BLOCKHAIN BASED PEER-TO-PEER LENDING AND BORROWING SYSTEM

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

in

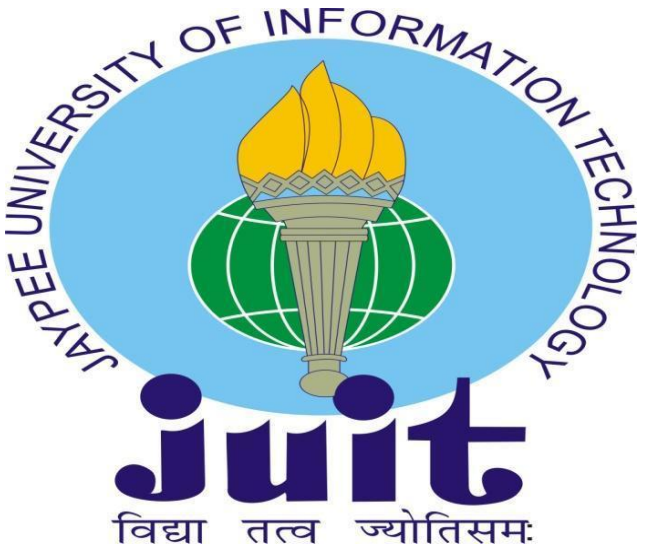
Computer Science and Engineering

By

GOURAV (191383)

Under the supervision of

Dr. MONIKA BHARTI



Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology,

Waknaghat, 173234, Himachal Pradesh

Candidate's Declaration

I hereby declare that the work presented in this report entitled **"Blockchain based Peer-to-Peer Lending and Borrowing System"** 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 February 2023 to May 2023 at NonceBlox Pvt. Ltd. under the supervision of **Mr. Arjun Dev**, Technical Program Manager, NonceBlox and **Dr. Monika Bharti,** Assistant Professor (SG), Department of Computer Science and Engineering, Jaypee University of Information Technology, Waknaghat, Solan.

I also authenticate that I have carried out the above-mentioned project work under the proficiency stream of Cloud Computing.

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

Gourav [191383]

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

Dr. Monika Bharti
Assistant Professor (SG),
Computer Science and Engineering
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:

<table>
<tr><th>Copy Received on</th><th>Excluded</th><th>Similarity Index (%)</th><th colspan="2">Generated Plagiarism Report Details (Title, Abstract & Chapters)</th></tr>
<tr><td rowspan="2"></td><td rowspan="4">• All Preliminary Pages
• Bibliography/Images/Quotes
• 14 Words String</td><td rowspan="2"></td><td>Word Counts</td><td></td></tr>
<tr><td rowspan="2">Character Counts</td><td rowspan="2"></td></tr>
<tr><td rowspan="2">Report Generated on</td><td rowspan="2">Submission ID</td></tr>
<tr><td rowspan="2"></td><td>Total Pages Scanned</td><td></td></tr>
<tr><td></td><td>File Size</td><td></td></tr>
</table>

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

ACKNOWLEDGEMENT

Firstly, I would like to express my heartiest thanks and gratefulness to almighty Gods for their divine blessings that made it possible to complete the project work successfully.

I am really grateful and wish my profound indebtedness to Dr. Vivek Kumar Sehgal, HOD, Department of Computer Science and Engineering and Information Technology, Jaypee University of Information Technology, Waknaghat, Solan. Deep knowledge & keen interest of my project supervisor Dr. Monika Bharti, Assistant Professor (SG), Department of Computer Science and Engineering. Her endless patience, guidance, constant encouragement, and supervision, valuable advice, reading many drafts and correcting them at all stages have made it possible to complete this project.

I would also like to thank Mr. Arjun Dev, Technical Program Manager, NonceBlox for his guidance throughout the project to overcome any technical difficulties as well as providing moral support at various times. I am also grateful for the support by my colleagues and the whole NonceBlox family for their kind support and encouragement to pursue this project.

I would also like to thank the various individuals, both teaching and non-teaching, for their convenient help and support throughout the working phase of the project.

Regards,

Gourav (191383)
Computer Science and Engineering

TABLE OF CONTENT

LIST OF ABBREVIATIONS:

1. **AKA**: also known as
2. **AI:** Artificial Intelligence
3. **ML:** Machine Learning
4. **DApp:** Decentralized Application
5. **DRI:** Decentralized Record Innovation
6. **BAT:** Basic Attention Token
7. **DEX:** Decentralised exchanges
8. **AMM:** Automated Market Making
9. **ETF:** Exchange-Traded Funds
10. **DeFi:** Decentralized Finance
11. **EVM:** Ethereum Virtual Machine
12. **PC:** Personal Computer
13. **TradFi:** Traditional Finance
14. **DAO**: Decentralised Autonomous Organisations
15. **ERC-20:** Ethereum Request for Comments 20
16. **NFT**: Non-fungible token
17. **BNB:** Binance Shrewd Chain
18. **HTML:** HyperText Markup Language
19. **URL:** Uniform Resource Locator
20. **MF-DCCA:** Multifractal detrended cross-correlations analysis
21. **CLI:** Command Line Interface
22. **GUI:** Graphical User Interface
23. **IDE:** Integrated Development Environment
24. **API:** Application Programming Interface

LIST OF FIGURES:

LIST OF TABLES:

ABSTRACT

The project explores the field of blockchain development by developing a peer-to-peer lending and borrowing platform using Solidity, Ganache, Web3.js, and Metamask. The platform leverages the power of blockchain technology to enable direct lending and borrowing transactions between users, eliminating the need for intermediaries and centralized financial institutions.

In the simplest terms, a blockchain is a decentralized and distributed digital ledger that records and verifies transactions across multiple computers or nodes across the globe. Once a block is added to the blockchain, it becomes permanent and cannot be altered retroactively without the consensus (a.k.a Proof of Work) of the network. This feature provides a high level of security, making it difficult for any outsider to manipulate or tamper with the recorded data.

Ethereum, a blockchain platform where smart contracts are hosted and carried out. A specific number of gas units are burned when a smart contract operation is carried out also known as gas usage. The overall amount of gas used depends on how much processing power is required to complete the function.

The power of blockchain technology can be used for revolutionizing lending and borrowing practices, making the way for a more inclusive and transparent financial system for the benefit of the society. The project was developed by the integration of Solidity, Ganache, Web3.js, and Metamask to build a Web application with Smart contracts integration, Ethereum wallet support and a express.js backend server.

CHAPTER-1

INTRODUCTION

1.1 Introduction

A new internet revolution was about to come when the Concept of Blockchain and cryptocurrency was first introduced in the Bitcoin Whitepaper published by Satoshi Nakamoto on October 31, 2008. Till this date, Satoshi Nakamoto remains anonymous to the public.

Blockchain technology can, at least, be anticipated to manage much more significant methods for trade going forward because it is at the core of Bitcoin and other virtual currency forms. However, the applications of blockchain technology go well beyond only virtual currencies.

The blockchain is an open and distributed ledger. It employs an append-only data format, which permits new transactions and data to be added to a blockchain but precludes the erasure of prior data. This results in the creation of a permanent and verifiable record of the data and interactions between two or more parties. Our social and economic systems may be enhanced, leading to more transparency and accountability.

A blockchain is created by running code and connecting multiple hubs. Since there are several blockchains, a blockchain isn't just one global component. Consider a group of linked PCs that are all housed in a very secure workplace and are connected to one another but not to the internet. In this way, a blockchain may have many related hubs while yet being totally distinguishable from other blockchains. For a range of hierarchical objectives, businesses and banks can design internal blockchains with their own peculiarities. 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 by "mining," and the reward system is a custom that awards an excavator with some Bitcoin if they successfully block. Mining is done by powerful PCs testing numerical puzzles.

As soon as a transaction has been validated and deemed true by the whole network, miners start working on the next block. As a result, a blockchain keeps growing (connecting every new block to the one before it).

All nodes instantaneously record, validate, and settle the transaction's details, including the price, asset, and owner, when a transaction is added to the blockchain. A verified change is instantaneously entered on all copies of a ledger that that change has been confirmed on.
Due to the open and permanent recording of every transaction across all ledgers, there is no need for external verification.

The third web age, known as Web3, is currently being developed. Sites and programs will definitely wish to handle data in a shrewd, human-like manner using enhancements like man-made brainpower (computer-based intelligence), AI (ML), huge information, decentralized record innovation (DLT), and others.

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 decentralization, 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.
Web3 was first referred to as the Semantic Web by Tim Berners-Lee, the inventor of the Internet. Its goal was to create a web that was more open,

intelligent, and free. A Web3 application must be able to analyze enormous volumes of data and give users accurate information and useful actions. Despite this, these apps are still in their infancy, which suggests there is much space for development and that they are quite unlike how Web3 applications could previously function.

Decentralized apps, or "dApps," are digital software that operate on a blockchain network of computers rather than depending on a single computer. Because they are decentralized, dApps are not governed by or influenced by a single authority. The benefits of dApps include user privacy protection, the lack of restriction, and development freedom.

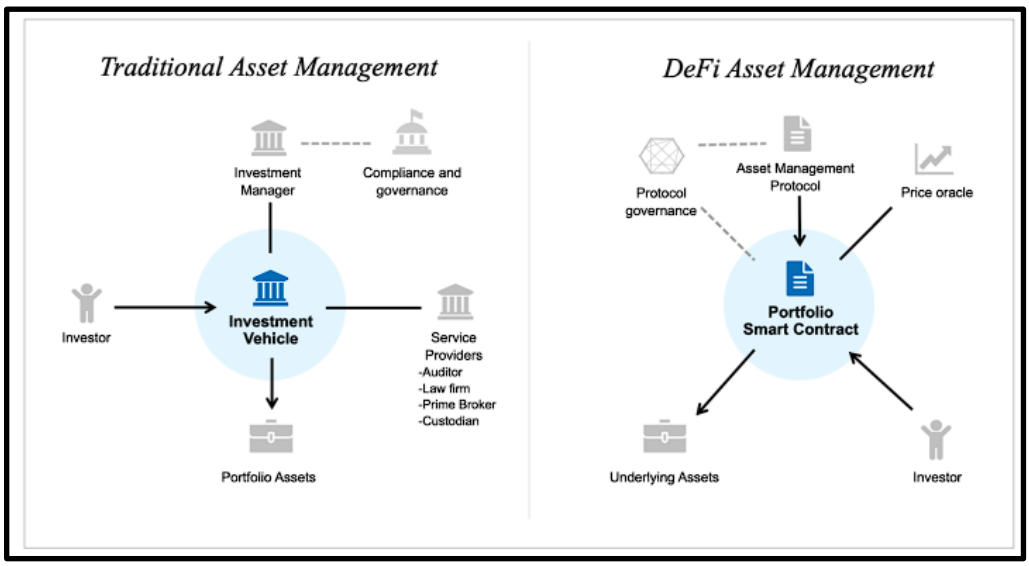


Figure 1.1.1: Decentralized Finance vs traditional Asset Management

A dApp is like any other program. It may be a website or a mobile app. An application known as a "dApps" is one that is created using a decentralized platform, such as Ethereum, as opposed to a conventional application. Writing a portion of a dApps backend code is analogous to creating one's own Ethereum smart contracts. Furthermore, even though dApps user interface resembles that of a standard app, Ethereum is utilized for all or some of the backend.

dApp = Frontend + backend {smart contracts}

Solidity, Serpent, and Vyper are the three programming languages most frequently used to create backend code for Ethereum.

Some of the most promising Ethereum currencies and dApps are laying the foundation for the future of the Internet, including:

- The Basic Attention Token (BAT) is a method for improving privacy and transferring value to users, publishers, and advertising. used in the Brave web browser. The Golem (GNT) programme allows code to be run on one or more distributed computing nodes. A social networking platform called Minds helps content creators and consumers better transfer value.
- Token Sets: utilised in automated tokenized asset management programmes for bitcoin assets.
- Aave: used for depositing bitcoin and borrowing cryptocurrency assets. A decentralised cryptocurrency exchange is called IDEX.

1.2 Problem Statement

Since the development of the Internet, the amount of information and interpersonal contact has increased. Our ability to create and consume knowledge is essentially boundless.

Unfortunately, the ability to control this information has become significantly more centralized over time. This includes information about social life, health, finances, and a variety of other topics. The people in charge of this data are its true owners, and they are allowed to utilize it anyway they see right.

These people are only intermediaries who keep private information on centralized servers to provide services like money management, website hosting, enabling communication with loved ones, etc. Additionally, by merely

clicking a button, they have the power to completely prevent you from accessing this (your) information and any linked services.

