# RENT BOOKS ON BLOCKCHAIN

## Decentralized Marketplace

Version 13

11/25/2020

**-Nabin Bhatta**

# CONTENTS

# ABSTRACT

The main plan of the project is to make the existing system more efficient and less expensive using the block chain technology. The current transaction is done in such a way it required many middle people to complete the transaction. Especially, when the transaction is done outside of the national boarder. My main idea of project is to make a smart contract to rent/sell a book which can be done by any individual with the help of the application. In the current situation, the renter takes some money as a deposit from the buyer and returns on completion of contract. This process is little expensive. So, a smart contract can be used to verify the contract between customer and any individual who wants to rent the book. There are lots of books which are not being recycled properly. If all the individual can rent or sell their books, it would be much easier for many other students since it would be cheaper that that provided by the other stores.

Blockchain is one of the prospective technologies that can change many things soon. The important and main thing is that the transactions can happen without any middle person or intermediaries. The efficient thing is that it is fully autonomous where most of the operations is automatically by the rules and policies programmed by the person. These are smart contract which is renewed by the smart contracts that is visible for all the users on the blockchain. Due to smart contracts there is no need of any thrusted third party. This means it is present in all the nodes.

Apart from the immense advantages of using blockchain over the conventional method of using transaction and other contracts agreement, blockchain has some controversies of using the smart contract which is visible to all the people within the network. Also, blockchain has a high latency for the time passes for each verified block of transactions to be added to the ledger. It can provide the universal ledger for trading around the globe. Its attribute is immensely fruitful. It is very transparent, immutable, and very secure which is driven by the consensus of people all over the distributed system. In the current circumstances, there are various areas like transportation where there are many middleperson/intermediaries when doing the transaction which takes a lot more time and capital to complete the task.

# PROBLEM DEFINITION

The main problem of including the third party in a system is that one must share the user information to a certain extent. Although there is a privacy to the data, it is better not to share the information to many third parties. Along with that, the transaction between two parties should have consensus which requires the check among them to avoid any fraud. The contract between the two parties should be recorded. This is required to be done in case there is a conflict regarding the transaction. Every process, rules and regulation are monitored. In this case, due date to return book. The process of including all the functionality is expensive.

With the blockchain technology, the important and main thing is that the transactions can happen without any middle person or intermediaries. The efficient thing is that it is fully autonomous where most of the operations is automatically by the rules and policies programmed by the person.

# SOLUTIONS

- The **immutability** on the blockchain is one of the main attributes to use the blockchain in the book renting application as the information cannot be changed so it remains **unaltered**.
- The proof of work (**POW**) ensures that the miners (the data processors) cannot lie about the transaction.
- The blockchain ledger is **distributed** in multiple sites which bring consensus from multiple people.
- The main attribute of blockchain over server/client is its **auditability,** as everyone can view the work and it is transparent to the users.
- The transaction processing part by a **third party is removed**. For example: there will be no need for middle-entity like VISA.
- The administration to regulate rules and regulation when renting and returning book is **replaced with a smart contract**.
- In the book renting application, the agreement between buyer and seller is **recorded.** This **prevents false allegation or fraud** among two parties.
- The transaction is recorded in the ledger which is validated and authenticated which is **secured by cryptography.**

# LITERATURE SURVEY

There is a work done on French airline (AXA) where they are using smart contract. They will give full details of delay of flight after two hours of delay and even the compensation to get the money back. The cons are that they may not pay full money back. [1]

“PolySwarm” is one of the first decentralized threat intelligence that runs on the logic behind smart contract. This encourages more advancement in cybersecurity with effective threat protection. “ChainLink” is also one of the decentralized networks of nodes that has partnered with Google to provide the required information from non-blockchain sources to the blockchain through smart contract via oracle. [1]

