Smart Lottery System

Team Number: 8

Saideep Ganireddy

Email ID: saideepg@buffalo.edu

Person Number : 50354553

Likhitha Dodda

Email ID: ldodda@buffalo.edu

Person Number : 50353571

Why a Decentralized Lottery System?

- Lottery games are dependent on luck, the outcome is unknown and there can be many scams and issues in the traditional centralized system.
- The participants should have to trust that the lottery owners declare the winners honestly, which actually is corrupted these days.
- And in a Centralized Lottery System the lottery owners or the companies should maintain the severs and staff for ticket counting creation and verification.
- To overcome all these issues a Block chain based decentralised lottery System is designed.
- This Block chain based decentralized lottery System can fix the issue by providing a platform that can't be hacked or corrupted.
- A smart contract is implemented in such a way that the winner is chosen randomly, and this can be proved.
- A distributed ledger is used to update the participants about all the changes and the updates i.e. The winner declarations in the lottery.
- So, the participants can trust that their money is treated fairly.

Building a Smart Lottery System:

- In order to build a smart lottery system here are the few things we need to consider:
  1. The participants should buy a lottery ticket.
  2. The owner should generate the winner randomly.
  3. The winning amount should be settled.

- Before the first step let's say the owner wants to keep a track of all the participants in his lottery. So, the owner wants to register every user who want to participate in his lottery.
- To buy a ticket, the participant should have to be registered by the owner first. So now the participant can buy any number of tickets he wants to buy and the smart contract randomly generates unique ticket numbers and assign it to the participant's Address.
- Here we can observe that this blockchain based system reduced the overhead of generating the tickets manually and selling them.
- Every lottery has a start time and end time. So, all the participants can register to the lottery system and buy any number of tickets until the lottery time ends
- Once the lottery time is ended any user/participant cannot buy lottery tickets. As a matter of transparency, the owner should be able to declare winners only after the lottery time ended and only owner should have the access of generating/declaring Winners.
- Once the winner is declared the distributed ledger is updated and all the users get to see the winner of the lottery and from the interface, they can click on seeWinner( ) function that is going to be created in the smart contract and see the address of the winner.
- The final step after declaring the winner is the payment settlement. The owner gets a gain from the lottery and the gain amount is settled to the owner's wallet and the winning amount is settled to the winner's wallet

ERC token:

- In the traditional centralised system the participants use their normal currency to buy the tickets and the winning amount is settled to the winner in the same currency.
- But as this is a blockchain based system a new ERC token is implemented in Ethereum for this smart lottery System.
- The name of the token is **"LWTcoin"** and the symbol of the token is **"LWT"** i.e. "**Lottery with Transparency**".
- LWT stands for lottery with transparency. We chose this as token symbol because it clearly reflects the main intention of the smart lottery system and thereby increasing the ease of usage of the token with understanding.
- Lottery transactions might not use decimal tokens but just for the convinence of the user 2 decimal points were added.

ERC token Smart Contract:

- In the ERC token Smart Contract we chose the maximum supply to be 5000000 as the ticket price is 5LWT and on an average there will not be more than 1million tickets sold at a time.
- Also there is an additional function to increase the total supply and the owner of the token can increase the supply of the tokens if necessary.

Here are the Functions used in the ERC token Smart Contract:

The functions from ERC20Interface is used and their definitions are overridden by making the required changes.

1. function totalSupply() external view returns(uint256)
2. function balanceOf(address account) external view returns(uint256);
3. function allowance(address owner, address spender) external view returns(uint256);
4. function transfer(address recipient, uint256 amount) external returns(bool);
5. function approve(address spender, uint256 amount) external returns(bool);
6. function transferFrom(address sender, address recipient, uint256 amount) external returns(bool);
7. event Transfer(address indexed from, address indexed to, uint256 value);
8. event Approval(address indexed owner, address indexed spender, uint256 value);

1. TotalSupply( ) :

- The total Supply is the total number of tokens we are deploying initially.
- This number is passed an argument while deploying the code and is assigned as a balance to the token owner's Account.
- We are giving the total Supply as 5000000 tokens. This function returns the total n umber of tokens deployed.

```
function totalSupply() public override view returns(uint256) {

return totalSupply_;

}
```

