain improvements and other data.

- On the Ethereum Virtual Machine, the permissioned Ethereum Smart ContractBlockchain hub known as Hyperledger Tunnel oversees exchanges and runs savvy contract code (EVM).

- A clever Confirmation of Slipped by Time agreement calculation is utilized by the undertaking level, permissioned, particular Blockchain stage known as Hyperledger Sawtooth.

-A Blockchain benchmarking tool called Hyperledger Caliper is used to assess the effectiveness of a particular Blockchain implementation.

-These Hyperledger-partnered projects stick to a plan worldview that advances interoperability, a particular and extensible methodology, and security highlights[9]. The ventures support no particular tokens or digital forms of money, however clients can fabricate them on a case by case basis.

The essential business components utilized by Hyperledger in its design are as per the following:

-The contract layer is responsible for laying out a comprehension of the grouping and approving the exactness of the assortment of exchanges that make up a block.

-Handling exchange demands and supporting just genuine exchanges are the obligations of the brilliant contract layer.

-The exchange of messages between peers is taken care of by the correspondence layer.

-To keep up with and approve clients' and frameworks' characters and construct trust on the Blockchain, personality the executives administrations are a necessity.

-The Blockchain can be interfaced with by third-party programmes and clients thanks to the API, or application programming interface.

An open, tried, undertaking grade appropriated record stage is Hyperledger Fabric[6]. It highlights complex protection controls so just the data you need shared will be imparted to the "allowed" (known) network clients.

The business processes you really want to robotize are filed in wise agreements with self-executing conditions between the social affairs that are written in lines of code. The appropriated, decentralized Blockchain network is contained the code and arrangements included therein[5]. Since exchanges are retraceable and irreversible, associations can trust each other. Because of the capacity to rapidly pursue better choices, organizations can save time, cash, and hazard.

## **Hyperledger Fabric**

Hyperledger Fabric is a permissioned Blockchain rather than a public Blockchain like Bitcoin and Ethereum where anybody can join and take part in the organization. Just a gathering of organizations related associations can join through a participation specialist co-op, and network is comprised of friends are claimed and contributed by those organizations. For records and chaincodes, peers act as hosts (Smart Contracts). The successive, impervious record of exchanges and state changes is known as a record. Chaincode summon causes an adjustment of state (exchange)[19]. Every exchange produces a bunch of key-esteem matches for resources that are then dedicated as manifestations, updates, or cancellations to the record.

Coming up next are a few different ways that Ethereum and Hyperledger Fabric are unique. Most importantly, just an assigned local area of members has approval to join the organization, while Hyperledger Fabric is a consortium Blockchain foundation. This recognizes Ethereum from Hyperledger Fabric as a public Blockchain stage. Conversely, Hyperledger Fabric offers a secluded plan with a qualification between hub responsibilities

(regarding model, endorsers and orderers) and configurable agreement and enrollment administrations, which brings about elevated degrees of versatility, vigor, and privacy. Second, neither fuel nor digital currencies are coordinated into Hyperledger Fabric (like Ether and gas in Ethereum). Third, the Hyperledger chaincode simply depicts an assortment of resources that are displayed as key-esteem coordinates and offers the capabilities for controlling the resources and changing their states.

A fitting and-play Blockchain design called Hyperledger Fabric fills in as a spine for making Blockchain-based products, administrations, and applications that are planned for use by confidential organizations[20]. In December 2016, the Linux Establishment presented Hyperledger, an open-source appropriated record framework intended for organizations. Decentralized record innovation (DLT) stage called Fabric was made by IBM for use in modern ventures.

Organizations can isolate data (like costs) in light of the fact that Hyperledger Fabric is private and expects approval to get to. Also, on the grounds that there are less hubs on the organization, exchanges can continue all the more rapidly. January 2020 saw the arrival of Fabric 2.0. Quicker exchanges, modernized Smart Contract innovation, and improved on information sharing are the primary elements of this form.

The Linux Foundation is now hosting Hyperledger Fabric, a collaborative cross-industry project that was started by Digital Asset and IBM. In March 2017, Fabric became the first Hyperledger project to move from the "incubation" stage to the "active" level.

The confidential exchanges and private agreements that are significant for organizations can't be upheld by customary Blockchain networks. Accordingly, the secluded, adaptable, and secure Hyperledger Fabric stage was made to give modern Blockchain arrangements. The open-source Blockchain motor known as Hyperledger Fabric handles the critical parts for evaluating and using Blockchain for business use cases.

Evident member personality is a pivotal prerequisite for private modern organizations. All organization members should have notable characters, and



Hyperledger Texture upholds participations in light of approval. Information security regulations command the upkeep of data about the various members and their separate admittance to different data of interest in numerous business regions, including medical care and money. Such permission-based membership is supported by Fabric.

Some example of Hyperledger are as follows:

On the off chance that a maker wishes to send chocolates to a particular store or market of retailers (all US shops, for instance) at a specific cost however doesn't have any desire to reveal that valuing in different business sectors, what might occur? (i.e., Chinese retailers).

If a basic execution of Blockchain innovation is used to empower this exchange, the confidential valuing might be unveiled to all closely involved individuals since the exchange of the ware might include extra gatherings, for example, customs, a transportation firm, and a supporting bank.

By maintaining private exchanges mystery on the organization, Hyperledger Fabric tackles this issue by guaranteeing that main those gatherings who require the data know about it[17]. On the Blockchain, information apportioning empowers specific information focuses to be open just to individuals who require them.

Other Hyperledger projects like Iroha, Indy, and Sawtooth contend with Hyperledger Fabric. It likewise faces rivalry from R3's Corda, a private, consent based DLT.

## **Hyperledger Sawtooth**

Under the Hyperledger brand, the open source Hyperledger Sawtooth project fills in as an undertaking level Blockchain development for building and running appropriated record associations and applications, famously for business use.

An open source try Blockchain-as-a-organization stage called Hyperledger Sawtooth licenses clients to run changed savvy contract without being have a lot of experience with the system's internal plan.

A broad based Blockchain improvement consortium called Hyperledger is maintained by associations like SAP, IBM, Intel, and the Linux Undertaking[19]. Different understanding techniques, as Practical Byzantine Variation to non-basic disappointment (PBFT) and Confirmation of Sneaked past Time, are maintained by Hyperledger Sawtooth (Essayist).

The middle arrangement thinking of Hyperledger Sawtooth, made by the Linux Foundation in association with IBM, Intel, and SAP, means to stay aware of the records really scattered and make sagacious agreements evidently more secured and as needs be useful for attempts. It uses Blockchain development as an assistance (BaaS).

Most of traditional Blockchain-based frameworks host and run their center and applications on a similar stage, which could cause execution issues and security challenges.

By separating the concealed record structure from the environment relevant to each application, Hyperledger Sawtooth makes it more clear to design applications while staying aware of system security[1]. With the help of this designing, an originator can create applications in their leaned toward programming language that can be worked with, made due, and used past the essential Blockchain network.

The middle structure picks the trade rules, picks the required permissioning framework, and decides the understanding computations that are used to complete the action of the electronic record with the end goal that best sponsorships the necessities of an endeavor. It in like manner permits applications to match on the comparable Blockchain.

## **Hyperledger Composer**

A set-up of open source instruments called Hyperledger Composer gives business people, administrators, and engineers a component to fabricate Blockchain applications and Smart Contract that address business issues or potentially help functional viability. It fills in as a delineation of a Blockchain-as-a-administration application for business (BaaS). One of the numerous Hyperledger drives facilitated by The Linux Establishment in relationship with colleagues is Hyperledger Author.

The Hyperledger Arranger project is in censured status as of August 2019, and that implies that even while it is still being used, the maintainers are not really effectively dealing with new elements or offering support. Hyperledger Texture v1.4+ now incorporates Arranger.

The programming language Javascript, which is stage autonomous and upholds the utilization of underlying libraries as well as the use of promptly accessible capabilities and contents to make the utility more adaptable and reusable, is utilized to make the Hyperledger Author[10]. Composer is an application improvement structure that makes it more straightforward and quicker to make Blockchain applications for the Hyperledger texture.

A financial specialist without specialized skill can undoubtedly work with a designer to fabricate specific elements utilizing Hyperledger Composer. They incorporate recognizing the resources that are exchanged Blockchain-based use cases, the business decides that will be utilized to handle exchanges, and the controls for members, their personalities, obligations, and access levels for doing the different kinds of exchanges.

The organization's computerized resources, exchange rationale, members, and access controls are among the critical parts of the Blockchain that an engineer may basically plan and redo while using Hyperledger Composer. Inside various ventures, Arranger works with the sharing, reusing, and scaling of parts. Utilizing Hyperledger Author, one may rapidly deliver the essential contents and APIs for business execution. It additionally offers continuous testing and use cases, which can be completed even without neighborhood establishments utilizing the online Author jungle gym.

An individual can fabricate and run a test Blockchain utilizing Hyperledger Composer, and they can give various clients limited admittance to it[12]. For example, it is easy to make a "Transient Products Organization" that empowers the trading of merchandise like foods grown from the ground, incorporates members like ranchers, transporters, and merchants, characterizes individual jobs for every member, characterizes and executes terms of understanding between the members, tracks shipments, recognizes, screens, and reports the situation with the merchandise at different places in the store network, and oversees installments.