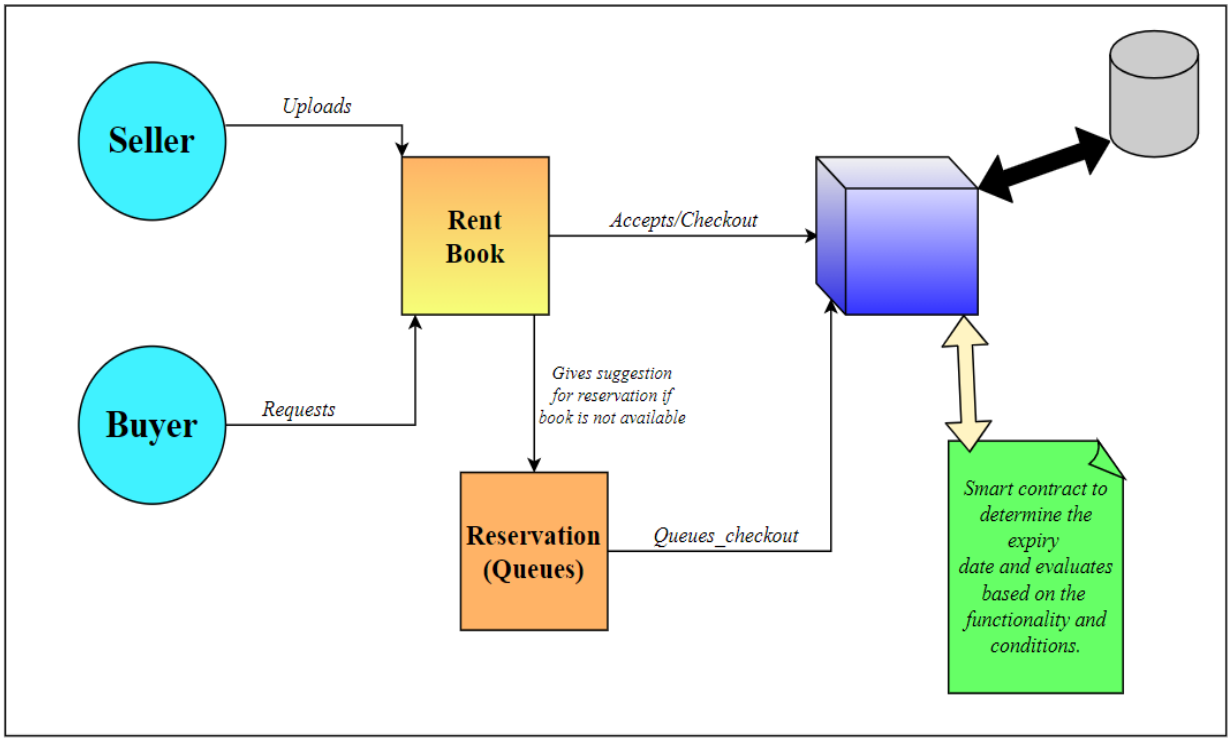
“Propy” is one of the first to make a real state selling overseas. They can even hold the property with some amount of money provided. So, if the seller refused to sell then the buyer can get their total money back since it is listed on the blockchain. So, my project to rent or sell books by individuals so that they can have their transaction recorded in ledger. So, if somebody refused to give them their book, the smart contract will give the money back. Similarly, “Populous” is also one the company that is that make easier for “buy and sell outstanding invoices on the blockchain using smart contracts”. This has various advantages from speed, cost, and reduction of errors.[2]

“Dai (DAI) is a decentralized stable coin running on Ethereum (ETH) that attempts to maintain a value of $1.00 USD. Unlike centralized stable coins, Dai is not backed by US dollars in a bank account. Instead, it is backed by collateral on the Maker platform.” [3] This may be very helpful while making a dapp which can be used for the transfer of money from one to another user.

There is also an idea of Ethereum Alarm clock which is used to set the timer of the event of the contract and as soon as the given timestamp is executed. A third party usually gives a day token which can be used by the user to set the timer to trigger the event. It is still difficult to use “sent later” button in MyEtherWallet [4] and use it on solidity without confusion.

# ARCHITECTURAL BLOCK DIAGRAM

This gives the overview of the decentralized program.

