

DECENTRALIZED STOCK EXCHANGE USING BLOCKCHAIN TECHNOLOGY

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering degree in Computer Science and Engineering

By
RONIE JOE J
(Reg No - 40731082)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

CATEGORY -1 UNIVERSITY BY UGC
Accredited with Grade "A++" by NAAC
JEPPIAAR NAGAR, RAJIV GANDHISALAI,
CHENNAI - 600119

APRIL – 2024

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **Ronie Joe J (40731082)** who carried out the Project entitled **"DECENTRALIZED STOCK EXCHANGE USING BLOCKCHAIN TECHNOLOGY"** Under my supervision from November 2023 to April 2024.


Internal Guide

Dr. S. L. JANY SHABU, M.Tech., Ph.D.


HEAD OF THE DEPARTMENT
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SATHYABAMA
Head of the Department
Dr. L. LAKSHMANAN, M.E., Ph.D.
(Rajiv Gandhi Salai,
Chennai - 600 119)

Submitted for Viva voce Examination held on 15-4-24


Internal Examiner


External Examiner

DECLARATION

I, Ronie Joe J (40731082) hereby declare that the Project Report entitled **DECENTRALIZED STOCK EXCHANGE USING BLOCKCHAIN** done by me under the guidance of **Dr.S.L.JANY SHABU, M.Tech., Ph.D.** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in **Computer Science and Engineering**.

DATE: 15/04/2024

PLACE: Chennai

A handwritten signature in blue ink, appearing to read 'Ronie Joe J', with a long horizontal flourish extending to the right.

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph. D, Dean**, School of Computing, **Dr.L. Lakshmanan M.E., Ph.D.**, Head of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. S. L. Jany Shabu M.Tech, Ph.D.**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Traditional stock exchanges operate within a centralized framework, where transactions are facilitated and regulated by intermediaries such as brokers, clearinghouses, and central authorities like stock exchanges and regulatory bodies. While this system has functioned well for decades, it is not without its limitations, including high transaction costs, lengthy settlement times, susceptibility to fraud, and limited accessibility for certain market participants. The advent of blockchain technology presents an opportunity to address these shortcomings by decentralizing the stock exchange process. Blockchain, a distributed ledger technology, enables the recording and verification of transactions across a network of computers in a transparent, immutable, and tamper-resistant manner. By leveraging blockchain, a decentralized stock exchange can operate without the need for intermediaries or central authorities, thus streamlining the trading process and reducing associated costs. Smart contracts, self-executing contracts with the terms of the agreement directly written into code, play a crucial role in facilitating transactions on a decentralized stock exchange. These smart contracts automatically enforce the terms of a trade once predefined conditions are met, eliminating the need for intermediaries to oversee transactions. This not only accelerates the settlement process but also minimizes the risk of human error and manipulation. Moreover, decentralization ensures that the stock exchange is resilient to single points of failure and immune to censorship or control by any single entity. Each transaction is validated and recorded by multiple nodes within the blockchain network, ensuring consensus and trust among participants. From an accessibility standpoint, a decentralized stock exchange opens up financial markets to a broader range of investors, including those in regions with limited access to traditional financial infrastructure. Individuals can participate in trading activities directly from their digital wallets, without the need for a third-party intermediary. However, it's essential to acknowledge the challenges and regulatory considerations associated with decentralized finance (DeFi) projects, including security vulnerabilities, compliance with existing financial regulations, and interoperability with traditional financial systems.

TABLE OF CONTENTS

Chapter No	TITLE	Page No.
	ABSTRACT	v
	LIST OF FIGURES	viii
	LIST OF ABBREVIATIONS	ix
1	INTRODUCTION	1
	1.1 Background and Motivation	2
	1.2 Problem Statement	2
	1.3 Objective and Scope	3
	1.4 Application of Decentralized Stock Exchange	4
2	LITERATURE SURVEY	7
	2.1 Overview Of Existing Research And Literature	7
	2.2 Inferences From Literature Survey	10
	2.2.1 Distributed Ledger (Blockchain) Technology	10
	2.2.2 Properties of Distributed Ledger	10
	2.2.3 Decentralized Exchanges	11
	2.3 Open problems in Existing System	12
3	REQUIREMENTS ANALYSIS	13
	3.1 Feasibility Studies/Risk Analysis of the Project	13
	3.1.1 DEX Front Running Risks	13
	3.1.2 Limited Recovery Ability of DEX	14
	3.1.3 Limited Support for DEX Trading	15
	3.2 Software Requirements Specification Document	15
	3.3 Ethereum Blockchain	19
4	DESCRIPTION OF PROPOSED SYSTEM	22
	4.1 Selected Methodology or process model	23
	4.2 Architecture / Overall Design of Proposed System	23
	4.3 Description of Software for Implementation and Testing plan of the Proposed Model/System	28

	4.4	Project Management Plan	31
	4.4.1	Transaction Cost	32
5		IMPLEMENTATION DETAILS	33
	5.1	Development and Deployment Setup	33
	5.2	Testing	36
6		OUTCOMES AND DISCUSSION	38
	6.1	Introduction	38
	6.2	IEEE Standards Followed in the Project	39
7		RESULT AND DISCUSSION	40
	7.1	Conclusion	40
		REFERENCES	41
		APPENDIX	43
		A. SOURCE CODE	43
		B. SCREENSHOTS	65
		C. RESEARCH PAPER	66

LIST OF FIGURES

FIGURE NO	FIGURE NAME	Page No.
3.1	Node javascript logo	9
3.2	Visual Studio Code IDE logo	10
3.3	Ethereum Block Chain Network	10
4.1	Architecture for Buyer Working	12
4.2	Architecture for Company Working	13
4.3	Overall System Architecture	14
B.1	Decentralized Exchange Output	17
B.2	Buy and Sell Contracts	51

LIST OF ABBREVIATIONS

ABBREVIATIONS

ETH
P2P
KYC
CEX
DEX
DSE

EXPANSION

Ether
Peer-to-Peer Know Your Customer
Centralized Exchange
Decentralized Exchange
Decentralized Stock Exchange

CHAPTER 1

INTRODUCTION

Since the 17th century when the Dutch East India company was the first listed company on a stock exchange, the world's economy was built around and relies upon stock exchanges where millions of trades are performed every day helping companies raise their value. A stock exchange market is an aggregation of buying and selling offers corresponding to an asset. An asset can represent equities or stocks of companies, bonds, or other securities. People who buy or sell the assets are called investors while the persons who perform the transactions are called brokers or traders. Modern stock exchanges are highly computerized and can handle vast number of transactions in a short amount of time, assuring the security, execution and authenticity of transactions at a cost of a transaction fee, usually directly proportional with the value of the auction. The traditional stock markets are implemented in a centralized application that gathers all the trading actions. This architecture has many benefits by having a central authority that ensures the authenticity, security and validity of the transactions. However, the centralization has also a lot of drawbacks, such as having a single point of failure, a possible performance bottleneck or susceptibility to attacks. Furthermore, the central authority oversees the fees and there is a lack of transparency of the auctioning process for the trader. This paper proposes decentralized stock exchange architecture to tackle the shortcomings identified above by using the new and emergent blockchain technology. The potential of the blockchain system can bring benefits to the entire system, by leveraging on full replication and verification by all the nodes in the system regarding the assets owned, the execution of the market orders and the correct settlement between the accounts. Furthermore, the ensured immutability of the ledger brings a valuable advantage with respect to the centralized system. Also, by decentralizing the system, there is no central authority or intermediate needed for placing and executing the orders. To demonstrate that a blockchain based solution can provide a better option regarding the order transaction fees we implement both a centralized and decentralized ethereum based prototype for a subset of rules of the Bucharest Stock Exchange (BVB), perform a set of transactions in centralized and decentralized fashion and compare the transaction fees for both approaches.

1.1 BACKGROUND AND MOTIVATION

The development of decentralized automation in stock exchanges is rooted in addressing inefficiencies and limitations inherent in traditional centralized systems. Historically, centralized exchanges have faced challenges such as processing delays, susceptibility to manipulation, and high barriers to entry for market participation. The emergence of blockchain technology, with its features of transparency, security, and decentralization, has provided a compelling solution to these issues. By harnessing blockchain's capabilities, decentralized automation enables trustless and efficient transactions through smart contracts, eliminates the need for intermediaries, and creates a more inclusive and accessible marketplace. Additionally, the decentralized nature of these exchanges offers resilience to regulatory changes and enhances market integrity, fostering a paradigm shift towards a more democratic and innovative financial ecosystem. The motivation for decentralized automation of stock exchanges lies in the pursuit of efficiency, transparency, and accessibility. By leveraging blockchain technology and smart contracts, decentralized exchanges offer faster transaction processing, reduced counterparty risk, and increased transparency through immutable, publicly accessible ledgers. This decentralized approach also enables greater resilience to regulatory changes and provides global access to a wider range of participants, ultimately democratizing access to the stock market and fostering innovation in financial markets.

1.2 PROBLEM STATEMENT

The problem statement for decentralized automation of stock exchanges revolves around addressing the inefficiencies, lack of transparency, and centralized control inherent in traditional exchange platforms. Current stock exchanges are susceptible to delays, manipulation, and single points of failure due to their centralized architecture. Additionally, the reliance on intermediaries introduces counterparty risk and increases transaction costs. Moreover, regulatory constraints and geographical barriers limit market access for many potential participants. Therefore, the need arises for a decentralized automation solution that leverages blockchain technology and smart

contracts to enable trustless, transparent, and efficient transactions while reducing counterparty risk, lowering barriers to entry, and ensuring global market access. This problem statement seeks to foster the development of decentralized exchanges that revolutionize the stock trading landscape by democratizing access, enhancing market integrity, and promoting innovation.

1.3 OBJECTIVE AND SCOPE

Objective

The objective of decentralized automation in the stock exchange domain is to create a transparent, efficient, and inclusive marketplace that leverages blockchain technology and smart contracts to automate trading processes while reducing reliance on intermediaries. This objective encompasses several key goals:

Enhance Efficiency: Develop a decentralized exchange platform that can execute transactions swiftly and securely, minimizing processing delays and increasing overall market efficiency.

Ensure Transparency: Implement blockchain technology to maintain a transparent and immutable ledger of all transactions, providing stakeholders with real-time access to trade data and fostering trust in the marketplace.

Reduce Counterparty Risk: Utilize smart contracts to automate trade settlements and eliminate the need for intermediaries, thereby reducing counterparty risk and ensuring the timely and secure transfer of assets.

Democratize Access: Create a decentralized exchange that offers global market access to a diverse range of participants, including retail investors, institutions, and traders from all geographical regions, thereby promoting financial inclusion.

Foster Innovation: Provide a platform for the trading of a wide variety of assets, including traditional securities, digital tokens, and other innovative financial instruments, fostering innovation and diversity in the marketplace.

Scope

The scope of this project includes the development of a decentralized exchange platform, the implementation of robust security measures to protect user assets, the integration of user-friendly interfaces for seamless trading experiences, and compliance with relevant regulatory frameworks to ensure legal compliance and market integrity. Additionally, ongoing research and development efforts will focus on optimizing the platform's performance, expanding its asset offerings, and enhancing its scalability to support increasing transaction volumes. Ultimately, the objective and scope of decentralized automation in the stock exchange domain aim to transform the way securities are traded, making markets more accessible, efficient, and innovative for all participants.

1.4 APPLICATIONS OF DECENTRALIZED STOCK EXCHANGE

Efficient Trading: Decentralized exchanges (DEXs) enable peer-to-peer trading of assets without the need for intermediaries. Through automation powered by smart contracts, transactions can be executed instantly and securely, eliminating the delays and inefficiencies associated with traditional exchanges.

Increased Transparency: Blockchain technology provides a transparent and immutable ledger of transactions on decentralized exchanges. This transparency enhances trust among participants by allowing them to verify transaction histories and asset ownership in real-time, thereby reducing the risk of fraud and manipulation.

Reduced Counterparty Risk: Smart contracts automate the settlement of trades on decentralized exchanges, eliminating the need for intermediaries and reducing counterparty risk. By ensuring that trades are executed only when predefined conditions are met, smart contracts mitigate the risk of default and enhance the security of transactions.

Global Access: Decentralized exchanges offer global access to financial markets, allowing users from around the world to trade assets without the need for geographical or regulatory barriers. This democratization of access enables greater liquidity and diversity of assets, fostering innovation and investment opportunities for a broader range of participants.

Innovative Financial Instruments: Decentralized exchanges support the trading of a wide range of assets, including traditional securities, digital tokens, and other innovative financial instruments. By providing a platform for the creation and exchange of new asset classes, decentralized automation encourages innovation in financial markets and enables the development of new investment opportunities.

Resilience to Centralized Failures: Decentralized automation reduces reliance on centralized infrastructure, making exchanges more resilient to failures and disruptions. By distributing transaction processing across a network of nodes, decentralized exchanges are less vulnerable to single points of failure, ensuring continuous operation and availability even in the face of unforeseen events.

CHAPTER 2

LITERATURE SURVEY

Blockchain is a distributed ledger that allows data to be stored on blocks, where each block is linked to the other using cryptography. The primary advantage of using blockchain is to achieve decentralized architectures.

2.1 OVERVIEW OF EXISTING RESEARCH AND LITERATURE

Y. Yuan et.al [12] a six-layer reference model framework for blockchain technology. The topmost layer is the application layer, which refers to the scope of blockchain, i.e. Finance, Healthcare, Education etc. The contract layer consists of various algorithms and self enforcing smart contracts which serve as the crux for any application. The next layer, the Incentive layer is optional in the framework; this layer ensures data verification and encourages various nodes in the network to participate in the verification of security. The consensus layer consists of multiple algorithms that guarantee consistency of data and ensure a fault-tolerant environment via algorithms such as Proof of Work, Proof of Stake etc. The network model emphasizes decentralized communication ensuring a P2P network, data transparency and data verification. The bottom-most layer, also known as the data layer is responsible for creating the blocks and linking them via cryptography. The data stored on each block is encrypted via algorithms such as hashing. The blocks in the network are chained one by one in chronological order and linked via cryptography. The above mentioned six layers make up the proposed reference model.

C. Pop et.al [3] a decentralized stock exchange and a continuous open market using blockchain which tackles the shortcomings of the traditional centralized stock exchange systems, such as high transaction fees, centralized governance susceptible to attacks and lack of openness. Reduction of transaction costs is the aim of any blockchain system. They make use of a single, smart contract for the Buyer, Seller and the Exchange. The proposed system has been implemented in Ethereum.

M. H. Miraz et.al [5] have a framework that uses blockchain to help make transactions on the exchange more secure. They have discussed the technical and legal implications of such a system. The proposed method uses a hybrid model of blockchain involving proof of stake for matching buyers to sellers while adopting a round-robin approach for settlement procedures. Their framework pays immense focus to the technology as well as legal aspects of such applications.

X. Xu et.al [9] explore the factors for a stock exchange to be implemented using blockchain. They state that while blockchain provides many advantages such as security and elimination of intermediaries, one issue that is yet to be tackled is data transparency. Every market participant is exposed to the blockchain, which thus creates a disadvantage for several investors. Such an issue creates a hurdle when trying to build the proposed system to the scale of traditional stock exchanges. Although an evolving concept, blockchains have certain limitations, hence, they do not fit all scenarios. The author emphasizes the need to evaluate the system requirements before designing a system.