## **Hyperledger Iroha**

A Blockchain platform called Hyperledger Iroha is intended to be quickly integrated into a variety of commercial applications that call for distributed ledger technology. The platform, for instance, may assist businesses and governments with identity management, including the creation of national IDs, and the financial services industry with bank-to-bank transfers.

Hyperledger seeks to develop distributed ledger technology, according to the company's website, "that enables enterprises to construct and execute reliable, industry-specific apps, platforms, and hardware systems to support their distinct business transactions."

A commercial Blockchain architecture called Hyperledger Iroha is intended to be implemented into infrastructure projects that require distributed ledger technology[13]. A Blockchain's distributed ledger function functions similarly to a public database, enabling the sharing of data. However, many companies may utilise a private Blockchain network as a foundation to create software programmes, or apps, for their own internal use or to provide customers with technology-based goods.

The platform offered by Hyperledger Iroha enables customers to create applications tailored to their own business requirements, notably for mobile apps. It uses a C++ design that is domain-driven, a language used by software

developers. Additionally, Iroha includes the YAC consensus method (for Yet Another Consensus algorithm). A coded, step-by-step process called an algorithm is created to solve issues and carry out a series of instructions.

Features :-

- When an application requires multiple signatures for transaction settlement, Hyperledger Iroha features multisignature (or multiple key) functionality for transactions.

- Support for employing programming languages like Java, JS, Python, and iOS to create apps for many platforms (such as mainframe and mobile).

- Many compatible operating systems, such as Linux, macOS and Windows.

- Plug-in, modular architecture makes it simple for developers to set up and use a Blockchain.

By separating the concealed record structure from the environment relevant to each application, Hyperledger Iroha makes it more clear to design applications while staying aware of system security[1]. A financial specialist without specialized skill can undoubtedly work with a designer to fabricate specific elements utilizing Hyperledger Iroha. With the help of this designing, an originator can create applications in their leaned toward programming language that can be worked with, made due, and used past the essential Blockchain network.

Iroha makes it simple to deploy and maintain applications, offers a wide variety of code libraries to developers to make it easy to create applications, secure control[4] and permissions over user roles and activities, simple asset management, participant identification, and a modular design architecture to support the Blockchain ecosystem.

For instance, the Blockchain of Hyperledger Iroha is being used to establish insurance contracts such as weather derivatives by the Japanese-based global casualty and property insurance business Sompo Japan Nipponkoa Holdings

Inc. These financial contracts, known as derivatives, are used to insure against or protect the insurer from weather-related losses.

**Table.1. Comparative Analysis of Blockchain Methodologies and their Limitations**

Author(s)	Journal year	Published by	Methodology	Disadvantage
Rasheed Mohammad Nassr, Megat F.	2017	ACM Press	Solidity, Goerli	Used Proof of Stake
Yadav, Nikhil, and V. Sarasvathi	2019	IEEE	Bytecode, Ethereum live network	Used proof of work
Saadat, M. Nazmus	2019	Indonesian Journal of Electrical Engineering and Computer Science	Solidity, Rinkeby network	Used proof of Burn

## **CHAPTER 03: SYSTEM DEVELOPMENT**

### **3.1. Proof of Stake (PoS)**

Proof-of-stake (PoS) is a consensus mechanism used by Blockchains to achieve distributed consensus in a more energy-efficient and scalable way compared to proof-of-work (PoW). In PoS, validators, also known as stakers, participate in the block validation process by locking up a certain amount of cryptocurrency, typically in the form of tokens, as collateral, which is referred to as "staking." This staked cryptocurrency acts as a guarantee or collateral that can be destroyed, or "slashed," if the validator behaves dishonestly or fails to fulfill their responsibilities.

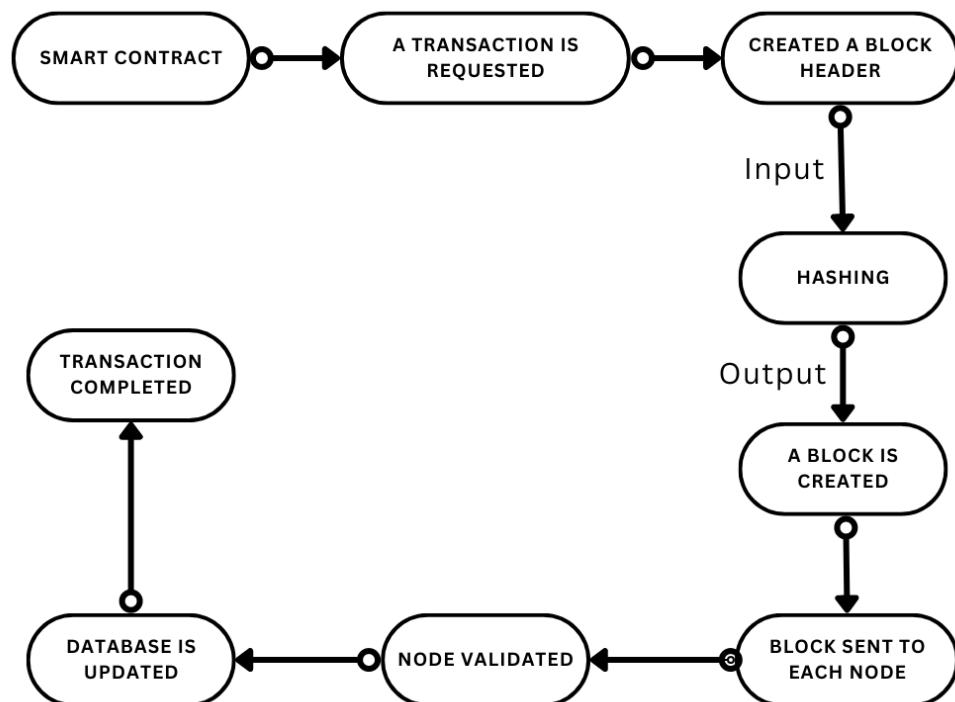
Ethereum, the second-largest cryptocurrency by market capitalization, is in the process of transitioning from PoW to PoS through its Ethereum 2.0 upgrade. In Ethereum's PoS mechanism, known as the Ethereum 2.0 Beacon Chain, validators explicitly stake capital in the form of Ether (ETH), the native cryptocurrency of the Ethereum Blockchain. Validators are required to lock up a certain amount of ETH as collateral in a Smart Contract on the Ethereum Blockchain, which they cannot access or spend during the staking period.

Validators in Ethereum's PoS system are responsible for checking the validity of new blocks propagated over the network and occasionally creating and propagating new blocks themselves. Validators take turns proposing and attesting to blocks, and the probability of being selected as a validator to propose a block is proportional to the amount of ETH they have staked. Validators are also required to actively participate in the consensus process and follow the rules of the Ethereum protocol. If a validator behaves

dishonestly or fails to fulfill their duties, their staked ETH can be slashed or destroyed as a penalty.

Proof-of-stake comes with a number of improvements to the now-deprecated proof-of-work system:

- fewer barriers to entry, lowered hardware requirements - there is no need for elite hardware to stand a chance of creating new blocks
- higher energy efficiency - there is no need to use lots of energy on proof-of-work computations
- Because proof-of-stake requires less energy, fewer ETH tokens need to be issued to encourage participation. Economic penalties for bad behaviour make 51% style attacks exponentially more expensive for an attacker compared to proof-of-work. The community can turn to social recovery of an honest chain if a 51% attack were to succeed despite the crypto-economic defences.



**Figure 5. Working of Blockchain protocol**



### 3.1.1 HOW A TRANSACTION GETS EXECUTED IN ETHEREUM POS

The accompanying gives a start to finish clarification of how an exchange gets executed in Ethereum confirmation of-stake.

- A client makes and signs an exchange with their confidential key, as a rule through a wallet or a library like ether.js or web3.js, and sends a solicitation to a hub utilizing the Ethereum JSON-RPC Programming interface. The client determines the gas charge they will pay as a tip to a validator for remembering the exchange for a block, with the tips going to the validator and the base expense getting scorched.
- The exchange is submitted to an Ethereum execution client, which confirms its legitimacy by checking in the event that the source has enough ETH and has marked the exchange accurately with the right key.
- In the event that the exchange is legitimate, the execution client adds it to its neighborhoodmempool, which is a rundown of forthcoming exchanges, and broadcasts it to different hubs over the execution layer tattle organization. Different hubs likewise add the exchange to their neighborhoodmempool. High level clients might decide to advance their exchange to particular block developers like Flashbots Closeout for ideal association of exchanges in impending blocks to boost benefits from MEV (Excavator Extractable Worth).
- One of the hubs on the organization is chosen as the block proposer for the ongoing opening utilizing RANDAO, a pseudo-irregular determination process. The block proposer is liable for building and broadcasting the following block to be added to the Ethereum Blockchain, as well as refreshing the worldwide state. The hub comprises of an execution client, an agreement client, and a validator client. The execution client bunches exchanges from the neighborhoodmempool into an "execution payload" and executes them locally to create a state change. This data is then passed to the agreement client, which wraps the execution payload as a component of a "reference point block" that incorporates different subtleties like prizes, punishments, slashings, and

verifications. These subtleties assist the organization with settling on the arrangement of blocks in the chain in view of the fork decision rules.