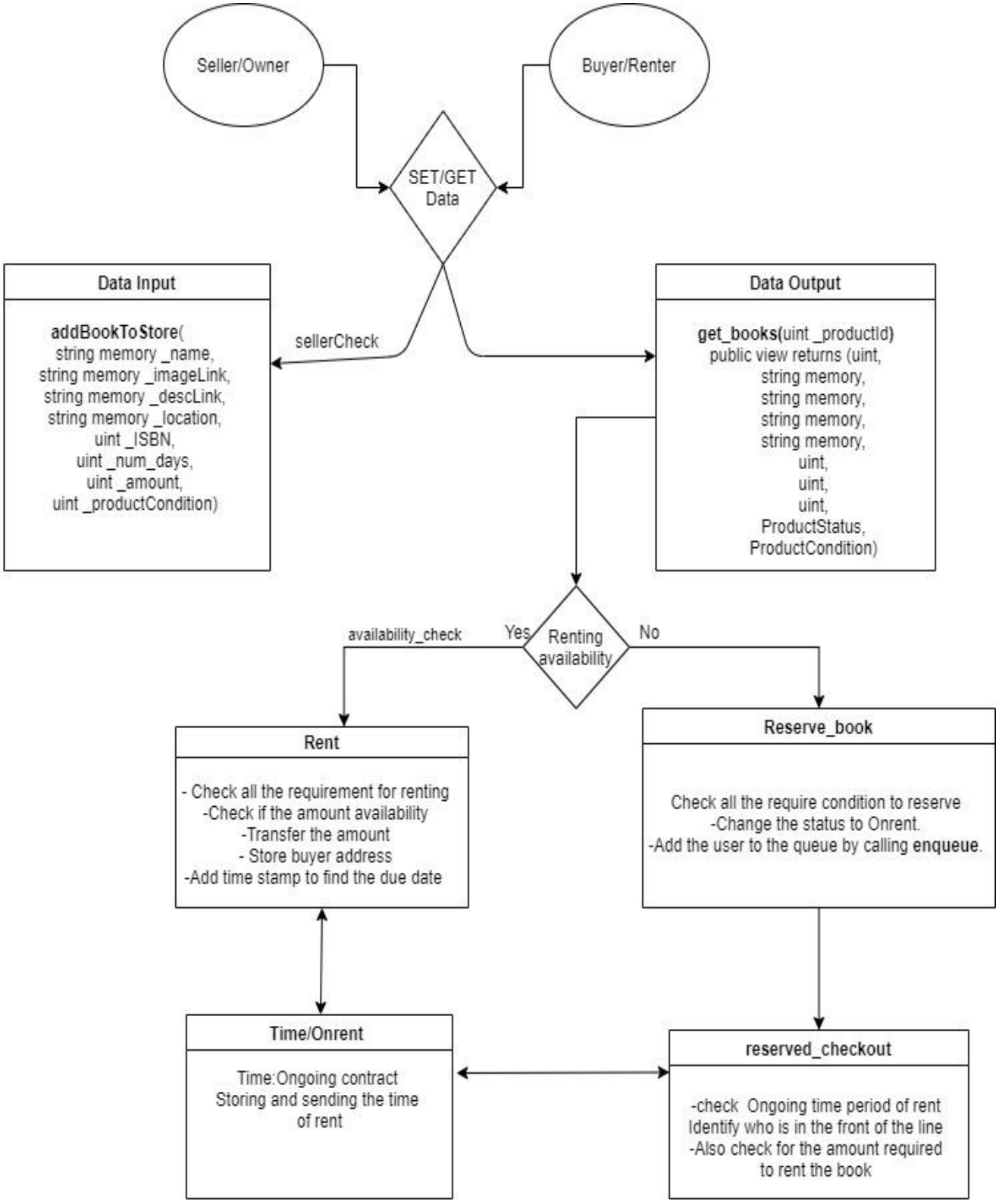



- When the seller uploads the information in blockchain the information is recorded in a block.

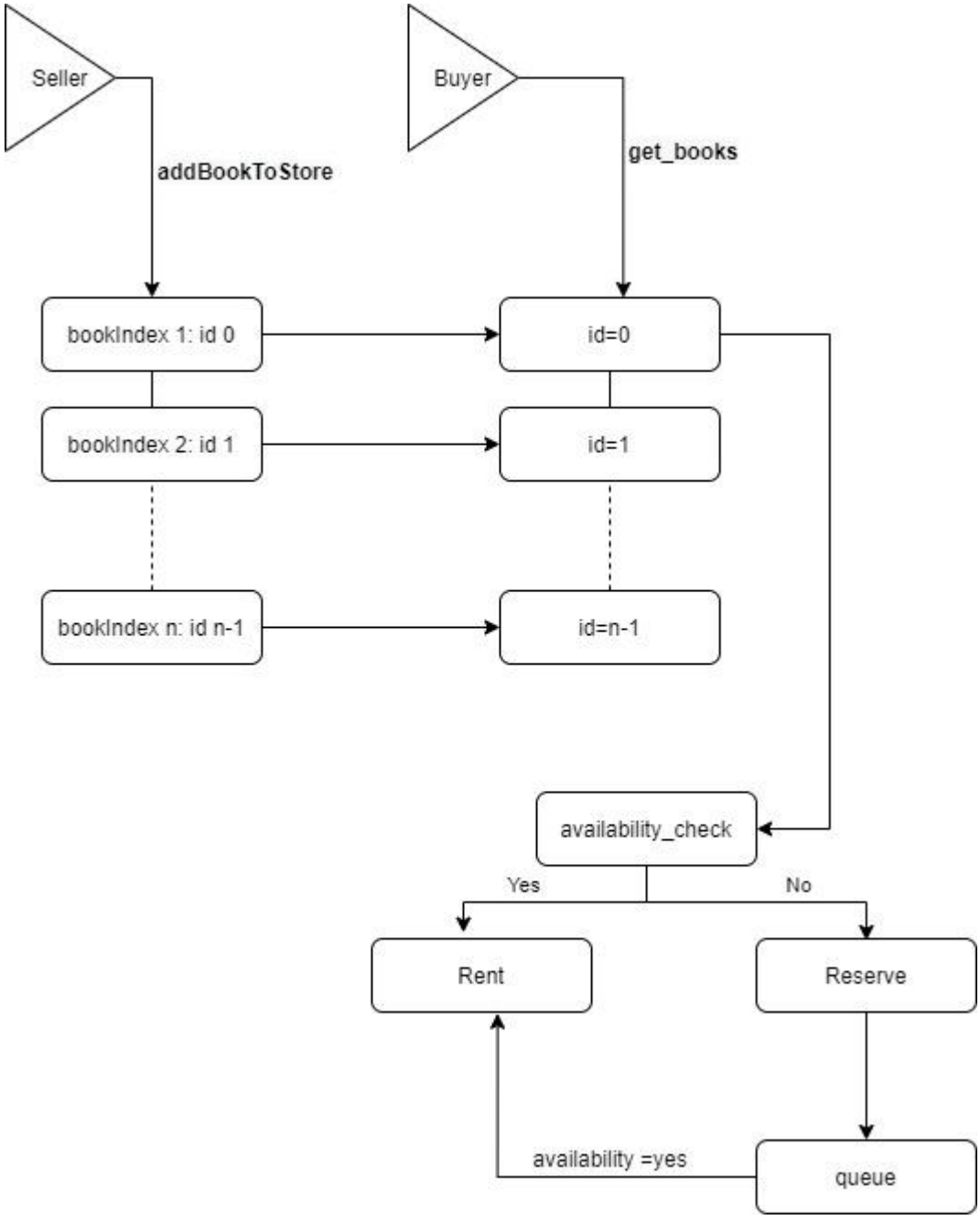
- The smart contract gets triggered when the buyer rents the book, and the transaction is deployed.