2. BalanceOf( ) :

- BalanceOf function takes the address of the account as parameter and returns the balance of LWT tokens of the account.

```
function balanceOf(address tokenOwner) public override view returns(uint256) {

    return balances[tokenOwner];

}
```

3. Allowance( ) :

- Allowance function gives the amount of tokens allowed by the token owner to be transferred to the spender's account i.e. the participant's account in this case.

```
function allowance(address owner, address delegate) public override view returns(uint256) {

    return allowed[owner][delegate];

}
```

4. Transfer( ) :

- Transfer function is used by the token owner to send tokens to the recipients.
- In this smart owner can use this function to send the tokens from his account to participant accounts while registering them or while transferring winning amount to winner.

```
function transfer(address recipient, uint256 numTokens) public override returns(bool) {

    require(numTokens <= balances[msg.sender]);

    balances[msg.sender] -= numTokens;

    balances[recipient] += numTokens;

    emit Transfer(msg.sender, recipient, numTokens);

    return true;

}
```

- This function first checks if the sender has enough balance to send the tokens and then updates the tokens of the sender and recipient accordingly.

5. Approve( ) :

- Approve function is used by the token Owner to approve the spender to transfer tokens from his account to spender's account.
- This can be used by the lottery owner to cross check the transfer of winning funds to the winner account before transferring them.

```
function approve(address delegate, uint256 numTokens) public override returns(bool) {

  allowed[msg.sender][delegate] = numTokens;

  emit Approval(msg.sender, delegate, numTokens);

  return true;

}
```

- This function updates the number of tokens that are approved from sender to receiver.

6. transferFrom( ) :

- transferFrom( ) function is used to transfer tokens from sender's account to receiver's account .
- This function can be used in the smart contract to transfer tokens between two addresses.

```
function transferFrom(address owner, address buyer, uint256 numTokens) public override
returns(bool){

  require(numTokens <= balances[owner]);

  require(numTokens <= allowed[owner][msg.sender]);

  balances[owner] -= numTokens;

  allowed[owner][msg.sender] -= numTokens;

  balances[buyer] += numTokens;

  emit Transfer(owner, buyer, numTokens);

  return true;

}
```

- This function first checks if the owner/sender has enough balance to send the tokens and then checks if there is an approval for the transfer of those specified value from the owner/sender.
- Only after both the conditions are satisfied the tokens are transferred from owner's wallet to participant's wallet/address.
- This function uses the Transfer event with parameters of owner address buyer/participant address and the number of tokens to record onto the blockchain digital wallet.

There is an additional Function added to the ERC smart contract:

7. Increase total Supply( ) :

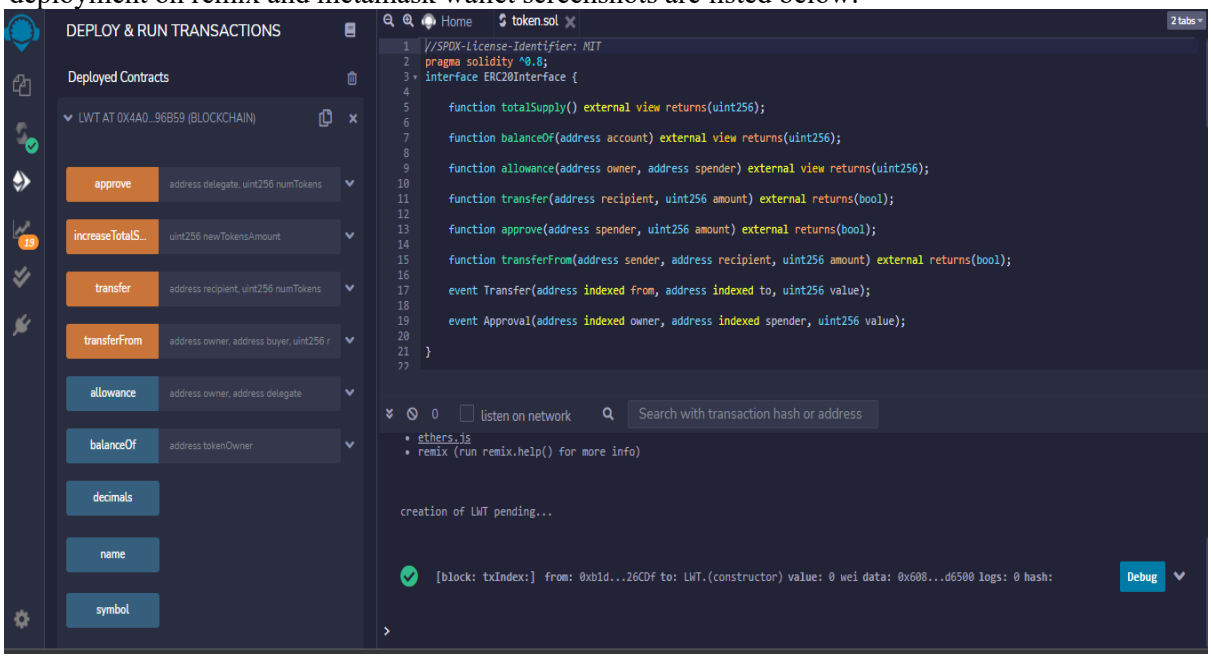
- This takes in the number of tokens that are to be increased as a parameter and adds that to the existing balance.

```
function increaseTotalSupply(uint newTokensAmount) public {

    totalSupply_ += newTokensAmount;

    balances[msg.sender] += newTokensAmount;

  }
```