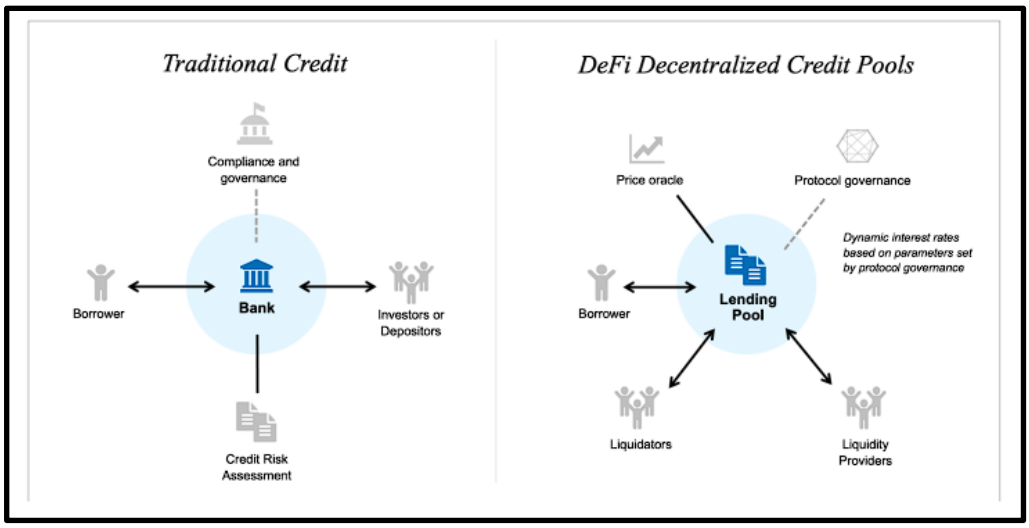


Figure 1.2.1: Traditional vs Decentralized Credit systems.

The services you use, the information you produce, and the information you consume are all subject to this monopoly. Web 3.0 is altering all of that, and Ethereum dApps are an important component of it. To put it simply, the way that international payments are being made through banking channels is a hotch-potch. It involves a multi-step process involving several intermediaries. Additionally, each cycle phase necessitates a significant financial expenditure.

The World Bank estimates that the average exchange cost for payments made overseas is close to 7%. That is a ton of it! Blockchain lightens the burden of unnecessary time delays and streamlines the whole interaction by eliminating all agents and obstinate systems.

When dealing with sensitive data like financial transactions and records, every set of books must be accurate. Additionally, the present procedures for payment settlement may be extremely costly and time-consuming.

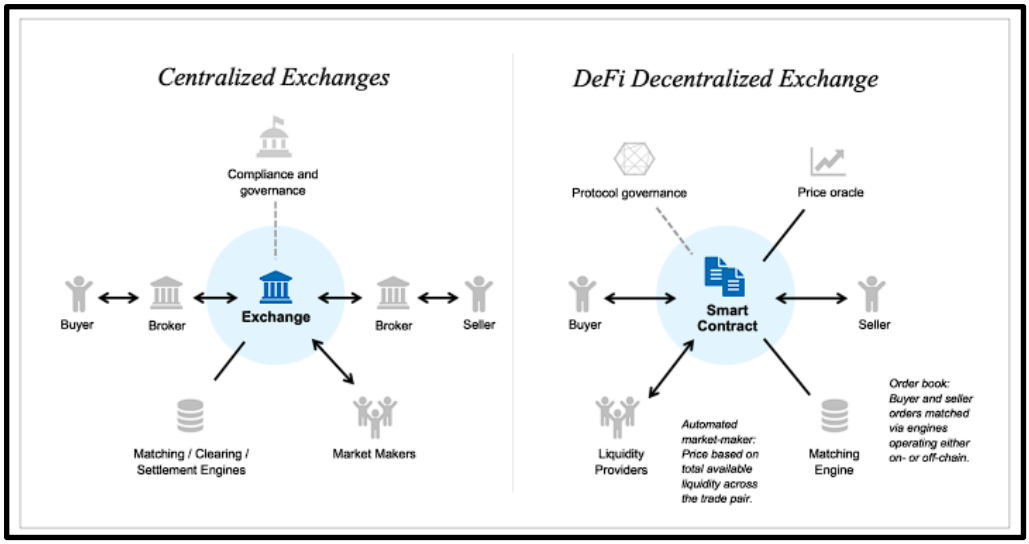


Figure 1.2.2: Centralized vs Decentralized Exchanges.

When an exchange is registered using its cutting-edge and secure distributed record, the money is sent to the recipient nearly immediately. Given that the exchange cannot be altered or swapped, it also provides greater accountability and security than the existing approach. When you consider the advantages of increased security and simplicity, blockchain is unquestionably a better option for international financial trades.

1.3 Objective

To understand the concepts of Blockchain, Blockchain Development, Ethereum Blockchain Network and Decentralized Financing to develop a Decentralized Web3 Application using Ethereum smart contracts that can be used for Peer-to-Peer money lending and borrowing on blockchain.

1.4 Methodology

Methodology includes the approach, techniques and technologies used to achieve the final state of the project.

1.4.1 Tech Stack Used

1. Ethereum Virtual Machine

A runtime environment for Ethereum's smart contracts is called the EVM, or Ethereum Virtual Machine. The provision of security and permitting the execution of unauthorised software over a worldwide network of open nodes are given a lot of priority. EVM, which also confirms that programmes cannot access one another's state and that communication is formed without any potential intervention, especially guards against denial-of-service attacks.

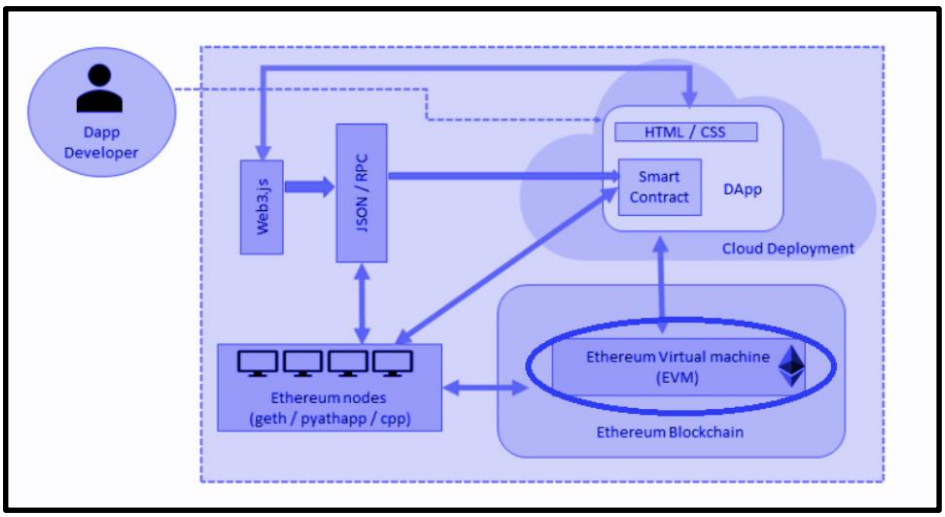


Figure 1.4.1.1: EVM Architecture for DApps.

The electronic voting machine (EVM) is a contemporary voting system built on a microcontroller. It eliminates the chance of illegitimate votes, upholds total voting data confidentiality, and enables quick and accurate vote counting. The voting data kept in EVMs is saved for years and may be erased if required.

The electronic voting machine is a trustworthy means of conducting the election when one candidate must be selected from a vast field of contenders. The one-post, one-vote electronic voting machine is designed for such purpose.

An EVM is the environment in which Ethereum smart contracts are executed. Given that Ethereum has its own Turing-complete programming language called Robustness, running this code is necessary. The Ethereum Virtual Machine (EVM) is a programme that may be used to carry out this task. Each hub decides what code should be executed at a given moment since it is built on top of the Ethereum organisation.

The Ethereum blockchain is commonly implemented using scripts that are performed by a software known as the Ethereum Virtual Machine (EVM). The Ethereum Virtual Machine makes it simple to create new tokens on the Ethereum Blockchain. In this usage, "script" refers to an algorithm or set of instructions that tells the computer what to do to make something work as intended. The EVM requires that one have access to any network node in order to execute the necessary commands and quickly produce new tokens on the blockchain. Two elements make up the Ethereum Virtual Machine (EVM):

- Solidity's EVM (the component that executes source code): The LLVM compiler is used by the EVM, which is built in C++. It is a fully functional virtual machine with every feature you could ask for in a general-purpose smart contract virtual machine, including support for a variety of programming languages, security features, runtime environments, and more. You can even create your own EVM bytecode with this.

- Uncles: These are little bits of data or smart contracts that are kept on the blockchain. This is a helpful feature because it enables you to save program-related metadata. The EVM assembly You can use EVM's bytecode as your programming language; it is presented here.

2. Solidity

All contract code is written in Solidity because it is the language that is used to deploy contracts on the blockchain network. A programming language called robustness is used to create clever decisions that are subsequently carried out on a blockchain network. These magnificent agreements serve as a representation of all interactions that take place across the company. 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. The development of smart contracts using this language is supported on the Ethereum blockchain as well as other blockchains.

JavaScript, a computer language that is widely used, and Solidity are comparable. It is recognised as a dialect of JavaScript. This suggests that if you are comfortable with JavaScript, understanding Solidity should be straightforward. There are significant similarities between Solidity and the C++ and Python programming languages.

Solidity is a high-level language, thus there is no need to type code made up of only ones and zeros. By mixing letters and numbers in a form that makes sense to them, humans can create programmes far more quickly and simply.

In addition to supporting complex user-defined types, libraries, and inheritance, Solidity is statically typed. Since Solidity is statically typed, the user must specify each variable. Data types allow the compiler to check if variables are being utilised appropriately. Usually, information types for robustness are separated into reference types and esteem types.

The key distinction between respect types and reference types (Ethereum Virtual Machine) is how they are delegated to factors and stored. Even if altering the value in one variable of a value type does not

affect the value in another variable, anybody referring to modified values in reference type variables may get updated values.

One of the crucial components that enables the execution of Solidity code is the EVM. On the blockchain, the EVM is referred to as a virtual computer that transforms user ideas into code used by blockchain applications.

Machine-level code is generated by Solidity and performed behind the scenes on the EVM. A compiler converts high-level, readable code into instructions that the processor can understand. On a number of platforms, including the Remix online compiler and a PC-downloaded command-like compiler, free Solidity compilation is accessible.

EVM smart contracts are subject to a number of limitations that need to be overcome. One of the most significant of them is the absence of widely accessible library methods for parsing JSON structures or carrying out floating-point computations.

Smart contracts for fungible and non-fungible tokens are created using Solidity. The Ethereum ecosystem uses a variety of standards to build both fungible and non-fungible tokens.

These allow for the creation of several use cases for blockchain users. Thanks to Solidity, tokens and non-fungible tokens are usable on Ethereum. Ethereum makes it possible to employ tokens for a variety of purposes, such as producing non-fungible tokens and adding them to yield farming pools for increased interest.

Decentralised Autonomous Organisations (DAOs) are another feature of Solidity. A DAO is a brand-new type of online organisational structure that is mostly constructed in Solidity. On an online voting platform,

different people may sign up to become members of DAOs and vote on the most crucial choices for the organisation.

Through Solidity, process automation is feasible within the DAO. Two examples of automated procedures in DAOs include voting on significant decisions and granting reputation to DAO members for their contributions to the organisation.

3. MetaMask

Clients may safely store their Ether and other ERC-20 tokens with the help of the encrypted wallet MetaMask. Additionally, wallet users may communicate with decentralised apps, or dApps.

MetaMask needs to be introduced before it can be used as an Ethereum wallet, much like with other programme modules. Clients can use any Ethereum address after they have stacked it to execute by storing ether and other ERC-20 tokens.

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

The decentralised web, or Web3 (DApps), is built on digital money and decentralised applications. However, you actually need a UI to make use of them. A nice, straightforward, and easy-to-use point of engagement is wonderful.
MetaMask, one of the most well-known digital currency wallets, focuses on programme integration and has a compelling strategy to act as a significant entry point into the Web3, decentralised finance (DeFi), and NFTs domains.

Two of the key benefits of cryptocurrencies are their ability to send and receive tokens safely and anonymously. These are the two areas where MetaMask excels. Clients just need to download the application and set up a new wallet in order to use MetaMask; they are not need to provide any explicitly identifying information, such as their name, address, or Government controlled retirement number. MetaMask is also a non-custodial wallet, which means that data is not maintained in a concentrated data set and that the company does not get customer information through wallet use. After the wallet is created, it can only be restored using the client's chosen secret key or the 12-word seed expression.

The MetaMask wallet is a flexible solution that supports multiple standards on many blockchains. Customers may link non-fungible tokens (NFTs) to business hubs like Open Sea and keep them in their wallets alongside more than 670,000 currencies that adhere to the Ethereum ERC-20 standard.