- After the given period (number of days), the smart contract checks the users are in queue and places the position based on FIFO (First in First Out).

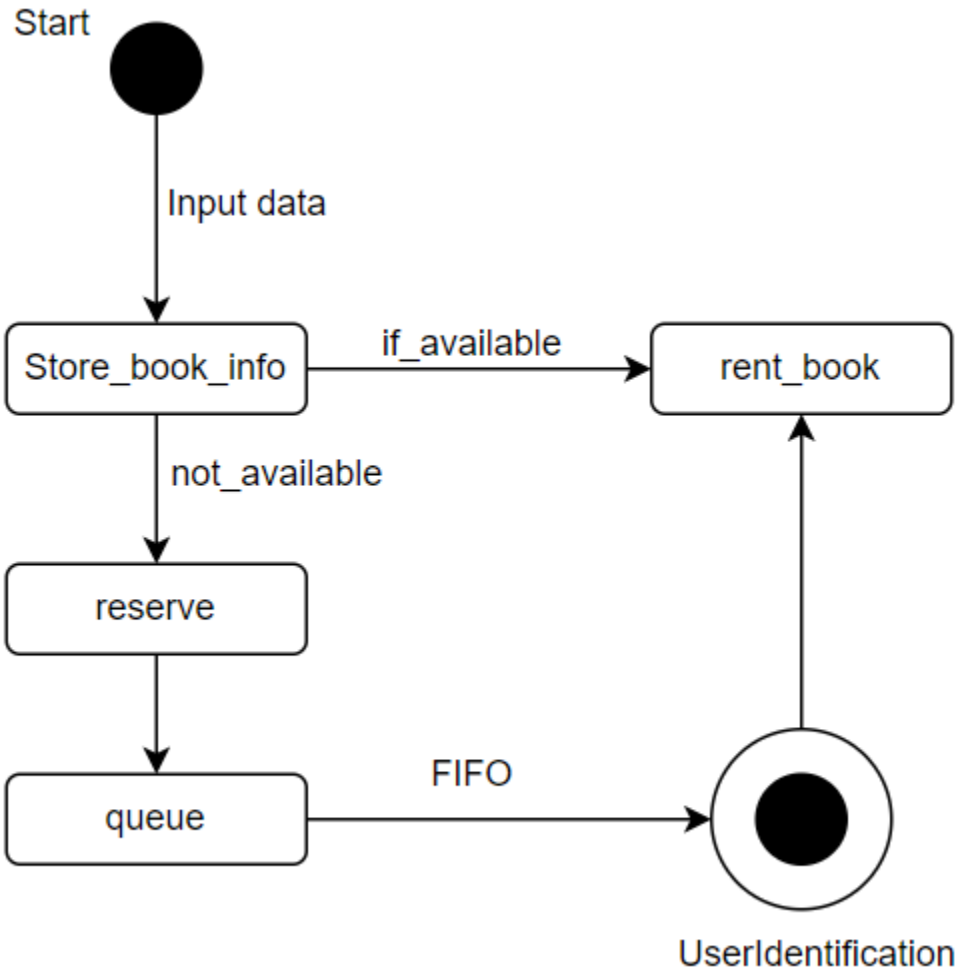
# SOFTWARE ARCHITECTURE



# EVENT FLOW

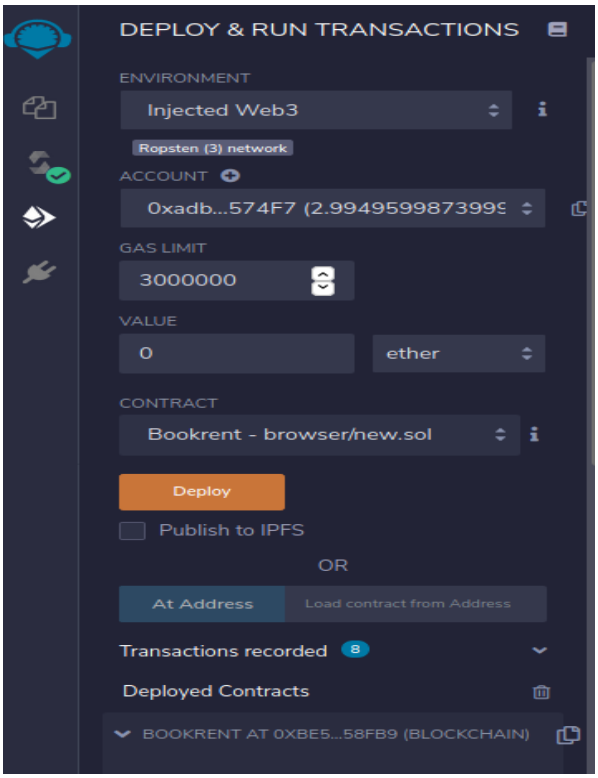

Seller
Buyer
addBookToStore
get_books
bookIndex 1: id 0
bookIndex 2: id 1
bookIndex n: id n-1
id=0
id=1
id=n-1
availability_check
Yes
No
Rent
Reserve
availability =yes
queue

## STATE DIAGRAM



## EXPERIMENTAL SETUP

The datasets of decentralized book rent are completely taken from user input. There will be 7 user input where the seller will upload their book information to the database. This will include name of the book, location, a 13-digit ISBN, amount for the book, and condition of the book. It also includes the image link and description link of the book. It also has the number of days that is allowed to rent for anyone per time period. **For the time sake, the number of days was set to number of seconds.**




The main test was carried out in Ropsten test network. The fake ether was used for the experiment. The fake ethers were retrieved from ether faucet. The main source code was in remix Ethereum IDE.

