International Journal of Multidisciplinary Research and Growth Evaluation
ISSN: 2582-7138
Received: 11-12-2020; Accepted: 16-01-2021
www.allmultidisciplinaryjournal.com
Volume 2; Issue 1; January-February 2021; Page No. 153-159

Distributed application (DApp) for securing user: Interaction logs for tasksusing ethereum block chain and smart contracts in solidity

A Sonya[1], Mohammed Shikeb[2]
[1, 2] B. S. Abdur Rahman Crescent Institute of Science and Technology, University, Chennai, Tamil Nadu, India

Corresponding Author: **A Sonya**

Abstract
The proposed project is targeted to implement a distributed application that can be utilized as task management system using Ethereum blockchain environment. Traditional task management systems pose a challenge of accountability while tracking tasks that need to be accomplished in a given amount of time. For the demonstration of the idea locally, we will be using ganache blockchain environment along with truffle framework with the Meta Mask tool to connect the distributed app to the Ethereum chain. The Ethereum platform is used to make smart contracts in solidity language. Smart contracts are pieces of instruction that once deployed on the blockchain cannot be tampered with. Traditional task managing system lack provenance and accountability in providing a motive to complete the task. The benefit of using the blockchain platform to achieve this goal is that the code once deployed on the blockchain restricts modification to the code. A normal web app interacts with the server to access data on the server-side database whereas a distributed app interacts with the web application that is connected to blockchain on a distributed network which executes smart contracts to produce results. All activities that happen on the network is stored in the blockchain, which can be used for provenance purpose.

Keywords: Distribute Application, Block Chain, Cryptography, Smart Contracts, Ethereum, Solidity

1. Introduction
A blockchain is a decentralized computation and information sharing platform, which allows multiple parties who may or may not trust each other to work together in a rational decision-making process. Unlike distributed platforms that rely on a single point of failure, the blockchain network follows a distributed ledger data structure. The latest copy of the ledger is present with all the nodes in the network. The public ledger contains the information about all the valid transactions that take place in the network. A block is not added to the chain unless verified by all the members in the network. Special nodes called miners are responsible to propose new blocks that are added to the blockchain using a consensus algorithm.
Blockchain helps achieve privacy of nodes, security and authenticity of transactions and integrity of data. Cryptographic hashing techniques are implemented in the block chain to ensure tamper proof nature of the system. Various consensus algorithms are available that help in establishing a reliable platform where only valid transactions can be put in the subsequent blocks of the block chain.
Bitcoin was the first breakthrough implementation of blockchain. Bitcoin was introduced in 2008 and was named the first fully digital currency, the launch was done is 2009. It is an implementation of permission less block chain environment, where anyone can join the network and start mining, which is to say that anyone can mine the blocks and earn bitcoins as the reward. It combines the cryptographic techniques with the economic incentives. Bitcoin reached $80 billion market capitalization in September 2017. Many alternate cryptocurrencies were introduced following the success of the bitcoin. The programming language used in early blockchains, for example the Bitcoin’s scripting language targeted more on reducing the complexity for having better security.
Ethereum is a public blockchain environment that is based on distributed computing and supports smart contract functionality. It is also open source. A modified version of Nakamoto consensus via transaction – based state transaction is also supported by Ethereum. Implementation of Turing complete language makes it theoretically compatible to use smart contracts in all practical computations. A smart contract is a piece of code that is stored on a blockchain and gets executed when a user interacts with the application is a way that will invoke the smart contract to produce output. With this enhanced functionality of the smart contracts came new security challenges relating to the design and secure programming practices.
Ethereum can be considered as a state machine. All the nodes present in an Ethereum network maintain a shared view of the

global state. Ether is the cryptocurrency used to invoke transactions and update the state. Ethereum supports two types of accounts: externally owned accounts and contract accounts. The account state consists of the following field

a. Nonce: The number of transactions sent by the externally controlled account or the number of contracts made by the contract accounts.
b. Balance: Ethers owned by the account
c. Storage Root: Merkle Patricia tree root of this account's storage.
d. Code Hash: Hash of this account's contract bytecode.

The address of the accounts is 160-bit which is a subset of the public key in case of the externally controlled accounts. For the contract accounts, the address is derived from the contract creator's address and its nonce.

Characteristics of a DApp have been discussed in [1]. The paper discusses about the advantages of blockchain technology that address the confidentiality, integrity and availability of the data by utilizing the platform.