Y. Wanjun et.al [2] have explored a framework for network trading systems based on blockchain technology. The proposed framework has three modules, a business center system, a payment center system and a third-party identity authentication center. The system provides data integrity and prevents any data loss during network transmission. The system uses a P2P network technology. Their system uses algorithms such as asymmetric encryption technology, identity authentication technology and consensus mechanism to ensure data integrity and prevent any data loss during network transmission.

Through a self-executing smart contract, it is possible to eliminate the need for third party agents and mediators. The price of executing a smart contract directly depends on the amount of gas that is consumed by the transaction.

W. Hu et.al [10] have an algorithm to reduce gas consumption using business process modelling notation (BPMN) models and Petri nets. They propose a rule to map the BPMN model to solidity code. Their method can write the optimized smart contract template, ensuring the integrity of the original business proposal.

E. Leka et.al [4] SLR(Systematic Literature Review) to explore the trajectory of recent studies relating to blockchains using smart contracts and their applications. They discuss the fundamental issues concerning these technologies while also providing areas of research for the future. Endpoint vulnerabilities, public and private key security, blockchain integration platforms are some such issues that have been discussed.

A. Dixit et.al [6] Belief-Desire-Intention (BDI) model-based Multi- Agent System (MAS) to create smart contracts that are not only more flexible but are also self-aware and legally viable. Traditional smart contracts lack flexibility and cannot be altered once they have been created. Such a contract is referred to as a Self-Aware Contract (SAC).

Z. Gao et.al [11] got a system that eliminates the need for users to run each contract before its execution individually. The traditional method of executing a contract inhibits the system's capacity to execute a more significant number of contracts. Their system uses a fair partition algorithm that allows multiple smart contracts to be executed in parallel, thus improving the system's efficiency. This paper aims to decentralize the stock exchange thereby eliminating the need for third parties by using self-enforcing smart contracts. A Decentralized Autonomous Organization(DAO) model is setup using Ethereum. Self-Aware Contracts are used to model the buyer, seller and central exchange.

2.2 INFERENCES FROM LITREATURE SURVEY

2.2.1 Distributed Ledger (Blockchain) Technology

A blockchain is a continuous chain of blocks that holds data in the form of a ledger, which are linked together following certain rules. These blocks contain a hash of the previous block's data, a timestamp, and transaction data. The timestamp proves that the transaction data existed when the block was published. As each block contains information about the previous block, each new additional block added will reinforce the chain, making it harder and harder to change the previous block's data. This is the reason why blockchains are considered tamper proof, as data once recorded cannot be altered without subsequently changing all future blocks. Blockchains are managed by a network of peer-to-peer participants as a publicly distributed ledger, where nodes follow a protocol to communicate and evaluate new blocks. Although blockchain records are immutable, new forks can lead to the creation of new blocks. The adoption of blockchain technology has the potential to significantly improve the quality of accounting information. By increasing the accuracy, transparency, efficiency, security, and auditability, a blockchain can help to ensure that financial information is reliable, trustworthy, and accessible to all stakeholder.

2.2.2 Properties of Distributed Ledger

Programmable is the ability of a system to be controlled or modified with code or software. In the context of a blockchain, smart contracts are an example of programmable elements that can be used to automate complex processes and enforce the terms of an agreement. Smart contracts are self-executing programs that serve as an agreement between the buyer and seller that are directly written into lines of code. The code and the agreements are deployed publicly on the blockchain network. Secure refers to the measures in place to protect against unauthorized access or attacks on a system.

In the context of a blockchain, this can include measures such as cryptography and consensus algorithms, which help ensure the ledger's integrity and security.

Cryptography is used to secure the contents of blocks on the blockchain and to ensure that transactions cannot be altered once they have been added to the blockchain. Consensus algorithms are used to ensure that all parties agree on the state of the ledger and to prevent unauthorized changes from being made to it.

Anonymous is the ability to use a system without revealing one's identity using a self-sovereign identity that can be achieved using pseudonymous addresses that do not reveal the identity of the user. While the transactions that are associated with a particular pseudonymous address can be seen on the blockchain, the identity of the user behind the address is not revealed.

2.2.3 Decentralized Exchanges

Decentralized exchanges or DEXs are exchanges built on the blockchain using smart contracts. It enables the peer-to-peer transfer of digital assets without the use of any intermediary and also offers a slew of other advantages. In DEXs, the users do not handover their funds or assets to any custodians or depositories, as the exchange is taken care of by smart contracts and is non-custodial. These exchanges are either partially or fully governed by a set of contracts.

Smart contracts are automatic contracts stored on blockchains. They are computer programs that run automatically when a certain condition is met or satisfied. They contain a set of instructions, which are usually written in the Solidity programming language if the Ethereum blockchain is being used. Ethereum is a blockchain protocol with smart contract functionality that allows developers to build decentralized applications on top of the blockchain. Smart contracts run on the EVM (Ethereum Virtual Machine). The smart contract system is designed in such a way that it cannot access the network, file system, or any other processes running on the EVM.

DEXs were built as a response to the problems faced by centralized cryptocurrency exchanges, or CEXs, such as susceptibility to external attacks causing the loss of user funds and several other problems faced by centralized exchanges. There are many ways to build a DEX . One way is to use the orderbook format of the CEX.

This is performed by building an on-chain order book where every order is recorded on the blockchain; however, this can be very expensive, as every transaction on the blockchain costs money. There are off-chain order books as well, which record the order off-chain and only use the blockchain as a settlement layer. These are not as expensive as on-chain order books, but they do not offer the same amount of security and decentralization as on-chain order books. Both on-chain and off-chain order books perform regularly in markets with high liquidity, but when the assets being traded are not highly liquid, they are not very useful. To solve this problem, some decentralized exchanges have adopted a new technique called automated market makers, or AMMs.

2.3 OPEN PROBLEMS IN EXISTING SYSTEM

Security Risks: Centralized exchanges are vulnerable to hacking and other security risks.

Lack of Privacy: Centralized exchanges may require users to provide personal information such as identity documents and banking details, which raises privacy concerns.

Centralized Control: Centralized exchanges are controlled by a single entity, which can have an impact on the operations and pricing.

Central Point of Failure: Centralized exchanges can become a single point of failure, causing market-wide disruptions if there are issues with the platform.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDIES/RISK ANALYSIS OF THE PROJECT

Feasibility studies for decentralized automation in stock exchanges involve assessing various key aspects. This includes analyzing the current market landscape of traditional and decentralized exchanges, evaluating the technical feasibility of blockchain technology and smart contracts, and considering legal and regulatory compliance requirements. Financial analysis is conducted to determine costs, potential revenue streams, and return on investment, while risk assessment identifies potential challenges and mitigation strategies. Operational feasibility is evaluated in terms of user experience, scalability, and reliability, and social and environmental impacts are also considered. By conducting thorough feasibility studies, stakeholders can make informed decisions about the viability and potential success of implementing decentralized automation in stock exchanges.

3.1.1 *DEX Front Running Risks*

People are worried about questions like “Can decentralized exchanges be hacked?” to such an extent that they forget crucial concerns. One of the foremost critiques of the existing centralized exchange model refers to the necessity of making payments for order flows. Investors have to pay for order flows in centralized exchanges, which is interestingly a notable mention among DEX risks. Miners or the mining pools could obtain a preview of transactions while confirming and validating them on decentralized exchanges. Front-running risks pose a significant challenge to decentralized automation in stock exchanges, particularly in the context of decentralized exchanges (DEXs). Front-running occurs when a trader exploits non-public information to execute trades ahead of other market participants, thereby profiting at their expense. In the decentralized setting, where transactions are transparent and visible to all participants, front-running risks can manifest through various means, such as manipulating transaction ordering or exploiting network latency. This undermines the fairness and integrity of the exchange and erodes trust among users. To mitigate front-running risks,

decentralized exchanges must implement robust mechanisms, such as randomizing transaction ordering, implementing anti-front-running protocols, and enhancing transaction privacy. Additionally, educating users about front-running risks and promoting transparency in trading practices can help mitigate the impact of front-running on decentralized automation in stock exchanges.

3.1.2 *Limited Recovery Ability of DEX*

The doubts about “are decentralized exchanges safe” could find some promising answers in the fact that DEXs do not have any ability to recover lost, misplaced, or stolen funds. The lack of a KYC process and the ability to cancel transactions generally leads to issues in recovering data or their assets. You could not rely on a support team for notifications about a lost private key or missing funds. Refunds are incompatible with the general smart contract-based model of DEX networks, thereby creating issues in recovery. The limited recovery ability of decentralized exchanges (DEXs) poses a challenge to decentralized automation in stock exchanges. Unlike centralized exchanges, which often have mechanisms in place to recover lost or stolen funds, DEXs operate on immutable blockchain where transactions are irreversible once confirmed. This means that if a user's funds are lost or stolen due to hacking, fraud, or other security breaches, it can be challenging or impossible to recover them. This limitation undermines user confidence and trust in DEXs and may deter investors from participating in decentralized trading. To address this challenge, DEX developers and users must prioritize security measures, such as robust authentication protocols, secure wallet management, and decentralized insurance solutions. Additionally, regulatory frameworks and industry standards can be established to promote accountability and enhance investor protection in the decentralized trading ecosystem. Despite these challenges, the inherent advantages of decentralization, such as transparency, censorship resistance, and user control over funds, continue to drive the adoption of DEXs, albeit with ongoing efforts to improve security and risk management practices.

3.1.3 Limited Support for DEX Trading

Another formidable setback associated with decentralized exchanges refers to the limited functionality for trading. How? The DEX risks of reduced transaction settlement speed could create obstacles for participating in trading on DEXs. Furthermore, decentralized exchanges have become popular only in recent times with a focus on simple purchasing and selling orders. Limited support for decentralized exchange (DEX) trading is a significant challenge facing decentralized automation in stock exchanges. Unlike centralized exchanges that often have extensive support for various trading features, liquidity pools, and trading pairs, DEXs may have limited liquidity and fewer trading options. This limitation arises due to factors such as lower trading volumes, fewer users, and the fragmentation of liquidity across different DEX platforms. As a result, traders may face difficulties executing large orders, finding suitable trading pairs, or obtaining competitive prices on DEXs compared to centralized exchanges. To address this challenge, efforts are underway to enhance liquidity aggregation, interoperability between DEXs, and the development of decentralized liquidity protocols. Additionally, initiatives to incentivize liquidity providers and improve user interfaces and trading experiences on DEX platforms can help attract more traders and increase support for DEX trading. Despite these challenges, the growing popularity of decentralized finance (DeFi) and the adoption of blockchain technology are driving innovation and growth in the DEX ecosystem, paving the way for broader support and adoption in the future.

3.2 SOFTWARE REQUIREMENTS SPECIFICATION DOCUMENT

3.2.1 NodeJS



Fig 3.1: Node javascript logo

Node.js is a cross-platform, open-source server environment that can run on Windows, Linux, Unix, macOS, and more. Node.js is a back-end JavaScript runtime environment, runs on the V8 JavaScript engine, and executes JavaScript code outside a web browser. Node.js lets developers use JavaScript to write command line tools and for server-side scripting. The ability to run JavaScript code on the server is often used to generate dynamic web page content before the page is sent to the user's web browser. Consequently, Node.js represents a "JavaScript everywhere" paradigm,[6] unifying web-application development around a single programming language, as opposed to using different languages for the server- versus client-side programming. Node.js has an event-driven architecture capable of asynchronous I/O. These design choices aim to optimize throughput and scalability in web applications with many input/output operations, as well as for real-time Web applications (e.g., real-time communication programs and browser games). The Node.js distributed development project was previously governed by the Node.js Foundation, and has now merged with the JS Foundation to form the OpenJS Foundation. OpenJS Foundation is facilitated by the Linux Foundation's Collaborative Projects program.

Asynchronous and Event-Driven: Node.js uses a non-blocking, event-driven architecture that allows it to handle a large number of concurrent connections efficiently. This makes it well-suited for building real-time applications such as chat applications, gaming servers, and streaming services.

Single-threaded and Non-blocking I/O: Node.js uses a single-threaded event loop model to handle asynchronous I/O operations. This enables it to handle multiple requests simultaneously without blocking the execution of other code, leading to better performance and resource utilization.

NPM (Node Package Manager): Node.js comes with a powerful package manager called NPM, which allows developers to easily install, manage, and share reusable code packages (modules) from the NPM registry. This vast ecosystem of modules provides solutions for a wide range of functionalities, making development faster and more efficient.

Cross-platform Compatibility: Node.js is cross-platform, meaning it can run on various operating systems such as Windows, macOS, and Linux. This allows developers to write code once and deploy it on multiple platforms without significant modifications, increasing productivity and reducing development time.

Scalability: Node.js is highly scalable and can handle large-scale applications with ease. Its lightweight and efficient architecture make it suitable for building micro services architectures and distributed systems that can scale horizontally to meet growing demands.

Community Support: Node.js has a vibrant and active community of developers who contribute to its ongoing development, share knowledge, and provide support through forums, tutorials, and online resources. This community-driven ecosystem fosters innovation and collaboration, driving the continuous improvement of Node.js and its associated tools and frameworks.

Server-side Rendering: Node.js enables server-side rendering of JavaScript applications, allowing developers to build isomorphic applications that can render both on the client and server-side. This improves performance, SEO, and user experience by delivering pre-rendered content to users quickly.

3.2.2 Visual Studio Code



Fig 3.2: Visual Studio Code IDE logo

Visual Studio Code, also commonly referred to as VS Code, is a source-code editor made by Microsoft with the Electron Framework, for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. Users can change the

theme, keyboard shortcuts, preferences, and install extensions that add functionality. In the Stack Overflow 2023 Developer Survey, Visual Studio Code was ranked the most popular developer environment tool among 86,544 respondents, with 73.71% reporting that they use it.

IntelliSense: IntelliSense is a feature that provides context-aware code completion suggestions, parameter hints, and code navigation capabilities as you type, helping to improve productivity and reduce errors.

Syntax Highlighting: VS Code offers syntax highlighting for a wide range of programming languages, making code easier to read and understand by highlighting different elements such as keywords, variables, and comments with distinct colors.

Code Snippets: Code snippets are predefined templates or shortcuts for common code patterns, allowing developers to quickly insert code blocks and avoid repetitive typing. VS Code supports both built-in and user-defined code snippets.

Integrated Terminal: VS Code includes an integrated terminal that allows developers to run command-line tools, compile code, execute scripts, and perform other tasks without leaving the editor. The terminal can be customized and configured to use different shells and command interpreters.

Version Control Integration: VS Code seamlessly integrates with version control systems such as Git, providing features like source control management, diff viewing, commit history visualization, and conflict resolution directly within the editor.

Debugging Support: Visual Studio Code offers built-in debugging capabilities for various programming languages and platforms. Developers can set breakpoints, inspect variables, step through code execution, and diagnose issues using the integrated debugger.

Extensions Marketplace: VS Code has a rich ecosystem of extensions available through its marketplace, offering additional features, language support, themes, and tools to customize and extend the editor according to specific development needs.

Task Automation: VS Code supports task automation through its built-in task runner system and integration with build tools such as Grunt, Gulp, and npm scripts. Developers can define custom tasks to automate common development tasks like compiling code, running tests, and deploying applications.