- Different hubs get the new signal block through the agreement layer tattle organization. They pass it to their execution client, which re-executes the exchanges locally to check the proposed state change. The validator client then, at that point, verifies that the block is legitimate and is the coherent next block in their perspective on the chain, meaning it expands on the chain with the best weight of verifications in light of the fork decision rules. The block is added to the neighborhood data set of every hub that bears witness to it.

- The exchange is thought of "settled" and irreversible in the event that it turns out to be important for a chain with a "supermajority connect" between two designated spots. Designated spots happen toward the start of every age, and a supermajority interface requires verification from 66% of the all out marked ETH on the organization.

## **3.2 Hashing**

Hashing is a cycle that converts input information of any length into a fixed-size string utilizing a particular calculation. For instance, in Bitcoin, the hash calculation utilized is SHA-256 (Secure Hashing Calculation 256 pieces). This calculation is a one-way cryptographic capability, implying that the first information can't be recovered through unscrambling.

### **3.2.1 How Hashing Works in Blockchain**

A hashing calculation takes an endless number of pieces as info and performs estimations to yield a decent number of pieces, bringing about a fixed-size hash no matter what the info information's length. The first information is alluded to as information, and the subsequent change is known as a hash. Information structures assume a significant part in understanding hashing, as they include pointers and connected records. Pointers are factors that demonstrate the area of different factors and give locations to the following block in the chain. Connected records, then again, comprise of hubs associated by pointers.

In the Blockchain, hashing is utilized to relegate a remarkable identifier to each impede, which has irreversible ramifications for changing the Blockchain. The block is recognized by data remembered for the header of the block. It comprises of such subtleties as:

- the adaptation number of the Blockchain
- UNIX timestamp
- hash pointers
- nonce, which is the worth the diggers need to make a block
- a hash of a Merkle root

This large number of components are vital for making a block, and while hashing happens in the Blockchain, the information is changed over into an exceptional string inside a block.

### **3.3 Smart Contracts**

Smart Contracts are executable codes that work on top of the Blockchain. Network mechanization and the ability to change over paper contracts into computerized ones were made conceivable by Smart Contract[9]. Rather than customary agreements, Smart Contract offered mechanized exchanges without the oversight of a focal power, empowering clients to formalize their arrangements and trust connections. Smart Contracts are duplicated to every hub of the Blockchain organization to forestall contract control. Human mistake could be diminished to bring down the probability of contentions including such agreements by empowering the execution of activities by machines and utilizing the abilities presented by Blockchain stages.

#### **3.3.1 Operational process of Smart Contracts**

A common understanding between at least two gatherings is known as a Smart Contract. Because of its pre-characterized capabilities, it stores information, handles inputs, and creates yield. One more illustration of a capability that can be determined in a Smart Contract is the fall to pieces capability. By utilizing

this capability, normally just the Smart Contract proprietor can obliterate the agreement.

To execute and control relevant occasions and exercises as per the guidelines of the agreement, a Smart Contract is normally a class that contains state factors, capabilities, capability modifiers, occasions, and designs. It can likewise call other brilliant agreements, furthermore. States and works are remembered for each Smart Contract. We might recognize two different state types: writable states, which can keep states in the Blockchain, and consistent states, which can never be changed. These last option codes can peruse or alter states. We might recognize two distinct kinds of capabilities: compose works and read-just capabilities. Compose capabilities need gas since state advances should be encoded in new Blockchain blocks, while read-just capabilities don't. Moreover, paying with cash is important to endlessly prevent shrewd agreements from running.

Users can call any available Smart Contract function by sending a transaction once a Smart Contract has been deployed and the creator has received the returned parameters (such as the contract address).

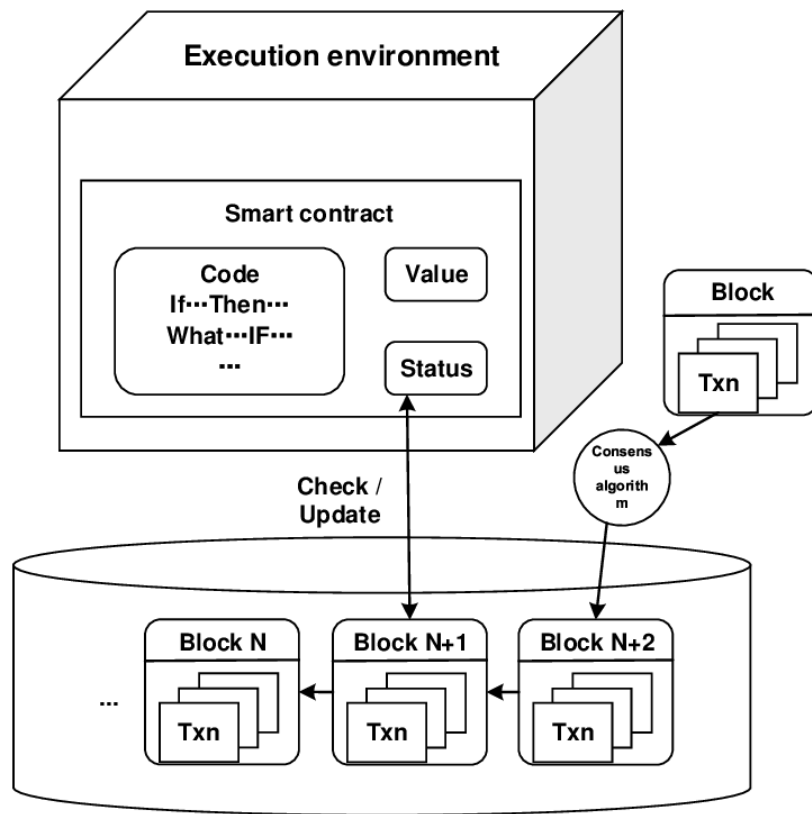
### **3.3.2 Operational Mechanism of Smart Contracts**

Smart Contracts consist of two main aspects: value and state. They utilize triggering condition statements, such as "If-Then" statements, to determine how the contract terms will react to specific triggering situations. All parties involved in the Smart Contract must agree to and sign it before it can be submitted as a transaction to the Blockchain network.

Once the Smart Contract transaction is submitted, it is broadcast across the peer-to-peer (P2P) network for verification by miners. After the transaction is verified by miners, it is included in the appropriate Blockchain block along with other transactions.

Clients can summon decreases by sending an exchange after the makers of the agreements get the returned boundaries, (for example, the agreement address).

Diggers will contribute their handling ability to approve the exchange in light of the framework's motivating force structure. The reaction activities are completely done in the event that the requirements are fulfilled. An exchange is packaged into another block after it has been endorsed. When everybody on the organization concurs, the new block is added to the Blockchain.



**Figure 6. Operational mechanism of Smart Contract <sup>[9]</sup>**

### 3.3.3 Ethereum:

Ethereum, an exchange based state machine, has turned into the most generally involved stage for creating Savvy Agreements. It begins with a beginning state and uses exchanges to change it into various end states, which are perceived as the "form" that is considered sanctioned in the Ethereum universe.

In Ethereum, there are two kinds of records: remotely claimed accounts (EOAs) and contract accounts. EOAs are constrained by confidential keys and don't have related code, while contract accounts are constrained by their agreement code and related code.

Clients start exchanges solely through EOAs, which can contain Ether and extra data (payload). On the off chance that the zero-account is the beneficiary of the exchange, a Savvy Agreement is made. The agreement's code is then executed in the neighborhood Ethereum Virtual Machine (EVM) in the event that the beneficiary is an agreement account, with the payload gave as information.

Each activity in Ethereum that includes programmable estimations, for example, making contracts, settling on capability decisions, getting to account capacity, and performing tasks in the EVM, requires an expense known as gas. This charge is paid as remuneration to excavators who contribute their figuring assets, and it is intended to forestall network misuse and keep away from issues that might emerge from Turing fulfillment, which is a property of Ethereum's prearranging language. Gas is the unit used to address the expenses related with these calculations.

#### **3.3.4. Hyperledger Fabric:**

Hyperledger Fabric, a Blockchain framework managed by The Linux Foundation, operates as a permissioned Blockchain. Unlike public Blockchains such as Bitcoin and Ethereum, which are open to anyone, Hyperledger Fabric requires businesses and organizations to join through a membership service provider to participate in the network. The network consists of peers that are owned and contributed by these businesses, creating a closed and controlled network.

In Hyperledger Fabric, peers play the role of hosting ledgers and chaincodes. A ledger is a record of transactions and state changes that are sequential and cannot be altered. Chaincode invocation, representing Smart Contracts in Hyperledger Fabric, triggers state changes through transactions. Each

transaction generates key-value pairs for assets, which are committed as creations, updates, or deletions to the ledger.

The permissioned approach of Hyperledger Fabric provides enhanced control and privacy for participating organizations, making it well-suited for enterprise and business use cases where confidentiality and restricted access are important requirements. It also allows for customization of governance and consensus rules based on the needs of the consortium of organizations involved, providing flexibility and adaptability in the network's operation.

Proposal: Application sends an exchange proposition to the companions supporting different associations (likewise called endorsers who approve exchanges against underwriting strategies and implement the approaches). Conjuring a chaincode capability is the thing the proposition is requesting to peruse or potentially compose information to the record. A reaction esteem, read set, and compose set are undeniably remembered for the exchange results. The arrangement of these qualities is sent back to the application as an exchange proposition reaction along with the endorsers' marks.

