LNMIIT
The LNM Institute of
Information Technology

BTECH PROJECT REPORT (SEM-VI 2020)
Course Instructor: Dr. Abhishek Sharma

Auction Bidding DApp

(A decentralised application in blockchain)

Submitted by:

Taruna Agrawal (17UCS170)

Hitesha Sharma (17UEC056)

ACKNOWLEDGEMENT

We have taken efforts in this project with our full dedication. Although, it would have been very difficult without your kind support and helping hand of many individuals and organizations. We would like to thank all of them for making our project prominent and well finished.

We are highly grateful to our instructor Dr. Abhishek Sharma for his guidance and for providing necessary information regarding the project, also for his support in completing the project.

We would like to express our gratitude towards our friends and teachers for their kind assistance and reassurance which helped us in the completion of this project within the limited time frame.

Our thanks and appreciations also go to our fellow mates in developing the project and people who have willingly helped us out with their abilities.

TABLE OF CONTENTS

1. Overview of our Auction Bidding application

The whole of us have known about Auction, a spot where we sell and purchase things by making bids. In spite of the fact that it is advantageous, it costs merchants about 10% of their income to pay the administration charge to the auction's overseeing organization.

On the off chance that the organization itself swindles a few exchanges and gets the cash, in what capacity will you realize that you have been cheated? Therefore, a decentralized arrangement is an ideal arrangement.

OUR AIM: We are building a decentralised bidding App having 4 objects from which we can choose to bid upon and all the transactions how it works using frontend, backend and metamask are shown.