For this experiment, ten account was setup where 1 ether was in every account and the transaction was done between the seller and renter. It was done in remix IDE where the environment was set to injected web3. This way the remix IDE can easily interact with account with the help of the browser extension called Metamask.

# METRICS:

The metrics were taken based on the decentralized renting application and its functionality on Ethereum based platform. They were tested on Ropsten network.

## [Book Rent] Deployment Transaction Vs Time: -

- This involves the deployment of contract with different account at different time.

## User's data Vs Time: -

- This is to test the size of book information, location of the buyer with respect to the gas consumed.

## User's information (size) Vs Gas consumed (Gas): -

- This is to test the size of book information, location of the buyer with respect to the gas consumed.

## Number of Users in queue Vs Gas consumed (Gas): -

- This is to test the various gas consumed based on the number of users in queues.

## Users in queue Vs Time: -

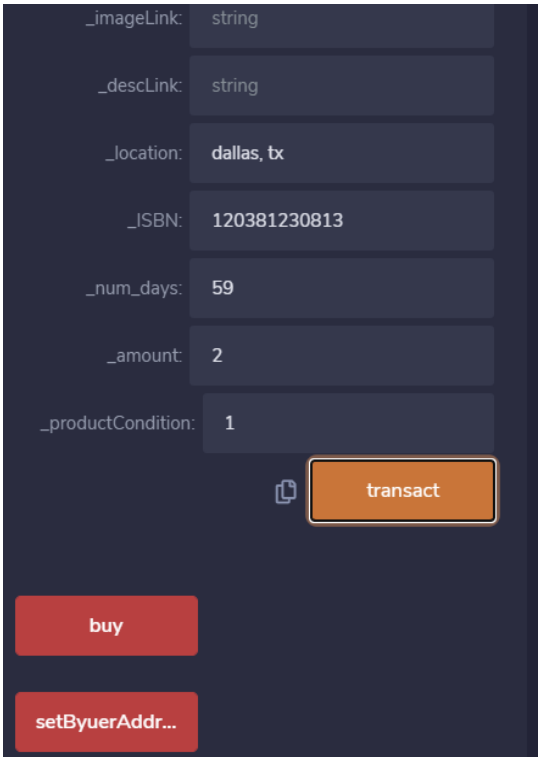
- This test involves the number of transactions done to reserve the book with respect with time. The effect of increase in reservations and time taken to complete the process.