Integrated Git UI: In addition to command-line Git integration, VS Code includes a user-friendly Git interface with features like staged changes, commit history, branch management, and conflict resolution, making it easier to work with version-controlled projects.

Extension Development: Visual Studio Code itself is highly extensible, allowing developers to create custom extensions and plugins to add new functionality or integrate with third-party services. The VS Code extension API provides access to various editor features and services for extension development.

Customization and Theming: VS Code offers extensive customization options, including themes, key bindings, editor settings, and UI layouts. Users can personalize their development environment to suit their preferences and workflow preferences.

3.3 Ethereum Blockchain Network



Fig 3.3: Ethereum Block Chain Network

Ethereum is a decentralized blockchain platform that establishes a peer-to-peer network that securely executes and verifies application code, called smart contracts. Smart contracts allow participants to transact with each other without a trusted central authority. Transaction records are immutable, verifiable, and securely distributed across the network, giving participants full ownership and visibility into transaction data. Transactions are sent from and received by user-created Ethereum accounts. A sender must sign transactions and spend Ether, Ethereum's native cryptocurrency, as a cost of processing transactions on the network.

Smart Contracts: Ethereum introduced the concept of smart contracts, which are self-executing contracts with the terms of the agreement directly written into code. Smart contracts automatically enforce the rules and conditions of the agreement, eliminating the need for intermediaries and enabling trustless transactions.

Decentralization: Ethereum is decentralized, meaning it operates on a network of computers (nodes) distributed worldwide, with no single point of control. This decentralization ensures censorship resistance, fault tolerance, and resilience against attacks or failures.

Ether (ETH) Cryptocurrency: Ether (ETH) is the native cryptocurrency of the Ethereum platform. It is used as a medium of exchange for transactions on the network, as well as a reward for miners who validate and secure blocks of transactions through a process called mining.

Consensus Mechanism: Ethereum currently uses a Proof of Work (PoW) consensus mechanism, similar to Bitcoin, where miners compete to solve complex mathematical puzzles to validate transactions and add them to the blockchain. Ethereum is transitioning to a Proof of Stake (PoS) consensus mechanism with the Ethereum 2.0 upgrade, which aims to improve scalability, energy efficiency, and security.

EVM (Ethereum Virtual Machine): The Ethereum Virtual Machine is a runtime environment that executes smart contracts on the Ethereum network. It allows developers to write smart contract code in high-level languages like Solidity and compile it into bytecode that can be executed by the EVM.

Interoperability and Standards: Ethereum supports interoperability with other blockchains and platforms through standards such as ERC (Ethereum Request for Comments) token standards. ERC-20, ERC-721, and ERC-1155 are examples of token standards that define common rules and interfaces for creating fungible tokens, non-fungible tokens (NFTs), and multi-token contracts, respectively.

Decentralized Applications (DApps): Ethereum enables the development and deployment of decentralized applications (DApps) that run on the blockchain. DApps leverage smart contracts to automate processes, manage digital assets, and facilitate peer-to-peer interactions without relying on centralized servers or intermediaries.

Immutable Ledger: The Ethereum blockchain maintains an immutable ledger of all transactions and smart contract interactions since its inception. Once data is recorded on the blockchain, it cannot be altered or deleted, providing a transparent and tamper-proof record of historical transactions.

Scalability Solutions: Ethereum has faced challenges with scalability due to limitations in transaction throughput and network congestion. Various scalability solutions are being developed and implemented, including layer 2 scaling solutions like Ethereum's Optimistic Rollups and Plasma, as well as sharding and off-chain scaling solutions.

Community and Ecosystem: Ethereum has a vibrant and diverse community of developers, entrepreneurs, researchers, and enthusiasts who contribute to its development and adoption. The Ethereum ecosystem encompasses a wide range of projects, tools, libraries, and services built on top of the Ethereum platform, driving innovation and growth in the blockchain space.

CHAPTER 4

DESCRIPTION OF PROPOSED SYSTEM

Automated market makers are nothing but a collection of smart contracts with instructions to execute trades. At the time of writing this paper, the most mature blockchain in terms of developers, users, and miners is Ethereum. The model proposed in this paper uses Ethereum as the reference blockchain. An AMM-based exchange consists mainly of two components. The first component is the asset that has to be traded. A decentralized stock exchange trades securities against a reserve currency. In Ethereum, the token standard that can be used to represent a reserve currency is ERC-20. The ERC-20 standard covers all the criteria needed to represent a currency. An alternate Ethereum token standard is needed to represent the security or stock token. These tokens require additional functionalities to better represent the underlying asset and to enforce regulations on the asset and the trader or the issuer of that asset. One of the standards that can be used to represent securities in Ethereum is ERC-1400. The second component is the AMM. The AMM that is used for such an application must either be for a general purpose or must be specific to stocks. Building an AMM specifically to trade stocks is out of the scope of this paper, as it would require more research into the mathematics involved in the game theory of financial markets. In the current work, an existing AMM formula has been used, as it has been tried and tested in the cryptocurrency markets. The AMM used for a decentralized stock exchange should be a general-purpose digital asset exchange tool because a stock or securities represent a fractional ownership or stake in a physical or digital entity. This ownership or stake can have different functionalities based on how it is implemented and what kind of entity will be using it. The stocks are bought or sold on exchanges using currencies that have their own characteristics. This AMM needs to have the capability to swap or exchange assets with fundamentally different natures.

4.1 SELECTED METHODOLOGY OR PROCESS MODEL

Operations of the proposed model is a six layer blockchain architecture. Each node is registered as either a buyer or seller in the network. On the buyer side, the cost of the shares bought should not exceed the wallet balance. On the seller side, the number of shares being sold should not exceed the number of shares being owned. All orders are verified and stored in a public distributed ledger (data layer). All the nodes in the network are interconnected. There is no central authority monitoring the transactions as in case of the P2P network(network layer). Each transaction is stored at each node's ledger. Ethereum handles broadcast and consensus of transactions at all nodes (consensus layer). The miners are paid a transaction fee based on the amount of gas consumed for executing the trade. This transaction fee is negligible (incentive layer). The exchange contract matches orders autonomously. The buyer and seller contract ensures the exchange of funds and stocks without human intervention (contract layer). Therefore the proposed model fits the architecture specification and can be viewed as a decentralized, selfmanaged DAO.

4.2 ARCHITECTURE / OVERALL DESIGN OF PROPOSED SYSTEM

A proposed system for decentralized automation in stock exchanges could consist of the following components:

Decentralized Exchange Platform: The core of the system would be a decentralized exchange platform built on blockchain technology. This platform would facilitate the peer-to-peer trading of securities, tokens, and other assets without the need for intermediaries. It would leverage smart contracts to automate trade execution, settlement, and asset management.

Blockchain Infrastructure: The system would rely on a secure and scalable blockchain infrastructure to record transactions, maintain a transparent ledger, and enforce smart contract logic. This infrastructure could be built on existing blockchain

platforms such as Ethereum, Binance Smart Chain, or developed as a custom blockchain solution tailored to the specific requirements of the decentralized exchange.

User Wallets: Users would interact with the decentralized exchange platform through digital wallets that support the storage and management of various assets. These wallets would provide secure access to funds, facilitate asset transfers, and enable users to participate in trading activities on the exchange.

Smart Contracts: Smart contracts would play a crucial role in automating various aspects of trading on the decentralized exchange platform. These programmable contracts would be responsible for executing trades, managing order books, settling transactions, and enforcing compliance with predefined rules and regulations.

Order Matching Engine: The system would include an order matching engine responsible for matching buy and sell orders submitted by users on the exchange. This engine would operate based on predefined algorithms and rules to ensure fair and efficient order matching and execution.

Liquidity Provision Mechanism: To address liquidity challenges commonly faced by decentralized exchanges, the system could incorporate mechanisms for incentivizing liquidity providers. This could involve liquidity mining programs, yield farming opportunities, or other incentive structures designed to attract liquidity to the platform.

Regulatory Compliance Tools: Given the regulatory complexities surrounding securities trading, the system would include tools and features to ensure compliance with relevant regulations and legal requirements. This could include KYC/AML verification processes, regulatory reporting functionalities, and integration with compliance frameworks.

User Interface: The system would feature an intuitive and user-friendly interface that allows users to easily navigate the exchange, place orders, monitor their portfolio, and access relevant information and analytics. The user interface would be accessible through web browsers, mobile applications, or other client interfaces.

Security Measures: Security would be paramount in the decentralized exchange system, given the potential risks associated with handling digital assets and conducting financial transactions. The system would implement robust security measures, such as encryption, multi-factor authentication, audit trails, and cold storage solutions, to protect user funds and data from unauthorized access and cyber threats.

Overall, the proposed system for decentralized automation in stock exchanges would provide a secure, efficient, and transparent platform for trading securities and other assets in a decentralized manner, empowering users with greater control over their investments and financial transactions.

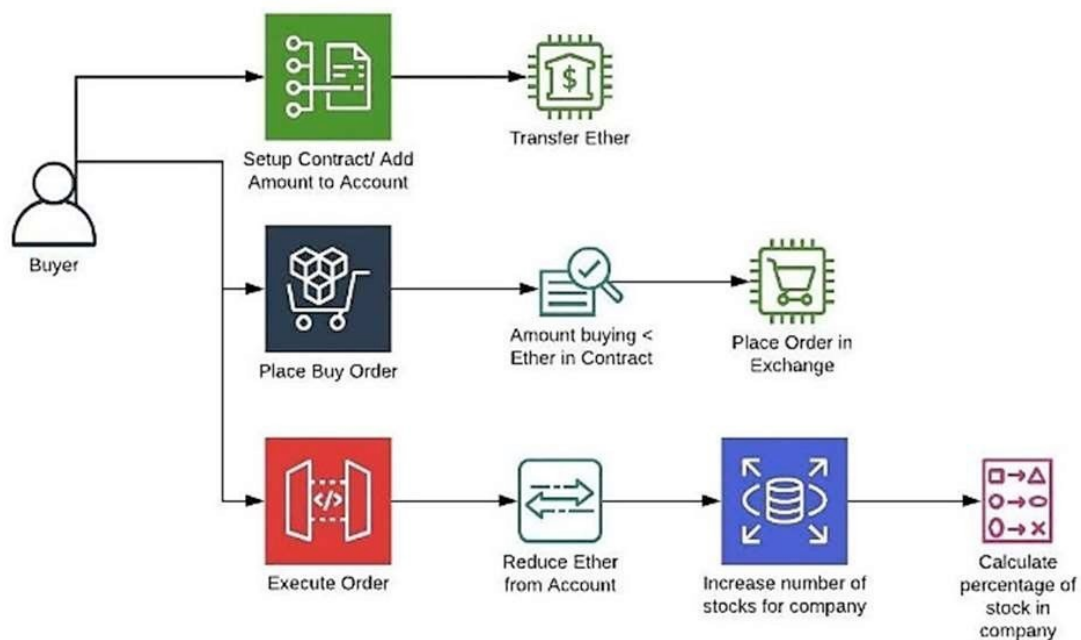


Fig 4.1: Architecture for Buyer Working

Fig 4.1. represents the functionalities of the buyer contract. The buyer is required to set up an account with an initial balance of ether. The buyer can only place an order if he has enough funds in his account for that particular transaction; this ensures that

fraudulent activities do not take place. Within the scope of this balance, the buyer is now allowed to buy/sell shares. This mechanism also eliminates the need for an escrow, since the exchange has locked the currency required for the transaction.

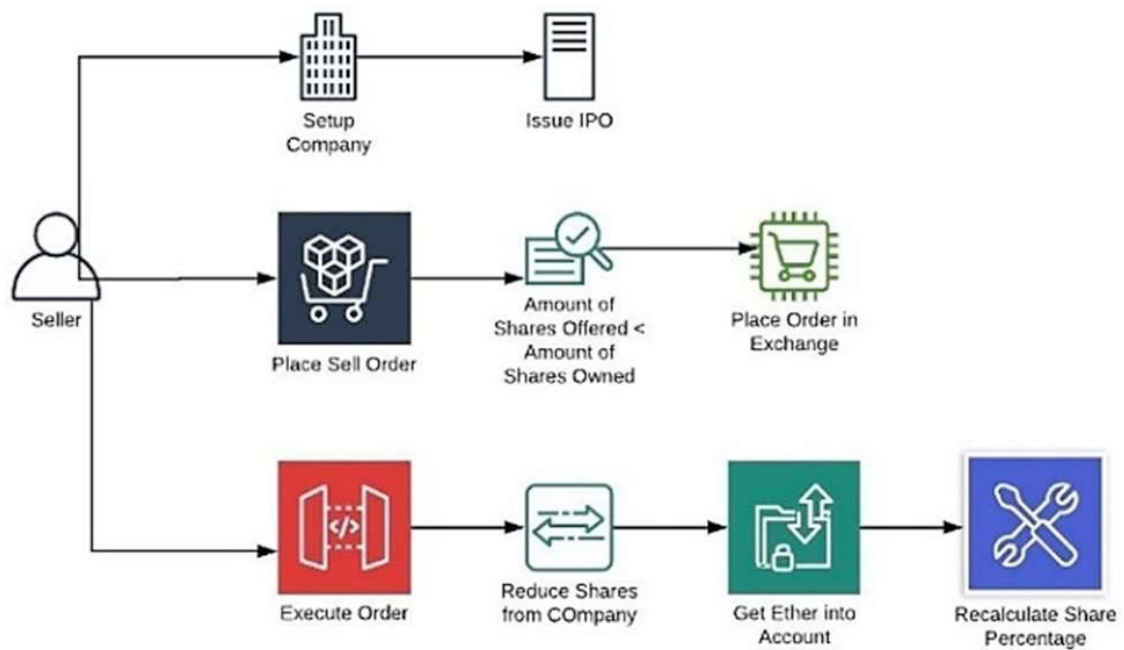


Fig 4.2: Architecture for Company Working

Fig 4.2 represents the functionalities of the company contract. To register itself on a stock exchange, a company should launch an Initial Public Offering (IPO), which denotes the number of stocks and the price to the public for sale.