ERC Token Deployment steps:

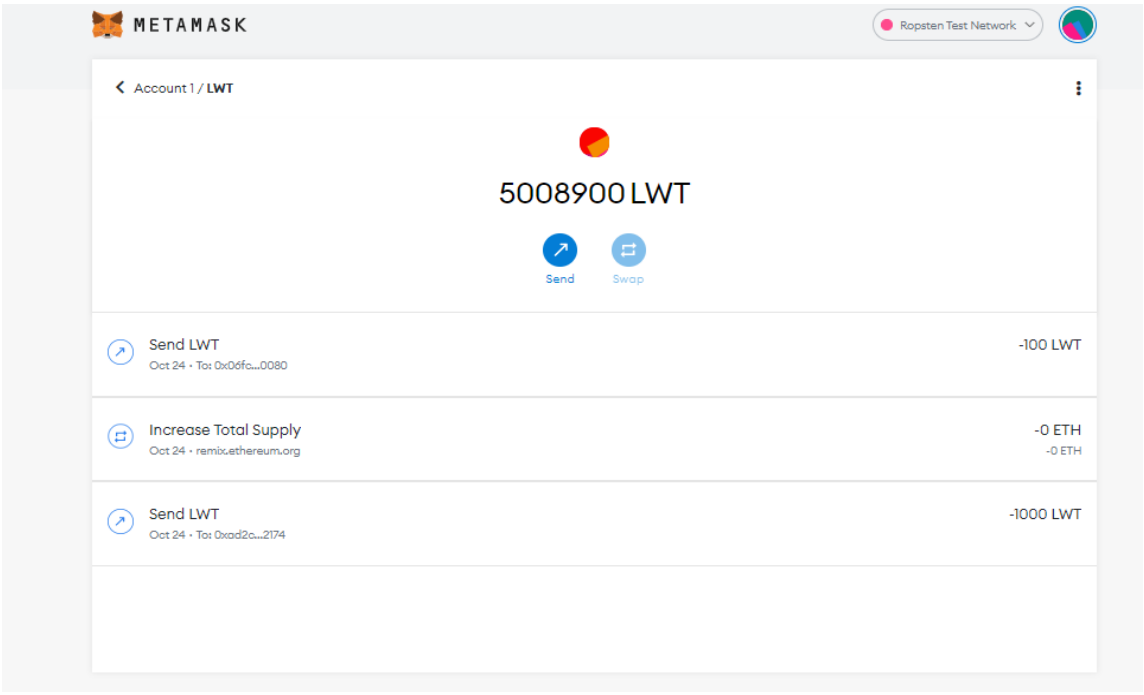
1) Copy the solidity code provided in the zip files to remix environment.
2) Compile the code in 0.8.9 compiler
3) Deploy the token.sol in remix environment using the metamask in Injected web3.
4) Smart contract address will be generated in the deployed Contracts sections.
5) Add the smart contract address onto metamask. The owner of smear contract will get all the total supply tokens onto his wallet and all others will get 0 LWT..
6) After importing the tokens you can see the LWT tokens added to your metamask wallet