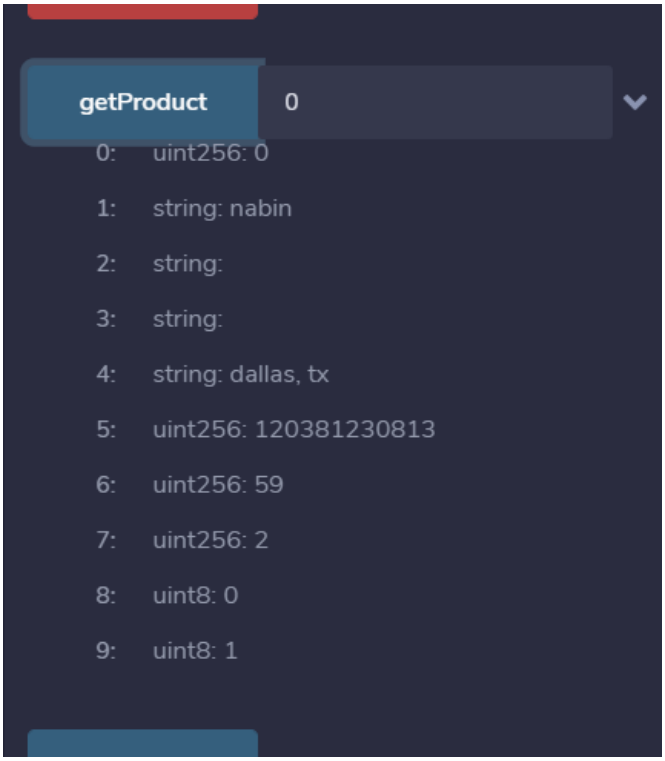
# TEST PLAN

My whole test plan is on remix Ethereum IDE. This provides the environment required to test the metrics on the injected web3. This will link to the Metamask account that enables the test on Ropsten network. First the required source code is uploaded to the remix IDE. Then, the Metamask (a browser extension) was opened where there were 10 accounts with each having one test ether.

Secondly, the environment in remix IDE was changed to injected web3. This way the current active account was linked to remix. In my renting application, I have a seller and renter. The contract is always deployed by the owner and the book information was uploaded by the user. This information will be stored in index depending upon the number of books he wants to upload. Then the renter can rent the book as the status was stated to be available (Open). As soon as the first person rents the book the status will be set to "Onrent".

In case no one has rented after the first rent period, the book modifiers will look for the queue and allow another user to rent the book.



## RESULTS:

The various test results based on metrics was tested and the following results were depicted and analyzed.

### [Book Rent] Deployment Transaction Vs Time: -

| Accounts | Time |
|---|---|
| Account 1 | 51 |
| Account 2 | 37 |
| Account 3 | 35 |
| Account 4 | 33 |
| Account 5 | 44 |
| Account 6 | 50 |
| Account 7 | 32 |
| Account 8 | 25 |
| Account 9 | 36 |
| Account 10 | 26 |

Averahe gas price:214292

The transaction based on the network was tested with 10 different accounts on Ropsten network. This test involves the number of transactions done to rent the book with respect to time. The effect of increase in transactions and time taken to complete the process was analyzed and there was no relationship of different account renting the book with time. This means any users any different probability of completing the transaction.

## User's data Vs Time: -

Higher the account, more information(text) was included in the test.

| Accounts | Time(In seconds) |
|---|---|
| Account 1 | 4 |
| Account 2 | 5 |
| Account 3 | 12 |
| Account 4 | 15 |
| Account 5 | 11 |
| Account 6 | 14 |
| Account 7 | 15 |
| Account 8 | 14 |
| Account 9 | 12 |
| Account 10 | 10 |