Packaging: The programme validates the consistency of the proposal responses and the endorsers' signatures. The application then sends the transaction to the ordering service (orderer) so that the ledger can be updated. The orderer groups batches of the transactions it has received from the network into a block that is prepared for delivery to all of the peers connected to it.

Validation: Every transaction in the block is verified by peers connected to the orderer to make sure it has been consistently approved by the relevant organisations in accordance with the endorsement policy. It is important to note that only during the proposal phase does the chaincode need to be performed. Following validation, each peer adds the block to the chain and updates the ledger.

Coming up next are a few different ways that Ethereum and Hyperledger Texture are unique. Most importantly, just an assigned local area of members has approval to join the organization, though Hyperledger Texture is a consortium Blockchain foundation. This recognizes Ethereum from Hyperledger Texture as a public Blockchain stage. Conversely, Hyperledger Texture offers a secluded plan with a qualification between hub responsibilities (regarding model, endorsers and orderers) and configurable agreement and enrollment administrations, which brings about elevated degrees of versatility, power, and classification. Second, neither fuel nor digital currencies are coordinated into Hyperledger Texture, (for example, Ether and gas in Ethereum). Third, the Hyperledger chaincode simply depicts an assortment of resources that are displayed as key-esteem coordinates and offers the capabilities for controlling the resources and changing their states. To wrap things up, Ethereum's contract code can be executed on neighborhood virtual machines by any digger who gets the exchange that contains it and engenders it around the P2P organization. Yet, with Hyperledger Texture, peer hubs really have the chaincode (peers). At the point when an exchange is made by the application, just certain companions are permitted to execute and sign the exchange (embracing peers). Every one of these supporting companions autonomously executes the exchange subsequent to getting it from the application by calling the chaincode that the exchange alludes to. Chaincode works in a holder climate (like Docker) for security and detachment.

It means quite a bit to take note of that Ethereum and Hyperledger are progressively meeting. For example, the Hyperledger Tunnel project, which utilizes the Tendermint agreement motor, presently upholds utilizing the Hyperledger Texture EVM chaincode module to run Ethereum Smart Contracts on Fabric.

### **3.3.5 Platforms for Smart Contracts**

Different Blockchain systems offer diverse capabilities for developing and deploying Smart Contracts. Examples of such systems include NXT,



Ethereum, and Hyperledger Fabric. These platforms provide unique features, such as contract programming languages, contract code execution mechanisms, and security levels, for creating Smart Contracts. Some platforms even allow the use of high-level programming languages for creating Smart Contracts.

Bitcoin, although having limited computational power, can be used for conducting cryptocurrency transactions on its public Blockchain network. Bitcoin uses a stack-based bytecode scripting language, which has limitations in building Smart Contracts with complex logic. Implementing Smart Contracts on the Bitcoin Blockchain would require significant adjustments to the mining processes and incentive programs.

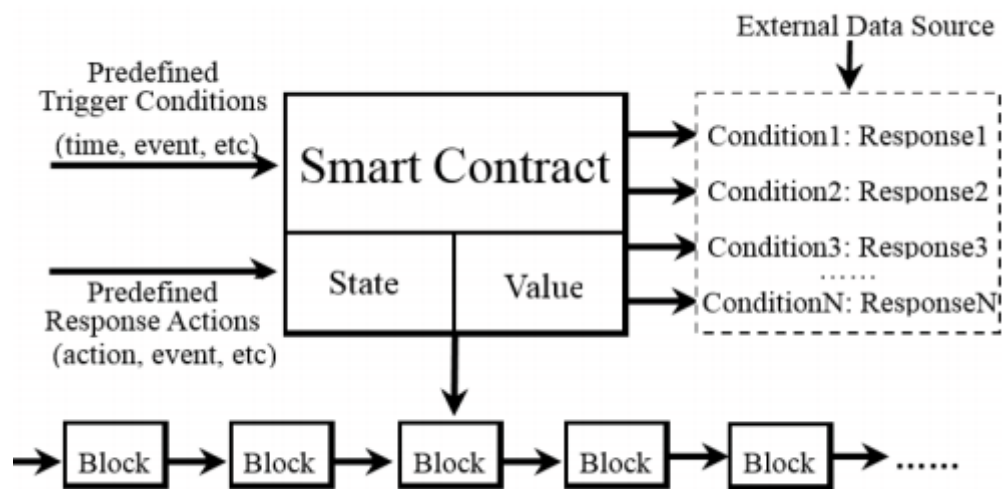
NXT, an open-source Blockchain platform, uses a proof-of-stake consensus protocol and includes several active Smart Contracts. However, it is not Turing-complete, which means only pre-existing templates can be used, and custom Smart Contracts are not supported.

Ethereum was the first Blockchain platform to introduce Smart Contracts. It supports sophisticated and customized Smart Contracts using a Turing-complete virtual machine called the Ethereum Virtual Machine (EVM). Smart Contracts on Ethereum are created using the high-level programming language Solidity, which is then translated into EVM bytecode and executed on the Blockchain. Ethereum is currently the most widely used Smart Contract development platform, allowing the creation of various types of decentralized apps (DApps) across different industries.

Hyperledger Fabric, on the other hand, is a permissioned Blockchain that requires organizations to join through a membership service provider. The network is owned and contributed to by these organizations, in contrast to public Blockchains like Bitcoin and Ethereum that allow anyone to participate. IBM's Hyperledger Fabric supports Smart Contracts and is an open-source

enterprise-grade distributed ledger technology platform that provides modularity, reliability, and scalability for different business use cases.

There are notable differences between Smart Contracts on Ethereum and Hyperledger Fabric. Hyperledger Fabric supports Smart Contracts written in Go, Java, and JavaScript, while Ethereum Smart Contracts are commonly written in Solidity. In Ethereum, any miner receiving a transaction containing contract code can execute it locally in their virtual machine.



**Figure 7. Taxonomy of Blockchain-enabled Smart Contract** <sup>[10]</sup>

### 3.3.6 Optimization Driven Smart Contract Improvement

Smart Contracts have emerged as a fresh and promising technique to develop fully decentralised apps without engaging a trustworthy third party. Smart Contracts have many advantages, but adoption is still hampered by a variety of issues, such as performance issues, security risks, and privacy concerns. In reality, new Smart Contract applications are more demanding in terms of contract execution costs, privacy, and security. We talk about optimization-driven Smart Contract improvement strategies in this area, which may be divided into security and performance optimization strategies.

### 3.3.7. Research framework of Smart Contract

We partition the life-cycle of a Smart Contract into five phases in light of the functional system of Smart Contract: 1) discussion; 2) improvement; 3) organization; 4) support; and 5) learning and implosion. We give a key exploration worldview to Smart Contract in light of this life-cycle. The structure likewise refers to various prior works of writing. To structure the finishes of the flow corpus of exploration on Blockchain innovation, Risius and Spohrer, for example, gave an examination system. To arrange and differentiate Blockchains and Blockchain-based frameworks, Xu et al. recommended a scientific categorization. The scientific categorization includes important Blockchain building blocks and the effects of various design choices, which aids in crucial design considerations about the value and calibre of Blockchain-based frameworks. Glaser made an exhaustive calculated structure for Blockchain frameworks and further isolated them into the texture layer and application layer, two particular coding levels.

A six-layer structure is used in the proposed research framework, with the infrastructures layer at the bottom and the contracts layer, operations layer, intelligence layer, manifestations layer, and applications layer at the top. The specifics are listed below.

- **Infrastructures Layer:** The infrastructures that support Smart Contracts and associated applications include, but are not limited to, trusted development environments, trusted execution environments, and trustworthy data feeds (Oracles). The choice of these infrastructures will somewhat affect the design patterns and contract characteristics of Smart Contracts.

**Trusted Development Environments:** An extensive variety of improvement instruments, as programming dialects, incorporated improvement conditions (IDEs), improvement systems, clients, wallets, and so forth, are utilized during the turn of events, sending, and setting off of shrewd agreements. Involving the wallet for instance, it commonly performs undertakings such being a boot hub, introducing an agreement, and executing an agreement as well as filling in as a device for overseeing computerized resources.

Trusted Execution Environments: Smart Contracts may be executed in a trusted environment thanks to Blockchain technology. The consensus process, incentive system, and peer-to-peer network of the Blockchain are all used in the execution of Smart Contracts, and the results of that execution will be recorded in the distributed ledger that is maintained by all nodes. The design pattern, execution effectiveness, and security of Smart Contracts will all be impacted by various consensus algorithms and incentive structures. For instance, when developing and deploying Smart Contracts in Ethereum, it is important to take fuel consumption into account in order to prevent denial-of-service attacks, excessively high costs brought on by the frequent invocation of dead code, opaque predicates, expensive operations in loops, and other gas-intensive operations, as well as the out-of-gas exception brought on by a gas shortage.

Trusted Data Feeds (Oracles): In order to maintain the security of the Blockchain network, Smart Contracts are often executed in SEE (for example, EVM in Ethereum and Docker container in Hyperledger Fabric). Smart Contracts require trusted data feeds (referred to as "Oracles") to provide external states about the real world in the form of a transaction in a secure and dependable manner, ensuring the deterministic nature of contract execution outcomes. This is necessary because information that is not generated by a transaction must be added as data to a transaction.