Only a select number blockchains are supported by MetaMask, including the BNB Chain (formerly known as the Finance Shrewd Chain), Polygon, Torrential slide, and many other test blockchains.

In comparison to more traditional security systems, MetaMask adopts a more user-anonymous approach to security. Since MetaMask is non-custodial software, no user information is saved by it. Rather, wallets are protected via a user-generated password during setup, mobile device biometric data, and a unique 12-word seed phrase for recovering them. Because customers can't obtain their passwords back from technical support, the seed phrase is much more important. If a user loses their seed phrase, they face the risk of never regaining access to their wallet.

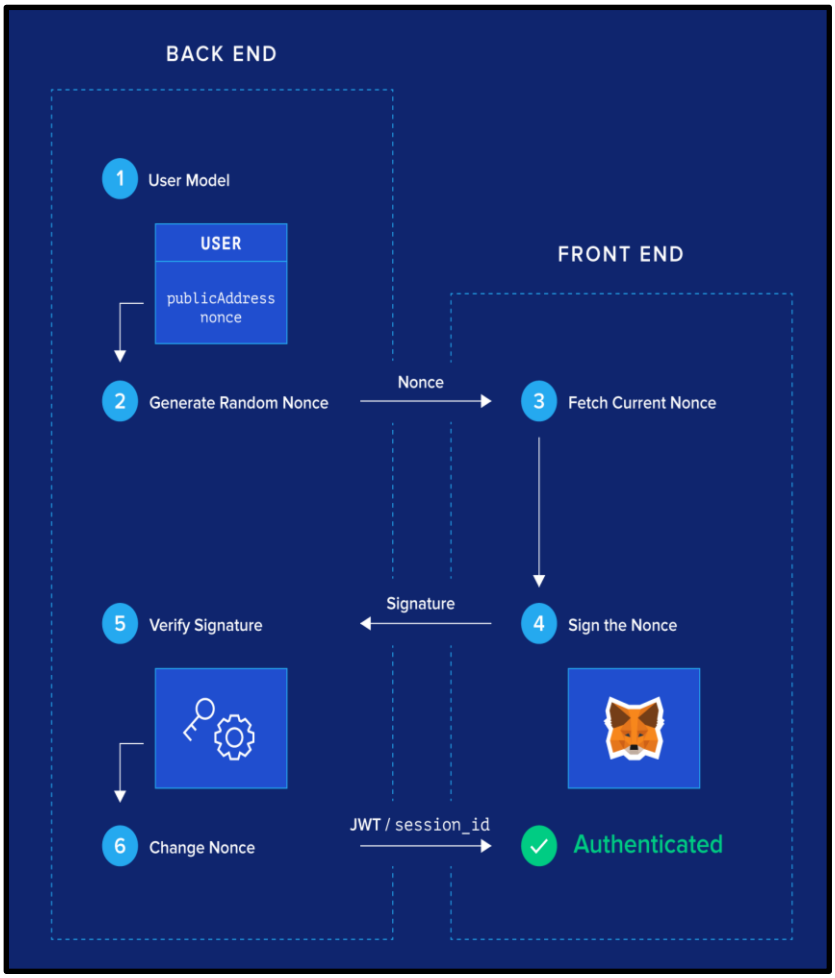


Figure 1.4.1.2: MetaMask wallet security architecture.

Since there is only one way to reinstall a single wallet across many devices, MetaMask is also defenceless against hackers, viruses like broker Trojans, social engineering projects, and other digital currency fakes. The most frequent attacks against MetaMask wallet users include infections that target digital currency wallets, disclosing the 12-word seed expression to an external or on a phishing website, and downloading a phoney MetaMask extension that grants programmers access to your wallet. Never give out your seed expression to someone you don't know and trust, and if you suspect your wallet has been hacked, contact MetaMask support immediately.

Unlike a bitcoin transaction, the wallet does not require users to provide any explicitly identifiable information in order to use it. Users have

complete control over the bitcoin they collect and trade using their wallets, including their secret seed phrase. Technical assistance will never ask for a user's secret phrase, and MetaMask does not save user data.

Customers are permitted to continue working and trading coins or NFTs while using their MetaMask wallet, but the item warns that blockchain activities are not entirely mysterious. Given everything, they label relationships as "pseudonymous." The wallet ID functions as a fictitious name for the client because every trade is publicly visible on the blockchain. If a customer links their wallet to their persona by utilising an NFT as their Twitter profile image or an Ethereum Name Administration space, their true identity may be exposed.

4. Smart Contracts

A smart contract is a self-executing contract in which the states of the agreement between the buyer and seller are directly encoded into lines of code. The plans and secret code are dispersed around the whole decentralised blockchain network. Exchanges are reversible and traceable, and the code controls how they are carried out.

Smart contracts enable trusted transactions and agreements to be established amongst dispersed, unidentified groups without the need for a central authority, a general set of rules, or an external implementation system. In a PC organisation, smart contracts are lines of code that afterwards confirm and complete the states of an agreement between a buyer and a seller.

American PC researcher Scratch Szabo, who created the virtual currency "Touch Gold" in 1998, defined smart contracts as electronic exchange norms that fulfil the requirements of an agreement. Exchanges are

traceable, easy to understand, and irrevocable thanks to blockchains that have amazing agreements incorporated.

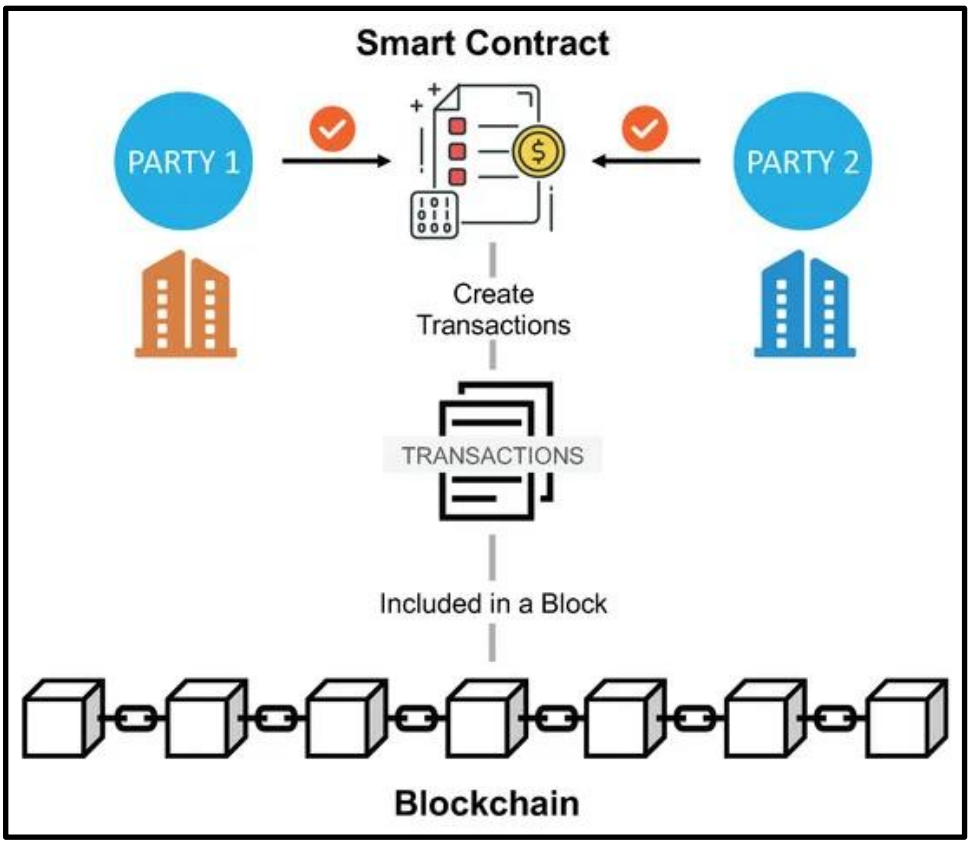


Figure 1.4.1.3: Asset transaction using Smart Contracts.

A smart contract, also known as a crypto contract, is a piece of computer software that, under certain circumstances, directly and directly coordinates the exchange of digital assets between the parties. A smart contract functions with programmed agreement requirements, much like a common agreement. Smart contracts are computer programmes that function exactly as they have been programmed or customised to do. Similar to how a traditional agreement is enforced by law, smart contracts are enforceable by code.

The bitcoin network was fast to use them by leveraging clever agreements to transmit value from one participant to the next. The associated smart contract makes use of crucial safeguards including

verifying that the amount of value to move is indeed existent in the originating account.

The Ethereum platform then appeared, and it was seen to be much more astounding since software developers and designers could reach extraordinary agreements using a Turing-complete language. It should be noted that Turing-fragmented languages were used to write the agreements for the Bitcoin network, which limits the possibility of carrying out brilliant agreements on the Bitcoin network. A few examples of platforms for amazing agreements include Hyperledger Texture, Solana, Polka dot, and Ethereum.

Fundamentals of a Smart Contract:

- Distributed: No party may change the smart contract's conditions, which are guaranteed to be available to everyone on the network. A smart contract is duplicated and distributed to all network nodes.
- Deterministic: Smart contracts can only perform the tasks for which they were designed when the circumstances required for their intended usage are satisfied. The outcome will be the same no matter who carries out the smart contract.
- Immutable: After being released, a smart contract cannot be changed; the only way to remove it is if the functionality has already been implemented.
- Autonomy: There is absolutely no outside intervention. The parties share the agreement that your draught. Bullying is decreased and the dealing partners have complete control because there are no middlemen. Additionally, every node on the network manages and executes the smart contract, removing any influence from a single party.
- Customizable: Before being put into use, smart contracts may be modified or personalised to do anything the user wants.

- Transparent: The code for smart contracts is always preserved on a public distributed ledger known as the blockchain, whether or not a person is a participant in the contract.

Featured Smart Contracts' Capabilities: -

Smart contracts are accurate to the extent that a programmer properly constructs them for execution.

- Automation: Smart contracts may be used to automate procedures and tasks that are now carried out manually.
- Speed: Smart contracts utilise computer code to automate operations, which reduces the time it takes to complete all activities, even those that need human interaction. Since everything is programmed, the time it takes for the smart contract's code to run is the same amount of time it takes to do all of the jobs.
- Reinforcement: This support is the best since every node on the blockchain maintains the shared record.
- Security: Cryptography can ensure the safety of the assets. The hacker will still need to modify each block that comes after the one that was modified, even if the encryption is cracked. Please be advised that owing to the assignment's great difficulty and computational demands, a small or medium-sized organisation would not be able to accomplish it.
- Savings: By eliminating the need for middlemen, smart contracts lower expenses. In addition, the paperwork costs very little or nothing.
- Manages information: Smart contracts keep track of user agreements and application-related data like domain registration and membership records.
- Multi-signature accounts: Once all parties have approved the agreement, smart contracts enable the distribution of funds using multi-signature accounts.

Hence, a smart contract is a computerised contract with blockchain security code. It contains certain instructions and consents that must take place in a particular request for the provisions of the beautiful agreement to be acknowledged. Cut-off times may be specified in the agreement owing to time constraints. Every clever contract has a blockchain address. The agreement may very well be achieved by utilising the organization's location, provided that it has been disclosed via the latter.

5. **ReactJS**

   React is a UI enhancement package for JavaScript. Facebook and an open-source enhancement locality place restriction on it. Despite not being a language, React is a well-known library for web development. Since its debut in May 2013, the library has become one of the most frequently used frontend libraries for web development.

   React has additional features, such as Motion and React Local, to facilitate the development of whole apps beyond simply the user interface.

   React is more well-known than any other front-end development framework in the modern day. The following features of react made it so popular as it is today:

1. React needs less code and offers higher utility, making it easier to create dynamic web-based apps than JavaScript, where pre-arranging sometimes becomes quite complex very quickly.
2. Web application development is sped significantly by React's usage of Virtual DOM. Virtual DOM analyses the previous states of the parts and updates just the elements in the Genuine DOM that were changed, as opposed to refreshing all of the parts again like conventional web applications do.
3. Each React application is built on parts, and even a single application usually uses a variety of parts. Due to the reuse of these components, each

of which has its own logic and controls, the application's development time may be drastically reduced.