The deployment on remix and metamask wallet screenshots are listed below:

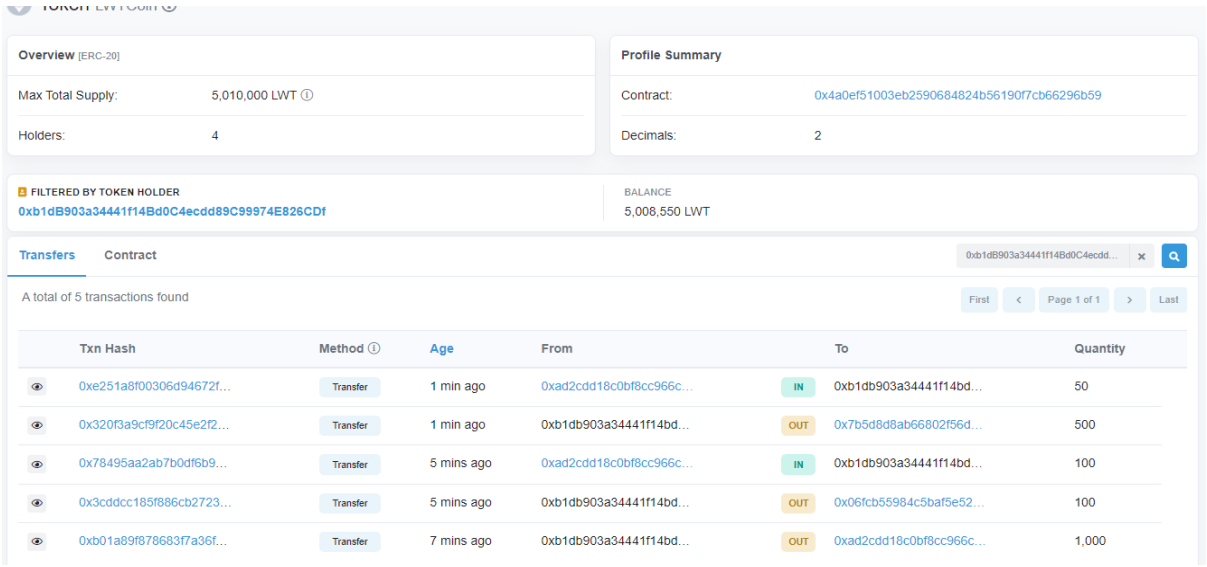


The address of the ERC smart Contract is: 0xAd4341b801b8101C89Dd2f456a5c7730129ed021

Metamask Wallet:



Etherscan Transactions:



Smart Contract:

The smartContractAddress is: `0x9dd69f77C4267C03a67f6bE7d82eC1f766E274ce`

Firstly, the functions involved in our smart contract for Smart Lottery System is:

1. Register( ) :

- This function is used by the owner to register the participants on to his lottery system.
- While registering the owner can give them some bonus tokens of say 50LWT for registering onto the lottery System.

2.StartLottery():

- This function is only accessible to owner to start the lottery.
- When the owner starts the lottery the start time is updated as the present timestamp time and the endtime is updated as start time + lottery time i.e the time dedicated for the lottery.

3. BuyTickets( ) :

- The lottery owner can fix the price of his lottery tickets let's say the price of each ticket is 5LWT and the participants can buy any number of tickets for the lottery.
- This function first checks if the participant is registered and if the lottery time is not ended with the help of modifiers and then check if the participant has enough tokens to buy the tickets.
- After purchasing the ticket the smart contract assigns a unique ticket number and stores then under that participant address.

4.UpdateNumberofWinners():

This function is used to update the number of winners that a smart contract can generate. By default the smart contract only generates 1 winner. But the owner has an options to update the number of winners before the lottery is started. So a modifier islotterystarted is added.

5. GenerateWinningTicket( ) :

- This function is used to randomly generate winner among all the tickets that are purchased by the participants.
- This function is only accessible to the lottery owner and the in order to declare winners the lottery time should be ended.
- And these conditions are checked with the help of a modifier.
- After generating the winner randomly the winner addresses are stored into an array.