- **Contracts Layer:** Contract terms, situation reaction rules, and association necessities are totally contained in the agreements layer alongside other static agreement information. Subsequently, this layer can be considered a static data set of Smart Contract that contains every one of the rules for contract conjuring, execution, and correspondence. While making a Smart Contract, all gatherings (project workers) should initially examine and settle on the boundaries of the understanding, which might incorporate lawful provisions,

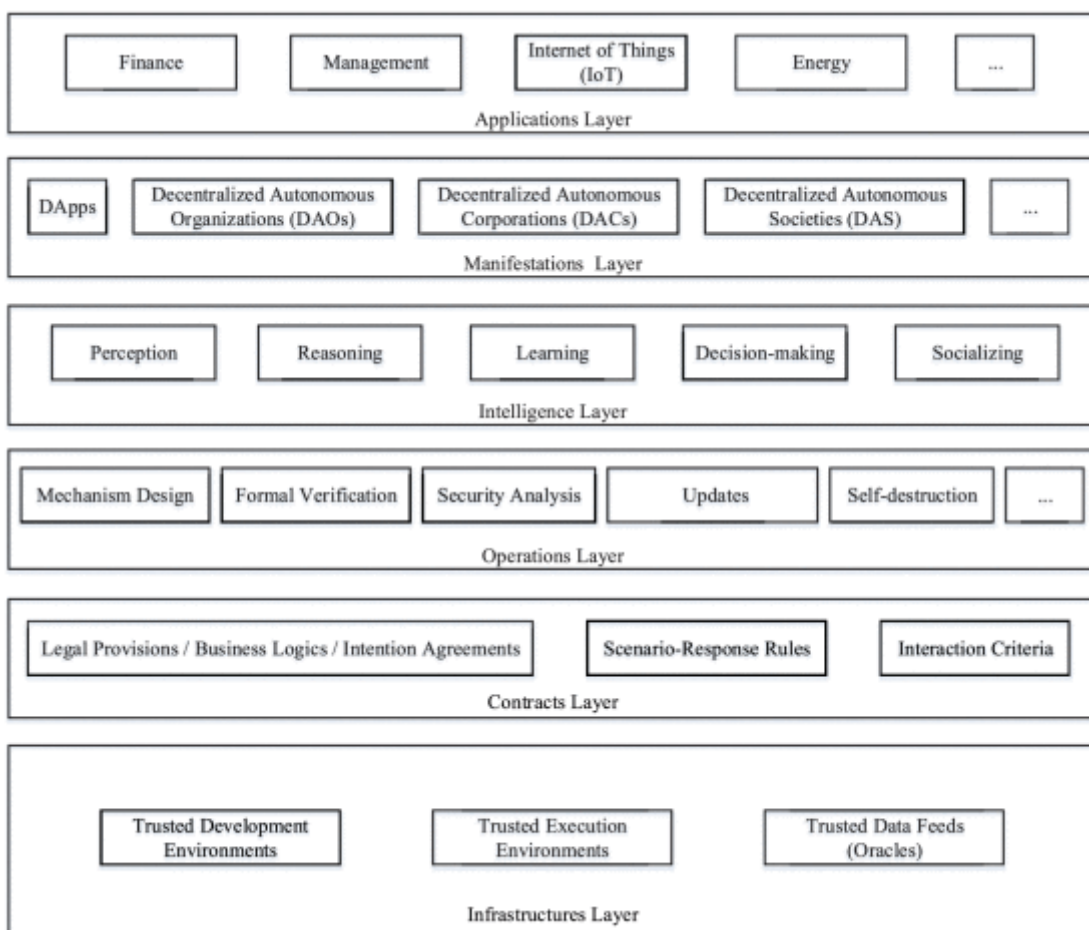
business rationales, and aim arrangements. Furthermore, as per the attributes of the improvement stages and the expectations of the workers for hire, collaboration standards (like access authority, correspondence mode, and so on) ought to likewise be executed in this layer for connections among agreements and clients (or endlessly contracts).

- **Operations Layer:** All dynamic operations on static agreements, for example, component plan, formal confirmation, security examination, updates, and implosion, are contained in the tasks layer. Since pernicious or weak savvy contracts can possibly incur critical monetary misfortunes to clients, the maintenance layer is vital for the right, safe, and powerful activity of brilliant agreements. Before the brilliant agreements are put onto the Blockchain, instrument plan strategies use data and impetus hypothesis to successfully empower contracts to complete their motivation. This is seen according to the viewpoint of the life-pattern of shrewd agreements, which ranges from discussion through implosion. The legitimacy and security of agreement codes are affirmed utilizing formal confirmation and security examination processes, which likewise ensure that the codes will be executed as per the software engineers' planned semantics. At the point when a Smart Contract's usefulness is challenging to match client demands or the agreement has patchable weaknesses after it has been distributed onto the Blockchain, updates can be hypothetically made, but all past updates are recorded on the Blockchain and can't be modified. Implosion is completed to guarantee network security towards the finish of the Smart Contract life cycle or at whatever point a high-risk weakness shows up.
- **Intelligence Layer:** The insight layer contains different sharp calculations that add knowledge to the Smart Contract made utilizing the first three levels. These calculations incorporate those for insight,

thinking, learning, direction, and mingling. It should be noticed that savvy contracts as they exist today need huge knowledge. Notwithstanding, we believe that future Smart Contract ought to highlight "Imagine a scenario where"- type derivation, calculation, and smart dynamic in unexpected situations as well as being self-implementing as per the preset On the off chance that assertions. As was at that point noted, Smart Contract that work on the Blockchain organization can be considered programming specialists that address their users. Agents will have some knowledge because of mental figuring, support learning, and different progressions in man-made brainpower (man-made intelligence) innovation, like discernment, thinking, and learning. Subsequently, those specialists are not just independent since they are equipped for choosing errands, focusing on undertakings, and participating in objective coordinated ways of behaving (otherwise called conviction want goal, or BDIA), yet they are additionally friendly since they participate in correspondence, collaboration, and discussion with each other. To improve contract plan and activity and at last make the truly "brilliant" contract, the learning and participation results will likewise be sent back to the agreements layer and the tasks layer.

- **Manifestations Layer:** For imminent applications, like DApps, decentralized independent associations (DAOs), decentralized independent organizations (DACs), and decentralized independent social orders, the indications layer embodies various sign kinds of Smart Contract (DASs). Blockchain's application interfaces, which permit it to implant different application situations, are undifferentiated from Smart Contracts, which catch the unpredictable ways of behaving of organization hubs. An assortment of DApps can be made, for example, by integrating legitimate statements, business rationales, and expectation arrangements into Smart Contracts.

- Applications Layer:** All application domains that were developed upon the manifestation layer are included in the applications layer. For instance, based on DAO, the Ethereum platform developed the Plantoid application (also known as the distributed autonomous art), which realised a truly aesthetic economy that connects creators, designers, works of art, and audiences into a symbiotic relationship, freeing art from monopolised and hierarchically organised capitalist markets. Smart Contracts can theoretically be applied in every sector, including finance, IoT, healthcare, supply chain, etc.



**Figure 8. Basic research framework of Smart Contracts** <sup>[4]</sup>

It is important to note that the suggested framework, particularly for the intelligence layer, is simply an ideal one. The fact that any activity in the

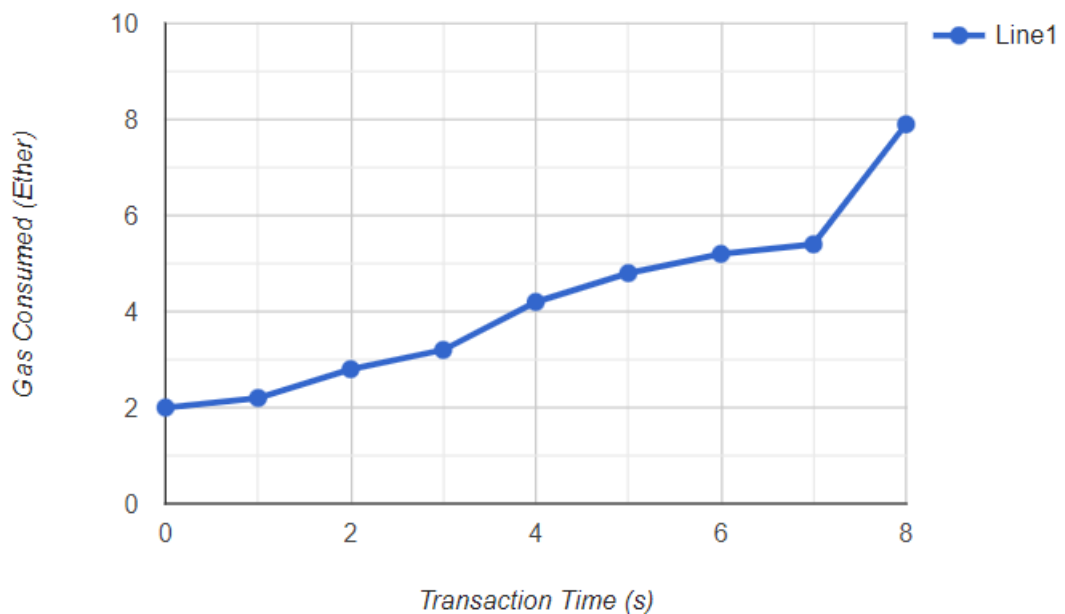
Blockchain must be initiated by a node controlled from outside the network, as Glaser also noted, is a functional limitation. As a result, Smart Contracts should implement autonomous mechanisms or intricate microservice interactions that, when combined, realise more complex service logic, such as an autonomous portfolio management service. Future Smart Contracts ought to be autonomous and intelligent in some way.