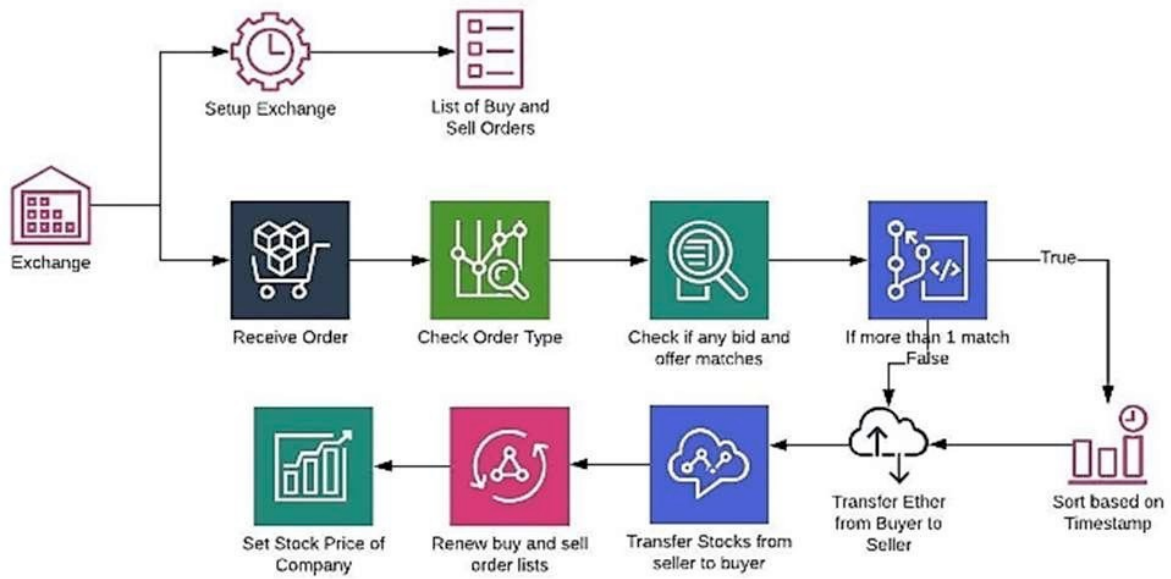


Fig 4.3: Overall System Architecture

Fig 4.3. represents the functionalities of the exchange contract. This contract records every transaction that takes place and thus acts as an order book for the exchange. It uses a priority queue dependent on the timestamp of the order to perform matching. i.e. partial or complete. After a successful transaction, if there are no more available shares for a particular company, that company is removed from the exchange. If during a transaction, the number of shares to be bought is more than the available stocks, then a partial match is made, and the order book is updated. Ether is transferred from the wallet of the buyer to the seller, followed by which the price of the stock is updated to the price of the last transaction executed.

4.3 DESCRIPTION OF SOFTWARE FOR IMPLEMENTATION AND TESTING PLAN OF THE PROPOSED MODEL/SYSTEM

To implement the system, a private Ethereum network is created on Ethereum core using Geth. Miner instances were set up on a machine having I7 8th Gen processor, 16GB of RAM and Nvidia GeForce GTX 1060 Graphics Processor. Smart contracts were created using Solidity and compiled using the online compiler Remix. The compiled code is deployed in the private network. Every node deployed in the network can represent either a buyer or a seller. The exchange contract is deployed on the system, there is only one exchange contract for the entire network. It keeps track of the order books. For a company to be created, a node should deploy the seller contract, and an IPO should be launched on the exchange. This registers the company on the exchange contract. The node that deploys the company contract is considered to be the majority shareholder/owner of the company. There is one seller contract per company. A buyer should refer to this contract to buy/sell shares in a company. For any buyer to purchase shares in a company, the buyer contract should be deployed. There will be a buyer contract for each buyer node. Buyer and seller contracts communicate via the exchange contract. Data from all three contracts are replicated at each node. But as data is stored as three separate contracts, data cannot be accessed without the key to the contract. This ensures security. But as data is stored in all nodes, it induces replicability and fault tolerance. To test the scalability of transactions, a private network is set up on kaleido.io, an online blockchain platform. Using the API endpoints of the smart contract deployments, transactions are executed. Daily trading data for Amazon from LOBSTER was used. LOBSTER is an open source for order books and transaction data from NASDAQ.

Blockchain Development Platforms: Platforms such as Ethereum, Binance Smart Chain, or Hyperledger Fabric would be used to develop the underlying blockchain infrastructure for the decentralized exchange. These platforms provide tools, libraries, and frameworks for building and deploying smart contracts, managing transactions, and interacting with the blockchain network.

Smart Contract Development Tools: Software tools like Solidity (for Ethereum), Vyper, or Truffle Suite would be used for developing smart contracts that power various functionalities of the decentralized exchange, such as trade execution, order matching, and asset management. These tools provide integrated development environments (IDEs), testing frameworks, and debugging tools for efficient smart contract development and testing.

Wallet Development Kits (Wallet SDKs): Wallet development kits, such as Web3.js (for Ethereum), Binance Chain SDK, or WalletConnect, would be utilized to integrate digital wallet functionalities into the decentralized exchange platform. These SDKs enable interactions with user wallets, such as account creation, fund transfers, and transaction signing, enhancing the user experience and security of the exchange.

Testing Frameworks: Testing frameworks like Truffle Test, Ganache, or Hardhat would be employed for writing and executing automated tests for smart contracts and other components of the decentralized exchange system. These frameworks allow developers to simulate various scenarios, validate contract behavior, and ensure the correctness and reliability of the system.

Security Tools: Security tools and libraries, such as OpenZeppelin for smart contract security, Ethlint for code linting, and MythX for smart contract analysis, would be utilized to enhance the security posture of the decentralized exchange platform. These tools help identify vulnerabilities, enforce best practices, and mitigate potential security risks in the codebase.

Monitoring and Analytics Tools: Monitoring and analytics tools like Etherscan (for Ethereum), BscScan (for Binance Smart Chain), or custom monitoring solutions would be employed to monitor the performance, health, and activity of the decentralized exchange network. These tools provide insights into transaction activity, network congestion, and contract interactions, enabling developers to optimize system performance and scalability.

User Interface Frameworks: User interface frameworks such as React.js, Angular, or Vue.js would be used to develop the frontend interface of the decentralized exchange platform. These frameworks provide tools and libraries for building responsive, interactive, and visually appealing user interfaces that enhance the user experience and engagement.

System Architecture Design: Define the overall architecture of the decentralized exchange system, including components, interactions, and data flows. Design the blockchain infrastructure, smart contracts, user interface, and integration points with external systems.

Smart Contract Development: Develop and deploy smart contracts for essential functionalities such as token issuance, order management, trade execution, and asset settlement. Implement security measures, compliance features, and error handling mechanisms in smart contracts.

Wallet Integration: Integrate digital wallet functionalities into the decentralized exchange platform using wallet development kits (SDKs). Implement features for wallet creation, account management, fund transfers, and transaction signing.

User Interface Development: Develop the frontend interface of the decentralized exchange platform using user interface frameworks. Design and implement features for account registration, asset management, order placement, and trade monitoring.

Backend Development: Develop the backend components of the decentralized exchange platform, including order matching engine, liquidity provision mechanisms, and regulatory compliance tools. Implement APIs for data exchange, transaction processing, and system integration.

Testing and Quality Assurance: Write and execute automated tests for smart contracts, frontend components, and backend functionalities using testing frameworks. Perform integration testing, regression testing, and security testing to ensure the reliability, correctness, and security of the system.

Deployment and Launch: Deploy the decentralized exchange platform to a test network or staging environment for final testing and validation. Conduct user acceptance testing (UAT) and performance testing to evaluate system readiness. Deploy the platform to the production environment and launch the decentralized exchange for public use.

Monitoring and Maintenance: Implement monitoring and analytics tools to monitor system performance, health, and activity in real-time. Establish processes for ongoing maintenance, updates, and security patches. Provide user support and address any issues or concerns raised by users.

Community Engagement and Growth: Engage with the community of users, developers, and stakeholders to gather feedback, address concerns, and drive adoption of the decentralized exchange platform. Continuously iterate on the platform based on user feedback and market trends to enhance usability, functionality, and scalability.

Compliance and Regulation: Ensure compliance with relevant regulatory requirements and legal frameworks governing securities trading and blockchain technology. Implement KYC/AML verification processes, regulatory reporting functionalities, and adherence to industry standards to maintain regulatory compliance and user trust.

4.4 PROJECT MANAGEMENT PLAN

4.4.1 Transaction Costs

Three different contracts, one for each of the following: buyer, seller and central exchange, is deployed. Buyer and seller accounts can be linked to Ethereum wallets of the account holder. Whenever an order has successfully matched a transfer of funds from the buyer to the seller wallet takes place through these accounts. The costs are measured in gas units. Gas is used to calculate the fees that need to be paid to the

miners to execute an operation. The decrease in gas on the seller side is because the array of orders is iterated only particular transaction. When a higher mining fee is paid, transactions get mined faster. Two different instances of the prototype are mined at 5 Gwei and 8 Gwei. The prototype proposed has an average of \$0.042228 for the seller and \$0.14656 for the buyer at 5 Gwei and \$0.084455 for the seller and \$0.2345 for the buyer at 8 Gwei. The parallelized execution results in costs being shared by both buyer and seller. The average principal value traded in the deployment of 635 transactions was \$76858. The brokerage fee for Merrill Edge for this principal value is calculated to be \$1134.83. For the same 635 transactions on the proposed system, the average gas consumed was 69564. Assuming mining takes place at 13 Gwei, \$0.19627 is to be paid as mining costs. This is a 99.98% decrease in fees for trading.

4.4.2 Transaction Time

The transaction time for mining each of the 635 transactions has been plotted. The transaction time is a maximum of eight seconds. It can be observed that the majority of the transactions take just four seconds, which is the duration for placing orders in the order book. Transactions which take longer to execute represent the transfer of ether from buyer to seller, when an order has been matched. Complete match of orders takes lesser time to execute than partial matches. The transaction time for matching of orders will reduce if the miners are paid more, or the number of miners are increased.

CHAPTER-5

IMPLEMENTATION DETAILS

5.1 DEVELOPMENT AND DEPLOYMENT SETUP

The development and deployment setup for a decentralized automation stock exchange involves several steps to ensure a smooth and efficient process. Here's a proposed setup:

Development Environment Setup:

- Choose a suitable development environment for blockchain development, such as Truffle Suite, Remix IDE, or Visual Studio Code with appropriate extensions.
- Install necessary development tools and frameworks, including Node.js, Git, and package managers like npm or yarn.
- Set up a local blockchain network for testing and development purposes using tools like Ganache or Hardhat.
- Configure the development environment to support smart contract development, testing, and deployment.

Smart Contract Development:

- Develop smart contracts for the decentralized exchange platform, including contracts for token issuance, order management, trade execution, and asset settlement.
- Write automated tests for smart contracts using testing frameworks like Truffle Test or Hardhat.
- Conduct thorough testing of smart contracts to ensure correctness, security, and reliability.

Backend Development:

- Develop backend components of the decentralized exchange platform, such as order matching engine, liquidity provision mechanisms, and regulatory compliance tools.
- Implement APIs for data exchange, transaction processing, and integration with external systems.
- Ensure scalability, performance, and security of backend systems through rigorous testing and optimization.

Frontend Development:

- Develop the frontend interface of the decentralized exchange platform using user interface frameworks like React.js, Angular, or Vue.js.
- Design and implement features for account registration, asset management, order placement, and trade monitoring.
- Ensure responsiveness, accessibility, and usability of the frontend interface across different devices and browsers.

Integration Testing:

- Integrate frontend and backend components of the decentralized exchange platform.
- Conduct integration testing to validate the interaction between different system components and ensure seamless functionality.
- Identify and resolve any issues or inconsistencies in the integrated system.

Deployment Setup:

- Choose a suitable blockchain network for deployment, such as Ethereum mainnet, testnets (Ropsten, Rinkeby, Kovan), or a private blockchain network.
- Set up deployment configurations and scripts for deploying smart contracts and backend services to the chosen blockchain network.

- Configure deployment parameters, such as gas limits, transaction fees, and contract addresses, based on network requirements and constraints.

Deployment Process:

- Deploy smart contracts to the blockchain network using deployment scripts or tools like Truffle or Remix IDE.
- Deploy backend services to the appropriate hosting environment, such as cloud platforms (AWS, Azure, GCP) or dedicated servers.
- Configure DNS settings, SSL certificates, and other security measures for the deployed services.

Testing and Validation:

- Conduct comprehensive testing of the deployed system to validate functionality, performance, and security.
- Perform user acceptance testing (UAT) to ensure that the decentralized exchange platform meets user requirements and expectations.
- Address any issues or bugs identified during testing and make necessary adjustments to the system.

Launch and Maintenance:

- Once testing is complete and the system is validated, launch the decentralized exchange platform for public use.
- Monitor system performance, security, and user feedback after the launch.
- Provide ongoing maintenance, updates, and support to address any issues, implement new features, and ensure the continued operation and success of the decentralized exchange platform.
- By following this development and deployment setup, you can effectively build, deploy, and maintain a decentralized automation stock exchange that provides a secure, efficient, and transparent platform for trading securities and other assets in a decentralized manner

5.2 TESTING

Testing for a decentralized automation stock exchange is crucial to ensure the reliability, security, and functionality of the platform. Here's a comprehensive testing approach:

Unit Testing:

- Test individual components such as smart contracts, backend services, and frontend modules in isolation.
- Use testing frameworks like Truffle, Hardhat, or Jest to write automated unit tests for smart contracts and backend code.
- Ensure that each unit of code behaves as expected and handles edge cases appropriately.

Integration Testing:

- Test the interaction between different components of the decentralized exchange platform, including smart contracts, backend services, and frontend interfaces.
- Conduct integration tests to validate the flow of data and transactions between components and ensure seamless functionality.

End-to-End Testing:

- Perform end-to-end testing to validate the entire workflow of the decentralized exchange platform, from user registration to trade execution and settlement.
- Use testing frameworks like Selenium or Cypress to simulate user interactions with the frontend interface and verify that the platform behaves as expected.

Security Testing:

- Conduct security audits and penetration testing to identify vulnerabilities and weaknesses in the platform.
- Perform code reviews, static code analysis, and dynamic security testing to identify and mitigate potential security risks.
- Test for common security vulnerabilities such as reentrancy attacks, integer overflows, and unauthorized access to sensitive data.

Performance Testing:

- Test the performance and scalability of the decentralized exchange platform under different load conditions.
- Use load testing tools like Apache JMeter or Gatling to simulate a large number of concurrent users and transactions and measure system performance metrics such as response time, throughput, and resource utilization.
- Identify and optimize performance bottlenecks to ensure that the platform can handle the expected volume of transactions efficiently.

User Acceptance Testing (UAT):

- Involve end-users, stakeholders, and domain experts in user acceptance testing to validate that the platform meets their requirements and expectations.
- Conduct usability testing, accessibility testing, and user feedback sessions to gather insights into user experience and identify areas for improvement.
- Address any issues or concerns raised during UAT and make necessary adjustments to the platform based on user feedback.

Regulatory Compliance Testing:

- Ensure that the decentralized exchange platform complies with relevant regulatory requirements and legal frameworks governing securities trading and blockchain technology.

CHAPTER - 6

OUTCOMES AND DISCUSSION

6.1 INTRODUCTION

The development of a decentralized stock exchange project utilizing MacBook, Node.js, TestRPC, and Ethereum presents a pioneering approach to revolutionize traditional financial systems. This project aims to address inherent challenges in centralized stock exchanges, such as massive transaction charges, susceptibility to hacking, and a transparency issue. By leveraging the power of blockchain technology through Ethereum, the system provides a secure and transparent platform for users to engage in buying, selling, and participating in Initial Public Offerings (IPOs). The use of Node.js enhances the efficiency and responsiveness of the platform, ensuring a seamless user experience. TestRPC serves as a local Ethereum blockchain for testing, streamlining the development and testing process. This decentralized stock exchange not only eliminates the need for intermediaries but also introduces a democratic and reliable decision-making process through consensus algorithms. The implementation on MacBook showcases the versatility of the system across different environments, making it accessible to a broader user base. Overall, this project represents a significant step towards creating a decentralized financial ecosystem that empowers users while ensuring the security of their assets and integrity of their transactions. The output represents how the platform works by having a easy to understand user interface. Users can engage with the buy or sell buttons for transaction orders as well as IPO function for buying newly listed company shares. A wallet balance demonstrator field helps user with keeping up with the gas they have for trading.