6. SeeWinner( ) :

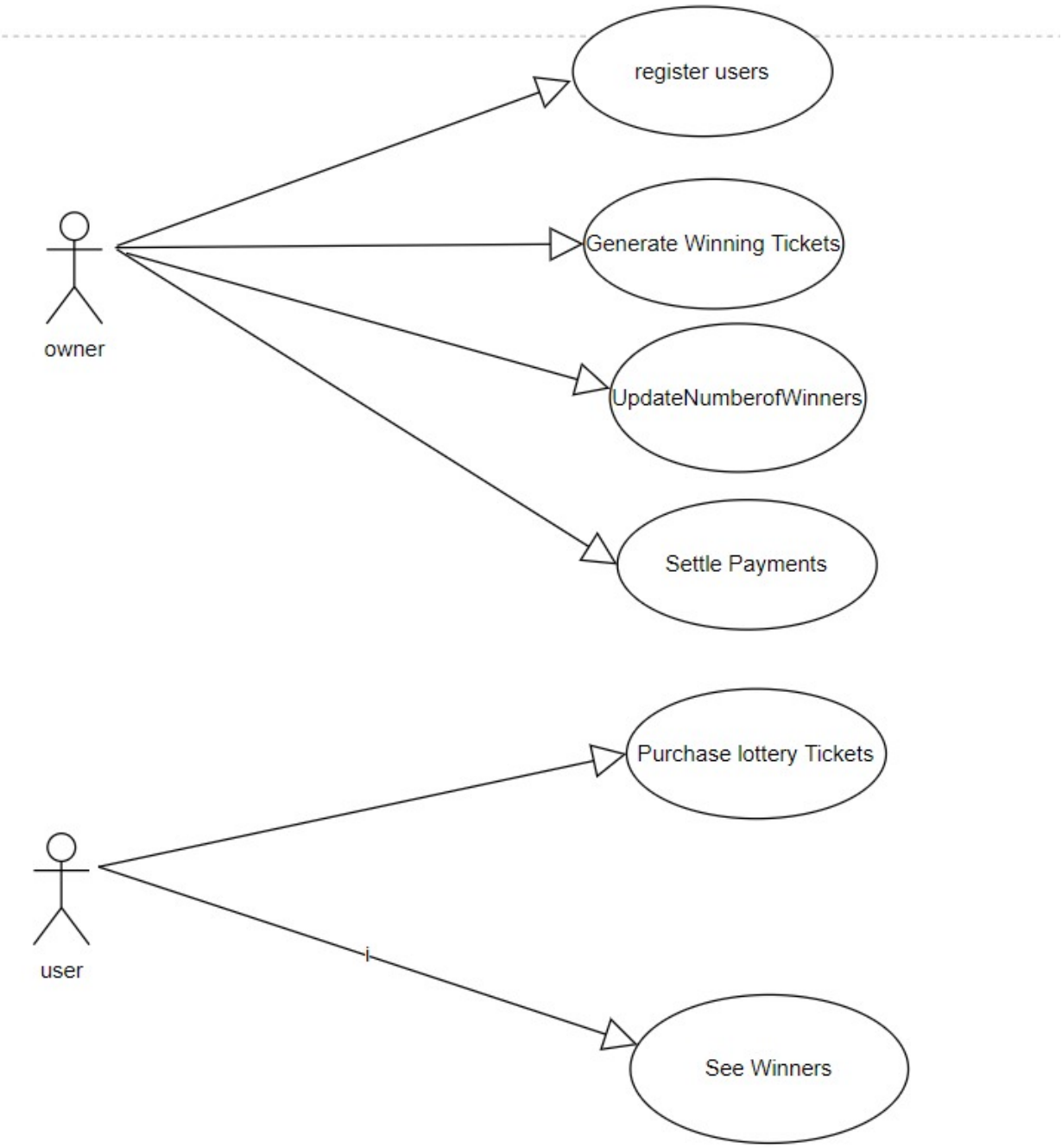
- This Function returns the Addresses of the winners but in order to return the winner the lottery should be ended so a modifier "islotteryended" is used in this smart contract.