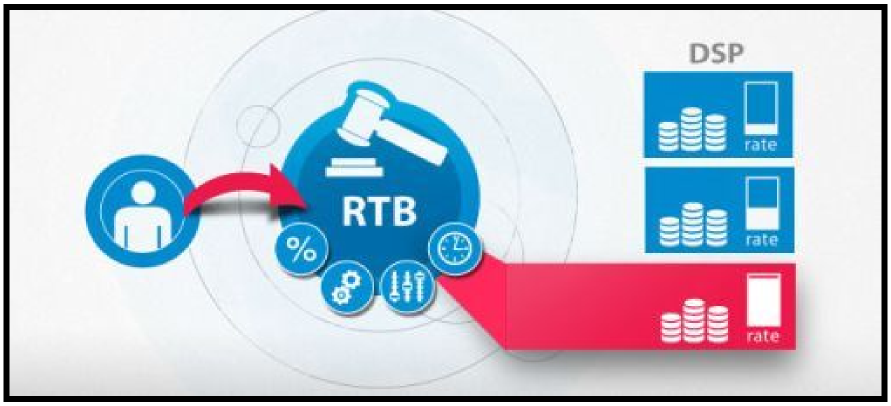


Fig 1: Real Time Bidding

1.1 Architecture Diagram

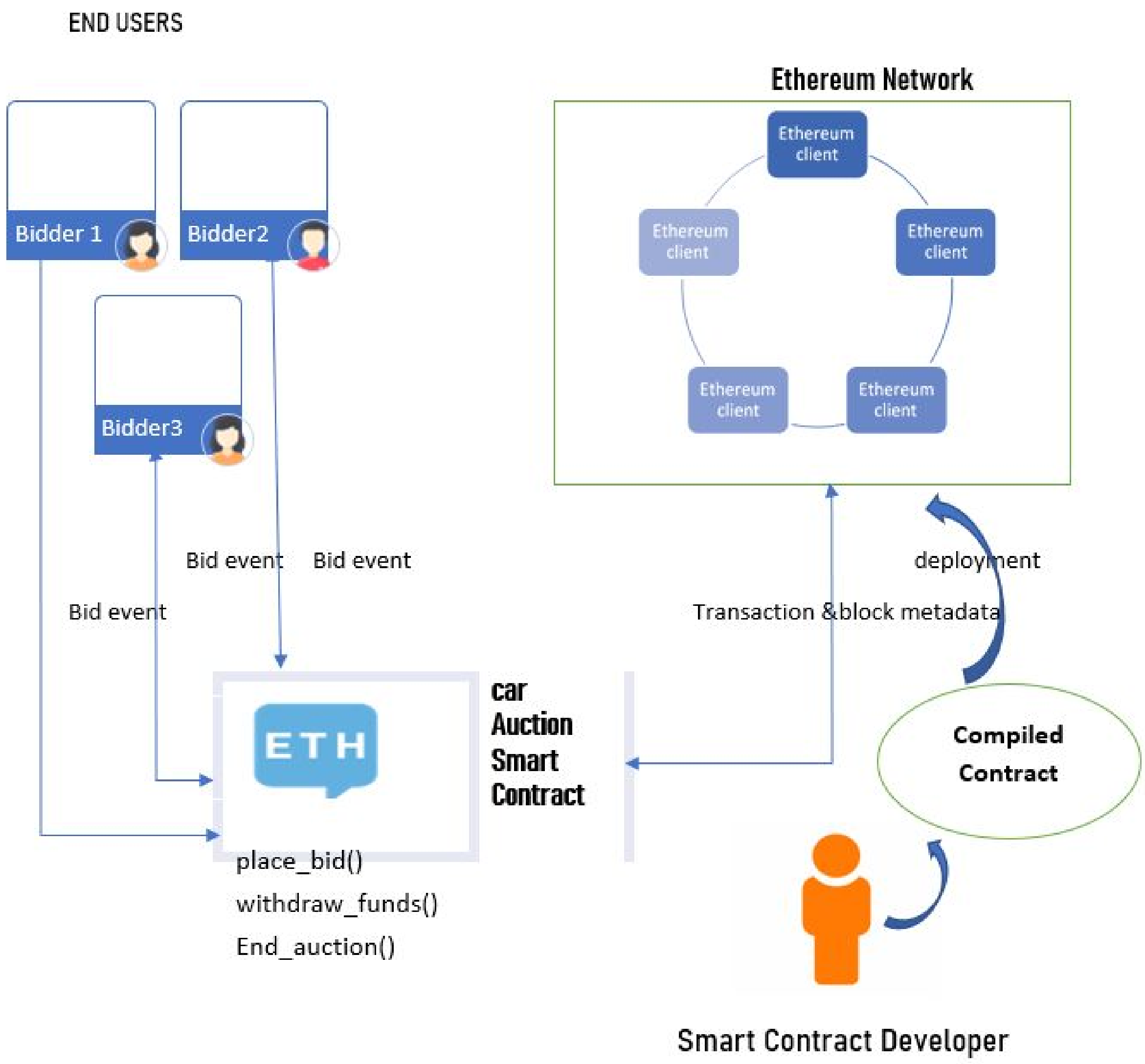


Fig 1.1 Auction Architecture Diagram

1.2 Flow Diagram

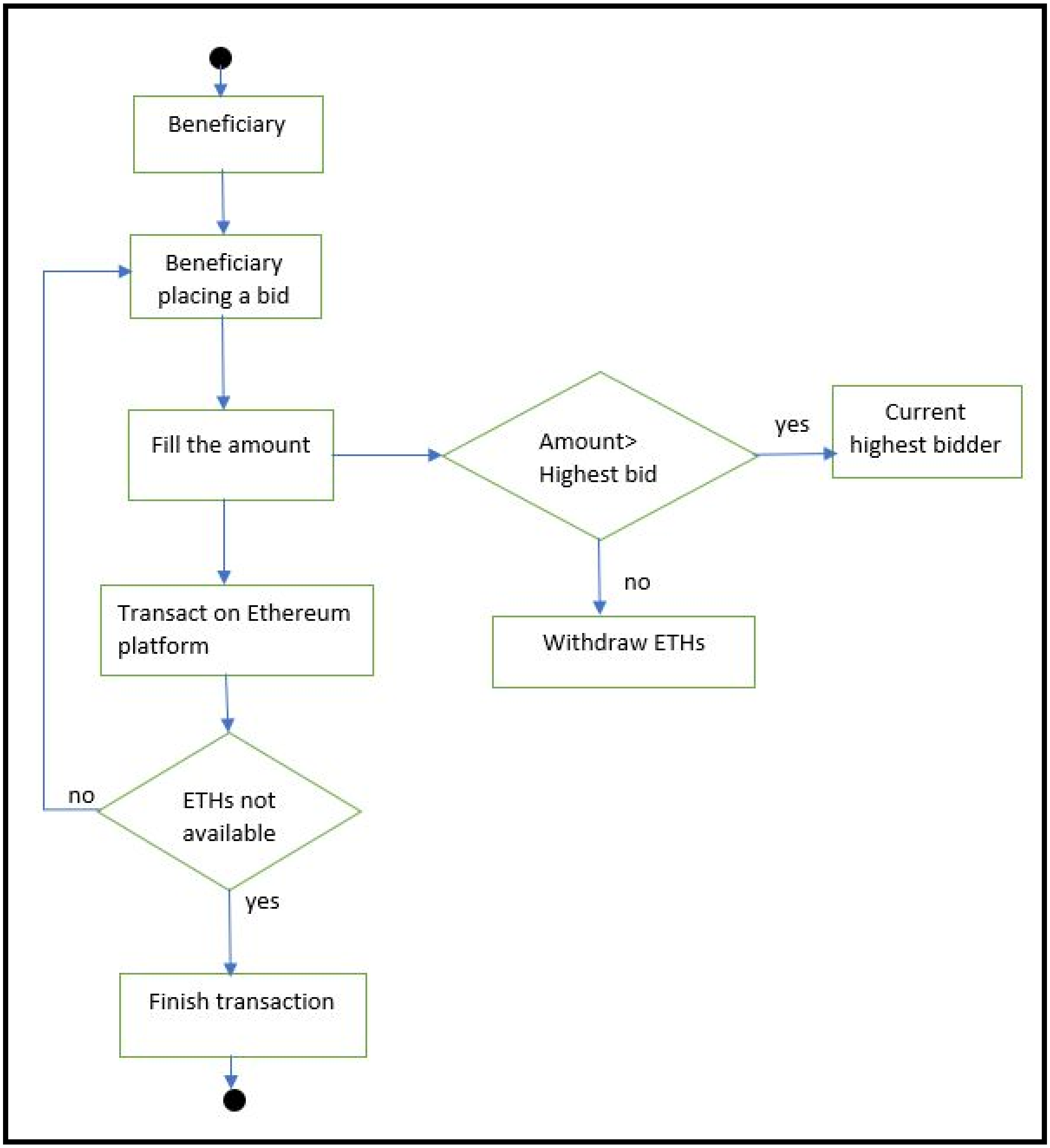


Fig 1.2 Activity Diagram of Smart Contract

1.3 Schema of work flow

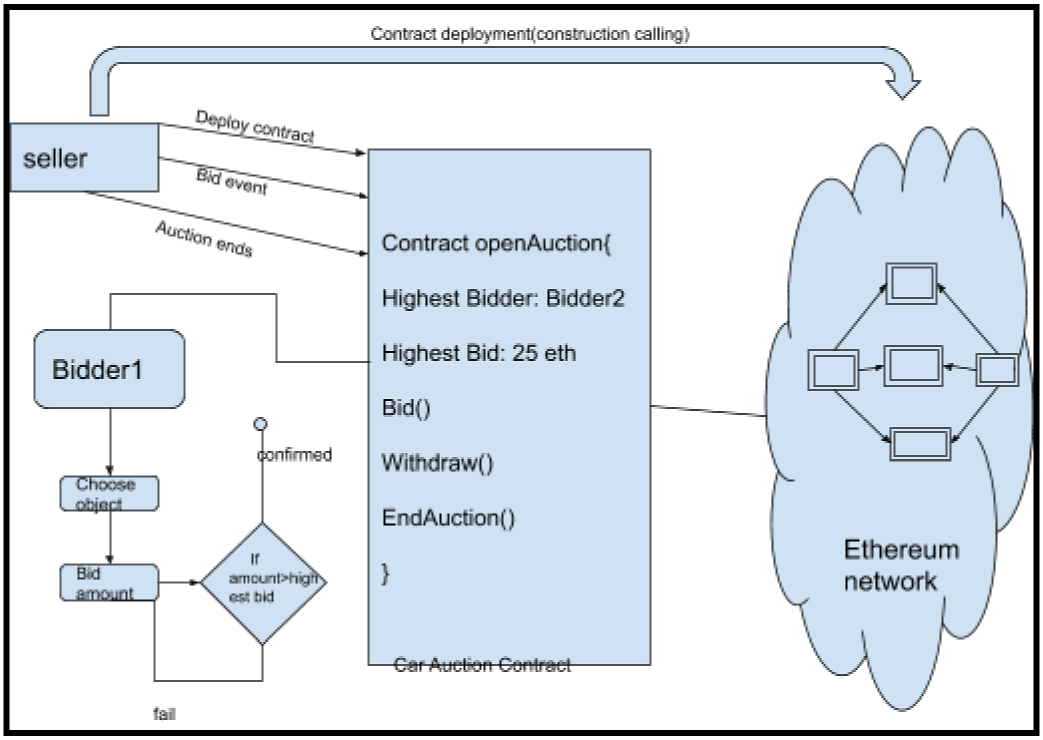


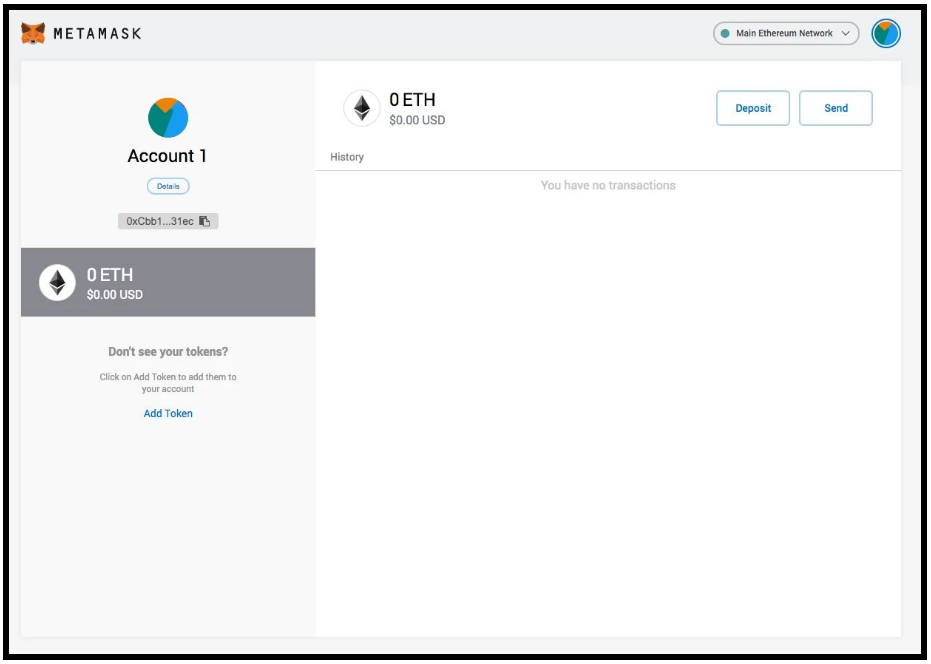
Fig 1.3 Schema of work flow

2.Installing all tools and technologies used

2.1 Metamask

What is Metamask?

MetaMask capacities as an entryway that permits clients to have direct access to any blockchain dApp on the Ethereum arrange directly from your work area program.