4. Unidirectional information stream: React adheres to this. As a result, while creating React apps, designers frequently nest child parts inside parent components. Information only flows in a single direction, making it simpler to identify problems in applications and narrow down their exact locations. Given that it primarily organises fundamental HTML and JavaScript principles for certain important parts, React has an unpretentious expectation to learn and adapt. To fully comprehend React's library, however, demands an investment, just like with other tools and systems.
5. It is frequently used in the creation of mobile and online applications: Despite the fact that we are aware that React is used to create web apps, there are other things for which it may be useful. React Local, which was derived from React itself, is a tremendously well-known invention that can be used to create beautiful portable apps. Thus, React may be used to create both online and mobile applications.
6. Examine React apps with the use of specialised tools: Facebook has provided a Chrome plugin. This improves the framework for debugging React web apps.

6. 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 seeking for a specific resource online, the WWW initialism was one of the first letters put 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 connections.

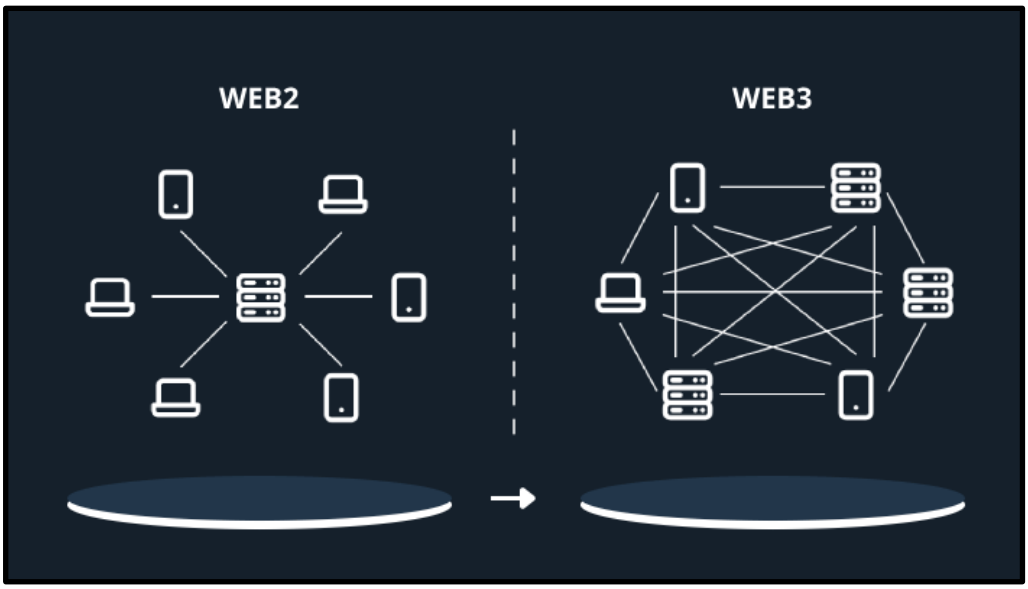


Figure 1.4.1.4: Basic Difference between Web 2 and Web 3.

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. The next phase of the web, which we are now building, is called Web 3.0, and it will be more decentralised, open, and useful than Web 2.0.

Web 2.0 has rapidly expanded thanks to innovations like smartphones, mobile internet access, and social networks. Sectors that have not adapted to the new web-based business model have been disrupted by Web 2.0.

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

The characteristics that best describe Web 3.0 include decentralisation, trust lessness and permissionlessness, artificial intelligence (AI), connectivity, and omnipresence.

Customers will genuinely desire to sell the information given by many and powerful technological assets, like smartphones, PCs, machines, automobiles, and sensors, through decentralised information organisations while still maintaining ownership control.

7. Ethereum

We can execute programmes on this Blockchain network because it is network-based. These programmes are known as smart contracts. Comparable in operation to a digital contract is a smart contract. It has the capacity to carry out reasoning and store value (such as money). The Ethereum network resists nonrepudiation attacks because it keeps track of transaction data.

The blockchain-based decentralised open-source platform Ethereum is used to conduct smart contracts, or projects that proceed exactly as intended without the risk of fraud, outside intrusion, limitation, or margin time. It provides a platform for over 2,60,000 unique digital currencies. Ethereum developers create a virtual currency known as ether, which is used as payment for computations conducted to secure the blockchain.

Since this framework is only a model, we didn't use the main Ethereum network; instead, we used the testnet, which functions very similarly to the main Ethereum organisation. We make use of the Goerli test network, a proof-of-authority blockchain, to facilitate client trades in this framework.

Being on the Goerli test network prevents us from mining Ether; instead, we must request it from the Goerli Test Faucet. The specifics of user transactions, regardless of whether they were successful or not, may be seen using the Etherscan API. Below is a description of the system architecture diagram.

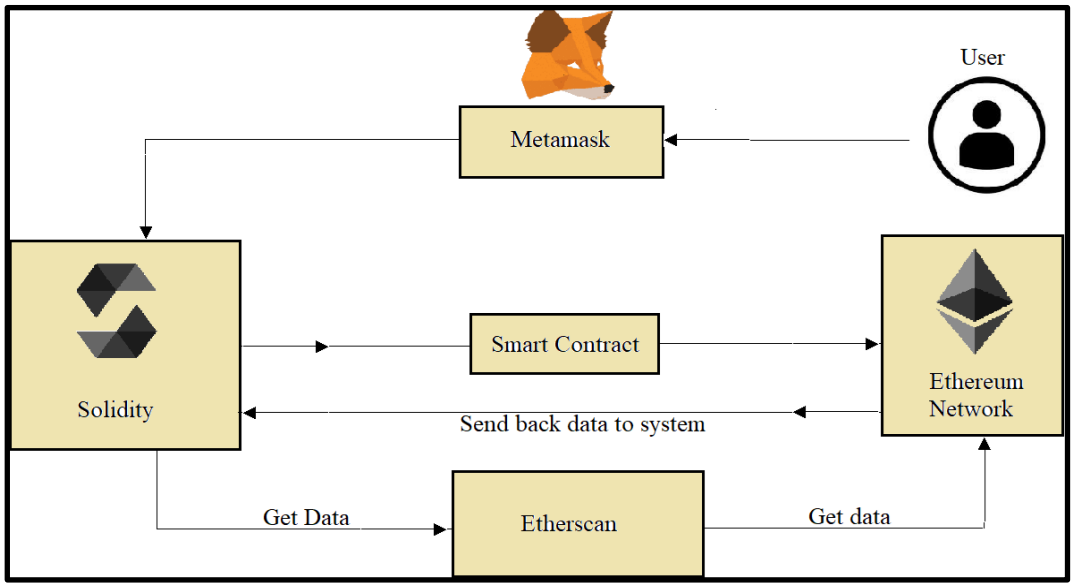


Figure 1.4.1.5: System Architecture for D-Apps.

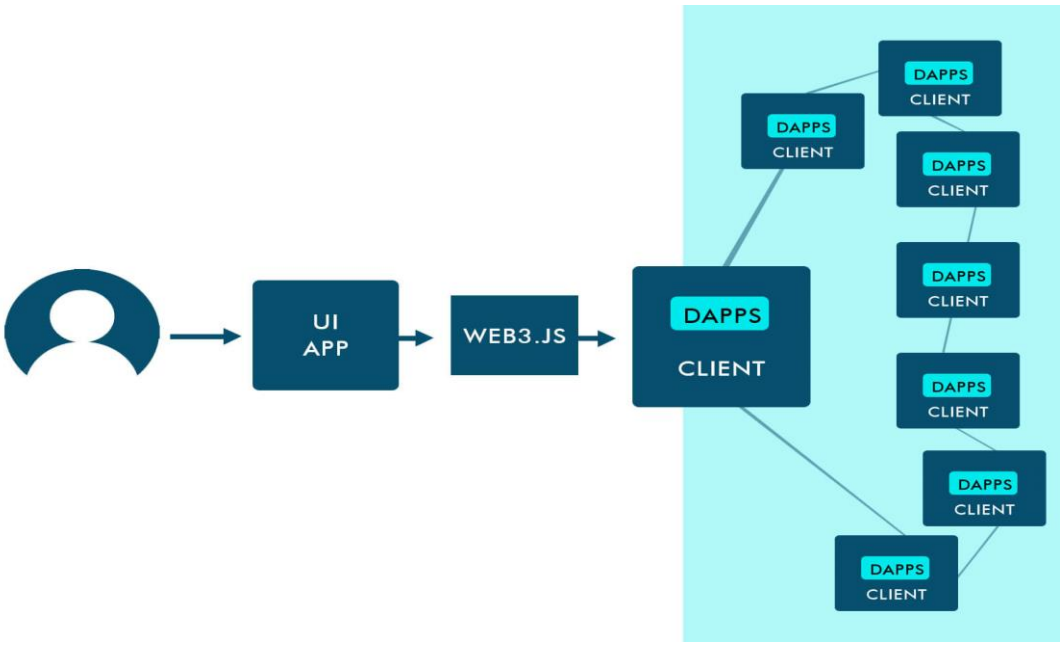


Figure 1.4.1.6: Web3.js as middleware to App UI and Smart Contracts.

CHAPTER-2

LITERATURE SURVEY

Decentralized finance (DeFi) refers to online financial services provided on a public blockchain. In various publications, reviews and detailed understanding of DeFi is provided in the form of research and development from around the world. According to the review of the literature, decentralized finance has a number of advantages over centralized finance, including closer financial inclusion, support for permission-free innovation, a reduction in the need for middlemen, assured transaction immutability, censorship elimination, and cuts costs for international trade. Some of the issues that are associated with them include the risk of smart contract execution, the risk of legal responsibility, the risk of data theft, the risk of interconnectivity, the risk of external data, and a greater propensity for illicit activity while using DeFi applications. An evaluation of the existing literature on the topic shows that there has been little research on DeFi, and many of those that have been are non-empirical. Studies frequently have a favourable opinion of DeFi.

The advantages of DeFi are emphasized in considerable detail, however neither the difficulties nor the results of any critical investigations on DeFi have been thoroughly examined. Decentralized finance is becoming a topic of increasing interest in Europe, the United States, Asia, and Oceania, according to observations on global DeFi trends. Regulating decentralized finance has raised questions about whether it will hinder the development of decentralized finance markets in Asia. Additionally, there are worries that outlawing crypto assets may stifle the development of decentralized finance in African nations where regulatory frameworks do not fully support blockchain-based cryptocurrencies. There are several DeFi-related policy concerns raised. The literature on decentralized finance is advanced by identifying potential areas for more study. [1]

Decentralized applications that are used to supply financial goods are exchanged on a DeFi market. All transactions in DeFi marketplaces are open and transparent, including the terms, conditions, and commitments of the financial service provider, the client, and the customer. A successful DeFi market requires a variety of components, including smart contracts, DeFi software protocols, decentralized apps (dApps), decentralized finance platforms, decentralized exchanges (DEXs), and decentralized finance lending platforms. In order to regulate digital assets and automate contract conditions between buyers and sellers, lenders and borrowers, and other parties, smart contracts are computer programmes that run on a blockchain. Blockchain protocols for decentralized finance are norms and regulations that are developed to control particular operations or activities. It permits communication between buyers, sellers, lenders, and borrowers.

Decentralised applications (DApps) are programmes that serve decentralized, Blockchain based, reliable services targeted towards end users. They can be utilised to trade in cryptocurrency assets, lend money, withdraw it, save it, spend it, trade derivatives, and lower risk. A decentralised finance platform is a consumer-facing financial interface that employs blockchain technology and bitcoin stakes. Without the requirement for a centralised authority, people can trade digital assets on decentralised exchanges (DEXs). A powerful algorithm that adjusts prices and executes deals based on available liquidity is designed to replace the market-making and custody tasks of exchanges in DEXs. Decentralised finance lending systems let cryptocurrency owners lend large sums of money to borrowers instantaneously and anonymously, provided they can provide enough security to fund a smart contract and settle the loan on time. [2]

There are several advantages to decentralized finance. Decentralised finance can increase financial inclusion, encourage intermediary-free innovation, eliminate the need for them, ensure transaction immutability, guarantee that all participants are bound by the same rules, prevent censorship, make it possible for anyone with internet access to audit transactions, reduce the cost of all

international transactions, encouraging trust less financial intermediation, and foster global participation. [3, 4]

Diverse opportunities are presented by decentralised finance. The first benefit of decentralised finance is greater efficiency. Decentralised finance can boost financial transactions' effectiveness. Decentralised finance can become more effective by replacing trust needs with smart contracts. Two parties willing to exchange digital assets in the form of tokens do not need a third party or financial intermediary in a decentralised financial environment. Instead, digital assets can be transferred using bilateral token transactions. Other efficiency advantages of token transfers include faster transactions, cheaper transactions, and less need for third-party audits. [5]

Decentralised financing advocates in Africa contend that it can improve access to financial services and hasten financial inclusion in the continent's nations. Arguments should also be made that decentralised blockchain technology and smart contracts can generate entire sectors in Africa with goods created to meet a variety of requirements in numerous African nations. It can also make smart contracts easier to use in a variety of tasks that are still done manually in African nations, including cross-border transactions, internet retail, freelance agreements, and tenancy and employment arrangements.[6]

Asia's high rate of digital literacy, widespread mobile internet access, and tech-savvy consumer demographic make the adoption of decentralised finance conceivable. In India, decentralised currency is gaining popularity as a crypto innovation. India is now ranked "sixth" in terms of decentralised finance adoption according to the 2021 Global DeFi Adoption Index. Decentralised finance (DeFi) proponents in India argue that it will provide access to decentralised financial services as well as low-cost borrowing and lending choices to millions of adult Indians who do not have bank accounts. There are concerns, meanwhile, that a ban by the government on all cryptocurrencies based on blockchain technology will stymie efforts by the private sector to grow India's decentralised finance sector.[7]

The discussion concerning the necessity for decentralised economies, the role of regulatory organisations, and whether bitcoin actually serves as a store of value has been revived in light of the recent expansion of COVID-19, ailing economies, and government money-pump interventions in the market. This explains the need for a new financial system and how blockchain technology and cryptocurrencies are essential to realising it. Oracles are a key component of blockchain applications since they allow them to communicate with the outside world. In this article, we've examined how oracles work before presenting a general architecture that can be used to create most financial instruments on blockchain.