7.SettlePayment():

- This function is used to settle payments for the winners the gain percentage is settled to the owner and the remaining amount is distributed among the winners.
- For the SettlePrize function to be called the lottery should be ended first so a modifier islotteryended is added

UML Diagrams:

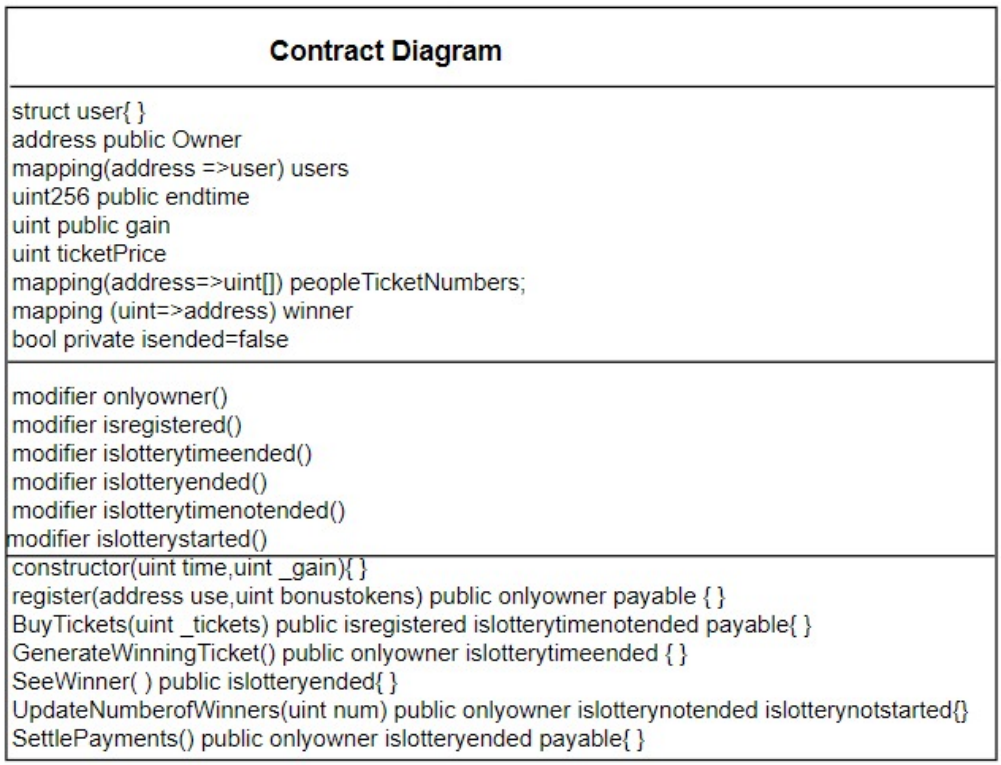
Use case Diagram:



- The owner has to perform three Functions i.e. Registering the users on to the lottery System, Generating a random winner and declaring the winner and Settling the winning amount to the winner.

- The user can perform three Functions i.e. buy LWT tokens if they need and they see the winner of the lottery they participated in.

Contract Diagram:



The data definitions used in the smart contracts are:

- Struct User{ }: A structure is defined for every user it has the Registration status of the user and the number of tickets they have purchased.
- address public Owner: This has the address of the Owner of the lottery.
- mapping(address =>user) users: This will have the address of all the participants registered on the lottery System.
- uint256 public end time: This variable will have the time at which the contract ends.
- uint public gain: This variable will have the gain % of the owner from the lottery
- uint ticketPrice: This variable will have the price of the ticket of the lottery.
- mapping(address=>uint[]) peopleTicketNumbers: This variable will have the ticket numbers assigned to the concerned Address of the participants.
- mapping (uint=>address) winner**:** This variable will have the address of the winner.

- bool private isended=false: This variable specifies if the lottery is ended or not.

Modifiers:

- onlyowner() : This checks if the operation is performed by owner or not
- isregistered(): This modifier checks if the participant is registered or not.
- islotterytimeended(): This modifier checks if the time allocated for lottery is ended or not.
- islotteryended(): This modifier checks if the lottery is ended or not.(i.e. if the winner is declared or not)
- islotterytimenotended(): This modifier checks if the time allocated for lottery is not ended.