Ethereum is vulnerable to denial – of – service attacks as the nodes participating in the network are unaware of the number of resources being utilized for validating a particular transaction. To prevent this the Ethereum protocol uses a pricing mechanism. It makes resource – intensive computations in smart contracts economically feasible. EVEM uses gas to price every computational step in a transaction. Market determines the price of the gas. For every transaction, the sender specifies the maximum amount of gas that a computation may consume, and also specifies the price the user has to pay per unit of gas. Miners can vote to gradually change the limit on the total amount of gas consumed in a block.

Ethereum uses the Proof-of-Work consensus to add blocks to the Ethereum block chain. All the nodes participate to solve a challenge posed by the network. Multiple miners can come up with a valid block and a valid solution to the challenge posed by the network. In case of multiple valid blocks mined the block with the heaviest chain is considered and the other blocks become the orphan blocks.

Ethereum smart contracts are written in the solidity programming language. EVM is a low-level Turing complete stack-based language operating on 256-bit words designed to be simple compared to general purpose. Solidity is a statically typed language with java-script like syntax.

Ethereum provides a flexible environment to develop Distributed Applications when compared to other blockchain platforms. The DApp developed in this project integrates the Web development technologies of html, css and java script integrated with the Ethereum Blockchain using the Meta Mask framework and Ganache local Ethereum Blockchain on the Ubuntu Platform.

2. Literature Survey

Blockchain innovation has pulled in gigantic consideration in both scholarly community and capital market. In any case, overpowering hypotheses on a large number of accessible digital forms of money and various beginning coin offering tricks have likewise welcomed infamous discussions on this rising innovation. This paper follows the advancement of blockchain frameworks to uncover the significance of decentralized applications (dApps) and the future estimation of blockchain. We study the cutting edge dApps and examine the bearing of blockchain advancement to satisfy the attractive attributes of dApps. The per users will increase an outline of dApp explore and get acquainted with ongoing improvements in the blockchain.

Blockchain innovations are pulling the strings of mega minds and mega brands, triggering a massive attention to the technology that is being implemented by a lot of ventures. Observations suggest that the financial industry is becoming an essential client for this technology. With the implementation of digital currency in the form of Bitcoin, it has also been observed that implementing the digital currency has hinderances like the gigantic cost base issue. On this, the budgetary emergency uncovered that even in money related administrations it won't be able to conceivably recognize the right present proprietor of an advantage. It is considerably even more an issue to remember possession over a more extended chain of changing purchasers in global financial exchange administrations: when, e.g., the US venture bank Bear Stearns flopped in 2008 and was totally procured by JP Morgan Chase, the quantity of offers offered to the acquirer was bigger than the offers extraordinary in the books of Bear Stearns. It was impractical to explain the book keeping blunders and JP Morgan Chase needed to bear the harm from overabundant (computerized) shares. While the issue of following back proprietorship in long exchange chains is as of now a basic viewpoint in budgetary markets, it is additionally significant for physical products, e.g., (blood) precious stones or broccoli. US retailer Wal-Mart with in excess of 260 million clients for every week is in scan for an innovation that assists with recognizing decisively those groups of vegetables that in a given case, e.g., are tainted by coliform microscopic organisms. Intermediation is the present overwhelming answer for confirming responsibility for and exchange handling. Mediators play out the cautious checking of each included gathering along a chain of mediators. Be that as it may, this can't tedious and expensive yet additionally bears a credit chance in the event that a mediator comes up short. The blockchain innovation vows to beat these basic angles, speaking to "a move from confiding in individuals to confiding in math" (Antonopoulos 2014) since human intercessions are no longer vital.

Ethereum, the notable blockchain stage, doesn't have any cutoff for square size, dissimilar to Bitcoin. Be that as it may, there are different snags in preparing boundless exchanges every second. Ethereum blockchain code runs by various customers, and they run on various speed and present the distinctive degree of execution. This paper contemplates Ethereum exchanges and it breaks down two most well-known Ethereum customers, Geth and Parity, on a private blockchain to acquire the better comprehension of the impact of various customers on Ethereum execution. The outcomes show that the exchanges are 89.8 percent on normal quicker in Parity customer in correlation with Geth customer, utilizing a similar framework arrangement.