For scholars and practitioners, the proposed paradigm has some theoretical and practical usefulness. On the one hand, the framework includes all of the essential components for a Smart Contract over its entire life. On the other side, the framework identifies the direction of the research and potential growth patterns.



## CHAPTER 04: EXPERIMENTS AND RESULT ANALYSIS

Metrics were collected on the Goerli Test Network. The Goerli Test Network provides ether to test applications, as well as in-depth statistics that can be used to track the amount of time a transaction takes to execute, as well as the amount of gas consumed by the transaction.



**Graph.1. Gas Consumption in comparison to Transaction Time**

The gas consumed, shown above, as expected, is proportional to the transaction time, and is also computed by the Ethereum network to be proportional to the computational complexity of a transaction. This is due to the propensity of nodes to prefer processing less computationally-heavy operations, and in turn, slow down transactions that require a large amount of gas to execute.

### **Susceptibility Comparison**

Three of the most prominent enablers for Blockchain are “Proof of Work”, “Proof of Stake” and “Proof of Authority”. Proof of work includes the members solving the complex problem without having a particular need for the solution (except as evidence, of course), which absorbs a large number of resources in turn. In proof-of-authority, machines earn the right to generate new blocks by passing a strict vetting process, which is discussed in detail in the next section. As a result, trustworthy validation machines protect PoABlockchains. These system moderators are preapproved participants who check blocks and transactions. In a proof of stake system, staking serves a similar function to proof of work’s mining, in that it’s the process by which a network participant gets selected to add the latest batch of transactions to the Blockchain and earn some crypto in exchange.

**Table2. Vulnerability Analysis of Consensus Mechanism against Common Blockchain Attacks**

<b>Attack</b>	<b>Is Proof of Work Vulnerable?</b>	<b>Is Proof of Stake Vulnerable?</b>	<b>Is Proof of Authority Vulnerable?</b>
DoS	Yes	Yes	Yes
Selfish Mining	Yes	Yes	Yes
Short-Range Attack	No	Yes	Yes
Long-Range Attack	No	Yes	No
Coin-Age accumulation	No	Yes	No
Pre-Computation Attack	No	Yes	No
Sybil Attack	Yes	Yes	Yes


Thus, it is seen that Proof of Stake is vulnerable to most of the attack while Proof of Work and Proof of Authority are vulnerable to some of the attack. Thus, our objective of building a secure decentralised application is achieved here.

### Transaction speed comparison

Performance metric, Transactions per second is considered here, to measure the number of records or transaction records sent and saved per second. We did two transactions, one using GoerliTestnet Faucet and the other using SepoliaTestnet Faucet.

#### Send ✕

**Confirmed** [Copy transaction ID](#)

**From** 0x003...CF69  0x07f...8F1F **To**

**Transaction**

Nonce	2
Amount	<b>-0 GoerliETH</b>
Gas Limit (Units)	21000
Gas Used (Units)	21000
Base fee (GWEI)	103.91662148
Priority fee (GWEI)	1.5
Total gas fee	0.002214 GoerliETH
Max fee per gas	0.000000155 GoerliETH
Total	<b>0.00221375 GoerliETH</b>




**+ Activity log**

- Transaction created with a value of 0 GoerliETH at 15:34 on 11/15/2022.
- Transaction submitted with estimated gas fee of 0.002 ETH at 15:34 on 11/15/2022.
- Transaction confirmed at 15:34 on 11/15/2022.

**Figure 9. Transaction done using GoerliTestnet Faucet**

**Send** ✕

**Confirmed** [Copy transaction ID](#)

**From**  0x003...CF69   **To** 0x07f...8F1F

**Transaction**

Nonce	2
Amount	<b>-0 SepoliaETH</b>
Gas Limit (Units)	21000
Gas Used (Units)	21000
Base fee (GWEI)	103.91662148
Priority fee (GWEI)	1.5
Total gas fee	0.002214 SepoliaETH
Max fee per gas	0.000000155 SepoliaETH
Total	<b>0.00221375 SepoliaETH</b>

**+ Activity log**

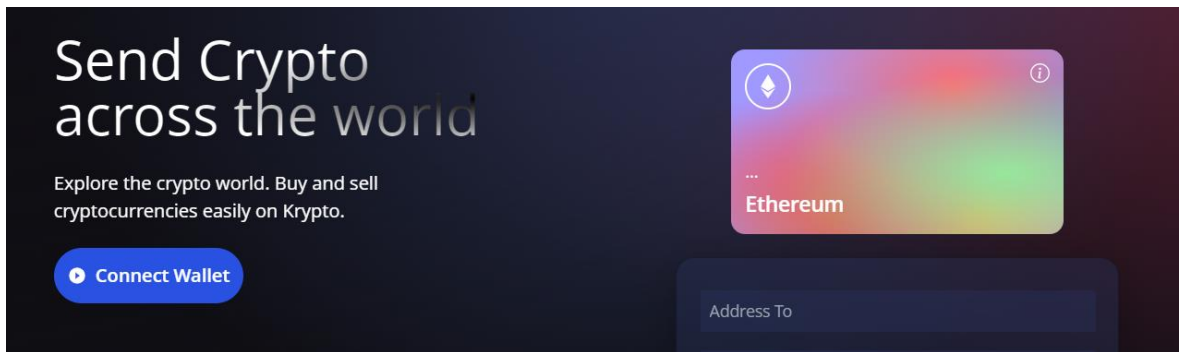
- Transaction created with a value of 0 SepoliaETH at 16:34 on 11/15/2022.
- Transaction submitted with estimated gas fee of 0.002 ETH at 16:34 on 11/15/2022.
- Transaction confirmed at 16:35 on 11/15/2022.

**Figure 10. Transaction done using SepoliaTestnet Faucet**

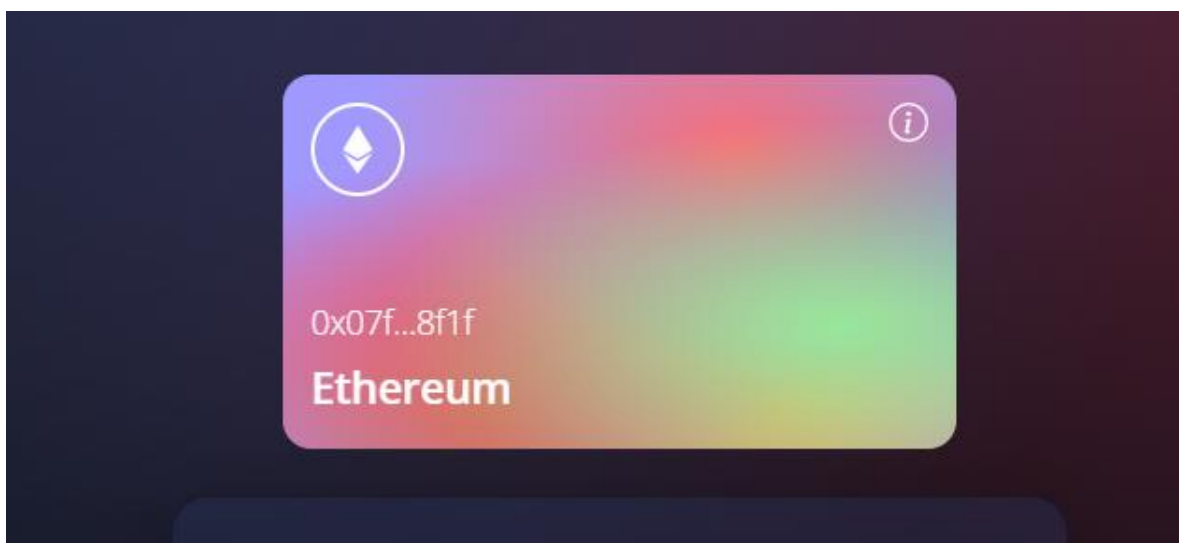
It is seen that the time per transaction is more in SepoliaTestnet Faucet as compared to GoerliTestnet Faucet (millions of transaction takes place in a second).

Hence, we've used GoerliTestnet Faucet that is a proof of stake Blockchain consensus to ensure faster transactions.