0xCbb1...31ec

0 ETH
$0.00 USD

You have no transactions

Don't see your tokens?

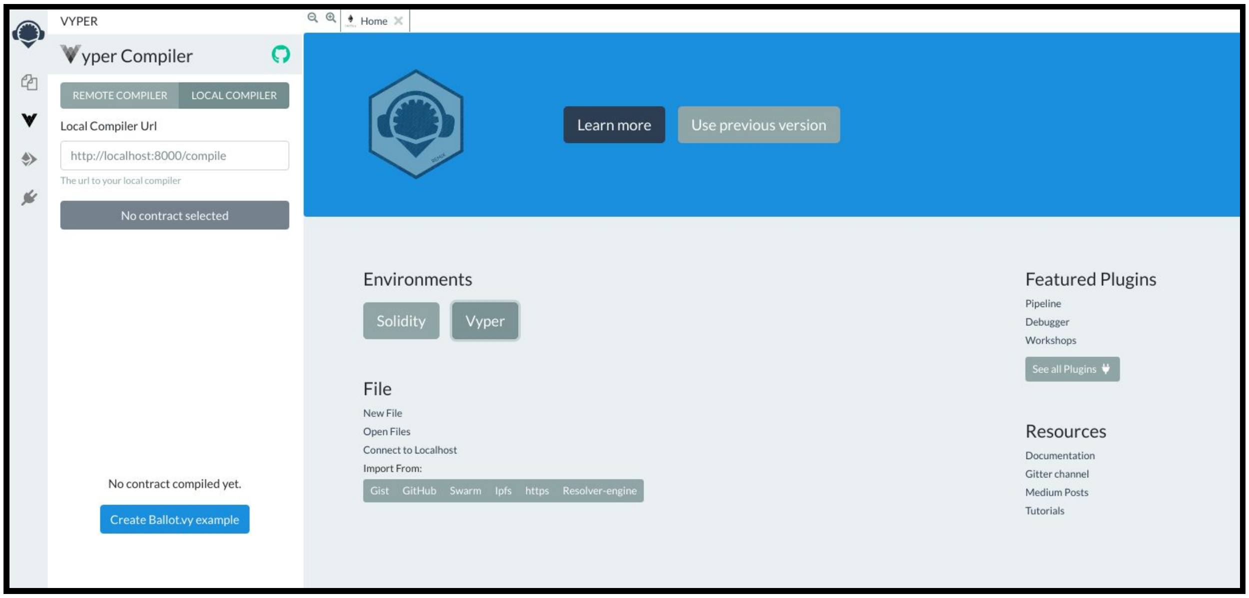
Click on Add Token to add them to your account

Fig 2.1: Metamask interface to initial setup

2.2 Remix IDE

Remix is an IDE that’s used to write, compile and debug code in different languages like solidity, vyper.

This is a remote compiler and here we have used Vyper as our programming language.



Local Compiler Url: http://localhost:8000/compile

The url to your local compiler

No contract compiled yet.