Functions:

- constructor(uint time, uint _gain){ }: This is called when the smart contract is deployed . This assigns endtime with the time and the gain with the gain values used by the Owner while deploying the Smart Contract.
- register(address use, uint bonustokens) public onlyowner payable { }: This function is used by the owner of the lottery to register the user. By registering a bonus token of 50LWT were giving to the users/Participants. This operation should be performed by the owner only so the modifier onlyowner is used.
- BuyTicket(uint _tickets) public isregistered islotterytimenotended payable{ }:This function is used by the users/participants to buy the lottery Tickets. In order to buy the ticket the lottery should not end and the user should be registered so the modifiers isregistered and islotterytimenotended are used.
- GenerateWinningTickets() public onlyowner islotterytimeended }: This function generates a random winner . In order to call this the lottery time should be ended and only the owner should call it. So the modifiers only owner and islotterytimeended are used.
- SeeWinner( ) public islotteryended{ }: This function is used to see the winner of the lottery. But in order to see the winner of the lottery the lottery should have been ended by then so the modifier islotteryended is used.
- SettlePayments()public onlyowner islotteryended payable {}:This operation settles the gain tokens to the owner and the winning tokens to the winner .In order to settle Payment the lottery should be ended so a modifier islotteryended and onlyowner is used.

Sequence Diagram:

Here are the sequence of steps performed in the Smart lottery System:

1. The owner registers the participants into the lottery System.
2. The participant has to buy the lottery tickets and the smart contract checks if the lottery time is ended or not and checks if the user is registered or not.
3. The owner decides to declare the winners. The smart contract checks if the lottery time ended or not and generates a random winner

4. The users can use the seeWinner function to get the address of the winner.The smart contract checks if the lotteryended or not and then returns the user with the winner's Address.

5. Settle Payment: During settle Payment Function the funds i.e. the gain is sent to the owner and the winning amount is sent to the winner using transfer function from ERC20 Smart Contract.