On a permissionless blockchain, smart contracts operate in a distrustful, hostile environment where security is provided through deterministic and sequential actions. In order to ensure that the nodes arrive at consensus, determinism is essential. Oracles serve as a connection between off-chain data and the blockchain, allowing the blockchain to operate to take into account information and events that occur outside of the network. Smart contracts cannot retrieve their data from servers like conventional programmes can since the internet is non-deterministic and evolves over time.

Oracles do supply this external data to the blockchain, but it's vital to understand that they serve as the layer that searches for, verifies, and authenticates the data before passing it on to the smart contract. [8]

Security Considerations that cross the mind when working with oracles are
Before passing the data on the chain, the data carrier can see the outcome. The smart contract for consumers must make sure that application fairness is not compromised by knowing the solution before the outcome is made public, the data provider could decide to only release the results that favoured it, the smart contracts can prevent this activity by withholding the money that a participant wagered in the game and if the data holder doesn't release the outcome, There is a chance that a considerable portion of the nodes could be controlled by an

enemy or imposters in order to disseminate false information while still being a part of the majority. The effectiveness of countermeasures depends on the reputation and quantity of involved nodes. A participating node has the ability to maliciously copy another node's input. By using a commit-and-reveal strategy, which encrypts information before each oracle submits it and decrypts it once a distinguished number of oracles have posted their replies, this problem is solved. These security concerns should be kept in mind and should be resolved with a solution to protect. [9]

Being able to make payments directly from one party to another without the need for a middleman has likely been the main objective of crypto and DeFi from the start. But as the industry developed, a number of other blockchains have appeared that aim to outperform the Bitcoin blockchain. The most popular of these at the time of writing for DeFi applications is Ethereum, followed by Binance and Tron.

Although the underlying blockchains provide the clearing and settlement infrastructure, it is important to keep in mind that stablecoins, which are cryptocurrencies that aim to maintain a stable value against a major currency (typically the US dollar) or other secure assets like gold, frequently serve as the unit of account for transactions (Arner et al., 2020). In reality, a TradFi anchor is necessary for DeFi to work effectively. Once more, it is impossible that DeFi would have ever gained any scale without (for the most part centralized) stablecoins acting as a bridging function to the fiat world.[10]

Table 2.1: Economic activities and roles in TradFi and DeFi

Function	**TradFi example**	**DeFi example**
Settling and clearing payments to ease commerce.	Payment methods, bank accounts, e-wallets, credit cards, and central counterparties.	Stablecoins, the Bitcoin network, other blockchains (such as Ethereum and Solana),

		and automated market making (AMM).
Pooling funds to carry out big projects.	Bonds, mutual funds, exchange-traded funds (ETFs), and stocks.	DApps for asset management, governance tokens, and DeFi tokens.
Managing risk and uncertainty.	Loans, insurance agreements, derivatives, and hedging techniques.	Derivatives, smart contracts, hedging techniques, and DeFi insurance.
Deal with incentive problems.	Risk management and recurrent encounters with established counterparties.	Smart contracts, overcollateralization.
To help decentralised decision-making be coordinated, provide price information.	Trading, exchanges, and derivatives.	Trading activities, DEXs (and CEXs), AMM, and cryptocurrency derivatives.

Despite the enormous amount of research for traditional financial assets, the body of knowledge regarding the relationship between return and volume in the wider cryptocurrency market has only recently started to grow. Koutmos (2018) looks into the connection between Bitcoin returns and transaction activity using recognisable Bitcoin addresses and transactions as proxies. Bivariate vector autoregressive (VAR) models are used to show that there is a significant relationship between Bitcoin returns and transaction activity between January 2013 and September 2017, with returns explaining a larger portion of the variation in transaction activity than vice versa. Alaoui et al. (2019) employ multifractal detrended cross-correlations analysis (MF-DCCA) to examine the price-volume connection in the Bitcoin market from July 2010 to May 2018.

It is discovered that there is a nonlinear link between Bitcoin price returns and trading volume, and that this relationship is multifractal. This finding suggests that investors might profit from taking trading volume into account when making investment decisions. A significant variety of cryptocurrencies' daily price and volume variations are investigated by Stosic et al. (2019) for their multifractal behaviour. However, the findings imply that price changes are more complicated than volume changes and that the dynamics of both variables differ. [11]

This article investigates the static and dynamic return relationships between four popular DeFi assets—Chainlink, Maker, Basic Attention Token, and Synthetix—and four significant fiat currencies—the Chinese Yuan, Japanese Yen, Euro, and Pound Sterling. the use of a framework that combines a time-varying parameter vector autoregression and a connectivity method based on generalised forecast error variance decomposition. The static connectedness analysis reveals that there is not much connectivity between the DeFi markets and the conventional currency markets. The results of the dynamic research demonstrate that return spill overs vary over time, with a sharp increase in connectedness between the DeFi and currency markets in early 2020, at the time of the pandemic's initial escalation.

The fact that the spill over from the Chinese Yuan to the system shows no rise due to the collapse triggered by COVID-19 highlights a pandemic-induced separation of the Chinese financial system from other centralised and decentralised markets. System spill overs to the DeFi marketplaces were at an unprecedented height at the beginning of the epidemic. However, we continue to notice that the DeFi marketplaces primarily operate as net innovation transmitters throughout the first COVID-19 year. Additionally, a pairwise-like relationship between the inversely symmetric profiles for the Maker-Euro, Basic Attention Token-Japanese Yen, and Chainlink-Pound Sterling pairs and the net return spill over profiles is discovered. Given the time-varying transmission-reception patterns for all markets, investors and policymakers can

use our spill over study to improve portfolio allocation and regulation choices. [12]

DApps (Decentralised applications), which are designed to carry out financial operations on blockchains, are the source of DeFi's decentralised architecture. Users can sidestep traditional financial institutions as well as cryptocurrency structures like crypto exchanges by using DApps. These structures had become important hubs for cryptocurrency trading but also had a long list of problems of their own. DApp solutions enable DeFi users to execute direct transactions through mechanisms made possible by smart contract technology.

The first organization that may be regarded as a DeFi platform was MakerDAO, which also created and currently manages the stablecoin DAI (the name DAO denotes "Decentralised Autonomous Organisation"). The DAO and MakerDAO launched about simultaneously in 2016 and 2015, respectively. While striving to maintain a peg with the US Dollar, MakerDAO allows users to take a credit liability on the DAI token. Initially, this technique of DAI transmission on a credit-basis marked a novel way of trading, but since 2020, there has been a flowering of various new models for DeFi. [13]

CHAPTER-3

SYSTEM DEVELOPMENT

3.1 System Design

The description of various libraries utilized in development of this project:

1. **Ganache**

Ganache is a popular software tool used in blockchain development, particularly for Ethereum-based projects. It provides a local, personal Ethereum blockchain that you can use for development, testing, and debugging purposes. Ganache simulates the behaviour of a real Ethereum network but operates in a local environment, making it easier and faster to iterate and experiment with smart contracts and decentralized applications (DApps).

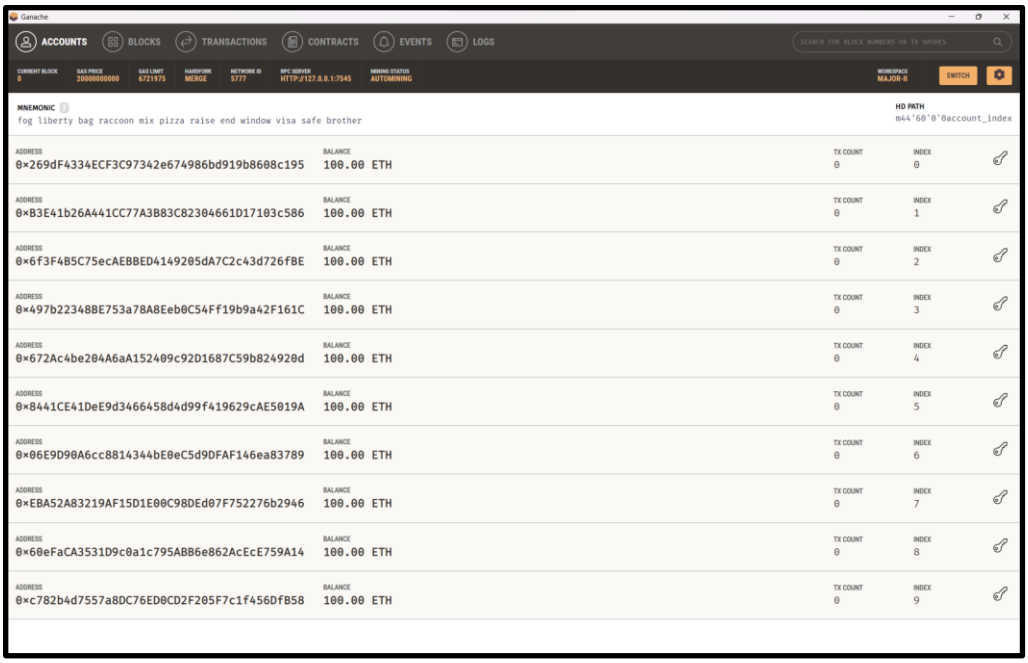


Figure 3.1.1.: Ganache tool usage in Ethereum project.

Following are the major functionalities of Ganache:

1. Ethereum Blockchain on local machine: We can interact with a local blockchain environment created by Ganache on your computer without connecting to the Ethereum main net or test net. This local blockchain is entirely under your control and is run on your computer.

2. Fast Generation of Blocks: We don't need to wait for block confirmations as you would on the open Ethereum network because Ganache generates blocks quickly. Additionally, it generates blocks in a deterministic manner, guaranteeing that the blocks are generated in the same order each time Ganache is restarted. For the sake of testing and debugging, this consistency is essential.

3. Accounts and Private Keys: We can test our smart contracts and DApps using a set of pre-funded accounts that Ganache provides. Private keys are linked to each account, enabling you to communicate with the blockchain and sign transactions. If you provide the private keys for your own accounts, you can import them as well.

4. Development of Ethereum Smart Contracts: Popular Ethereum development frameworks like Truffle and Remix are compatible with Ganache. On the nearby blockchain, we can deploy and test your smart contracts. Before publishing your smart contracts to the live Ethereum network, you may use Ganache to simulate various scenarios and interactions with them.

5. Logs of transactions and bugs: All transactions, including gas costs, contract addresses, and input/output values, are meticulously logged by Ganache. These logs can be quite helpful for troubleshooting and comprehending how your contracts act in various transactions.

6. Customizing working environment and networks: The gas limit, block time, and chain ID are just a few of the local blockchain network customization options available with Ganache. We can simulate various network situations and test edge cases thanks to this versatility.

7. User-Friendly GUI: Ganache has an intuitive graphical user interface that presents key details about the blockchain, such as accounts, transactions, and blocks. Additionally, it offers options for handling accounts, importing/exporting keys, and changing network configurations

8. Dual interaction options: Ganache is available in two versions: Ganache has a GUI (Graphical User Interface) and a CLI (Command Line Interface). While the GUI version offers a visual interface for interaction, the CLI version is used via the command line and is much more used and appreciated by the Blockchain Developers.

2. REMIX IDE

Most IDEs (Integrated Development Environments) are installed on a developer's local machine to write code and develop software. But unlike them, Remix is an online IDE designed for developing, testing and deploying Ethereum smart contracts on the Ethereum blockchain. Remix IDE provides an industry standard GUI that allows developers to code, compile and deploy smart contracts directly from any supported web browser.