Fig 2.2: Remix.org web interface for Vyper language to deploy smart contract

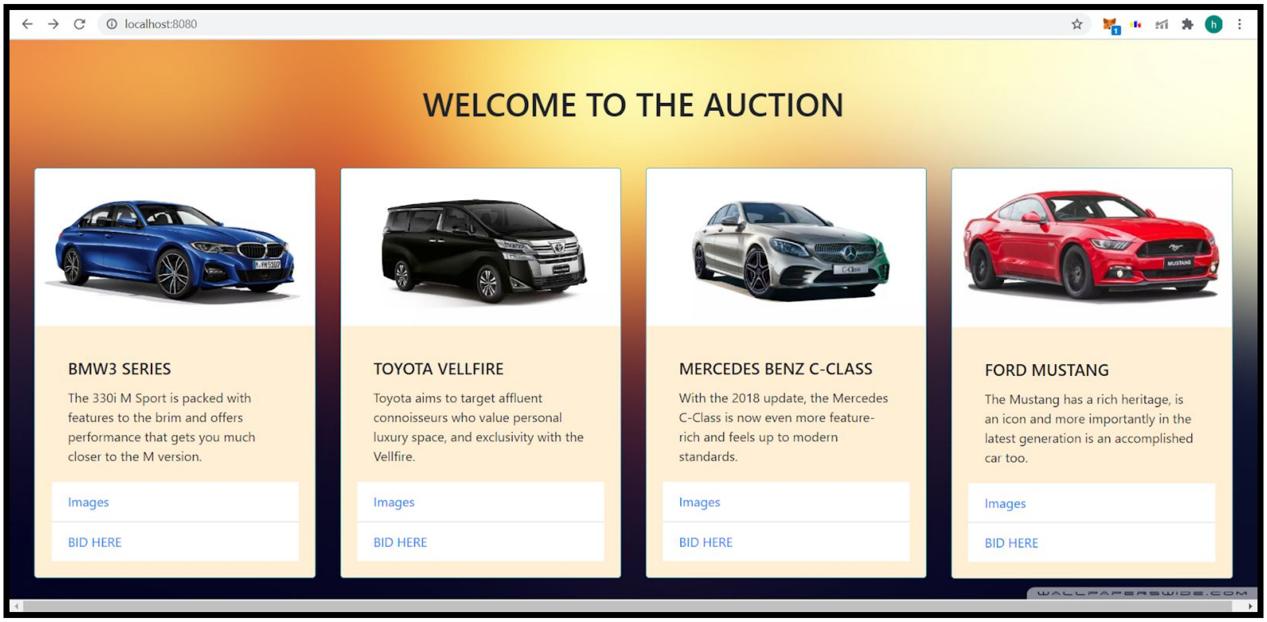
2.3 Bootstrap

An open-source CSS framework directed at responsive, mobile-first front-end web development, it basically is used to improve the data user experience better.

3.Creating User Interface for the application

3.1 Frontend via Bootstrap

We have used HTML, CSS, JS and Bootstrap for building the web application and later we have connected it to the open_auction smart contract.



WELCOME TO THE AUCTION

BMW3 SERIES

The 330i M Sport is packed with features to the brim and offers performance that gets you much closer to the M version.

TOYOTA VELLFIRE

Toyota aims to target affluent connoisseurs who value personal luxury space, and exclusivity with the Vellfire.

MERCEDES BENZ C-CLASS

With the 2018 update, the Mercedes C-Class is now even more feature-rich and feels up to modern standards.

FORD MUSTANG

The Mustang has a rich heritage, is an icon and more importantly in the latest generation is an accomplished car too.

Fig 3.1: Web user interface for our website for user experience

There are 4 objects:

1. BMW3 series
2. Toyota Vellfire
3. Mercedes Benz C-class
4. Ford Mustang

4. Creating Auction Smart Contract

The programming language we have used for writing smart contracts in ethereum is **Vyper,** which is derived from python3.

4.1 WORKFLOW:

Declaring of variables for:

- beneficiary address, auction duration time(Both start time and end time)
- Highest Bidder and highest bid

4.1.1 Constructor Function

Here constructor(__init__()) is used for initialising variables beneficiary and auction time in smart contract.

Pseudo Code:

```
beneficiary <- _benificiary address

auctionStart <- current time

auctionend <- auctionStart + _bidding_time
```

```
#constructor

@public

def __init__(_benificiary : address, _bidding_time: timedelta):
   self.beneficiary = _benificiary
   self.auctionStart = block.timestamp
   self.auctionend = self.auctionStart + _bidding_time
```

4.1.2 Default Function

Used for placing bids by transferring ether to the smart contract.Given below the code we have used:

Pseudo Code:

```
if(self.highestBidder !=NULL)
    pendingReturns of highestBidder <-  highestBid
highestBidder <- sender
highestBid <- Bid value
```

```
@public
@payable
def __default__():
    assert block.timestamp < self.auctionend
    assert msg.value > self.highestBid
    if(self.highestBidder !=ZERO_ADDRESS):
        self.pendingReturns[self.highestBidder]
+= self.highestBid
    self.highestBidder = msg.sender
    self.highestBid = msg.value
```

4.1,3 Withdraw function

The function used for refunding those bid amounts which fails the open auction. It is written after the default function in order to prevent malicious activity because placing bids involves transfer of ethers (ETH). Highest Bid and Highest Bidder is being set in bid function when current weigh value is greater than till highest bid.

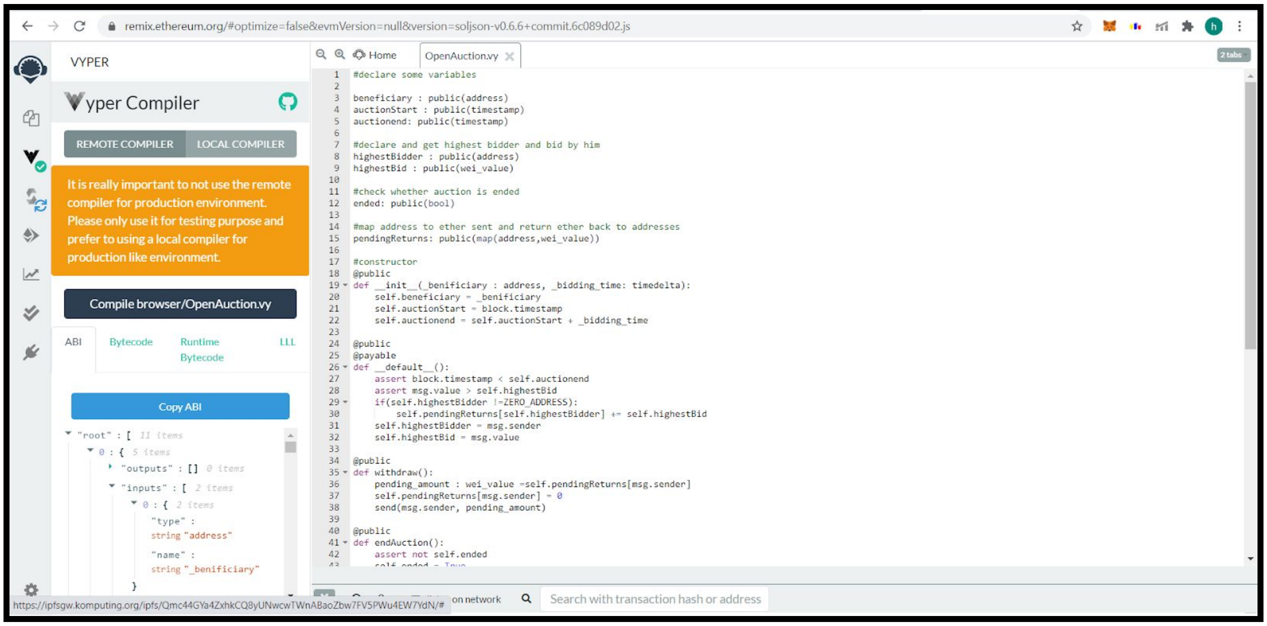
Pseudo Code:

```
pending_amount : wei_value = pendingReturns of the sender
pendingReturns of sender <- 0
send (msg.sender, pending_amount)
```

```
@public

def withdraw():
    pending_amount : wei_value = self.pendingReturns[msg.sender]
    self.pendingReturns[msg.sender] = 0
    send(msg.sender, pending_amount)
```

4.2 To connect smart contract copy ABI and smart contract address

Compile the code by clicking on compile in left.


remix.ethereum.org/#optimize=false&evmVersion=null&version=soljson-v0.6.6+commit.6c089d02.js
VYPER
Vyper Compiler
REMOTE COMPILER
LOCAL COMPILER
Compile browser/OpenAuction.vy
ABI
Bytecode
Runtime Bytecode
LLL
Copy ABI
Home
OpenAuction.vy
2 tabs
on network
Search with transaction hash or address
https://ipfsgw.komputing.org/ipfs/Qmc44GYa4ZxhkCQ8yUNwcwTWnABaoZbw7FV5PWu4EW7YdN/#

It is really important to not use the remote compiler for production environment. Please only use it for testing purpose and prefer to using a local compiler for production like environment.