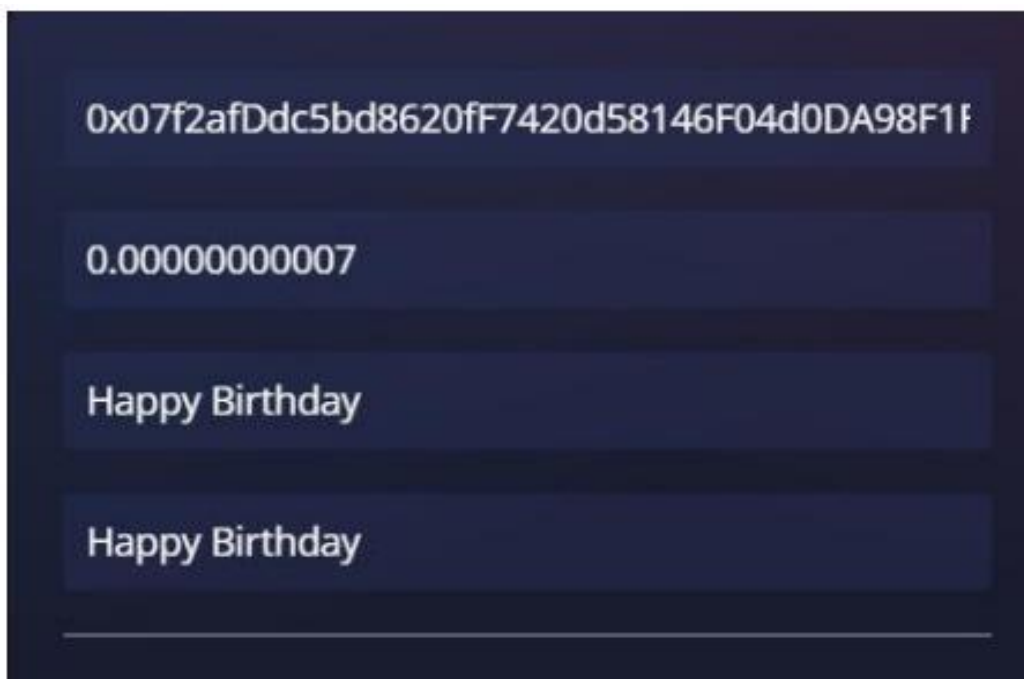
**Results:-**



**Figure 11. On clicking “Connect Wallet”, our Metamask wallet will be connected.**



**Figure 12. Our Metamask Wallet is now connected**



**Figure 13. Making a transaction with sender's address, ethers, gif and message**

From: 0x003...CF69

To: 0x07f...8F1F

Amount: 7e-11 ETH

Message: Happy Birthday

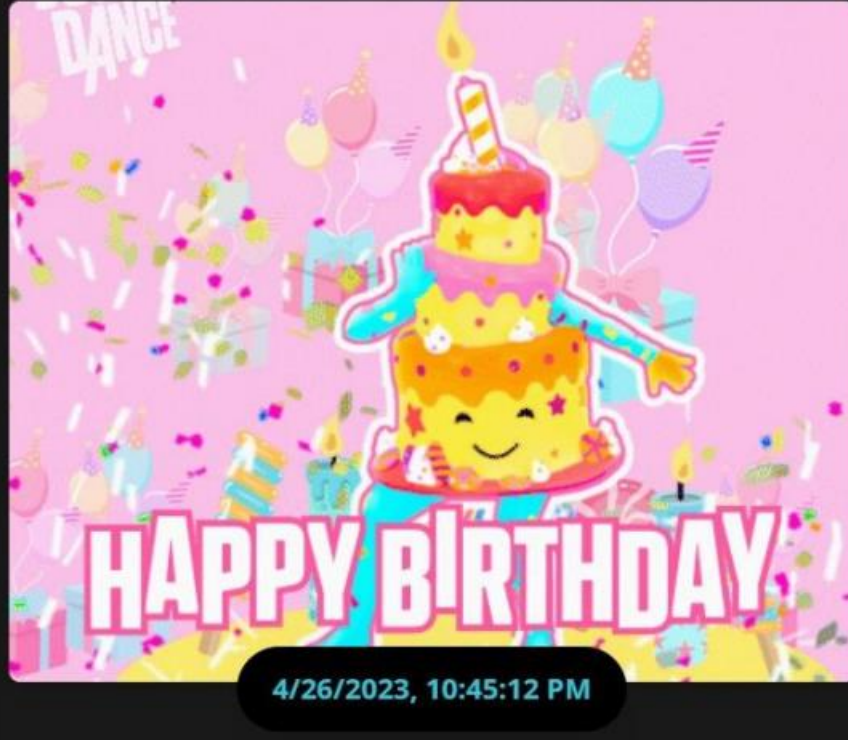


Figure 14. Recent Transaction

---

Transaction Hash:	0x38b0f79e770c4a0bc516706be7ea6d3bef29f93987a4052945e4dfa2d217d0ac <a href="#">🔗</a>
Status:	<span>Success</span>
Block:	3366283 <span>28 Block Confirmations</span>
Timestamp:	6 mins ago (Apr-26-2023 05:15:12 PM +UTC)

---

From:	0x003861Ea22907d9F6fd5e5CeaF2d1a5ADD81CF69 <a href="#">🔗</a>
To:	0xF893f066bC3897Ea98437887C59dEE894BCd619e <a href="#">🔗</a> <span>✓</span>

---

Value:	0 ETH (\$0.00)
Transaction Fee:	0.000387319449388864 ETH (\$0.00)
Gas Price:	2.231000008 Gwei (0.000000002231000008 ETH)

---

**Figure 15. Details of transaction on Etherscan**



## **CHAPTER 05: CONCLUSIONS**

### **5.1 Conclusion**

In conclusion, the completion of the "building and deploying a modern web 3.0 Blockchain app" project marks a significant achievement in showcasing our ability to develop a functional Blockchain application. Through the use of web3.js and Metamask, we have successfully created a web application that allows users to transfer Ether and view their latest transactions in a secure and user-friendly manner.

With the use of proof of work consensus algorithm, the project is vulnerable to most of the attack making it more secure. This project has not only provided valuable hands-on experience with cutting-edge technologies in the field of Blockchain and decentralized applications, but it also highlights the potential of web 3.0 to revolutionize traditional online interactions with enhanced security, transparency, and decentralization.

### **5.2 Future Scope**

The future scope of your "building and deploying a modern web 3.0 Blockchain app" project is promising, with potential opportunities for further development and expansion. One potential area for future scope is adding additional functionalities, such as support for other cryptocurrencies or implementing additional transaction types. Enhancing the user experience through improvements to the user interface (UI) and user experience (UX), strengthening security measures, optimizing performance and scalability, and exploring new use cases in industries beyond cryptocurrency transactions are also potential future scope opportunities. Engaging with the Blockchain community, seeking feedback from users, and exploring collaborations and partnerships with other projects or organizations can provide valuable insights and opportunities for growth. Carefully assessing the feasibility, relevance, and potential impact of future scope opportunities, and regularly updating the

project roadmap, can help ensure the continued success and competitiveness of your Blockchain application in the rapidly evolving Blockchain landscape. We could also consider adding additional features such as support for other cryptocurrencies, for example, Polkadot.

### **5.3 Applications**

1. Financial Services: This Blockchain app could be used for transferring digital assets, including cryptocurrencies, in a secure and transparent manner. It could facilitate peer-to-peer (P2P) transactions, remittances, and cross-border payments without the need for intermediaries, providing faster and more cost-effective financial services.

2. Supply Chain Management: Blockchain technology can enable transparent and traceable supply chain management. This app could be used to track and verify the origin, movement, and authenticity of goods, helping to prevent counterfeiting, ensure product quality, and improve supply chain transparency and efficiency.

3. Decentralized Finance (DeFi): This app could be used for developing and deploying decentralized finance (DeFi) applications, such as decentralized lending and borrowing, decentralized exchanges, and automated market makers, providing decentralized and permissionless financial services.

## REFERENCES

1. Freedman, Demon Nutting,(2015). The Foundations of Online Crowdfunding. In Equity Crowdfunding for Investors.
2. Dannberg, 'Advantages and disadvantages with crowdfunding : - and who are theusers?', Dissertation, 2017.
3. Schwienbacher, Armin and Larralde, Benjamin,Crowdfunding of Small Entrepreneurial Ventures (September 28, 2010). HANDBOOK OF ENTREPRENEURIAL FINANCE, Oxford University Press, Forthcoming.
4. Macht, Stephanie and Weatherston, Jamie (2014) The Benefits of Online Crowdfunding for Fund-Seeking Business Ventures. Strategic Change.
5. Schlueter (2015). Underlying Benefits and Drawbacks of Crowdfunding from the Perspective of Enterepreneurs in Germany. In: 5th IBA Bachelor Thesis Conference. [online] Enschede, University of Twente.
6. Gabison(2015), Understanding Crowdfunding and its Regulation.
7. Ramos, & James. (2014). Crowdfunding and the Role of Managers in Ensuring the Sustainability of Crowdfunding 60Platforms (Rep.). Publications Office of the European Union. doi:10.2791/76003Thakurdesai PA, Kole PL & Pareek RP (2004), Evaluation of the quality and contents of diabetes mellitus patient education on Internet. Patient Education and Counseling 53, 309–313.
8. Cumming, Douglas and Hornuf, Lars and Karami, Moein and Schweizer, Denis, Disentangling Crowdfunding from Fraudfunding (August 24, 2016). Max Planck Institute for Innovation & Competition Research Paper No. 16-09.
9. Mollick, Ethan, The Dynamics of Crowdfunding: An Exploratory Study (June 26, 2013). Journal of Business Venturing, Volume 29, Issue 1, January 2014.
10. Zheng, Xie, Dai, Chen andWang, "An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends," 2017 IEEE International Congress on Big Data (BigData Congress), Honolulu, HI, 2017.