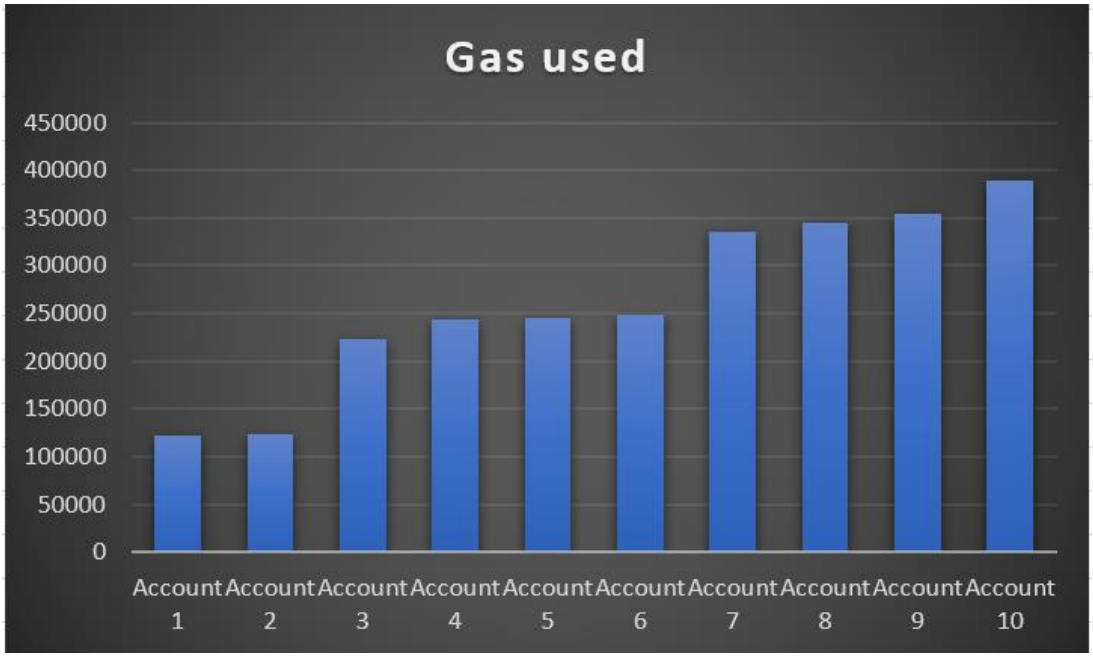
♦0.000465

The average gas price was 287234. Higher the number of the account, more data was added in the blockchain. With the increase in information, there was more time taken to complete the transaction. However, the data and time taken were not always proportional. More input data cost more gas and ether, but the time was highly depending on time.

## User's information (size) Vs Gas consumed (Gas): -

- This is to test the size of book information, location of the buyer with respect to the gas consumed.

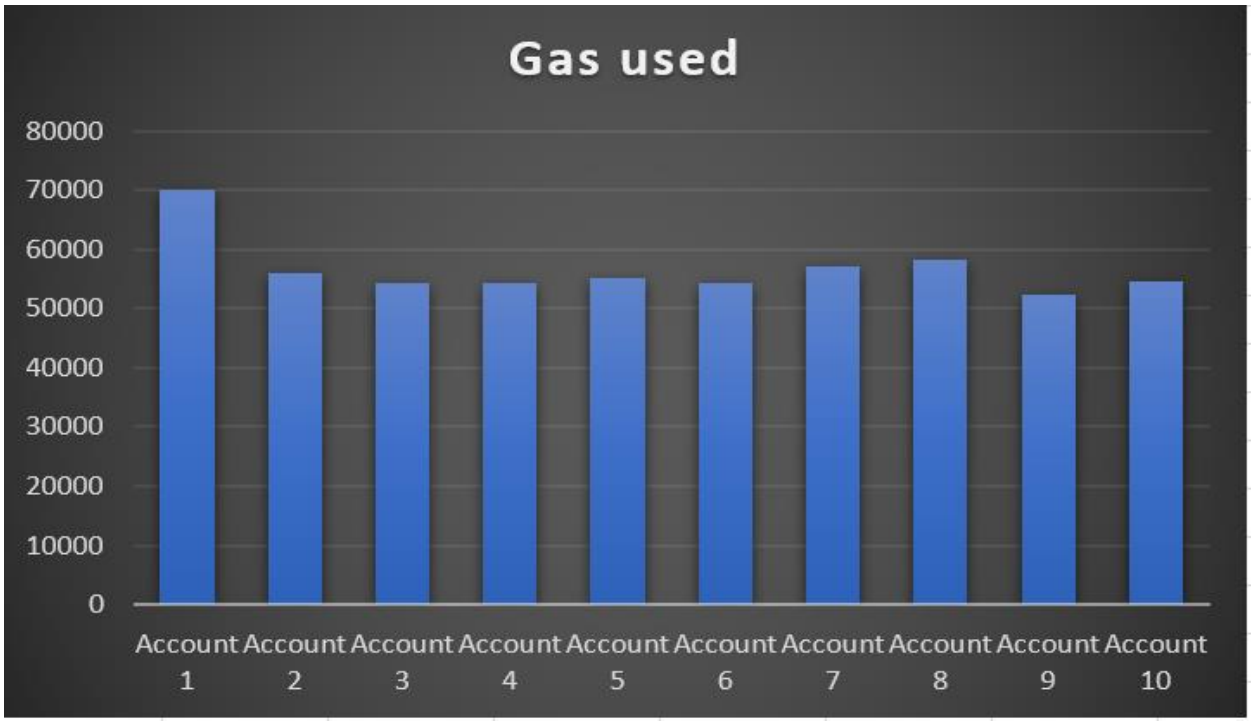
| Accounts | Gas used |
|---|---|
| Account 1 | 122432 |
| Account 2 | 123545 |
| Account 3 | 223434 |
| Account 4 | 243545 |
| Account 5 | 245453 |
| Account 6 | 248565 |
| Account 7 | 335434 |
| Account 8 | 345454 |
| Account 9 | 354544 |
| Account 10 | 388944 |