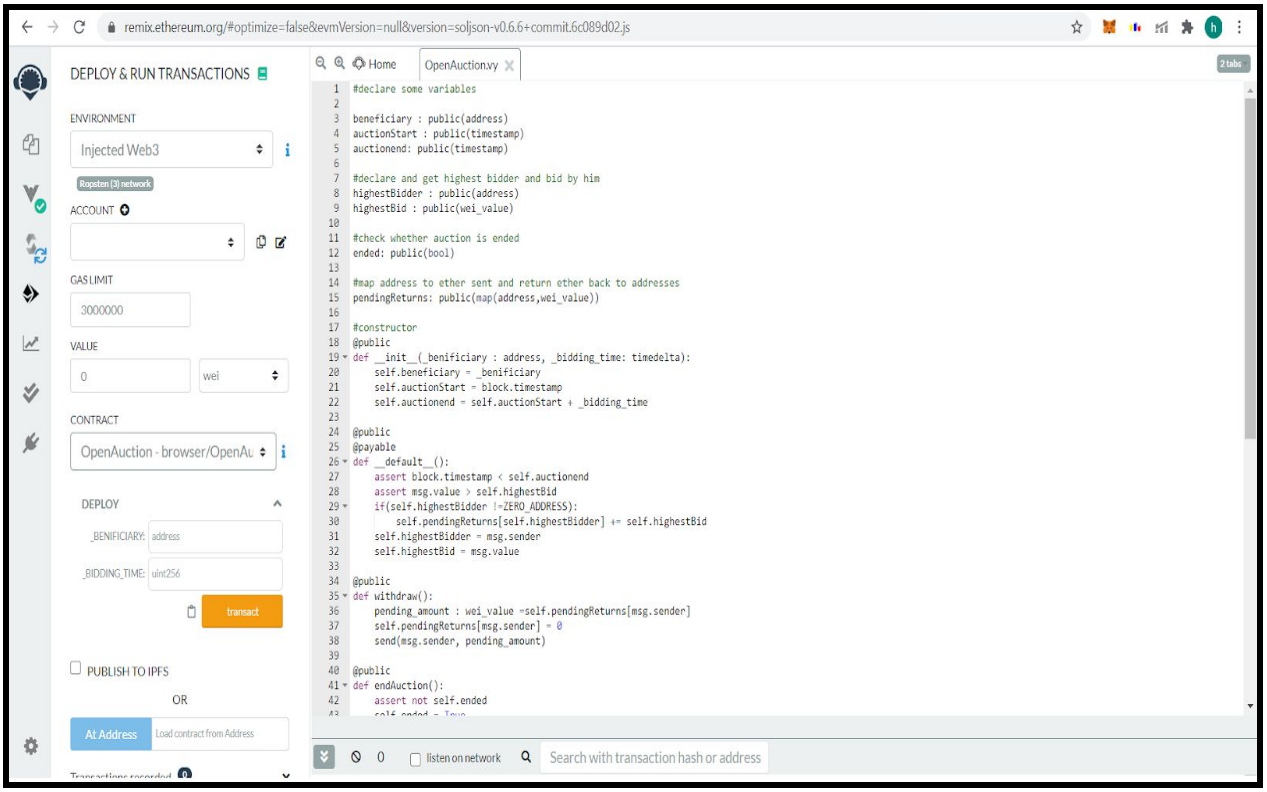
```
"root" : [ 11 items
  0 : { 5 items
    "outputs" : [] 0 items
    "inputs" : [ 2 items
      0 : { 2 items
        "type" :
        string "address"
        "name" :
        string "_benificiary"
      }
```

```
#declare some variables

beneficiary : public(address)
auctionStart : public(timestamp)
auctionend: public(timestamp)

#declare and get highest bidder and bid by him
highestBidder : public(address)
highestBid : public(wei_value)

#check whether auction is ended
ended: public(bool)

#map address to ether sent and return ether back to addresses
pendingReturns: public(map(address,wei_value))

#constructor
@public
def __init__(_benificiary : address, _bidding_time: timedelta):
    self.beneficiary = _benificiary
    self.auctionStart = block.timestamp
    self.auctionend = self.auctionStart + _bidding_time

@public
@payable
def __default__():
    assert block.timestamp < self.auctionend
    assert msg.value > self.highestBid
    if(self.highestBidder !=ZERO_ADDRESS):
        self.pendingReturns[self.highestBidder] += self.highestBid
    self.highestBidder = msg.sender
    self.highestBid = msg.value

@public
def withdraw():
    pending_amount : wei_value =self.pendingReturns[msg.sender]
    self.pendingReturns[msg.sender] = 0
    send(msg.sender, pending_amount)

@public
def endAuction():
    assert not self.ended
    self.ended = True
```

Fig 4.1: Remix IDE which displays the compiled smart contract and displays the Abi

- Now it is time to deploy the transaction by clicking on ***Deploy and run transaction.***

- In the environment field select ***Injected Web3 provider Deploy*** and select the smart contract from dropdown.



remix.ethereum.org/#optimize=false&evmVersion=null&version=soljson-v0.6.6+commit.6c089d02.js