Features of Remix Online IDE:

1. Remix Code Editor: Remix allows us to create smart contracts in Remix's code editor using the programming language Solidity. One can produce clear, error-free code with the help of the code editor's syntax highlighting, auto-completion, and error checking features.

2. Inbuilt Solidity Compiler: One can easily compile their smart contracts with Remix's integrated Solidity compiler without the need for any further setup. One has to select the target EVM (Ethereum Virtual Machine) version and other compilation settings.

3. Deployment of Smart Contracts: One can deploy their smart contracts using Remix on a variety of Ethereum networks, including the main Ethereum network, testnets (like Ropsten or Rinkeby), and the local development network. Within Remix, we can set the deployment parameters and directly communicate with deployed contracts.

4. Interaction with different Ethereum Smart Contracts: For easy communication with deployed smart contracts, one can easily rely on Remix’s user interface. The contracts' state variables can be viewed, their functions can be called, and we can create instances of them. Remix IDE makes it way simpler to test and confirm contract behaviour by creating user interfaces (UI) based on the functions and events of your contract.

5. Testing and Debugging: One may go through their smart contracts line by line with Remix's robust debugging functionality, analyse different variables, and create alternative breakpoints. Our contracts' execution flow can be observed, allowing us to spot any problems or faults and address them.
6. Integrated Plugins: Remix IDE supports various additional plugins that extend its overall functionality. These plugins can provide additional features, such as code analysis, security checks, and integration with external services. These can be installed form the IDE’s Plugin manager.
7. File Management System: You may step through your smart contracts line by line with Remix's robust debugging functionality, analyse variables, and create breakpoints. Your contracts' execution flow can be observed, allowing you to spot any problems or faults and address them.

Additionally, Remix supports events and logs, enabling you to examine the events that were released during contract execution.

8. Interaction and Integration with other software: Remix easily connects with various well-liked development frameworks and tools. The Truffle framework, for instance, is supported, enabling the import of Truffle projects and the use of Truffle-specific working features in Remix.

9. Advanced Collaboration and Sharing capabilities: You can share your projects on Remix and work together with other developers. You can create a URL for your project so that other users of Remix can view and engage with your contracts.

3.2 Tools and technologies used:

1. **Truffle**: A framework for Ethereum, that contains a suite of tools and libraries for smart contracts development.
2. **Ganache:** A personal blockchain emulator that provides a local testing environment for Ethereum development.
3. **Solidity:** A high-level programming language specifically designed for writing smart contracts on the Ethereum.
4. **React.js:** A JavaScript library for building user interfaces for web applications with ease.
5. **Express.js**: A framework for Node.js, that acts as a middleware that simplifies the development of backend and APIs.
6. **Web3.js**: A JavaScript library that serves as an interface for interacting with the Ethereum smart contracts, sending transactions, and retrieving blockchain data.

3.3 Smart Contract Flowcharts:

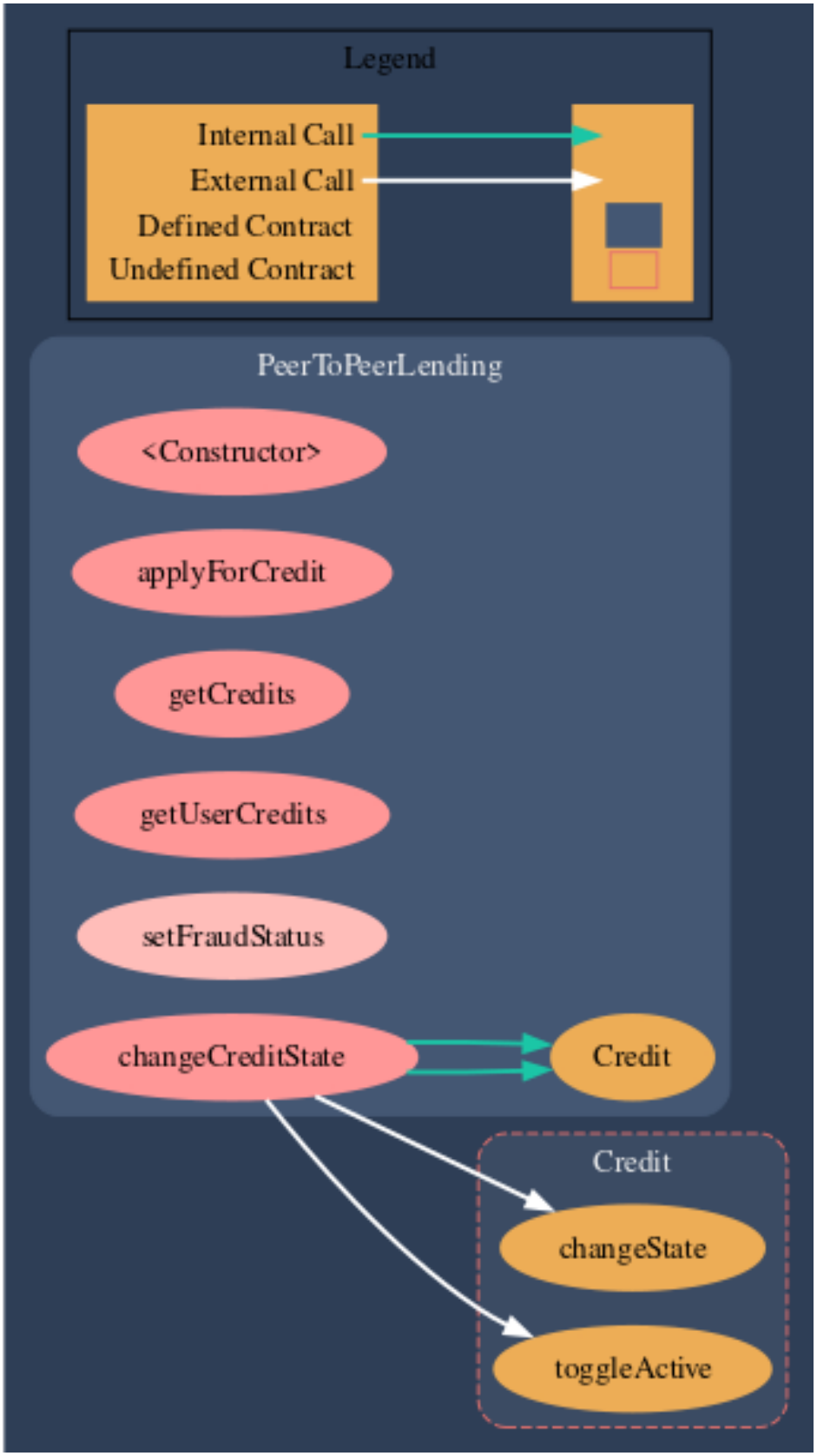


Figure 3.2.1: Peer-to-Peer Smart Contract flow chart.

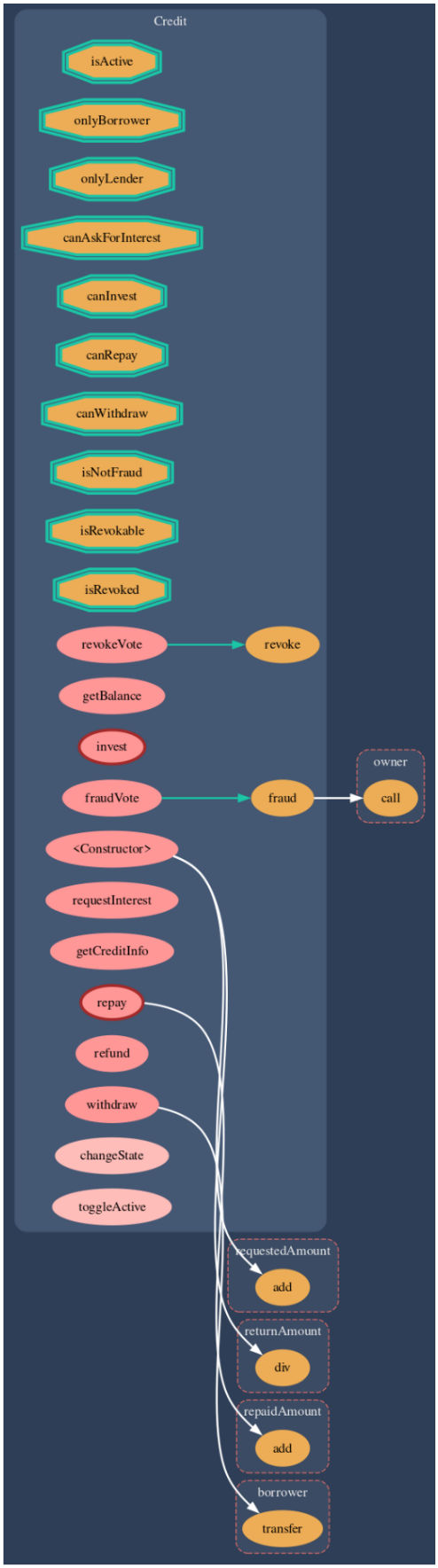


Figure 3.2.2: Credit Smart Contract Flowchart.

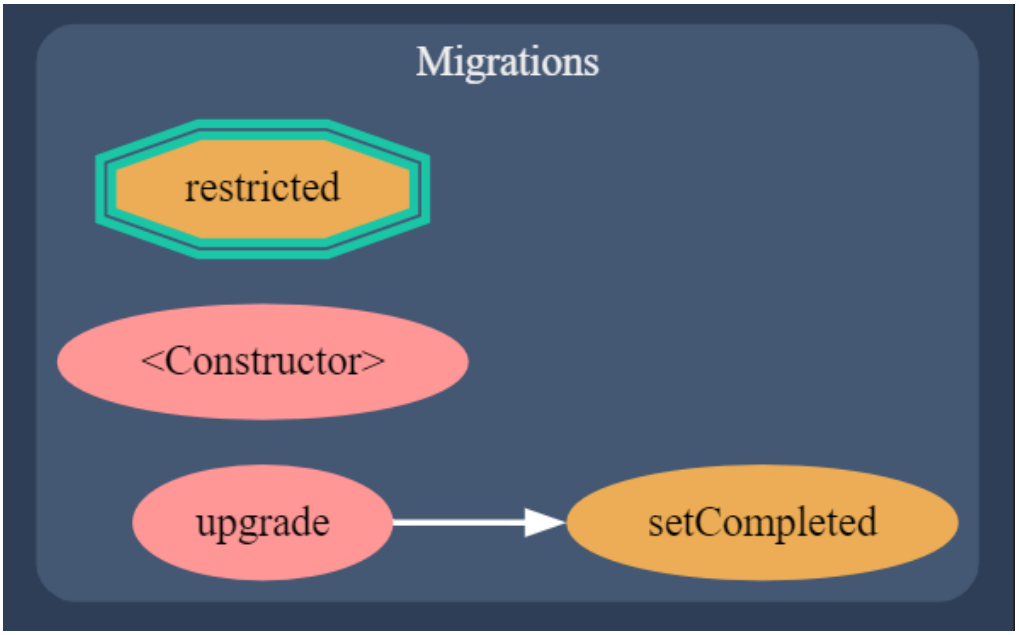


Figure 3.2.3: Migrations Smart Contract flow chart.

3.4 Web Application Hosting:

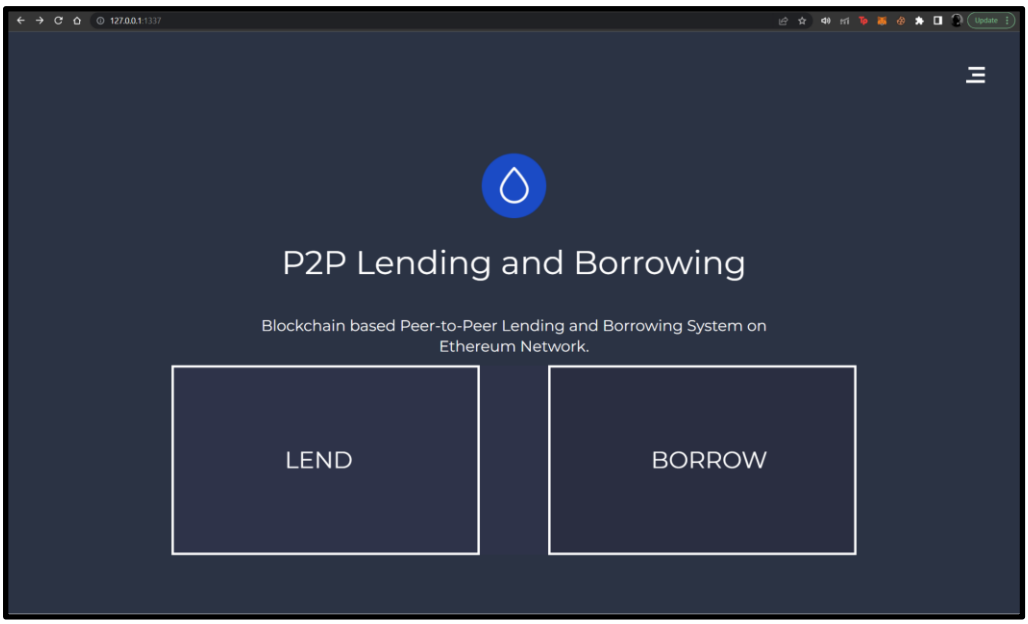


Figure 3.3.1: Application GUI hosted on localhost.