6.2 IEEE STANDARDS FOLLOWED IN THE PROJECT

In the development of a decentralized stock exchange project, adherence to relevant IEEE standards is crucial to ensure robustness, interoperability, and security. Several IEEE standards are applicable to various aspects of such a project:

1. **IEEE 802.11**: This standard addresses wireless LAN technology, which could be relevant for communication between nodes in the decentralized network, ensuring reliable and efficient data transmission.
2. **IEEE 802.1X**: This standard provides port-based network access control, ensuring that only authorized entities can access the network, thereby enhancing security in the decentralized exchange ecosystem.
3. **IEEE 802.3**: This standard pertains to Ethernet networks, providing guidelines for wired network communications which could be integral in the backend infrastructure of the decentralized exchange.
4. **IEEE 802.15.4**: This standard focuses on low-rate wireless personal area networks (LR-WPANs), which could be useful for communication between devices in the exchange, particularly in the context of Internet of Things (IoT) devices used for monitoring and data collection.
5. **IEEE 802.16**: This standard addresses broadband wireless access, which could be relevant for ensuring high-speed internet connectivity for the decentralized exchange platform, facilitating real-time trading and data transmission.
6. **IEEE 2410**: This standard specifically focuses on blockchain and distributed ledger technologies, providing guidelines for the implementation and interoperability of blockchain systems, which are fundamental to the operation of decentralized exchanges.

CHAPTER – 7

RESULT AND DISCUSSION

6.1 CONCLUSION

In conclusion, the development and implementation of our decentralized stock exchange represent a significant leap forward in stock market transparency. Through cutting-edge technologies such as Solidity, Ethereum, and NodeJS, we have successfully crafted a versatile and user-friendly tool that demonstrates the potential of decentralized exchanges (DEX) within the stock market platform. This project not only showcases the power of modern technology but also paves the way for a future where seamless transparency between the public and stock exchanges becomes the norm. Moving forward, the continuous refinement and expansion of this project promise to contribute to the ongoing evolution of blockchain technologies. By focusing on reducing transaction costs and leveraging blockchain's decentralized ledger system, we facilitate direct transactions without relying on banks or payment processors, thus eliminating associated charges. This not only enhances efficiency but also fosters greater accessibility for a wider range of investors. Our decentralized stock exchange embodies the principles of decentralization, transparency, and inclusivity, reshaping the traditional stock market landscape for the better. As we further refine and innovate upon this project, we remain committed to advancing the boundaries of blockchain technology and ushering in a new era of financial democratization and empowerment.

REFERENCES:-

- [1] A. Dixit and A. Norta, "A Self-Aware Contract for Decentralized Peer-to-Peer (P2P) Commerce", 2018 IEEE 3rd International Workshops on Foundations and Applications of Self* Systems (FAS*W), Trento, pp. 17-19, Sept. 2018
- [2] A. Kaushik, A. Choudhary, C. Ektare, D. Thomas, and S. Akram, "Blockchain Literature survey," in Proc. 2nd IEEE Int. Conf. Recent Trends Electron., Inf. Commun. Technol. (RTEICT), May 2017, pp. 21452148.
- [3] C. Pop et al., "Decentralizing the Stock Exchange using Blockchain An Ethereum-based implementation of the Bucharest Stock Exchange", 2018 IEEE 14th International Conference on Intelligent Computer Communication and Processing (ICCP), Cluj-Napoca, pp. 459-466, Sept. 2018
- [4] J. Jaoude and R. Saade, "Blockchain applications Usage in different domains," IEEE Access, vol. 7, pp. 45372 – 45373, 2019, doi: 10.1109/ACCESS.2019.2902501.
- [5] M. Conoscenti, A. Vetro, and J. C. De Martin, "Blockchain for the Internet of Things: A systematic literature review," in Proc. IEEE/ACS 13th Int. Conf. Comput. Syst. Appl. (AICCSA), Nov. 2016, pp. 16.
- [6] M. Demir, M. Alal, O. Turetken, and A. Ferworn, "Security smells in smart contracts," in Proc. IEEE 19th Int. Conf. Softw. Qual., Rel. Secur. Companion (QRS-C), Jul. 2019, pp. 442449.
- [7] M. Samaniego, U. Jamsrandorj, and R. Deters, "Blockchain as a service for IoT," in Proc. IEEE Int. Conf. Internet Things (iThings) IEEE Green Comput. Commun. (GreenCom) IEEE Cyber, Phys. Social Comput. (CPSCom) IEEE Smart Data (SmartData), Dec. 2016, pp. 433 436.
- [8] S. Rouhani and R. Deters, "Security, performance, and applications of smart contracts: A systematic survey," IEEE Access, vol. 7, pp. 5075950779, 2019.

- [9] S.Wang, L. Ouyang,Y.Yuan, X. Ni, X. Han, and F.-Y.Wang, ``Blockchain enabled smart contracts: Architecture, applications, and future trends," IEEE Trans. Syst., Man, Cybern. Syst., vol. 49, no. 11, pp. 2266 2277,Nov. 2019.
- [10] T. Ahram, A. Sargolzaei, S. Sargolzaei, J. Daniels, and B. Amaba,"Blockchain technology innovations," in Proc. IEEE Technol. Eng. Manage. Conf. (TEMSCON), Jun. 2017, pp. 137 141.
- [11] T. T. A. Dinh, R. Liu, M. Zhang, G. Chen, B. C. Ooi, and J. Wang,"Untangling blockchain: A data processing view of blockchain systems," IEEE Trans. Knowl. Data Eng., vol.30, no. 7, pp. 13661385,Jul. 2018.
- [12] Y.-H. Chen, S.-H. Chen, and I.-C. Lin, ``Blockchain based smart contract for bidding system," in Proc. IEEE Int. Conf. Appl. Syst. Invention (ICASI), Apr. 2018, pp. 208 211.

APPENDIX

A. SOURCE CODE

HTML CODE:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/logo.png" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#FFFFFF" />
    <meta
      name="description"
      content="DSE - Decentralized Stock Exchange. DApp by Adarsh Shrivastava"
    />
    <meta
      name="og:image"
      content="%PUBLIC_URL%/logo.png" />
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo.png" />
    <!--
      manifest.json provides metadata used when your web app is installed on a
      user's mobile device or desktop. See
      https://developers.google.com/web/fundamentals/web-app-manifest/
    -->
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />

    <link rel="preconnect" href="https://fonts.gstatic.com">
    <link
      href="https://fonts.googleapis.com/css2?family=Lato:wght@100;300;400;500;600;700
      &display=swap" rel="stylesheet">
    <!--
      Notice the use of %PUBLIC_URL% in the tags above.
      It will be replaced with the URL of the `public` folder during the build.
      Only files inside the `public` folder can be referenced from the HTML.

      Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC_URL%/favicon.ico" will
      work correctly both with client-side routing and a non-root public URL.
      Learn how to configure a non-root public URL by running `npm run build`.
    -->
    <title>DSE - Decentralized Stock Exchange</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
    <!--
```

This HTML file is a template.

If you open it directly in the browser, you will see an empty page.

You can add webfonts, meta tags, or analytics to this file.

The build step will place the bundled scripts into the <body> tag.

To begin the development, run `npm start` or `yarn start`.

To create a production bundle, use `npm run build` or `yarn build`.

```
-->
</body>
</html>
```

CSS CODE:

```
html, #root, body, .MainContainer{
  height: 100%;
  width: 100vw;
}

body {
  margin: 0;
  padding: 0;
  font-family: 'Lato', -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto',
'Oxygen',
  'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',
  sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  position: relative;
  overflow: hidden;
}

*{
  padding: 0;
  margin: 0;
}

:root{
  --main-color: #014F86;
  --sec-color: white;
  --bg-color: #335F8A;
}

.MainContainer{
  background-color: var(--bg-color);
  background-image: url('./resources/carbon-fibre.png');
  background-repeat: repeat;
  position: relative;
}
```

```
.MobileContainer{
  position: absolute;
  background-color: var(--sec-color);
  box-shadow: 1px 1px 2px 2px rgba(0, 0, 0, 0.25);
}
```

```
.MobileContainer::after{
  position: fixed;
  bottom: 0;
  left: 0;
  width: 100%;
  height: 20px;
  font-size: 0.8em;
  background-color: #5a189a;
  color: white;
  content: "By: Adarsh Shrivastava";
  text-align: center;
  vertical-align: middle;
}
```

```
.MobileContainerFlow{
  display: flex;
  flex-direction: column;
  justify-content: flex-start;
  align-items: flex-start;
}
```

```
.MobileContainerScroll{
  width: calc(100% - 40px);
  height: calc(100% - 60px);
  padding: 20px;
  padding-bottom: 40px;
  position: absolute;
  top: 0;
  left: 0px;
  display: flex;
  flex-direction: column;
  justify-content: flex-start;
  align-items: flex-start;
  overflow-x: hidden;
  overflow-y: scroll;
}
```

```
::-webkit-scrollbar-track{
  background-color: white;
  border: 0.5px solid gray;
}
```

```
::-webkit-scrollbar-thumb{
  background-color: var(--main-color);
  border-radius: 3px;
}
```

```

}

::-webkit-scrollbar{
  width: 5px;
  position: relative;
}

@media (max-width: 500px) {
  .MobileContainer{
    left: 0;
    top: 0;
    width: calc(100vw - 40px);
    height: calc(100% - 50px);
    padding: 20px;
    padding-bottom: 30px;
  }
}

@media (min-width: 501px) {
  .MobileContainer{
    left: 50%;
    top: 50%;
    width: calc((80vh * 0.5625) - 40px);
    height: calc(80vh - 60px);
    transform: translateX(-50%) translateY(-50%);
    padding: 20px;
    padding-bottom: 40px;
  }
}

```

JAVASCRIPT CODE:

```

import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';

```

```

ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById('root')
);

```

```

import React from 'react';
import {
  BrowserRouter as Router,
  Switch,

```

```

Route,
withRouter
} from 'react-router-dom';
import CompanyScreen from './screens/CompanyScreen';
import HomeScreen from './screens/HomeScreen';
import SplashScreen from './screens/SplashScreen';
import StocksScreen from './screens/StocksScreen';

class App extends React.Component{

  constructor(props){
    super(props);
    this.state = {};
  }

  render(){
    return(
      <Router >
        <Switch >
          <Route path="/Stocks">    <StocksScreen />    </Route>
          <Route path="/Dashboard">  <HomeScreen />    </Route>
          <Route path="/Company">    <CompanyScreen />  </Route>
          <Route path="/" exact>     <SplashScreen />  </Route>
        </Switch>
      </Router>
    );
  }
}

```

```
export default App;
```

```

import React, { Fragment } from 'react';
import { withRouter } from 'react-router';
import styled from 'styled-components';
import RefreshFloatBtn from '../components/RefreshFloatBtn';
import TransactionsEntry from '../components/TransactionsEntry';
import BackImg from '../resources/back.svg';
import RefreshImg from '../resources/refresh_black.svg';
import CompanyABI from '../Artifacts/CompanyABI.json';
import BuyDialog from '../components/BuyDialog';
import SellDialog from '../components/SellDialog';
import Web3 from 'web3';

```

```

class CompanyScreen extends React.Component{
  constructor(props){
    super(props);
    this.state = {
      currentTab: "desc", // desc, tran
      description: {
        id: null,
        name: "",

```

```

        rate: 0,
        volume: 0,
        maxrate: 0,
        minrate: 0,
        desc: ""
    },
    holdings: 0, // a number
    listingsRate: 0, // rate at which holdings are held
    listings: 0, // a number
    contractAddress: null,
    transactions: null, // {from, to, amount, rate}
    dialog: "", // buy, sell or "
  }
}

OnTabClicked(tab){
  if(tab == 'desc' && (this.state.transactions == null || this.state.transactions.length
== 0)){
    this.LoadTransactions();
  }

  this.setState({
    currentTab: tab
  });
}

GoBack(){
  this.props.history.goBack();
}

componentDidMount(){
  // check if all initialized else return to splash
  if(window.portisAccount == null){
    console.log('Invalid data, getting back to splash screen');
    this.props.history.push('/');
    return;
  }

  console.log('Company Page mounted : ' + new Date().getTime());
  if(this.state.description.id == null)
    this.LoadData();
}

LoadTransactions(){
  if(this.state.contractAddress == null)
    return;

  this.companyContract = new window.web3.eth.Contract(CompanyABI.abi,
this.state.contractAddress);
  this.companyContract.getPastEvents('Transaction')
    .then(events => {

```

```

    console.log('events', events);
    this.state.transactions = [];
    events.forEach(event => {
      if(event.returnValues._to !== null && event.returnValues._from !== null){
        this.state.transactions.push({
          to: event.returnValues._to,
          from: event.returnValues._from,
          amount: event.returnValues.amount,
          rate: event.returnValues.rate,
        });
      }
    });
    console.log('Final TRANSACTIONS', this.state.transactions);
    this.forceUpdate();
  })
  .catch(err => {
    console.log('Error loading transaction events', err);
    alert(`Error loading transaction events: ${err}`);
  });
}

```

```

LoadData(){
  var data = {};
  var query = this.props.location.search.replace('?', '');
  query.split('&').forEach(pair => {
    data[pair.split('=')[0]] = pair.split('=')[1];
  })
}

```

```

if(!data['stockId']){
  this.props.history.goBack();
  return;
}

```

```

window.exchangeContract.methods.getCompanyDetail(data['stockId'])
  .call({from: window.portisAccount})
  .then(details => {
    console.log('details', details);
    this.state.description = {
      id: details[0],
      name: details[1],
      rate: details[4],
      volume: details[3],
      maxrate: details[5],
      minrate: details[6],
      desc: details[7]
    };

    this.setState({
      holdings: details[8],
      listingsRate: details[10],
      listings: details[9],
    });
  });

```

```

        contractAddress: details[2]
      }, () => {
        if(this.state.transactions == null)
          this.LoadTransactions();
      });
    })
    .catch(err => {
      console.log('Error loading company details', err);
      alert('Error loading company details: ' + err + '. Please try again');
    })
  }

  RefreshPressed(){
    if(this.state.currentTab == 'desc')
      this.LoadData();
    else
      this.LoadTransactions();
  }

  BuyStocks(stocks, maxrate){
    try{
      var stocksVal = parseInt(stocks, 10);
      var rateVal = parseInt(maxrate, 10);
      if(stocksVal <= 0 || rateVal <= 0){
        alert('While buying stocks, Amount of Stocks and rate should be positive whole numbers');
        return;
      }

      var value = stocksVal * rateVal;

      window.exchangeContract.methods.buyStock(this.state.description.id,
rateVal, stocksVal)
        .send({from: window.portisAccount, value: Web3.utils.toWei(value + "
"ether"))})
        .on("transactionHash", (hash) => {
          alert('Stock buy request send. Stocks will reflect in your account in some time! In case stock buy request is rejected, you will get a full refund.');
```

time! In case stock buy request is rejected, you will get a full refund.');