Smart Contracts that development on blockchain advances is accepting extraordinary consideration in new business applications and mainstream researchers, since they permit untrusted gatherings to show contract terms in program code and in this manner wipe out the requirement for a confided in outsider. The creation procedure of composing admirably performing and secure agreements in Ethereum, which is the present most conspicuous shrewd agreement stage, is a troublesome errand. Research on this theme has as of late began in industry and science. In view of an examination of gathered information with Grounded Theory strategies, we have expounded a few basic security designs, which we portray in detail based on Solidity, the ruling programming language for Ethereum. The introduced designs depict answers for commonplace security issues and can be applied by Solidity engineers to moderate run of the mill assault situations.

New Cryptographic protocols which use the properties of public key cryptosystems to full advantage are now evolving. For public key distribution and for digital signatures several protocols are compared to each other for conventional alternatives.

To figure out how to utilize Solidity and the Ethereum venture second just to Bitcoin in showcase capitalization. Blockchain technology has caught the attention of the world on a large scale, moreover the Turing – complete scripting language Solidity that is paired with Ethereum has gained wide popularity for writing smart contracts. This book presents the blockchain wonder in setting; at that point arranges Ethereum in a world spearheaded by Bitcoin. See why experts and non-experts the same are sharpening their aptitudes in keen agreement designs and dispersed application improvement. You'll audit the basics of programming and systems administration, nearby first experience with the new order of crypto-financial matters. You'll at that point convey keen agreements of your own, and figure out how they can fill in as a back-end for JavaScript and HTML applications on the Web. Numerous Solidity instructional exercises out there today have a similar imperfection: they are composed for cutting edge JavaScript engineers who need to move their aptitudes to a blockchain situation. Acquainting Ethereum and Solidity is available with innovation experts and devotees all things considered. You’ll find energizing example code that can push ahead certifiable resources in both the scholarly and the corporate fields. Discover now why this book is an amazing door for imaginative technologists of numerous types, from idea to organization. What You’ll Learn See how Ethereum (and different cryptographic forms of money) work Compare appropriated applications (dapps) to web apps Write Ethereum keen agreements in Solidity Connect Ethereum brilliant agreements to your HTML/CSS/JavaScript web applications Deploy your own dapp, coin, and blockchain Work with essential and middle shrewd agreements Who This Book Is For Anyone who is interested about Ethereum or has some commonality with software engineering Product supervisors, CTOs, and experienced JavaScript developers Experts will locate the propelled test extends right now in light of the influence of Solidity

Ethereum, the notable blockchain stage, doesn't have any cutoff for square size, not at all like Bitcoin. Be that as it may, there are different impediments in handling boundless exchanges every second. Ethereum blockchain code runs by various customers, and they run on various speed and present the diverse degree of execution. This paper examines Ethereum exchanges and it breaks down two most well-known Ethereum customers, Geth and Parity, on a private blockchain to acquire the better comprehension of the impact of various customers on Ethereum execution. The outcomes show that the exchanges are 89.8 percent on normal quicker in Parity customer in correlation with Geth customer, utilizing a similar framework arrangement.

Ethereum, the notable blockchain stage, doesn't have any cutoff for square size, not at all like Bitcoin. Be that as it may, there are different impediments in handling boundless exchanges every second. Ethereum blockchain code runs by various customers, and they run on various speed and present the diverse degree of execution. This paper examines Ethereum exchanges and it breaks down two most well-known Ethereum customers, Geth and Parity, on a private blockchain to acquire the better comprehension of the impact of various customers on Ethereum execution. The outcomes show that the exchanges are 89.8 percent on normal quicker in Parity customer in correlation with Geth customer, utilizing a similar framework arrangement.

A smart contract is a PC program that can be accurately executed by a system of commonly doubting hubs, without the need of an outside confided in power. Since smart contracts handle and move resources of extensive worth, other than their right execution it is additionally pivotal that their usage is secure against assaults which target taking or altering the advantages. We study this issue in Ethereum, the most notable and utilized structure for keen agreements up until now. We break down the security vulnerabilities of Ethereum smart contracts, giving a scientific classification of regular programming entanglements which may prompt vulnerabilities. We show a progression of assaults which misuse these vulnerabilities, permitting an enemy to take cash or cause other harm.