DEPLOY & RUN TRANSACTIONS

ENVIRONMENT: Injected Web3

Ropsten (3) network

ACCOUNT

GAS LIMIT: 3000000

VALUE: 0 wei

CONTRACT: OpenAuction - browser/OpenAu

DEPLOY

_BENIFICIARY: address

_BIDDING_TIME: uint256

transact

□ PUBLISH TO IPFS

OR

At Address | Load contract from Address

Transactions recorded 0

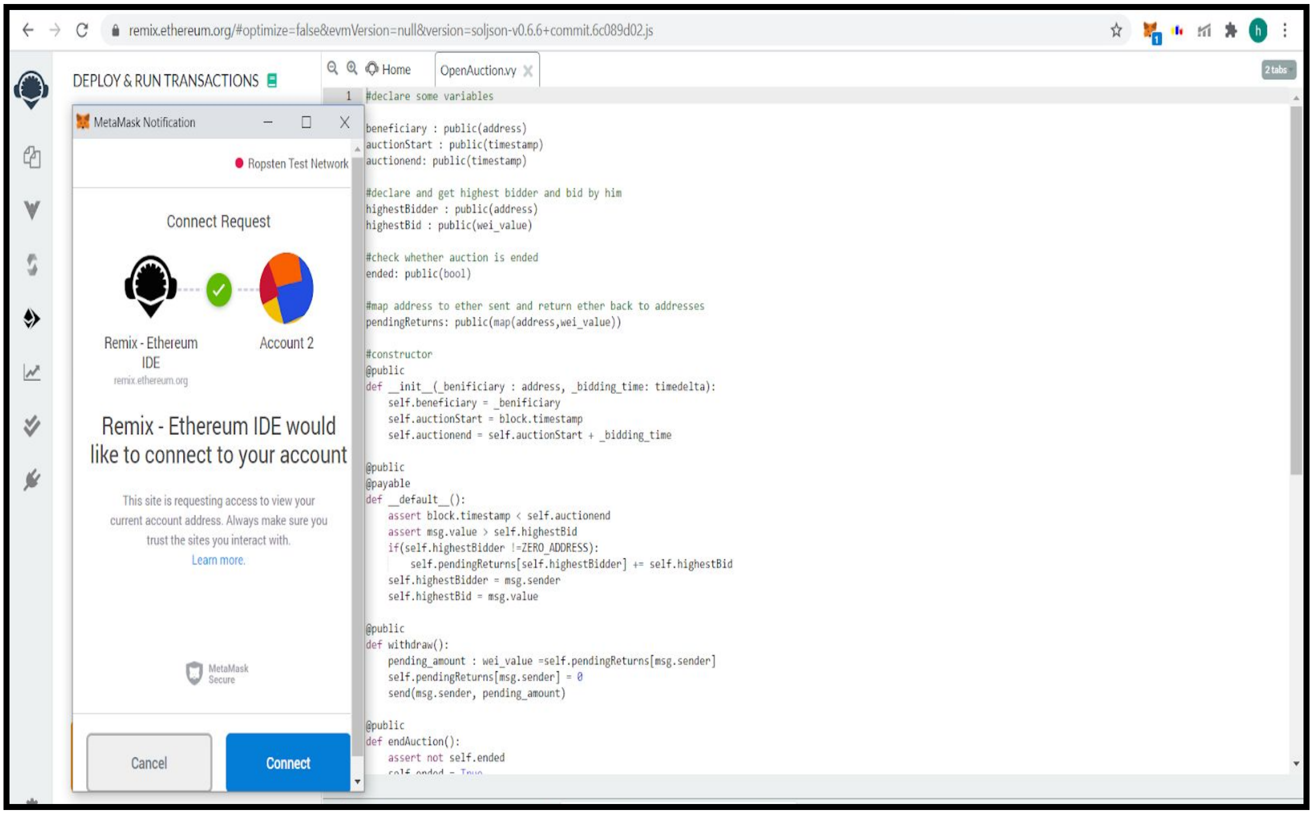
Home | OpenAuction.vy

```
#declare some variables

beneficiary : public(address)
auctionStart : public(timestamp)
auctionend: public(timestamp)

#declare and get highest bidder and bid by him
highestBidder : public(address)
highestBid : public(wei_value)

#check whether auction is ended
ended: public(bool)

#map address to ether sent and return ether back to addresses
pendingReturns: public(map(address,wei_value))

#constructor
@public
def __init__(_benificiary : address, _bidding_time: timedelta):
    self.beneficiary = _benificiary
    self.auctionStart = block.timestamp
    self.auctionend = self.auctionStart + _bidding_time

@public
@payable
def __default__():
    assert block.timestamp < self.auctionend
    assert msg.value > self.highestBid
    if(self.highestBidder !=ZERO_ADDRESS):
        self.pendingReturns[self.highestBidder] += self.highestBid
    self.highestBidder = msg.sender
    self.highestBid = msg.value

@public
def withdraw():
    pending_amount : wei_value =self.pendingReturns[msg.sender]
    self.pendingReturns[msg.sender] = 0
    send(msg.sender, pending_amount)

@public
def endAuction():
    assert not self.ended
    self.ended = True
```

Fig 4.2: Remix interface displaying the injected web3 Js setup

- Click on the ***Deploy*** button and mention all the parameters the ***beneficiary address*** and the ***bidding time(secs)*** and finally transact it.

- Popup of Metamask will appear which will deploy the smart contract on Ropsten Test Network.



Remix - Ethereum IDE would like to connect to your account

This site is requesting access to view your current account address. Always make sure you trust the sites you interact with.

```
#declare some variables

beneficiary : public(address)
auctionStart : public(timestamp)
auctionend: public(timestamp)

#declare and get highest bidder and bid by him
highestBidder : public(address)
highestBid : public(wei_value)

#check whether auction is ended
ended: public(bool)

#map address to ether sent and return ether back to addresses
pendingReturns: public(map(address,wei_value))

#constructor
@public
def __init__(_benificiary : address, _bidding_time: timedelta):
    self.beneficiary = _benificiary
    self.auctionStart = block.timestamp
    self.auctionend = self.auctionStart + _bidding_time

@public
@payable
def __default__():
    assert block.timestamp < self.auctionend
    assert msg.value > self.highestBid
    if(self.highestBidder !=ZERO_ADDRESS):
        self.pendingReturns[self.highestBidder] += self.highestBid
    self.highestBidder = msg.sender
    self.highestBid = msg.value

@public
def withdraw():
    pending_amount : wei_value =self.pendingReturns[msg.sender]
    self.pendingReturns[msg.sender] = 0
    send(msg.sender, pending_amount)

@public
def endAuction():
    assert not self.ended
    self.ended = True
```

Fig 4.3: Remix IDE displaying the Metamask popup where we confirm the connection request for deployment