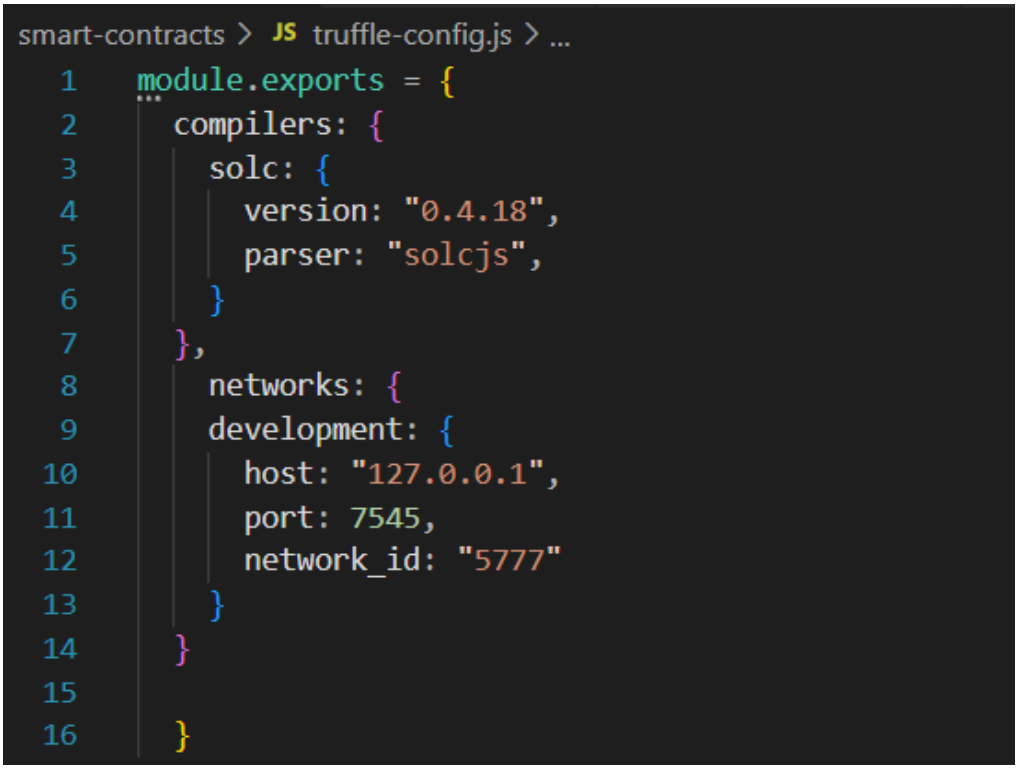


Figure 3.3.2: Truffle Config for the Solidity Smart Contracts.

In truffle configuration file, under the compilers section, the sold object is used to configure the Solidity compiler, the version of the Solidity compiler is set to "0.4.18".the network specifies that the project should connect to a local development environment running on the local host "127.0.0.1" address with the port "7545" and network ID 5777, used by Ganache for Ethereum Blockchain development.

CHAPTER-4

PERFORMANCE ANALYSIS

4.1 Results at various stages of the Decentralized Application.

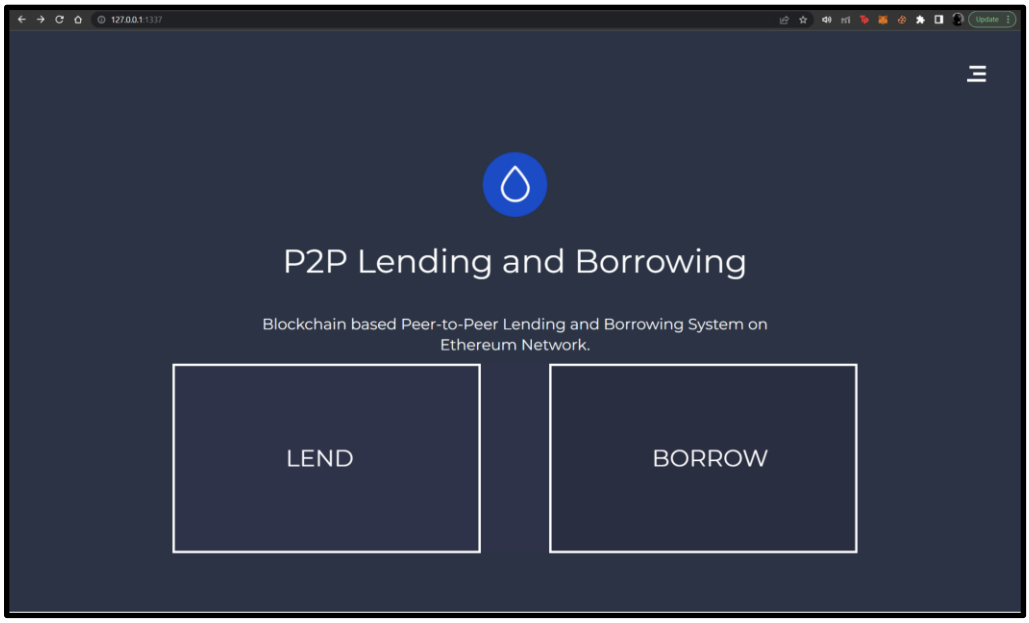


Figure 4.1.1: Decentralized Application GUI.

Figure 4.1.2: Application hamburger menu.

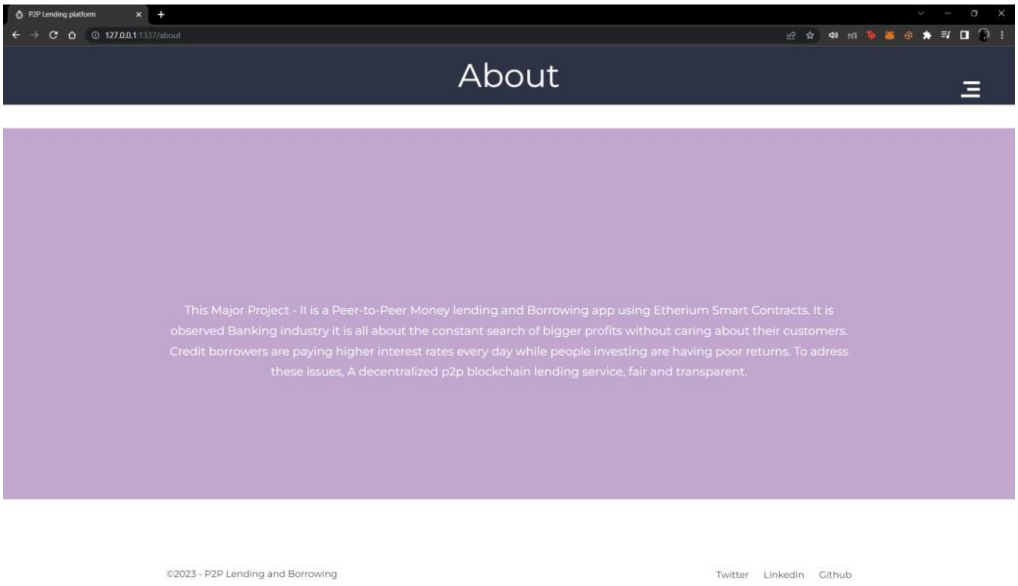


Figure 4.1.3: About section in the Submenu of the Application.

Figure 4.1.2: First Deployment of Smart Contract Migration.

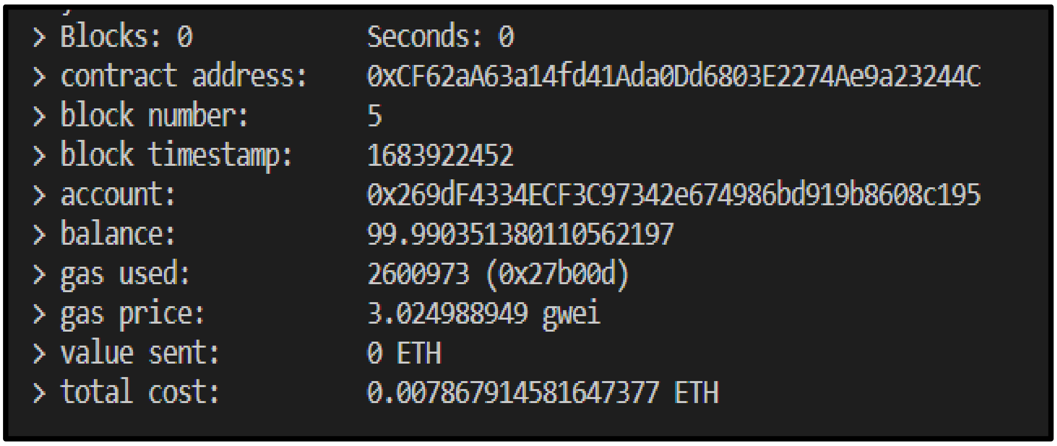


Figure 4.1.3: First Smart Contract Deployment details.

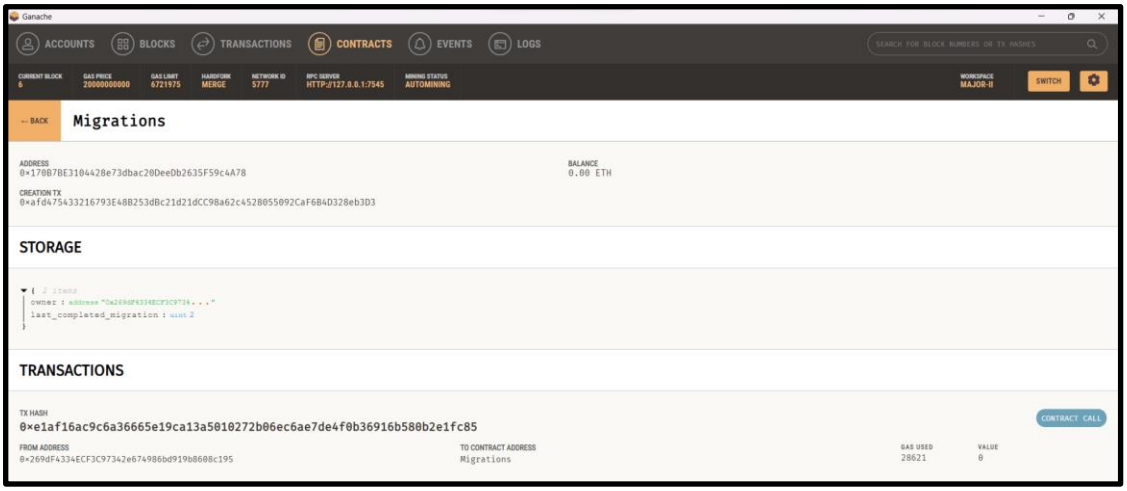


Figure 4.1.4: Ganache view for migrations Deployment.

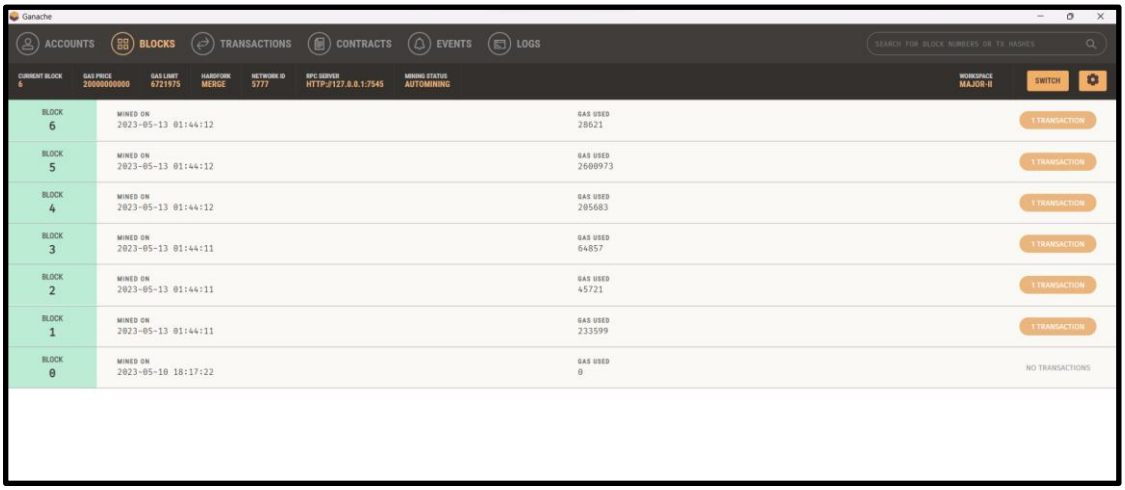


Figure 4.1.5: Blockchain view after first contract deployment.