From the test result, it was very clear that the more the amount of information added to the blockchain, more the gas was consumed. Very little to no information or input data was of low cost. The bar-graph depicts the amount of information and gas price shown above. Higher the index of the account, more information was added in the user input to know the gas price of each transaction.

## Adding Users in queue Vs Gas consumed (Gas): -

| Accounts | Gas used |
|---|---|
| Account 1 | 70000 |
| Account 2 | 56032 |
| Account 3 | 54343 |
| Account 4 | 54232 |
| Account 5 | 55233 |
| Account 6 | 54231 |
| Account 7 | 57232 |
| Account 8 | 58232 |
| Account 9 | 52434 |
| Account 10 | 54545 |



Ten different account were tested with the functionality queue in the blockchain. The first account in the queue was more costly. It took more gas used that that of other accounts. The first account around 70000 gas while the second took around 55000 gas. We can clearly analyze the data from the above bar-graph based on network it took relatively less gas price when the second person was on queue.

## Users in queue Vs Time: -

- This test involves the number of transactions done to reserve the book with respect with time. The effect of increase in reservations and time taken to complete the process.

| Accounts | Time(In sec) |
|---|---|
| Account 1 | 22 |
| Account 2 | 24 |
| Account 3 | 25 |
| Account 4 | 20 |
| Account 5 | 17 |
| Account 6 | 14 |
| Account 7 | 16 |
| Account 8 | 18 |
| Account 9 | 26 |
| Account 10 | 24 |


This result is the second part of involves the same gas price as it is a part of number of users in reservation and gas price. As we analyzed in the previous metrics, there was decline in gas use from the first user to second user and the rest of the users were around the gas price of second users.

However, the time of execution of reserve function was not related to gas price but rather on network so there was variance in time for reservation by various accounts.

# CONCLUSIONS AND FUTURE WORK

From the above experiments and test results, the execution time and gas price are independent of each other. For the same gas price, there might be different time of execution. Gas price was more dependent on function and the amount of information added to the blockchain. As more amount of information was added to the blockchain, There was more gas price during the deployment of contract by a large amount as compared to adding data in the blockchain. For the same contract, the gas price was constant at any time in a network. Adding the data or the information of the book (name, image link, description link, ISBN, location, number of days and status of the product) was less costly than that of deploying the contract but more costly than that of using the rent, reserve and reserve checkout functionality.

The time execution was more dependent on network. Sometime the all the transaction was completed fast while sometime the time would increase by three folds. For the result, it was clear that the gas consumption and time was independent when renting and reserving the book from the blockchain.

The **book rent smart contracts** fulfills various functionality of the blockchain like **amount transfer, time stamp, and adding users in the queue**. There are various modifiers to check and balance among the user to rent and reserve the book. The modifiers also check the dequeue process allowing the user in the first line to rent the book before others.

Although all the functionality works properly, there is a **place for improvement**. This rent smart contract contains timestamp which adds the current time and the number of days allowed to rent. However, a user must go to the reserve checkout function and complete by themselves. There is not a functionality in solidity to automatically. There is an Ethereum Alarm Clock which does the automatic transition. This could be done using the third-party day token. In this rent contract, if there is an alarm clock to trigger the event then, after one contract is over, the book can be transferred to next user in the line.

# REFERENCES

[1]https://medium.com/polyswarm/5-companies-already-brilliantly-using-smart-contracts-ac49f3d5c431

[2]https://www.stateofthedapps.com/stats

[3]https://www.coinbase.com/price/dai

[4]https://blog.chronologic.network/schedule-your-transaction-now-using-mycrypto-and-myetherwallet-17b48166b412