```

          setTimeout(() => this.LoadData(), 2500);
        })
        .on("error", (error, receipt) => {
          alert("Error buying stocks : " + error);
          console.log('Error buying stocks ', error, receipt);
        });
      }catch(ex){
        console.log('error value buy stock', ex);
        alert('While buying stocks, Amount of Stocks and rate should be positive whole numbers');
      }

      this.setState({dialog: ""});
    }
  }

```

```

}

SellStocks(stocks, rate){
  try{
    var stocksVal = parseInt(stocks, 10);
    var rateVal = parseInt(rate, 10);
    if(stocksVal <= 0 || rateVal <= 0){
      alert('While selling stocks, rate should be positive whole numbers');
      return;
    }
    if(stocksVal > this.state.holdings){
      alert(`You only own ${this.state.holdings} stocks. So you can't sell more than
that`);
      return;
    }

    console.log(this.state.description.id, stocksVal, rateVal);
    window.exchangeContract.methods.listStocks(this.state.description.id,
stocksVal, rateVal)
      .send({from: window.portisAccount})
      .on("transactionHash", (hash) => {
        console.log('Stocks listed', data);
        alert('Your stocks have been listed for sale. They will be added to your
account as holdings in some time.')
        setTimeout(() => this.LoadData(), 2500);
      })
      .on("error", (error, receipt) => {
        console.log('Stocks listing error: ', error, receipt);
        alert('Error Selling stocks : ' + error);
      });

  }catch(ex){
    console.log('error value buy stock', ex);
  }
  this.setState({dialog: ""});
}

UnlistStocks(){
  window.exchangeContract.methods.unlistStocks(this.state.description.id)
    .send({from: window.portisAccount})
    .on("transactionHash", (hash) => {
      console.log('Stocks unlisted', data);
      alert('Your stocks have been unlisted for sale. As soon as they are bought,
your wallet will be credited with the amount. Also, You could anytime unlist your stocks
and keep them with you.')
      setTimeout(() => this.LoadData(), 2500);
    })
    .on("error", (error, receipt) => {
      console.log('Stocks unlisting error: ', error, receipt);
      alert('Error Unlisting stocks : ' + error);
    });
}

```

```

    }

    render(){
      return(
        <div className="MainContainer">
          <div className="MobileContainer MobileContainerFlow">
            <TopBar>
              <Icon onClick={() => this.GoBack()} src={BackImg}/>
              <Icon onClick={() => this.RefreshPressed()} src={RefreshImg}/>
            </TopBar>

            <CompanyName>{this.state.description.name !== "" ?
            `${this.state.description.name},           ${this.state.description.id}` :
            "Loading..."}</CompanyName>
            <CurrentRate>{this.state.description.rate ? this.state.description.rate :
            "..."} Ether</CurrentRate>

            <TabContainer>
              <TabLabel selected={this.state.currentTab == 'desc'} onClick={() =>
this.OnTabClicked('desc')}>Description</TabLabel>
              <TabLabel selected={this.state.currentTab == 'tran'} onClick={() =>
this.OnTabClicked('tran')}>Transactions</TabLabel>
            </TabContainer>

            {this.state.currentTab == 'desc' ?
              this.GetDescriptionTabView()
              :
              this.GetTransactionTabView()}

            <ButtonPanel>
              <BottomButton   onClick={()   =>   this.setState({dialog:   'sell'})}
disabled={this.state.holdings == 0}>SELL</BottomButton>
              <BottomButton   onClick={()   =>   this.setState({dialog:   'buy'})}
>BUY</BottomButton>
            </ButtonPanel>

            {this.state.dialog === 'buy' &&
              <BuyDialog
                currentRate={this.state.description.rate}
                onClose={() => this.setState({dialog: ""})}
                onBuyClick={(stocks, rate) => this.BuyStocks(stocks, rate)}
              />
            }

            {this.state.dialog === 'sell' &&
              <SellDialog
                currentRate={this.state.description.rate}
                stocks={this.state.holdings}
                onClose={() => this.setState({dialog: ""})}
                onSellClick={(stocks, rate) => this.SellStocks(stocks, rate)}
              />
            }
          </div>
        </div>
      )
    }
  }
}

```

```

    }
  </div>
</div>
);
}

GetDescriptionTabView(){
  return(
    <TabView>
      <DescGrid>
        <span style={{gridArea: "item1"}}>
          <StatTitle>Your Holdings</StatTitle>
          <StatValue>{this.state.holdings} Stocks</StatValue>
        </span>
        <span style={{gridArea: "item2"}}>
          {this.state.listings > 0 &&
            <Fragment>
              <StatTitle clickable onClick={() => this.UnlistStocks()}>Listings(click
to unlist)</StatTitle>
              <StatValue>{this.state.listings} Stocks</StatValue>
            </Fragment>
          </span>

          <span style={{gridArea: "item3"}}>
            <StatTitle>Volume</StatTitle>
            <StatValue>{this.state.description.volume} Stocks</StatValue>
          </span>
          <span style={{gridArea: "item4"}}>
            <StatTitle>Market Cap</StatTitle>
            <StatValue>{this.state.description.rate * this.state.description.volume}
Ethers</StatValue>
          </span>

          <span style={{gridArea: "item5"}}>
            <StatTitle>Max</StatTitle>
            <StatValue>{this.state.description.maxrate} Ethers</StatValue>
          </span>
          <span style={{gridArea: "item6"}}>
            <StatTitle>Min</StatTitle>
            <StatValue>{this.state.description.minrate} Ethers</StatValue>
          </span>

          <span style={{gridArea: "desc"}}>
            <StatTitle>Description</StatTitle>
            <StatValue style={{marginBottom: "60px"}}>
              {this.state.description.desc}
            </StatValue>
          </span>
        </DescGrid>
      </TabView>
    )
  }
}

```

```

    }

    GetTransactionTabView(){
      return(
        <TableView>
          {this.state.transactions && this.state.transactions.length > 0 ?
            this.state.transactions.map(trans => {
              return(
                <TransactionsEntry
                  to={trans.to}
                  from={trans.from}
                  rate={trans.rate}
                  stocks={trans.amount}/>
              );
            })
            :
            <NoTransactions>No Transactions Yet</NoTransactions>
          }
        </TableView>
      )
    }
  }
}

```

```

const TopBar = styled.div`
  width: 100%;
  display: flex;
  flex-direction: row;
  justify-content: space-between;
  align-items: center;
`;

```

```

const Icon = styled.img`
  width: 20px;
  height: 20px;
  padding: 5px;
  object-fit: contain;

  :hover{
    cursor: pointer;
  }
`;

```

```

const CompanyName = styled.p`
  font-size: 1.2em;
  color: #6B6B6B;
  font-weight: 400;
  margin-top: 20px;
`;

```

```

const CurrentRate = styled.p`
  font-size: 2.2em;

```

```

    color: var(--main-color);
    font-weight: 600;
`;

const TabContainer = styled.div`
  width: 100%;
  display: flex;
  flex-direction: row;
  justify-content: flex-start;
  align-items: flex-end;
  flex-wrap: wrap;
  margin-top: 20px;
  margin-bottom: 20px;
`;

const TabLabel = styled.button`
  font-size: 1.1em;
  color: ${props => props.selected ? "black" : "#AEAEAE"};
  border: none;
  background-color: transparent;
  border-bottom: ${props => props.selected ? "1px solid black": "none"};
  padding-bottom: 4px;
  margin-right: 15px;
  outline: none;

  :hover{
    cursor: pointer;
    border-bottom: 1px solid black;
  }
`;

const TabView = styled.div`
  flex: 1;
  width: 100%;
  display: flex;
  flex-direction: column;
  overflow-x: hidden;
  overflow-y: auto;
`;

const DescGrid = styled.div`
  width: 100%;
  display: grid;
  grid-template-areas: 'item1 item2' 'item3 item4' 'item5 item6' 'desc desc';
  grid-column-gap: 10px;
  grid-row-gap: 10px;
`;

const StatTitle = styled.p`
  font-size: 0.8em;
  color: #8A8A8A;

```

```

font-weight: 600;
margin-bottom: 5px;

: hover{
  cursor: ${props => props.clickable ? "pointer": "text"};
}
`;

const StatValue = styled.p`
  font-size: 1em;
  color: black;
  font-weight: 700;
  margin-bottom: 10px;
`;

const ButtonPanel = styled.div`
  width: calc(100% + 40px);
  display: flex;
  flex-direction: row;
  justify-content: flex-start;
  align-items: center;
  margin-top: 10px;
  margin-left: -20px;

  @media (max-width: 500px){
    margin-bottom: -10px;
  }
  @media (min-width: 501px){
    margin-bottom: -20px;
  }
`;

const BottomButton = styled.div`
  color: var(--sec-color);
  background-color: ${props => props.disabled ? "gray" : "var(--main-color)"};
  font-size: 1em;
  font-weight: 400;
  flex: 1;
  outline: none;
  border: 0.5px solid gray;
  text-align: center;
  padding: 10px;

  : hover{
    cursor: pointer;
  }
`;

const NoTransactions = styled.p`
  font-size: 1.4em;
  color: #9e9e9e;

```

```

    font-weight: 700;
    align-self: center;
    margin-top: 40px;
  },

export default withRouter(CompanyScreen);

import React from 'react';
import { withRouter } from 'react-router';
import styled from 'styled-components';
import MyHoldingEntry from '../components/MyHoldingEntry';
import RefreshFloatBtn from '../components/RefreshFloatBtn';
import Web3 from 'web3';

class HomeScreen extends React.Component{
  constructor(props){
    super(props);
    this.state = {
      walletId: window.portisAccount,
      balance: null,
      netWorth: null,
      holdings: [], // {name, id, rate, stocks}
      refreshing: true,
    };
  }

  LoadPortfolio(){
    window.exchangeContract.methods.getUserPortfolio()
      .call({from: window.portisAccount})
      .then(data => {
        console.log('portfolio', data);
        const ids = data[0];
        const names = data[1];
        const rates = data[2];
        const stocks = data[3];
        this.state.holdings = [];
        for(let i = 0; i < ids.length; i++){
          this.state.holdings.push({
            id: ids[i],
            name: names[i],
            rate: rates[i],
            stocks: stocks[i]
          });
        }
        this.CalculateNetWorth();
        this.forceUpdate();
        setTimeout(() => {
          this.setState({
            refreshing: false,
          });
        }, 1000);
      });
  }
}

```

```

    }).catch(err => {
      console.log(`Error loading portfolio ${err}`);
    });
  }

  LoadBalance(){
    console.log('Loading balance...');
    window.web3.eth.getBalance(window.portisAccount)
      .then(bal => {
        console.log('balance', bal);
        this.setState({
          balance: `${Web3.utils.fromWei(bal, "ether")} Ethers`
        });
      }).catch(err => {
        console.log('Error loading balance', err);
        this.setState({
          balance: `0 Ethers`
        });
      });
  }

  CalculateNetWorth(){
    this.state.netWorth = 0;
    this.state.holdings.forEach(holding => {
      this.state.netWorth += holding.rate * holding.stocks;
    });
    this.state.netWorth = `${this.state.netWorth} Ethers`;
  }

  componentDidMount(){
    // check if all initialized else return to splash
    if(window.portisAccount == null){
      console.log('Invalid data, getting back to splash screen');
      this.props.history.push('/');
      return;
    }

    console.log('Home mounted : ' + new Date().getTime());
    if(this.state.balance == null || this.state.netWorth == null){
      this.LoadPortfolio();
      this.LoadBalance();
    }
  }

  ShowAllStocks(){
    this.props.history.push('/Stocks');
  }

  MyHoldingClicked(id){
    this.props.history.push(`/Company?stockId=${id}`);
  }

```

```

RefreshClicked(){
  this.setState({
    refreshing: true,
  });

  this.LoadBalance();
  this.LoadPortfolio();
}

render(){
  return(
    <div className="MainContainer">
      <div className="MobileContainer MobileContainerFlow">
        <div className="MobileContainerScroll">
          <AllStocksBtn      onClick={()      =>      this.ShowAllStocks()}>ALL
STOCKS</AllStocksBtn>
          <span style={{marginTop: "40px"}}>
            <StatTitle>Wallet ID</StatTitle>
            <StatValue small>{this.state.walletId}</StatValue>
            <StatTitle>Wallet Balance</StatTitle>
            <StatValue>{this.state.balance      ?      this.state.balance      :      "...
Ethers"}</StatValue>
            <StatTitle>Portfolio Net Worth</StatTitle>
            <StatValue>{this.state.netWorth      ?      this.state.netWorth      :      "...
Ethers"}</StatValue>
          </span>

          <HoldingTitle>Your Holdings</HoldingTitle>
          {this.state.holdings.map(holding => {
            return(
              <MyHoldingEntry
                key={holding.id}
                companyName={holding.name}
                stockCount={holding.stocks}
                rate={holding.rate}
                onClick={() => this.MyHoldingClicked(holding.id)}>
            );
          })}

          {this.state.holdings.length == 0 &&
            <NoHoldings>No Holdings Found!</NoHoldings>
          }
        </div>
        <RefreshFloatBtn      onClick={()      =>      this.RefreshClicked()}
refreshing={this.state.refreshing} />
      </div>
    </div>
  )
}
}

```

```

const AllStocksBtn = styled.button`
  border: none;
  color: var(--main-color);
  font-weight: 700;
  font-size: 0.9em;
  align-self: flex-end;
  background-color: transparent;

  :hover{
    cursor: pointer;
  }
`;

const StatTitle = styled.p`
  font-size: 0.8em;
  color: #8A8A8A;
  font-weight: 600;
  margin-bottom: 5px;
`;

const StatValue = styled.p`
  /* font-size: ${!(props) => props.small ? "0.6em" : "1em"}; */
  font-size: 1em;
  color: black;
  font-weight: 700;
  margin-bottom: 10px;
  max-width: 70%;
  overflow-wrap: break-word;
`;

const HoldingTitle = styled.p`
  font-size: 1.2em;
  font-weight: 700;
  color: var(--main-color);
  align-self: center;
  margin-top: 20px;
  margin-bottom: 20px;
`;

const NoHoldings = styled.p`
  font-size: 1.4em;
  color: #9e9e9e;
  font-weight: 700;
  align-self: center;
  margin-top: 40px;
`;

export default withRouter(HomeScreen);

```

```

import React from 'react';
import PortisImage from '../resources/power_portis.svg'
import Logo from '../resources/logo.png';
import styled from 'styled-components';
import LoadingSpinner from '../components/LoadingSpinner';
import { withRouter } from 'react-router';
import Portis from '@portis/web3';
import Web3 from 'web3';
import Exchange from '../Artifacts/ExchangeABI.json';
import Company from '../Artifacts/CompanyABI.json';
import Config from '../Config.json';

class SplashScreen extends React.Component{
  constructor(props){
    super(props);
    this.state= {
      loaded: false,
      refreshing: false,
    }
  }

  componentDidMount(){
    this.loadWeb3AndPortis();
  }

  async loadWeb3AndPortis(){
    try{
      window.portis = new Portis(Config.portisAppId, {
        nodeUrl: Config.nodeUrl,
        chainId: Config.chainId
      });
      window.web3 = new Web3(window.portis.provider);
    }catch(err){
      alert('Error Logging in : ' + err + '. Please Retry. ');
      this.loadWeb3AndPortis();
    }

    window.web3.eth.getAccounts().then(accounts => {
      window.portisAccount = accounts[0];
      console.log(`Account: ${window.portisAccount}`);
      window.exchangeContract = new window.web3.eth.Contract(Exchange.abi,
Config.exchangeContractAddress);
      setTimeout(() => {
        this.props.history.push('/Dashboard');
      }, 2000);
    }).catch(err => {
      alert('Error Logging in : ' + err + '. Please Retry. ');
      this.loadWeb3AndPortis();
    })
  }
}

```