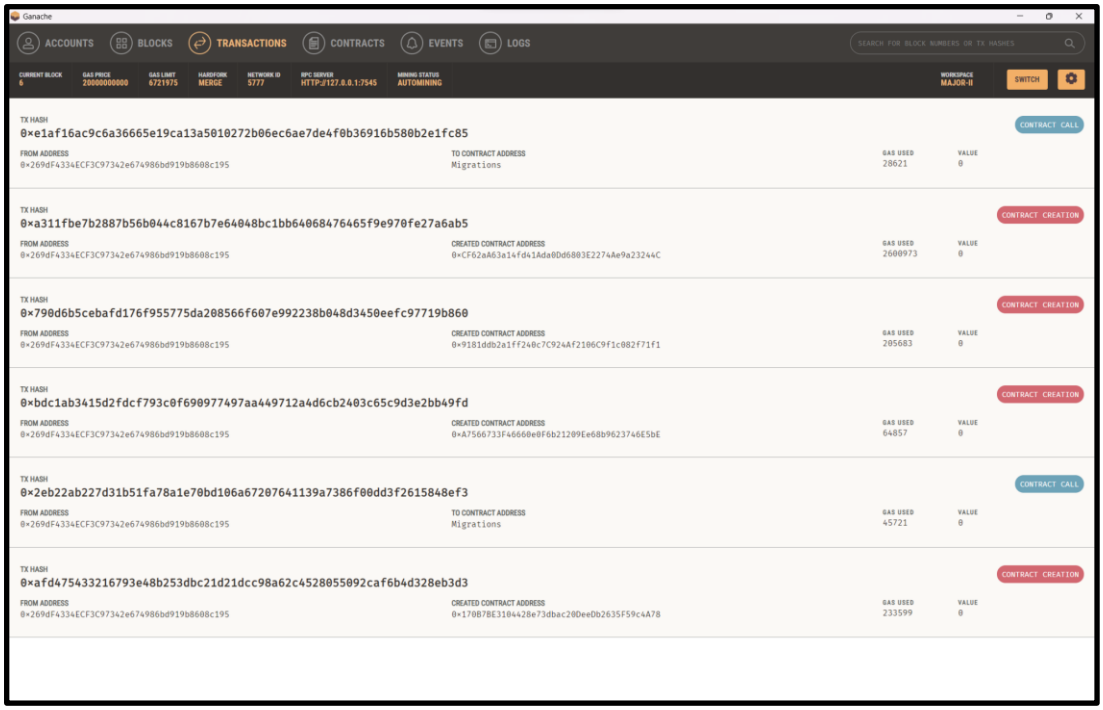


Figure 4.1.6: Blockchain Transaction History.

CHAPTER-5
CONCLUSIONS

A peer-to-peer lending and borrowing platform built on Web3.js and Ethereum Blockchain using smart contracts provides advantages of financial transparency, security, efficiency, and flexibility. As more and more of these factors continue to evolve, the financial system is likely to undergo an interplay between traditional institutions and decentralized platforms, shaping the future of finance. On the other hand, conventional banking credit systems provide a well-developed framework with extensive services and consumer protection.

The traditional banking credit systems offer a well-established infrastructure and regulatory framework. They provide a range of financial services, including insurances, loans, credit and debit services, with a long history and established mechanisms for risk management. However, Blockchain technology's transparency and immutability also increase the security and confidence of lending processes. With less need for manual paperwork and a lower danger of fraud or manipulation, smart contracts assure automated execution of loan agreements. Because blockchain transactions are transparent, lenders and borrowers can verify the legitimacy and track record of counterparties, which lowers the risk of default and enhances credit score.

It would be premature to claim that platforms for peer-to-peer lending and borrowing based on Blockchain networks will totally replace the established banking system, given the fast-evolving nature of the financial technology landscape. P2P platforms have indisputable benefits, including inclusion, openness, and efficiency, but they also encounter difficulties with scalability, regulatory frameworks, and building trust for now.

5.2 Future Scope

The usage of these peer-to-peer credit platforms might get popularized in the years to come, but their existence would most likely be an alternative to the traditional system, rather than their total replacement soon.

The future development of this project though, would be to include the following functionalities :

1. Working on Bugs faced during testing of the application UI.
2. Automated building of smart contracts, without the using Command line interface to build them using truffle.
3. Hosting the application on Cloud Server, so that the accessibility and functionalities can be improved and upgraded.
4. Integrating more Ethereum smart contracts to introduce some more financial tools.
5. Building a mobile application, launching app to work on real Ethereum blockchain network.
6. Adding more capabilities to the application as new features and functionalities are introduced for Solidity programming language.

REFERENCES

[1] Ozili, P.K., “Decentralized finance research and developments around the world.” , J BANK FINANCE TECHNOL 6, 117–133, Aug 2022.

[2] De Meijer, “DeFi and regulation: the European approach.” Finextra B, Jun 2021.

[3] Ozcan R . (2021, Jun15). Financial ecosystem and strategy in the digital era [Online]. Available: https://link.springer.com/book/10.1007/978-3-030-72624-9

[4] Y. Chen and C. Bellavitis, "Blockchain Disruption and Decentralized Finance: The Rise of Decentralized Business Models," in J. Bus. Ventur. Insights, vol. 13, p. e00151, 2020.

[5] F. Schär, "Decentralized Finance: On Blockchain- and Smart Contract-Based Financial Markets," FRB of St. Louis Review, vol. 103, no. 2, pp. 153-174, 2021.

[6] A. Ijaoba, "How DeFi Can Facilitate Better Access to Financial Services in Africa," A Benjamin Dada Report, July 14, 2021.

[7] R. Bhuvana and P. S. Aithal, "RBI Distributed Ledger Technology and Blockchain - A Future of Decentralized India," in Int. J. Manage. Technol. Soc. Sci. (IJMTS), vol. 5, no. 1, pp. 227-237, 2020.

[8] M. Kumar, N. Nikhil and R. Singh, "Decentralising Finance using Decentralised Blockchain Oracles," in 2020 International Conference for Emerging Technology (INCET), Belgaum, India, 2020, pp. 1-4, doi: 10.1109/INCET49848.2020.9154123.

[9] J. Heiss, J. Eberhardt, and S. Tai, "From Oracles to Trustworthy Data On-chaining Systems," 2019. Available at: http://www.redaktion.tuberlin.de/fileadmin/fg308/publications/2019/Heiss-et-aloracles_preprint.pdf

[10] M. Aquilina, J. Frost, A. Schrimpf, and A. Schrimpf, "Decentralised Finance (DeFi): A Functional Approach," Jan. 16, 2023. [Online]. Available: http://dx.doi.org/10.2139/ssrn.4325095

[11] J. Chu, S. Chan, and Y. Zhang, "An analysis of the return-volume relationship in decentralised finance (DeFi)," Int. Rev. Econ. Finance, vol. 85,

pp. 236-254, 2023. [Online]. Available: https://doi.org/10.1016/j.iref.2023.01.006.

[12] I. Yousaf, R. Nekhili and M. Gubareva, “Linkages between DeFi assets and conventional currencies: Evidence from the COVID-19 pandemic”, Int. Rev. of Financial Analysis, vol. 81, ISSN 1057-5219, 2022. [Online]. Available: https://doi.org/10.1016/j.irfa.2022.102082.

[13] Chohan, Usman W., “Decentralized Finance (DeFi): An Emergent Alternative Financial Architecture” Critical Blockchain Research Initiative (CBRI) . Jan 26, 2021. [Online]. Available at SSRN: https://ssrn.com/abstract=3791921

\

APPENDICES

SOURCE CODE :

A. Smart Contracts :

PeerToPeerLending.sol | Migrations.sol

smart-contracts > contracts > Migrations.sol > ...

report | graph (this) | graph | inheritance | parse | flatten | funcSigs | uml | draw.io

```
1   pragma solidity >=0.4.18 <0.4.26;
2
    UnitTest stub | dependencies | uml | draw.io
3   contract Migrations {
4       address public owner;
5       uint public last_completed_migration;
6
7       modifier restricted() {
8           if (msg.sender == owner) _;
9       }
10
        ftrace | funcSig
11      function Migrations() public {
12          owner = msg.sender;
13      }
14
        ftrace | funcSig
15      function setCompleted(uint completed↑) public restricted {
16          last_completed_migration = completed↑;
17      }
18
        ftrace | funcSig
19      function upgrade(address new_address↑) public restricted {
20          Migrations upgraded = Migrations(new_address↑);
21          upgraded.setCompleted(last_completed_migration);
22      }
23  }
```

PeerToPeerLending.sol | *Destructible.sol*

smart-contracts > contracts > common > Destructible.sol > ...

report | graph (this) | graph | inheritance | parse | flatten | funcSigs | uml | draw.io

```
1   pragma solidity >=0.4.18 <0.4.26;
2   import "./Ownable.sol";
3
    UnitTest stub | dependencies | uml | draw.io
4   contract Destructible is Ownable {
        ftrace | funcSig
5       function Destructible() public payable {}
6
        ftrace | funcSig
7       function destroy() public onlyOwner {
8           selfdestruct(owner);
9       }
10
        ftrace | funcSig
11      function destroyAndSend(address _recipient↑) public onlyOwner {
12          selfdestruct(_recipient↑);
13      }
14  }
```

PeerToPeerLending.sol | Ownable.sol ●

smart-contracts > contracts > common > Ownable.sol > ...

report | graph (this) | graph | inheritance | parse | flatten | funcSigs | uml | draw.io

```
1   pragma solidity >=0.4.18 <0.4.26;
2
    UnitTest stub | dependencies | uml | draw.io
3   contract Ownable {
4       address public owner;
5
6       event LogOwnershipTransfered(
7           address indexed _currentOwner,
8           address indexed _newOwner
9       );
10
11      modifier onlyOwner() {
12          require(msg.sender == owner);
13          _;
14      }
15
        ftrace | funcSig
16      function Ownable() public {
17          owner = msg.sender;
18      }
19
20  }
```

PeerToPeerLending.sol | SafeMath.sol ●

smart-contracts > contracts > common > SafeMath.sol > SafeMath > div (complex: 0 state: □)

report | graph (this) | graph | inheritance | parse | flatten | funcSigs | uml | draw.io

```
1   pragma solidity >=0.4.18 <0.4.26;
2
    UnitTest stub | dependencies | uml | draw.io
3   library SafeMath {
        ftrace | funcSig
4       function mul(uint256 a↑, uint256 b↑) internal pure returns (uint256) {
5           if (a↑ == 0) {
6               return 0;
7           }
8           uint256 c = a↑ * b↑;
9           assert(c / a↑ == b↑);
10          return c;
11      }
12
        ftrace | funcSig
13      function div(uint256 a↑, uint256 b↑) internal pure returns (uint256) {
14          uint256 c = a↑ / b↑;
15          return c;
16      }
17
        ftrace | funcSig
18      function sub(uint256 a↑, uint256 b↑) internal pure returns (uint256) {
19          assert(b↑ <= a↑);
20          return a↑ - b↑;
21      }
22
        ftrace | funcSig
23      function add(uint256 a↑, uint256 b↑) internal pure returns (uint256) {
24          uint256 c = a↑ + b↑;
25          assert(c >= a↑);
26          return c;
27      }
28  }
29
```

PeerToPeerLending.sol truffle-config.js

smart-contracts > truffle-config.js > ...

```
module.exports = {
  compilers: {
    solc: {
      version: "0.4.18",
      parser: "solcjs",
    }
  },
    networks: {
    development: {
      host: "127.0.0.1",
      port: 7545,
      network_id: "5777"
    }
  }

  }
```

B. Web Application Middleware (express.js)

express.js

client-app > server > config > express.js > ...

```
const express = require('express')
const bodyParser = require('body-parser')
const handlebars = require('express-handlebars')
const helpers = require('handlebars-helpers')()

module.exports = (app) => {

    app.engine('handlebars', handlebars({
        defaultLayout: 'main'
    }))
    app.set('view engine', 'handlebars')

    app.use(bodyParser.urlencoded({ extended: true }))

    app.use(express.static('public'))

    console.log('Express ready!')
}
```