3. Existing System

The existing task management systems pose an unreliability in tracking the performance of the person handling a task. It uses a centralized database, which means they are vulnerable to a single point of failure. Moreover, there is no proper and reliable storage of log entries, and even if they are able to store the logs, they are not tamper-proof. Any individual with a little knowledge on how to access the logs can easily tamper with the logs. Furthermore, improper log entry management practices are practiced in various domains that make auditing the performance of the employee difficult.

4. Proposed System

The proposed task management system handles all these issues by utilizing the power of blockchain platform using the Ethereum blockchain environment. All the users which may be employees, students or higher-grade officials will be the nodes of the network. Individual interfaces will be provided to all and proper smart contracts will be implemented according to the hierarchy. The person on the top of the hierarchy will be able to monitor the task of all. Manual verification is eliminated as the smart contracts are executed only if certain conditions are fulfilled. Based on the time taken by the user to complete the task, the smart contract will assign subsequent task to the user and notify it to the person on top of hierarchy. The logs will be stored on the blockchain thus making them tamper-proof. This also provides a more flexible way to audit the performance of the users. The problem of non-repudiation can also be solved as one cannot deny assigning a task and the user cannot deny that the task was assigned to them which is the case of many task management systems where there are issues where the users clear any logs related to assignment of task that cannot be traced back. This problem can be easily countered using the blockchain as the user will require a really powerful machine to tamper with the logs.

5. Design Process

Figure 1 depicts the architecture diagram of the system. The web app is deployed in the local environment using node.js. Ganache local Ethereum block chain system is used to create a local block chain and the meta mast frame work connects the block chain to the DApp using the private key of the useraccount. When an event occurs, in this case completion or addition of tasks, the smart contract is executed automatically and a transaction log is added to the block of the block chain. The log can be further referred to for its time stamp value to analyze the behavior of the staff regarding their interaction with the UI for a specified task and reward VM Ware Workstation PRO:

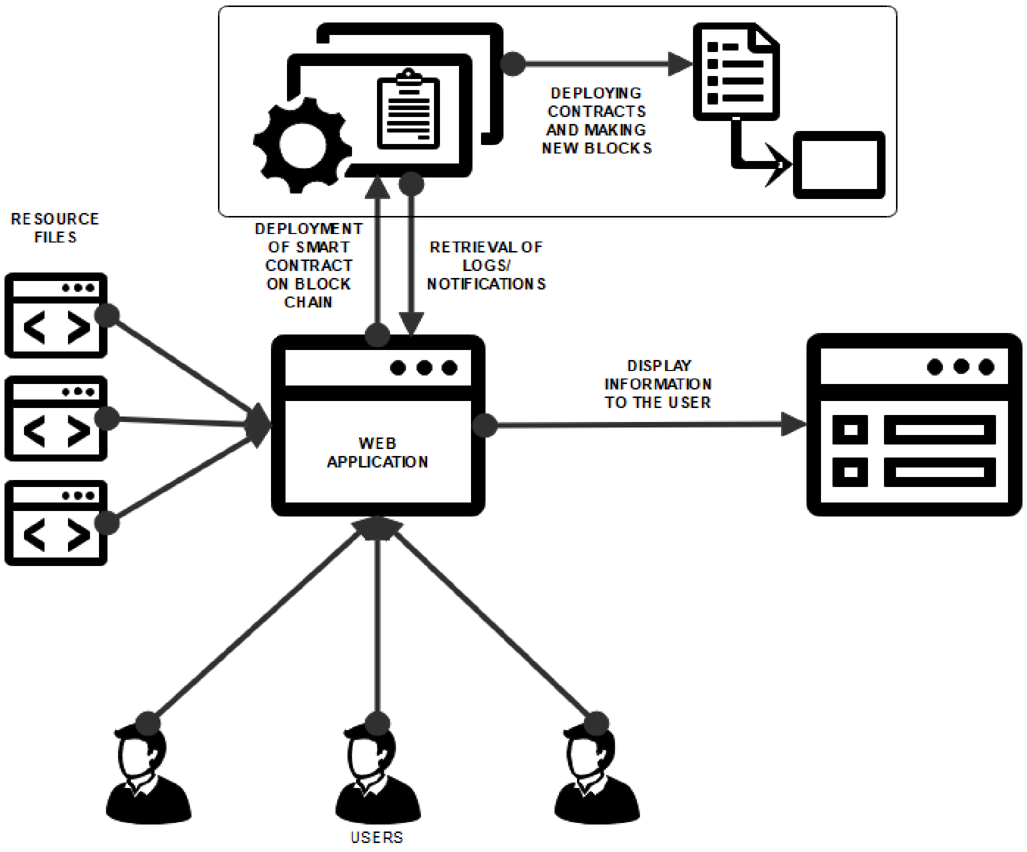

DEPLOYING CONTRACTS AND MAKING NEW BLOCKS
RESOURCE FILES
DEPLOYMENT OF SMART CONTRACT ON BLOCK CHAIN
RETRIEVAL OF LOGS/ NOTIFICATIONS
DISPLAY INFORMATION TO THE USER
WEB APPLICATION
USERS

Fig 1: Architecture Diagram

The entire project is run on the Ubuntu 18.04 virtual machine using the VW ware workstation PRO

Ubuntu
Provides safe and flexible environment to write and test block chain applications using the ganache local Ethereum block chain and truffle framework.

Ganache
A local Ethereum block chain environment that is preconfigured to be utilized for developing block chain applications, it provides accounts, private and public keys thus reducing the complexity to generate the accounts and keys using public key cryptographic algorithms

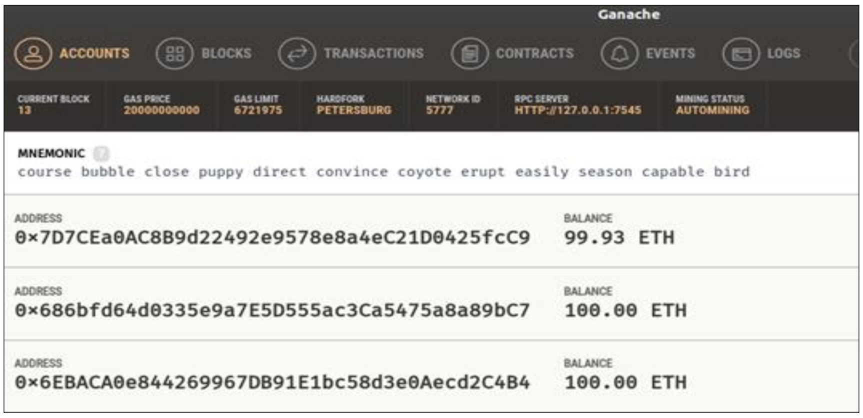
Firefox
Open-source browser used for deploying the project and running it locally.

Meta Mask
The meta mask FireFox extension provides a way to let the smart contracts communicate with the Ethereum block chain and use the local Ethereum wallet to perform transactions using Ethereum gas. It also helps the client-side application to communicate with the block chain, thus integrating the modules to the block chain.

6. Implementation
The client-side application is made in such a way that it communicates with the block chain application directly. Client - side application gets connected to the Ethereum block chain using the Ethereum node. The block chain also gets connected to the personal local Ethereum account and provides an Ethereum wallet.
The environment setup begins by installing the VM Ware workstation pro which will provide us with a virtual machine of Ubuntu 18.04 LTS for working with the block chain.
Download VM Ware Workstation
The virtual machine of the image that is supported with the system can be obtained from the official Linux website and turned on in the VM Ware workstation.
Download VM Ware image for Ubuntu 18.04 LTS.
Once the virtual machine is up using the terminal to provide necessary updates to the OS.
Next step is to install few dependencies for the project.
The Ganache Personal Block chain is local development block chain that mimics the behavior of public block chain. It allows the developers to deploy develop applications, deploy smart contracts and perform tests.
Download Ganache from.
100 Ether are added to each account. This saves a lot of time as building a personal block chain network from scratch and creating accounts manually will consume lot of resource and