11. Miraz, Dr. Mahdi & Ali, Maaruf. (2018). Applications of Blockchain Technology beyond Cryptocurrency. *Annals of Emerging Technologies in Computing (AETiC)*.
12. Chen, Guang, Bing, Manli& Chen, Nian-Shing. (2018). Exploring blockchain technology and its potential applications for education. *Smart Learning Environments*
13. Angrish, Craver, Hasan and Starly, "A Case Study for Blockchain in Manufacturing: “FabRec”: A Prototype for Peer-to-Peer Network of Manufacturing Nodes", *Procedia Manufacturing*.
14. Friedlmaier, Maximilian and Tumasjan, Andranik and Welp, Isabell M., *Disrupting Industries with Blockchain: The Industry, Venture Capital Funding, and Regional Distribution of Blockchain Ventures* (September 22, 2017). *Proceedings of the 51st Annual Hawaii International Conference on System Sciences (HICSS)*, January 2018.
15. Alharby, Maher & van Moorsel, Aad. (2017). *Blockchain Based Smart Contracts: A Systematic Mapping Study*.
16. Delmolino, Kevin & Arnett, Mitchell & Kosba, Ahmed & Miller, Andrew & Shi, Elaine. (2016). *Step by Step Towards Creating a Safe Smart Contract: Lessons and Insights from a Cryptocurrency Lab*.
17. Gebert, Michael. (2017). *APPLICATION OF BLOCKCHAIN TECHNOLOGY IN CROWDFUNDING*.
18. Ming Wang (2017). *CrowdBC: A Blockchain-based Decentralized Framework for Crowdsourcing*. IACR.
19. Dhokley, Er Gupta, Saurabh Pawar, Ganesh Shaikh, Abrar, “Crowdsourcing and Crowdfunding Platform using Blockchain and Collective Intelligence,” *International Journal of Computer Sciences and Engineering*, 2016.
20. Zheng, Xie Dai, Chen and Wang, “An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends,” *IEEE International Congress on Big Data (Bigdata Congress)*, Honolulu, HI, 2017.

## APPENDICES

### FUNCTIONALITY CODE:

```
contract Transactions {
  uint256 transactionCount;

  event Transfer(address from, address receiver, uint amount,
    string message, uint256 timestamp, string keyword);

  struct TransferStruct {
    address sender;
    address receiver;
    uint amount;
    string message;
    uint256 timestamp;
    string keyword;
  }
}
```

Figure 16. Contract Structure

```
function addToBlockchain(address payable receiver, uint amount, string memory message,
string memory keyword) public {
  transactionCount += 1;
  transactions.push(TransferStruct(msg.sender, receiver, amount, message, block.timestamp,
  keyword));

  emit Transfer(msg.sender, receiver, amount, message, block.timestamp, keyword);
}
```

Figure 17. Function to add transaction to blockchain

```
const createEthereumContract = () => {
  const provider = new ethers.providers.Web3Provider(ethereum);
  const signer = provider.getSigner();
  const transactionsContract = new ethers.Contract(contractAddress, contractABI, signer);

  return transactionsContract;
};
```

Figure 18. Function to create Ethereum Contract

```

const getAllTransactions = async () => {
  try {
    if (ethereum) {
      const transactionsContract = createEthereumContract();

      const availableTransactions = await transactionsContract.getAllTransactions();

      const structuredTransactions = availableTransactions.map((transaction) => ({
        addressTo: transaction.receiver,
        addressFrom: transaction.sender,
        timestamp: new Date(transaction.timestamp.toNumber() * 1000).toLocaleString(),
        message: transaction.message,
        keyword: transaction.keyword,
        amount: parseInt(transaction.amount._hex) / (10 ** 18)
      }));
    }
  }
};

```

**Figure 19. Function to get all the transactions**

```

const checkIfWalletIsConnect = async () => {
  try {
    if (!ethereum) return alert("Please install MetaMask.");

    const accounts = await ethereum.request({ method: "eth_accounts" });

    if (accounts.length) {
      setCurrentAccount(accounts[0]);

      getAllTransactions();
    } else {
      console.log("No accounts found");
    }
  } catch (error) {
    console.log(error);
  }
};

```

**Figure 20. Function to check if the wallet is connected**

```

const sendTransaction = async () => {
  try {
    if (ethereum) {
      const { addressTo, amount, keyword, message } = formData;
      const transactionsContract = createEthereumContract();
      const parsedAmount = ethers.utils.parseEther(amount);

      await ethereum.request({
        method: "eth_sendTransaction",
        params: [{
          from: currentAccount,
          to: addressTo,
          gas: "0x5208",
          value: parsedAmount._hex,
        }],
      });
    }
  }
};

```

**Figure 21. Function to send transaction**

## Code:

```

import { ethers } from "ethers";

import { contractABI, contractAddress } from "../utils/constants";

export const TransactionContext = React.createContext();

const { ethereum } = window;

const createEthereumContract = () => {
  const provider = new ethers.providers.Web3Provider(ethereum);
  const signer = provider.getSigner();
  const transactionsContract = new ethers.Contract(contractAddress, contractABI, signer);

  return transactionsContract;
};

export const TransactionsProvider = ({ children }) => {
  const [formData, setformData] = useState({ addressTo: "", amount: "", keyword: "", message: "" });
  const [currentAccount, setCurrentAccount] = useState("");
  const [isLoading, setIsLoading] = useState(false);
  const [transactionCount, setTransactionCount] = useState(localStorage.getItem("transactionCount"));
  const [transactions, setTransactions] = useState([]);

  const handleChange = (e, name) => {
    setformData((prevState) => ({ ...prevState, [name]: e.target.value }));
  };
};

```

```

const getAllTransactions = async () => {
  try {
    if (ethereum) {
      const transactionsContract = createEthereumContract();

      const availableTransactions = await transactionsContract.getAllTransactions();

      const structuredTransactions = availableTransactions.map((transaction) => ({
        addressTo: transaction.receiver,
        addressFrom: transaction.sender,
        timestamp: new Date(transaction.timestamp.toNumber() * 1000).toLocaleString(),
        message: transaction.message,
        keyword: transaction.keyword,
        amount: parseInt(transaction.amount._hex) / (10 ** 18)
      }));

      console.log(structuredTransactions);

      setTransactions(structuredTransactions);
    } else {
      console.log("Ethereum is not present");
    }
  } catch (error) {
    console.log(error);
  }
};

```

```

const checkIfWalletIsConnect = async () => {
  try {
    if (!ethereum) return alert("Please install MetaMask.");

    const accounts = await ethereum.request({ method: "eth_accounts" });

    if (accounts.length) {
      setCurrentAccount(accounts[0]);

      getAllTransactions();
    } else {
      console.log("No accounts found");
    }
  } catch (error) {
    console.log(error);
  }
};

```



```

const checkIfTransactionsExists = async () => {
  try {
    if (ethereum) {
      const transactionsContract = createEthereumContract();
      const currentTransactionCount = await transactionsContract.getTransactionCount();

      window.localStorage.setItem("transactionCount", currentTransactionCount);
    }
  } catch (error) {
    console.log(error);

    throw new Error("No ethereum object");
  }
};

```

```

const connectWallet = async () => {
  try {
    if (!ethereum) return alert("Please install MetaMask.");

    const accounts = await ethereum.request({ method: "eth_requestAccounts", });

    setCurrentAccount(accounts[0]);
    window.location.reload();
  } catch (error) {
    console.log(error);

    throw new Error("No ethereum object");
  }
};

```

```

const sendTransaction = async () => {
  try {
    if (ethereum) {
      const { addressTo, amount, keyword, message } = formData;
      const transactionsContract = createEthereumContract();
      const parsedAmount = ethers.utils.parseEther(amount);

      await ethereum.request({
        method: "eth_sendTransaction",
        params: [{
          from: currentAccount,
          to: addressTo,
          gas: "0x5208",
          value: parsedAmount._hex,
        }],
      });

      const transactionHash = await transactionsContract.addToBlockchain(addressTo, parsedAmount, message, keyword);

      setIsLoading(true);
      console.log(`Loading - ${transactionHash.hash}`);
      await transactionHash.wait();
      console.log(`Success - ${transactionHash.hash}`);
      setIsLoading(false);

      const transactionsCount = await transactionsContract.getTransactionCount();
    }
  }
};

```

```

    setIsLoading(true);
    console.log(`Loading - ${transactionHash.hash}`);
    await transactionHash.wait();
    console.log(`Success - ${transactionHash.hash}`);
    setIsLoading(false);

    const transactionsCount = await transactionsContract.getTransactionCount();

    setTransactionCount(transactionsCount.toNumber());
    window.location.reload();
  } else {
    console.log("No ethereum object");
  }
} catch (error) {
  console.log(error);

  throw new Error("No ethereum object");
}
};

```

```

useEffect(() => {
  checkIfWalletIsConnect();
  checkIfTransactionsExists();
}, [transactionCount]);

return (
  <TransactionContext.Provider
    value={{
      transactionCount,
      connectWallet,
      transactions,
      currentAccount,
      isLoading,
      sendTransaction,
      handleChange,
      formData,
    }}
  >
    {children}
  </TransactionContext.Provider>
);
};

```