```

    }

    render(){
      return(
        <div className="MainContainer">
          <MobileContainer className="MobileContainer MobileContainerFlow"
style={{justifyContent: "space-evenly", alignItems: "center"}}>
            <span style={{marginTop: "40px", textAlign: "center"}}>
              <SplashImage src={Logo} />
              <TitleSpan>
                <h2>DECENTRALIZED</h2>
                <h3>STOCK EXCHANGE</h3>
              </TitleSpan>
            </span>
            <LoadingSpinner show={!this.state.loaded} />
            <PoweredPortis src={PortisImage} />
          </MobileContainer>
        </div>
      );
    }
  }
}

```

```

const MobileContainer = styled.div`
  justify-content: flex-end;
  align-items: center;
`;

```

```

const PoweredPortis = styled.img`
  width: 100px;
`;

```

```

const TitleSpan = styled.span`
  text-align: center;
  color: var(--main-color);
  margin-top: 20px;
`;

```

```

const SplashImage = styled.img`
  width: 200px;
`;

```

```

export default withRouter(SplashScreen);

```

```

import React from 'react';
import styled from 'styled-components';
import BuyDialog from '../components/BuyDialog';
import RefreshFloatBtn from '../components/RefreshFloatBtn';
import SellDialog from '../components/SellDialog';
import StockEntry from '../components/StockEntry';
import TransactionsEntry from '../components/TransactionsEntry';

```

```

import BackImg from '../resources/back.svg';
import RefreshImg from '../resources/refresh_black.svg';
import {withRouter} from 'react-router-dom';

class StocksScreen extends React.Component{
  constructor(props){
    super(props);
    this.state = {
      stocks: [],
      dataLoaded: false,
    }
  }

  componentDidMount(){
    // check if all initialized else return to splash
    if(window.portisAccount == null){
      console.log('Invalid data, getting back to splash screen');
      this.props.history.push('/');
      return;
    }

    console.log('AllStocks mounted : ' + new Date().getTime());
    if(!this.state.dataLoaded)
      this.LoadAllStocks();
  }

  LoadAllStocks(){
    window.exchangeContract.methods.getCompanyListDetailed()
      .call({from: window.portisAccount})
      .then(data => {
        var ids = data[0];
        var names = data[1];
        var volumes = data[2];
        var rates = data[3];

        this.state.stocks = [];
        for(let i = 0; i < ids.length; i++){
          this.state.stocks.push({
            id: ids[i],
            name: names[i],
            volume: volumes[i],
            rate: rates[i]
          });
        }

        this.state.dataLoaded = true;
        this.forceUpdate();
      })
      .catch(err => {
        console.log('Error loading stocks ',err)
        alert(`Error loading stocks: ${err}. Please try again`);
      })
  }
}

```

```

    })
  }

  render(){
    return(
      <div className="MainContainer">
        <div className="MobileContainer MobileContainerFlow">
          <TopBar>
            <Icon onClick={() => this.props.history.goBack()} src={BackImg}/>
            <Title>ALL STOCKS</Title>
            <Icon onClick={() => this.LoadAllStocks()} src={RefreshImg} />
          </TopBar>

          <StockList>
            {this.state.stocks.map(stock => {
              return(
                <StockEntry
                  key={stock.id}
                  companyName={stock.name}
                  rate={stock.rate}
                  marketCap={stock.volume * stock.rate}
                  onClick={()
this.props.history.push(`/Company?stockId=${stock.id}`)}/>
                )
              })}
          </StockList>
        </div>
      </div>
    );
  }
}

```

```

const TopBar = styled.div`
  width: 100%;
  display: flex;
  flex-direction: row;
  justify-content: space-between;
  align-items: center;
`;

```

```

const Title = styled.p`
  flex: 1;
  font-size: 1.2em;
  font-weight: 600;
  color: black;
  margin-left: 10px;
  margin-right: 10px;
`;

```