time also, we need to credit each account with ether to do so. The next on the list of dependencies is the Node.JS, we need the version 8.x.x to work with our project.
Download node
Truffle framework can be downloaded using the command "$ npm install -g truffle@5.0.2"
Truffle framework is a suite of tools for writing Ethereum smart contracts in Solidity programming language. Truffle helps in smart contract management, automated testing, deployment and migrations, network management, development console, script runner and client-side development.
Metamask extension is the final dependency that we require to setup our environment. Most of the major web browsers currently do not support block chain, this extension allows the user to connect to the local block chain.
Create the project directory, the project implemented uses the home directory and the contents are placed in the directory called "Project".
Get inside the directory and run the command "$ truffle init" it will initialize a new truffle project.
Create a file "$ touch package.json" we create this file to list out some development dependencies.
Next, we run the command "$ npm install" this will install the necessary dependencies that were mentioned in the package. json file.
Various directories that get created for our project are contracts directory, migrations directory, node_modules directory, test directory and truffle – config.js file. The contracts directory contains all the contracts file. The migrations of the block chain are handled by special migration contracts.
The migrations files are present are in the migrations directory. The migrations are required to change the state of a database. Every time we deploy a smart contract on the block chain, we are updating the state of the block chain and hence we need the migrations
The node_ modules directory holds all the node dependencies installed.
Test directory is where all the test is written for the smart contracts
The main configuration file for our project is the truffle – config.js file.
Create a smart contract in the solidity format using the command "$ touch ./contracts/Project.sol" All the code of the smart contract is inside this file. All the state variables needed for the project are present here. After completing the writing of the smart contract use the command "$ truffle compile" to compile the smart contract to ensure no errors are present in the code. Once we compile the solidity file a file by the name "Project.json" will be created in " ./build/ contracts/ Project. json". It is called ABI file, which stands for "Abstract Binary Interface". The byteconde version of the smart contract code is present in it that runs on the Ethereum Virtual Machine or the Ethereum node. The smart contract functions are represented in JSON to be executed by external client – side Javascript applications.
Now to connect the application to the block chain we need to create migrations file. It will deploy the smart contract to the personal block chain after the migrations script are executed. We have to edit the truffle – config,js file to make it recognize the host IP and port so that it connects to the local Ganache block chain.
Use this command "$ touch migrations/2_ deploy_ contracts. js" to create migrations file We are numbering the files so that truffle recognizes which contract to deploy and the order or deployment of the same.
Use the command "$ truffle migrate" to deploy the smart contracts or change the state of the block chain. Notice, it will consume some ether or gas from your local account.
Solidity language provided by the truffle framework v.5.0 is used to write the smart contracts that will be automatically executed for a certain event and log the transaction in the block chain for the specific account. HTML, CSS and Javascript are used to develop the UI. Ganache is used to import local account and with the help of meta mask framework the account is connected to the browser and the DApp. Once the DApp is up and running events triggered are automatically logged, in this case, addition of tasks and completion of tasks are the events.

7. Observations
Once the Distributed application is connected to the local Ethereum block chain with the help of Ganache and Meta Mask, with every interaction of the user with the DApp the transactions are logged in the block chain and can be referred to for further analysis. The logs are completely tampering proof as they utilize the block chain technology.
Fig. 2 shows the Ganache interface, it includes the user address and the balance ether the user has. Each time the user interacts with the block chain and makes a transaction a certain amount of ether is deducted from the account balance to log that transaction on the block chain. This can be a vital step as each employee can be given a certain amount of ether based on their roles in the industry and they can be allocated with extra benefits like bonus and holidays based on the amount of ether they possess at the end of the deadline of the task. The amount of ether left in their account is sign of how many times they had to perform the task to reach to the requirements of the same. More ether remaining signifies that the user performed the task more efficiently within the given deadline.
Fig. 3 shows the confirmed transactions pop – up that indicates if the transactions failed or were successfully added to the block chain. In case of any network failure or technical issues the transactions will not be confirmed, however, the failed transaction will also be logged in the block chain, in this way it can also prevent non repudiation and the truthfulness of the user can also be judged, if they really interacted with the DApp for completing the task.
Fig. 4 shows the ether scan that displays all the logs including confirmed and failed transactions during the interaction with the DApp.

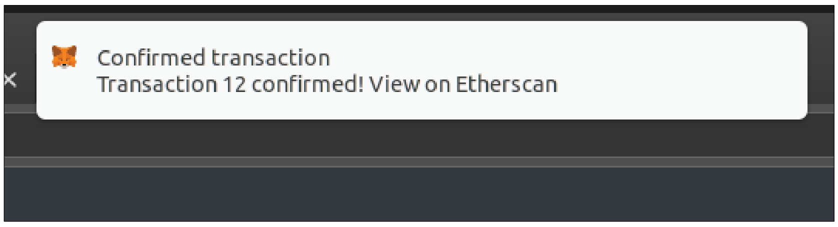


CURRENT BLOCK 13 | GAS PRICE 20000000000 | GAS LIMIT 6721975 | HARDFORK PETERSBURG | NETWORK ID 5777 | RPC SERVER HTTP://127.0.0.1:7545 | MINING STATUS AUTOMINING