5. Results

5.1 Computational View:

5.1.1 Gas details of each attribute in the smart contract:

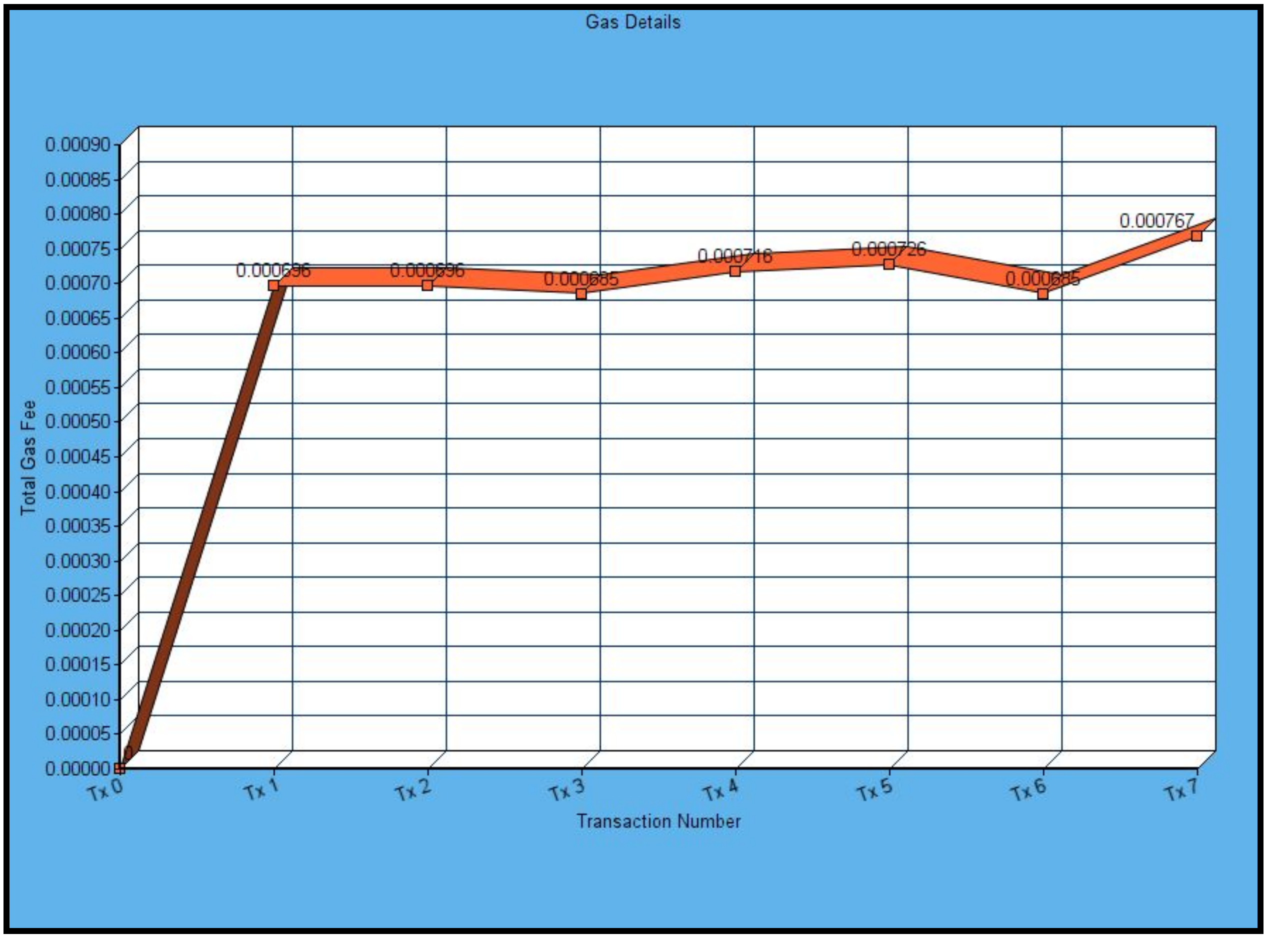
Function Name	Gas Limit
Auction end :	1271
Auction start:	1241
Beneficiary:	1211
endAuction:	72639
ended:	1361
highest bid:	1331
highest bidder:	1301
pending returns:	1545
withdraw:	56154

Table 5.1: Function Gas Limit

5.1.2 Gas details followed in the transaction:

Transac. Number	0	1	2	3	4	5	6	7
Total Gas fee (in ethers)	**NUll**	**0.000696**	**0.000696**	**0.000685**	**0.000716**	**0.000726**	**0.000685**	**0.000767**

Table 5.1: Transaction Gas Details



Graph 5.1: Gas details

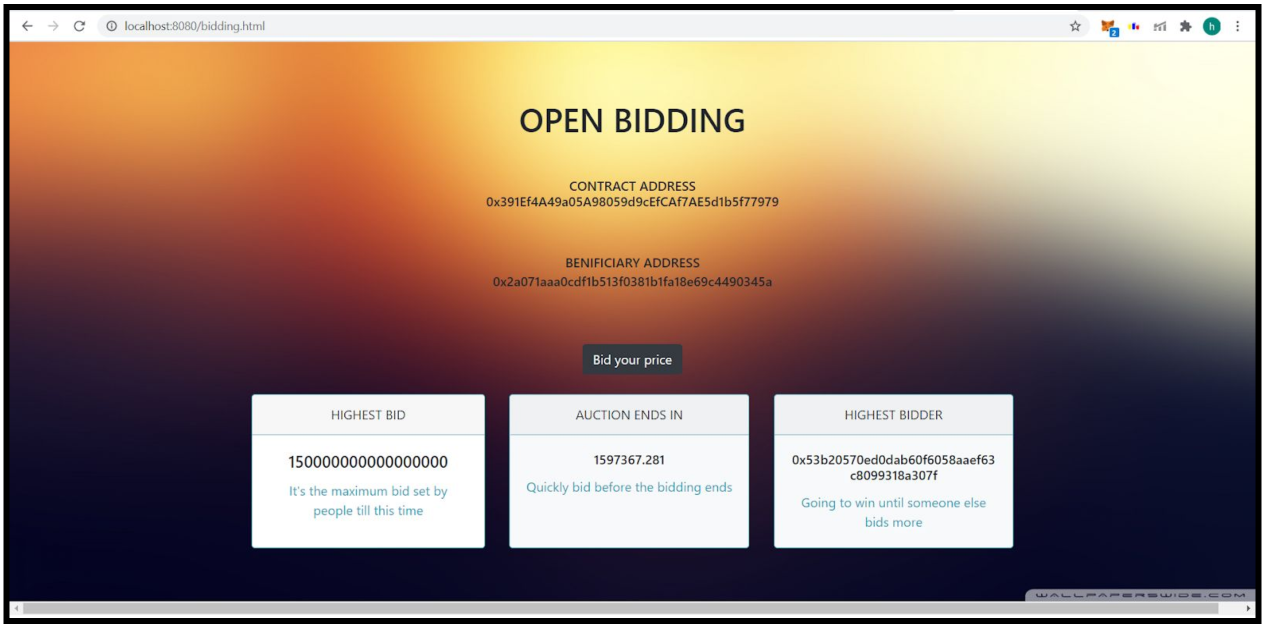
5.2 Observational view:

5.2.1 Table showing the bidders details:

Account No.	Bidder Address
1	0x040f9099cA0e1EF4B9A5352549e0fc85B660f9c1
2	0x53b20570Ed0dAb60F6058aaef63c8099318a307f
3	0x30C768CdE8Cf2ECEd47A604589F9B8AE76570Dc8
4	0xd48c4e17A0452022619dF80Db62239784Fe23Bc8
5	0x2A071aaa0CDF1b513F0381B1fa18e69c4490345a

Table 5.3: Bidder account details

We are showing the details for Toyota Vellfire:



OPEN BIDDING

CONTRACT ADDRESS
0x391Ef4A49a05A98059d9cEfCAf7AE5d1b5f77979

BENIFICIARY ADDRESS
0x2a071aaa0cdf1b513f0381b1fa18e69c4490345a

HIGHEST BID
150000000000000000
It's the maximum bid set by people till this time

AUCTION ENDS IN
1597367.281
Quickly bid before the bidding ends

HIGHEST BIDDER
0x53b20570ed0dab60f6058aaef63c8099318a307f
Going to win until someone else bids more

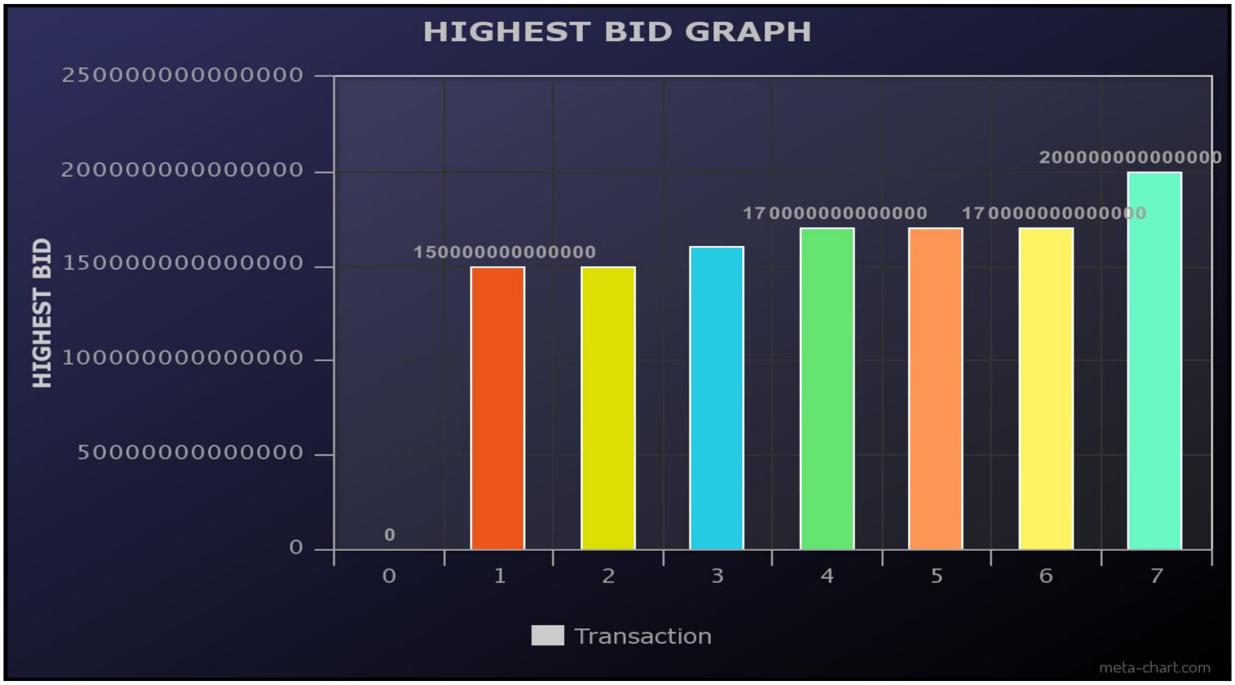
Fig 5.1: Web user interface for Open Bidding Page for user experience

5.2.2 Transactions Details:

Transaction Number	**Account Number** (metmask)	**Bidding Amount**	**Transaction**	**Highest Bid**	**Highest Bidder**
0	NULL	0	-	0	NULL
1	2	150000000000000	Confirmed	150000000000000	2
2	3	150000000000000	Failed (The bidding amount is not greater than the highest bid)	150000000000000	2
3	3	160000000000000	Confirmed	160000000000000	3
4	5	170000000000000	Confirmed	170000000000000	5
5	4	180000000000000	Failed (Doesn’t possess enough ether)	170000000000000	5
6	3	160000000000000	Failed (The bidding amount is not greater than the highest bid)	170000000000000	5
7	1	200000000000000	Confirmed	200000000000000	1

Table 5.4: Bidder transaction details

5.2.3 Highest bidding amount graphical representation:



Graph 5.2: Highest Bid graph

- Ultimately by going through all the procedures of metamask providing the interface connection between the frontend and backend. When we click on the Bid Your Price Icon a pop-up of metamask notification comes up which provides us the option of bidding through whichever account we want to bid.
- And the open bidding web interface provides us with the result that who one in the whole bidding won.
- Highest Bidder shows the blockchain account of the person who has given the highest bid till this time.
- Highest Bid shows the amount which was set by the highest bidder.

6. Conclusion and future scopes

We are finally able to create a decentralised Auction app performing bidding so that owners can have direct entry to this occurrence.As it is decentralized and scrambled, no single network can control the record, anybody can see it, making it straightforward and consistent.

Blockchain has a great future worldwide.So far we have seen the exponential rise in scope of blockchain technology.

Digital Advertising- By and by, we are confronting a great deal of difficulties in space extortion and absence of straightforwardness in the computerized publicizing field.Blockchain is still in its starting stage, all promoting organizations are beginning to assemble trust on this innovation on how they can upgrade their business.

Cyber Security and Forecasting - Using blockchain information is confirmed and encoded utilizing creative cryptographic innovation. The blockchain innovation is set to adjust the total procedure for research, counseling, examination and estimating.

IoT and Networking- Blockchain is utilized by numerous MNCs to build up a disseminated system of IoT gadgets which will expel prerequisites of a focal area to oversee correspondence between them.

- Future scope for this particular project is that we can build it for more number of objects and bidder accounts can be increased.
- It definitely is providing an ease to people by decentralised technology and will be exempt from any following malfunctions due to wrong people.
- It can be used to bid for people by providing them new technology gadgets and any accessories which are not used for rich people(who want to bid) and can be provided to others so that they can bid on an object of their choice without any fear of mishappenings and fraud.

7. References

- https://www.shiksha.com/it-software/articles/scope-of-blockchain-technology-blogId-26037
- https://vyper.readthedocs.io/en/latest/
- https://www.coinbureau.com/smart-contracts/beginners-guide-vyper-language/
- https://www.leewayhertz.com/ethereum-smart-contract-tutorial/