```

const Icon = styled.img`
  width: 20px;

```

B. SCREEN SHOTS

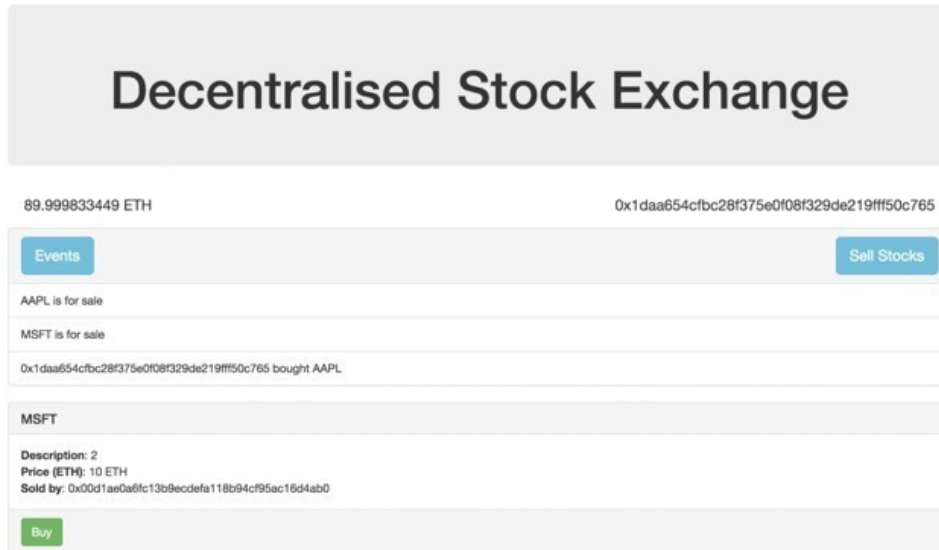


Fig B.1 Decentralized Stock Exchange

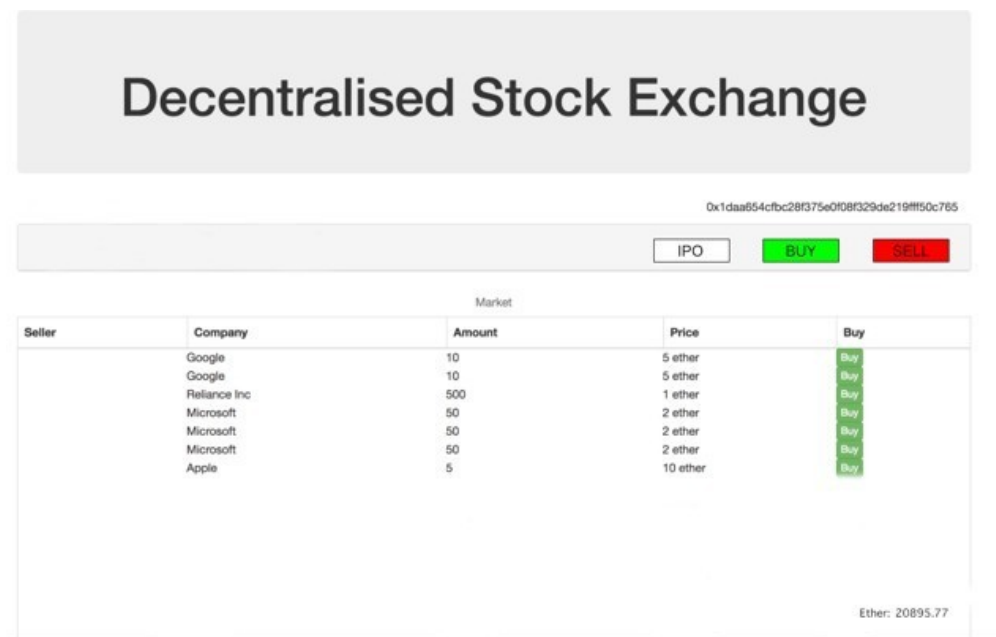


Fig B.2 Buy and Sell Contracts

C. RESEARCH PAPER

DECENTRALIZED STOCK EXCHANGE USING BLOCKCHAIN NETWORK

1st Ronie Joe.J
Department of Computer Science and
Engineering,
Sathyabama Institute of Science and
Technology
Tamil Nadu 600119, India
ronniejoe_j@icloud.com

2nd Pooja R.A
Department of Computer Science and
Engineering,
Sathyabama Institute of Science and
Technology
Tamil Nadu 600119, India
poojaamamath1533@gmail.com

3rd Dr.S.L.Jany Shabu
Department of Computer Science and
Engineering,
Sathyabama Institute of Science and
Technology
Tamil Nadu 600119, India
janyshabu.cse@sathyabama.ac.in

Abstract— This research endeavors to contribute to the ongoing evolution of decentralized financial systems by proposing a novel architecture for a decentralized stock exchange. Drawing inspiration from the successes and challenges of established decentralized exchanges in the cryptocurrency space, the project leverages blockchain technology to address identified shortcomings. The envisioned architecture aims to provide a decentralized solution that ensures asset ownership replication, transparent market order execution, and accurate settlement verification across all network nodes. By eliminating the need for a central authority or intermediary in order placement and execution, the proposed system seeks to enhance security, transparency, and operational resilience. The study also involves the practical implementation of a prototype on an Ethereum-based platform, with a focus on a subset of rules derived from established decentralized exchanges. Through comparative analysis, particularly in terms of transaction fees, with reference to existing decentralized exchange models, this project aspires to showcase the potential benefits and advancements that can be achieved in the realm of decentralized stock trading. By building on the foundations laid by existing decentralized exchanges and addressing their limitations, this research seeks to contribute valuable insights and innovations to the development of a decentralized stock exchange, paving the way for a more secure, transparent, and inclusive financial ecosystem.

Keywords — *Centralized Governance, Blockchain, Transparency.*

I. INTRODUCTION

In the ever-evolving landscape of modern financial systems, this research project aspires to spearhead a transformative era by presenting a revolutionary architecture for the development of a decentralized stock exchange. While traditional stock exchanges have undeniably played a pivotal role in shaping the global economy, the emergence of decentralized financial systems, inspired by the successes and challenges witnessed in established cryptocurrency exchanges, signifies a paradigm shift in the way we envision and engage with financial markets. Conventional stock markets operate within centralized applications that consolidate all trading actions, offering advantages such as a central authority ensuring transaction authenticity, security, and validity. However, this well-established architecture is not without its drawbacks, including a susceptibility to a single point of failure, potential bottlenecks, and a lack of transparency in the auctioning process. Against this backdrop, the proposed decentralized stock exchange capitalizes on the transformative potential of blockchain technology to address these inherent shortcomings. This research endeavors to bridge the time-tested practices of established decentralized exchanges with the strengths of traditional stock markets, fostering a symbiotic relationship between the old and the new. By synthesizing the transparency and security offered by decentralized systems with the efficiency and market infrastructure of traditional exchanges, the project aims to cultivate a harmonious convergence. Through the lens of a carefully crafted prototype implemented on an Ethereum-based platform, this study will scrutinize a subset of rules derived from established decentralized exchanges, ensuring alignment with industry standards and practices. In essence, this research seeks to offer not just an alternative but a holistic evolution in the financial ecosystem by proposing a decentralized stock exchange that amalgamates the resilience and transparency of decentralized systems with the robust market mechanisms of traditional exchanges. The outcome aspires to be a blueprint for a future where financial markets seamlessly integrate the best attributes of both worlds, fostering a more secure, transparent, and efficient avenue for market participants.

II. LITERATURE SURVEY

Blockchain technology, as a foundational element, has garnered extensive attention within academic and industry circles. Numerous studies highlight its potential to revolutionize traditional financial systems by providing a decentralized and tamper-resistant ledger. Scholars emphasize the immutability, transparency, and security afforded by blockchain, underscoring its applicability to asset ownership, settlement processes, and market order execution in financial transactions (Swan, 2015; Tapscott and Tapscott, 2016).

In their work, Badr Bellaj et.al [2] introduced a framework for distributed ledger technologies. The intricacy of comprehending various types blockchain networks or designing and adopting the appropriate blockchain network solution poses a challenge for decision-makers. In response to this challenge, they propose a DCEA reference model. DCEA aims to provide a conceptualization and in-depth examination of blockchain and globally distributed ledger technologies (DLT) through a layer

typed framework. This framework is constructed of four essential layers: the data layer, consensus layer, execution layer,

application layer. Such a layered approach enhances the understanding and decision-making processes related to blockchain technology.

Claudia Pop et.al [3] have achieved a milestone by successfully implementing a blockchain based stock exchange through the application of blockchain technology. This innovative approach effectively overcomes the limitations of traditional stock exchange systems, including massive transaction fees susceptibility to hacking due to centralized governance, and a lack of transparency. Their blockchain system primarily focuses on reducing transaction costs and is notable for using a smart contract for the roles of Traders and Exchange, streamlining the process. The implementation is specifically on the Ethereum platform.

In a related effort, N.S. Akhilesh et.al [10] have presented a framework aimed at enhancing the security of transactions on the exchange through blockchain technology. Their work meticulously explores both the legal and technical ramifications of decentralized market. The method adopts a hybrid blockchain model, incorporating proof of stake and a round-robin approach. This framework places significant emphasis on addressing the technological and legal aspects of blockchain-based stock exchange systems, showcasing a comprehensive strategy to tackle the diverse challenges associated applications.

Bhavarth Joshi et.al [3] investigate the elements influencing the implementation of a stock exchange by blockchain technology. They assert that while blockchain offers numerous capabilities, such as enhanced protection and the removal of intermediaries, there remains an unresolved issue related to data transparency. The blockchain exposes every market participant, creating a potential disadvantage for multiple investors. This challenge poses an obstacle when attempting to scale the system to the magnitude of centralized stock exchanges. Despite being a dynamic concept, blockchains come with specific limitations, making them unsuitable for all scenarios. The author underscores the importance of thoroughly assessing system requirements before initiating the design process.

Y. Wanjun et.al [16] developed a module for network trading systems utilizing decentralized technology. The framework consists of a few modules called business center system, payment center system, third-party identity authentication. These few systems ensures information integrity and prevents any information loss during transmission, employing P2P network technology. Various algorithms, including asymmetric encryption technology are employed to guarantee information integrity and prevent any form of information loss throughout network transmission.

The implementation of self-executing smart contracts presents the opportunity to eliminate the need for third-party and middleman, as the execution process is automated. The cost of executing a contract is directly tied to the amount of power consumed during the transaction.

In a foundational development by Tien Tuan Anh Dinh et.al [14], a simple replicated state machine model was implemented to facilitate the transfer of coins between addresses. This marked the inception of blockchain's expansion beyond cryptocurrencies, evolving to support user-defined states machine model. Ethereum, for instance, enables the creation of decentralized and replicated applications known as smart contracts. Notably, industry has driven the development of new decentralized platforms tailored for private configurations where people undergo authentication.

In a parallel exploration, Sara Rouhani et.al [12] highlight the substantial attention garnered by blockchain in both study application and practical application. Smart contracts, considered programmable transactions capable of sophisticated tasks, automatic execution, and storage on the blockchain, have been a focal point. The authors delve into fundamental issues associated with these technologies while also outlining areas for future research. Noteworthy issues discussed include endpoint vulnerabilities, security of public and private keys, and the integration platforms for blockchain technology.

Abhishek Dixit et.al [1] have developed a Multi-Agent System (MAS) to design smart contracts that not only exhibit enhanced flexibility but also possess self-awareness and legal viability. Traditional contracts lack flexibility and resist alterations, whereas these newly introduced contracts are termed Self-Aware Contracts (SAC).

In a separate study, Maher Alharby et.al [6] conducted a comprehensive mapping analysis encompassing all technology-oriented smart contract research. The papers were categorized into six groups, covering security, privacy, software engineering, application, performance & scalability, and other smart contract-related topics. The majority of these publications centered around 64% applications and 21%. Notably, differentiate to their 2017 survey, they observed an eightfold increase in the number of relevant articles, with a significant shift towards applications of smart contracts.

III. EXISTING SYSTEM AND THEIR PROBLEMS

Decentralized stock exchanges were still emerging and gaining traction. Decentralized finance (DeFi) platforms, which include decentralized exchanges (DEXs), have been more prominent in the cryptocurrency space. These platforms enable users to trade digital assets directly without a centralized authority.

A. Popular Decentralized Exchanges for Cryptocurrencies

- **Uniswap:** Uniswap functions as a decentralized automated market maker (AMM), enabling users to directly exchange a variety of Ethereum-based tokens from their wallets. Its operations are conducted on the Ethereum blockchain.
- **SushiSwap:** Similar to Uniswap, SushiSwap is a decentralized exchange that also incorporates additional features such as yield farming and staking.
- **Balancer:** Balancer is a decentralized liquidity protocol that enables users to create and administer liquidity pools for various tokens. It provides automated portfolio management.
- **Curve Finance:** Curve is designed for stablecoin trading and aims to provide low slippage for stablecoin swaps. It is optimized for stablecoin trading pairs.
- **1inch:** 1inch operates as a decentralized exchange aggregator, consolidating liquidity from different decentralized exchanges (DEXs) to offer users optimal trading rates.

It's important to note that these platforms primarily focus on decentralized trading of cryptocurrencies and tokens rather than traditional stocks.

B. Potential Technologies in Decentralized Stock Exchange

There were ongoing efforts and discussions about the development of decentralized technologies for stock trading, but no widely adopted and fully decentralized stock exchanges were in operation at that time. The decentralized finance (DeFi) movement had primarily focused on cryptocurrency and token trading. For decentralized stock exchanges, several challenges needed to be addressed, including regulatory compliance, liquidity, and the integration with traditional financial systems. Decentralized technologies were being explored, but it's crucial to verify the latest developments as the landscape evolves. Some potential technologies and concepts that might be explored or integrated into decentralized stock exchanges include:

- **Blockchain Technology:** The underlying technology for many decentralized systems. Blockchain can provide transparency, security, and immutability to the trading process.
- **Decentralized Identity and Authentication:** Ensuring secure and verifiable identity is crucial in financial systems. Decentralized identity solutions could enhance security and privacy in decentralized stock exchanges.
- **Oracles:** Decentralized oracles could be used to bring external data (such as real-time stock prices) onto the blockchain in a trustworthy manner, facilitating accurate pricing on decentralized exchanges.
- **Interoperability Standards:** Standards that allow different blockchain networks to communicate and share information. This could be crucial for connecting decentralized stock exchanges with each other or with traditional financial systems.

D. Open Problems in Existing System

- **Security Risks:** Centralized exchanges are exposed to security risks, making them vulnerable to hacking and other threats that can lead to the compromise of user funds.
- **Lack of Privacy:** Users on centralized exchanges may be required to disclose personal information, including identity documents and banking details, raising concerns about privacy and data security.
- **Centralized Control:** Centralized exchanges are governed by a single entity, potentially impacting operations and pricing. This centralized control goes against the decentralized ethos of cryptos.
- **Central Point of Failure:** Centralized exchanges represent a single point of failure, and any issues with the platform can lead to market-wide disruptions, affecting all users connected to the platform.

III. PROPOSED SYSTEM

In today's evolving financial world, the need for decentralized and transparent solutions is more pronounced than ever. This proposal outlines the development of a decentralized stock exchange platform utilizing Node.js, TestRPC, and Ethereum. This platform aims to revolutionize the traditional stock exchange model by leveraging the benefits of blockchain technology, ensuring security, transparency, and accessibility.

A. Technology Stack

- **Node.js:** Leverage the asynchronous, event-driven nature of Node.js for efficient and scalable server-side development.
- **TestRPC:** Use TestRPC for local testing and development, providing a simulated Ethereum environment for smart contract deployment.
- **Ethereum:** Utilize the Ethereum blockchain for building and deploying smart contracts that govern the rules of the decentralized stock exchange.
- **Web3.js:** Integrate Web3.js to interact with Ethereum smart contracts from the client-side, enabling a smooth user experience.
- **React:** Develop the user interface using React to create a responsive and interactive front-end for the decentralized stock exchange platform.

IV. WORKING

In the working section of this decentralized stock exchange project, the integration of P2P networks and artificial nodes plays a pivotal role in creating a robust and resilient system. The utilization of Ethereum blockchain, coupled with the efficiency of TestRPC, ensures a secure and decentralized environment for financial transactions. One of the notable features of the system is its remarkably reduced transaction time, enhancing the overall user experience. The implementation of Initial Public Offering (IPO) functionality introduces a dynamic element to the exchange, allowing companies to raise capital in a decentralized manner. Smart contracts, a cornerstone of the project, facilitate self-executing agreements, eliminating the need for intermediaries and

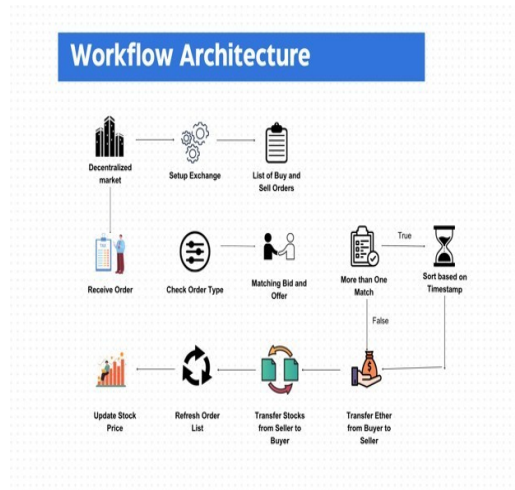


Fig 1 Workflow Diagram of Decentralized Stock Exchange

ensuring the automatic validation of transactional processes. This comprehensive approach, integrated seamlessly within the Ethereum blockchain, showcases the project's commitment to providing a decentralized stock exchange platform that is not only technologically advanced but also user-friendly and efficient.

The above figure demonstrates the functions of a decentralized stock exchange in a clear manner. Architecture has been split into three levels as shown in fig 1. This Exchange keep track of every order placed and processed. Orders with more than one match will be timestamped for sort listing. Once the order is matched the locked ether from buyers account will be deposited to the sellers account and the shares will be transferred to the buyer account. After a sale the order list will be refreshed without latency for stock price updating.

A. Implementation

The implementation of the decentralized stock exchange project on a MacBook, equipped with an M2 processor, 8GB RAM, and unified graphics, represents a sophisticated and high-performance environment for blockchain development. Leveraging the Truffle compiler ensures efficient smart contract deployment and management within the Ethereum blockchain. The utilization of artificial nodes further enhances the scalability and resilience of the network. The MacBook's powerful hardware capabilities contribute to a smooth development and testing process, allowing for seamless execution of complex blockchain operations. The inclusion of Alpaca API for stock market data integration adds real-time and accurate financial information to the decentralized exchange, enriching the user experience and ensuring the platform reflects the current market conditions. Moreover, the use of data stored in all nodes enhances redundancy and data integrity, promoting a distributed and secure ecosystem for stock trading. This implementation aligns with the project's goal of creating a decentralized stock exchange that combines cutting-edge technology with user-friendly features, providing a seamless and secure trading experience.

IV.RESULT AND DISCUSSION

The development of a decentralized stock exchange project utilizing MacBook, Node.js, TestRPC, and Ethereum presents a pioneering approach to revolutionize traditional financial systems. This project aims to address inherent challenges in centralized stock exchanges, such as massive transaction charges, susceptibility to hacking, and a transparency issue. By leveraging the power of blockchain technology through Ethereum, the system provides a secure and transparent platform for users to engage in buying, selling, and participating in Initial Public Offerings (IPOs). The use of Node.js enhances the efficiency and responsiveness of the platform, ensuring a seamless user experience. TestRPC serves as a local Ethereum blockchain for testing, streamlining the development and testing process. This decentralized stock exchange not only eliminates the need for intermediaries but also introduces a democratic and reliable decision-making process through consensus algorithms. The implementation on MacBook showcases the versatility of the system across different environments, making it accessible to a broader user base. Overall, this project represents a significant step towards creating a decentralized financial ecosystem that empowers users while ensuring the security of their assets and integrity of their transactions.

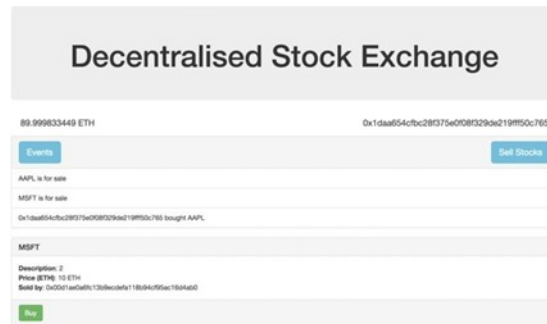


Fig 2 DSE Events Portal

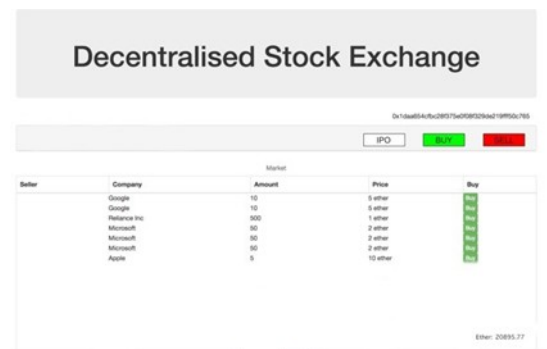


Fig 3 Decentralized Exchange Output

VII. CONCLUSION

In conclusion, the development and implementation of our decentralized stock exchange represent a significant leap forward in stock market transparency. Through cutting-edge technologies such as TestRPC, Ethereum, NodeJS we have successfully crafted a versatile and user-friendly tool that can project a clear picture of how a DEX can be used in stock market platform. This project not only showcases the powers of modern technology but also opens doors to a future where seamless transparency between public and stock exchange becomes the norm. Moving forward, the continuous refinement and expansion of this project promise to contribute to the ongoing evolution of blockchain technologies. The project also focuses on reducing transaction cost, blockchain functions through a decentralized ledger system, facilitating direct transactions without relying on banks or payment processors. As a result, this eliminates associated charges.

REFERENCE

- [1] Abhishek Dixit and A. Norta, "A Self-Aware Contract for Decentralized Peer-to-Peer (P2P) Commerce", 2018 IEEE 3rd International Workshops on Foundations and Applications of Self* Systems (FAS*W), Trento, pp. 17-19, Sept. 2018
- [2] Badr Bellaj, Aafaf Ouaddah, Emmanuel Bertin, Noel Crespi, Abdellatif Mezrioui, "DCEA: A Reference model for Distributed Ledger Technologies", IEEE, 2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), 03-06 May 2021
- [3] Bhavarth Joshi, Rutvi Shah, Siddhi Rane, Shilpa Kalantri, Priyanka Ghule, "Decentralized Stock Market using Blockchain", IEEE, 2021 International Conference on Intelligent Technologies (CONIT), 25-27 June 2021.
- [4] Claudia Pop; Cristian Pop, Antal Marcel, Andreea Vesa, Teodor Petrican, Tudor Cioara, Ionut Anghel, Ioan Salomie, "Decentralizing the Stock Exchange using Blockchain An Ethereum-based implementation of the Bucharest Stock Exchange", 2018 IEEE 14th International Conference on Intelligent Computer Communication and Processing (ICCP), Cluj-Napoca, pp. 459-466, Sept. 2018
- [5] J. Jaoude and R. Saade, "Blockchain applications Usage in different domains," IEEE Access, vol. 7, pp. 45372 45373, 2019, doi: 10.1109/ACCESS.2019.2902501.
- [6] Kaushik, A. Choudhary, C. Ektare, D. Thomas, and S. Akram, "Blockchain Literature survey," in Proc. 2nd IEEE Int. Conf. Recent Trends Electron., Inf. Commun. Technol. (RTEICT), May 2017, pp. 21452148.
- [7] Maher Alharby, Amjad Aldweesh, Aad van Moorsel, "Blockchain-based Smart Contracts: A Systematic Mapping Study of Academic Research", IEEE, ISSN 7281-1277, 15-17 November 2018.
- [8] M. Conoscenti, A. Vetro, and J. C. De Martin, "Blockchain for the Internet of Things: A systematic literature review," in Proc. IEEE/ACS 13th Int. Conf. Comput. Syst. Appl. (AICCSA), Nov. 2016, pp. 16.
- [9] M. Demir, M. Alal, O. Turetken, and A. Ferworn, "Security smells in smart contracts," in Proc. IEEE 19th Int. Conf. Softw. Qual., Rel. Secur. Companion (QRS-C), Jul. 2019, pp. 442449.
- [10] M. Samaniego, U. Jamsrandorj, and R. Deters, "Blockchain as a service for IoT," in Proc. IEEE Int. Conf. Internet Things (iThings) IEEE Green Comput. Commun. (GreenCom) IEEE Cyber, Phys. Social Comput. (CPSCom) IEEE Smart Data (SmartData), Dec. 2016, pp. 433 436.
- [11] N.S. Akhilesh, M.N. Aniruddha, K.S. Sowmya, "Implementation of Blockchain for Secure Bank Transactions", IEEE 2020 International Conference on Mainstreaming Block Chain Implementation, 21-22 February 2020.
- [12] Sara Rouhani and R. Deters, "Security, performance, and applications of smart contracts: A systematic survey," IEEE, vol. 7, 2019.
- [13] T. Ahram, A. Sargolzaei, S. Sargolzaei, J. Daniels, and B. Amaba, "Blockchain technology innovations," in Proc. IEEE Technol. Eng. Manage. Conf. (TEMSCON), Jun. 2017, pp. 137 141.
- [14] Tien Tuan Anh Dinh, R. Liu, M. Zhang, G. Chen, B. C. Ooi, and J. Wang, "Untangling blockchain: A data processing view of blockchain systems," IEEE, vol.30, no. 7, pp. 13661385, Jul. 2018.
- [15] Y.-H. Chen, S.-H. Chen, and I.-C. Lin, "Blockchain based smart contract for bidding system," in Proc. IEEE Int. Conf. Appl. Syst. Invention (ICASI), Apr. 2018, pp. 208 211.
- [16] Yu Wanjun, Wu Yuan, "Research on Network Trading System Using Blockchain Technology", 2018 International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS), IEEE, 21-24 October 2018