MNEMONIC
course bubble close puppy direct convince coyote erupt easily season capable bird

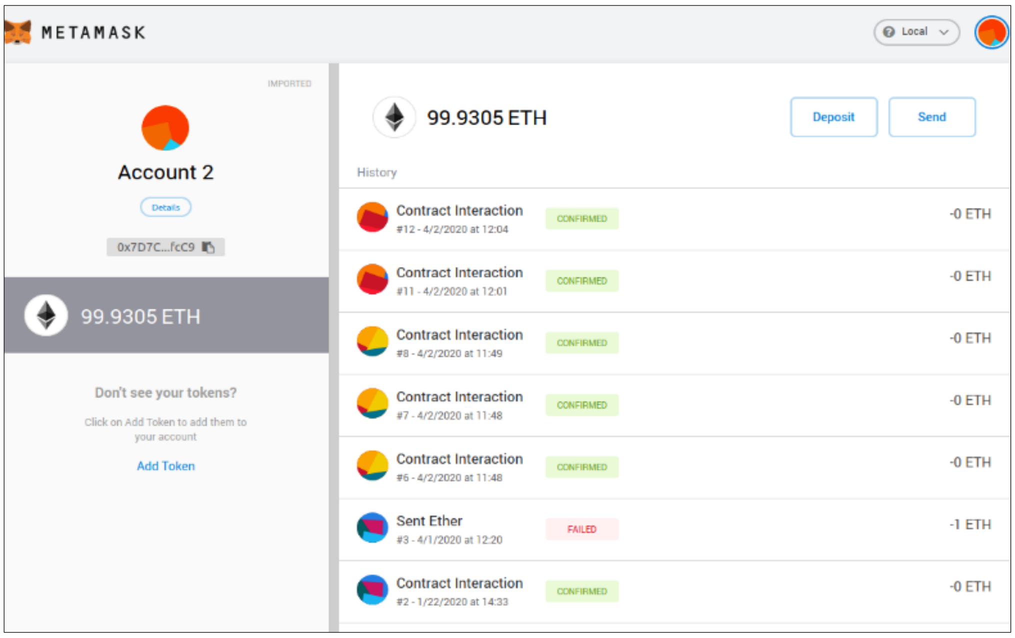
ADDRESS	BALANCE
0×7D7CEa0AC8B9d22492e9578e8a4eC21D0425fcC9	99.93 ETH
0×686bfd64d0335e9a7E5D555ac3Ca5475a8a89bC7	100.00 ETH
0×6EBACA0e844269967DB91E1bc58d3e0Aecd2C4B4	100.00 ETH

Fig 2: Ganache Local Block Chain



Confirmed transaction
Transaction 12 confirmed! View on Etherscan

Fig 3: Confirmation Dialogue



Account 2
0x7D7C...fcC9
99.9305 ETH

Don't see your tokens?
Click on Add Token to add them to your account

99.9305 ETH

History

Contract Interaction #12 - 4/2/2020 at 12:04	CONFIRMED	-0 ETH
Contract Interaction #11 - 4/2/2020 at 12:01	CONFIRMED	-0 ETH
Contract Interaction #8 - 4/2/2020 at 11:49	CONFIRMED	-0 ETH
Contract Interaction #7 - 4/2/2020 at 11:48	CONFIRMED	-0 ETH
Contract Interaction #6 - 4/2/2020 at 11:48	CONFIRMED	-0 ETH
Sent Ether #3 - 4/1/2020 at 12:20	FAILED	-1 ETH
Contract Interaction #2 - 1/22/2020 at 14:33	CONFIRMED	-0 ETH

Fig 4: Ether Scan showing user interaction with the DApp

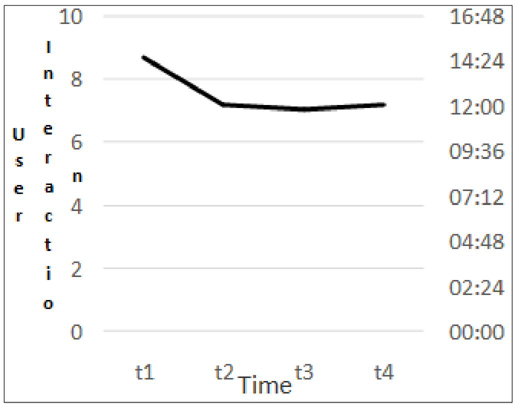


Fig 5: Show the Number of Times the User interacted with respect to time

8. Conclusion

The implemented project can serve as a good model to provide a tamper proof environment for storing logs of users work on the block chain. More precise research can be done on the behavior of users to a certain task of work by analyzing the logs of completed, pending or missed tasks. This can help in providing ways to improve human efficiency in work place. The UI can be further enhanced to utilize the functions of public and private block chain distinctively.

It can utilize more sophisticated databases like the Side DB and Couch DB that provide a flexible database support to the block chain implementation.

Also, from fig. 5 we can get the data of how many times the user interacted with the UI for a particular task. The graph can be plotted using the data stored in ether scan in fig. 4. It can act as a vital source of information for analyzing the performance of the user.

9. Challenges

Block chain is still in development phase and fully functional product ready applications still need more resource and planning to be implemented using the technology. Moreover, to ensure the integrity of the information stored in the blocks, miners have to be employed and rewarded which will add to the overall cost of maintenance and development.

It will require systems with high end configuration to deploy the applications on the industrial level.

Cryptocurrency is still not accepted in many parts of the world, which is an essential element to run the project using this technology.

10. References

1. Cai W, Wang Z, Ernst JB, Hong Z, Feng C, Leung VCM. Decentralized Applications The Blockchain-Empowered Software System. In IEEE Access. 2018; 6:53019-53033.doi: 10.1109/ACCESS.2018.2870644
2. Nofer M, Gomber P, Hinz O, Schiereck D. Blockchain,' Bus. Inf.Syst. Eng. 2017; 59(3):183187.
3. Rouhani S, Deters R. Performance analysis of Ethereum transactions in private blockchain, in Proc. 8th IEEE Int. Conf. Softw. Eng. ServiceSci. (ICSESS), 2017, 7074.
4. Wohrer M, Zdun U. Smart contracts: Security patterns in the Ethereum ecosystem and solidity, in Proc. Int. Workshop Blockchain Oriented Softw. Eng. (IWBOSE), 2018, 28.
5. Merkle RC. Protocols for public key cryptosystems, in Proc. IEEESymp. Secur. Privacy, 1980, 122.
6. Dannen C, Introducing Ethereum and Solidity: Foundations of Cryptocurrency and Blockchain Programming for Beginners. New York, NY, USA:Apress, 2017.
7. Rouhani S, Deters R. Performance analysis of Ethereum transactionsin private blockchain, in Proc. 8th IEEE Int. Conf. Softw. Eng. ServicESci. (ICSESS), 2017, 70-74.
8. Wohrer M, Zdun U. Smart contracts: Security patterns in theEthereum ecosystem and solidity, in Proc. Int. Workshop Blockchain Oriented Softw. Eng. (IWBOSE), 2018, 2-8.
9. Atzei N, Bartoletti M, Cimoli T. A survey of attacks onEthereum smart contracts (SoK), in Proc. 6th Int. Conf. PrincSecur. Trust, vol. 10204. New York, NY, USA: Springer-Verlag, 2017, 164-186.
10. Merkle RC. Protocols for public key cryptosystems, in Proc. IEEE Symp. Secur. Privacy, 1980, 122.
11. Nicola Atzei, Massimo Bartoletti, and TizianaCimoli. A survey of attackson Ethereum smart contracts (sok). In POST, volume 10204of Lecture Notes in Computer Science, pages 164{186. Springer, 2017.
12. Rouhani S, Deters R. Performance analysis of Ethereum transactions in private blockchain, in Proc. 8th IEEE Int. Conf. Softw. Eng. ServiceSci. (ICSESS), 2017, 70-74.
13. Dannen C. Introducing Ethereum and Solidity: Foundations of Cryptocurrency and Blockchain Programming for Beginners. New York, NY, USA: Apress, 2017.
14. Christidis K, Devetsikiotis M, Blockchains and smart contracts forthe Internet of Things, IEEE Access. 2016; 4:2292-2303.
15. Natoli C, Gramoli V. The blockchain anomaly, in Proc. IEEE 15th Int. Symp. Netw. Comput. Appl. (NCA), Oct./Nov, 2016, 310-317.