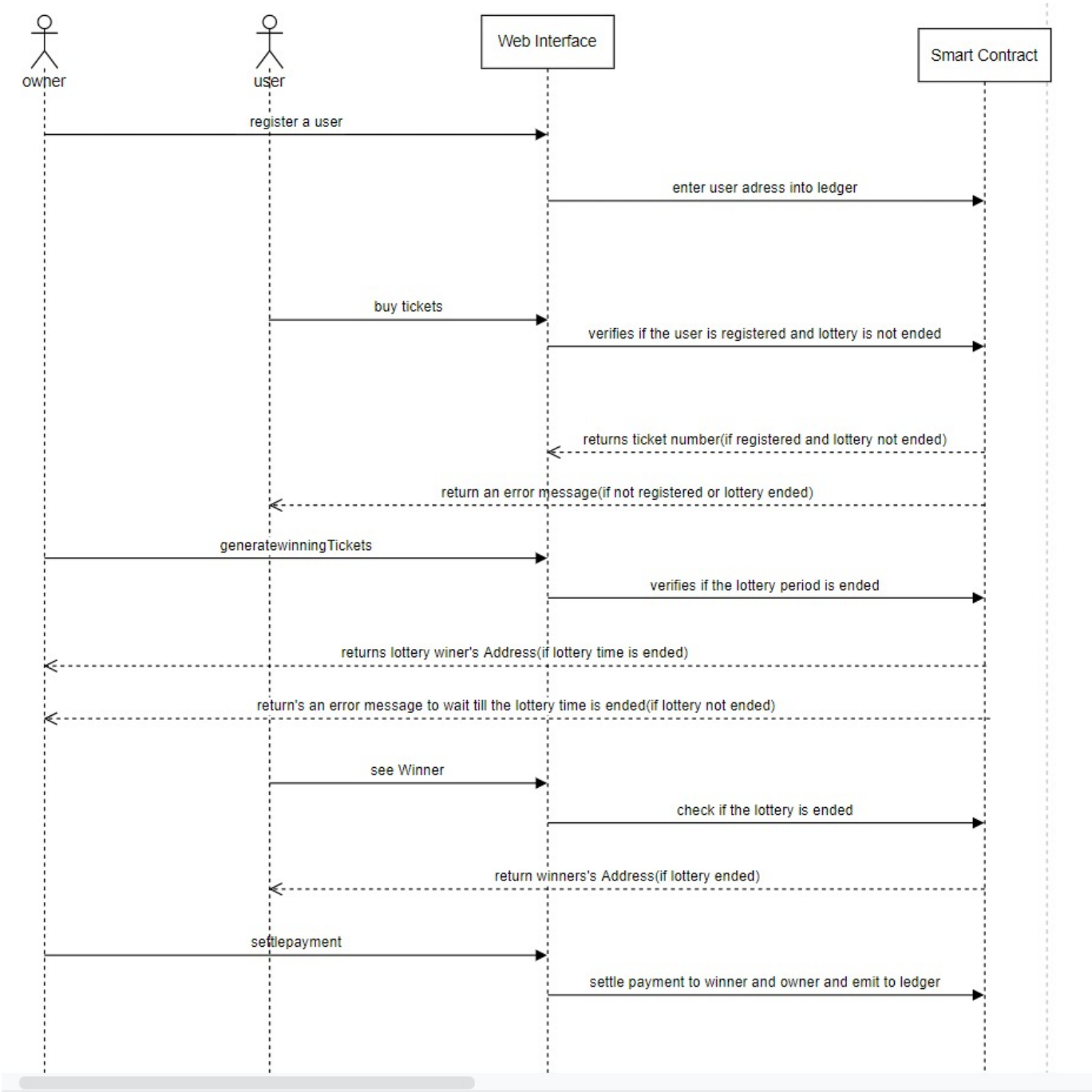
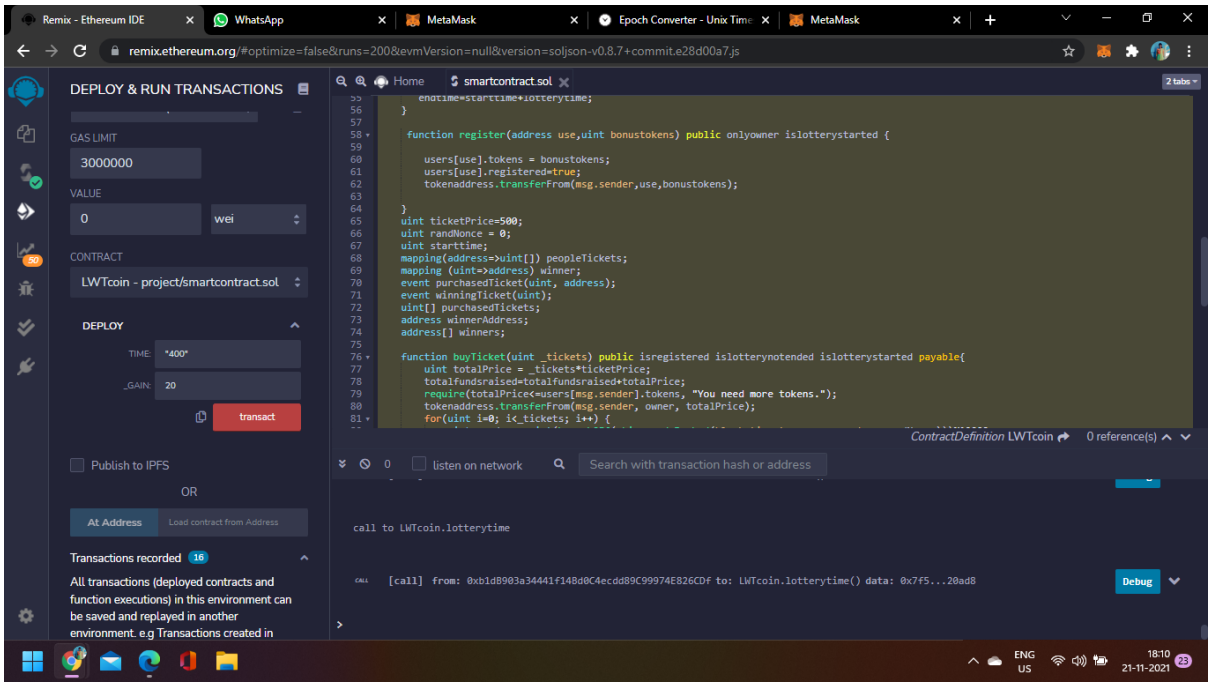


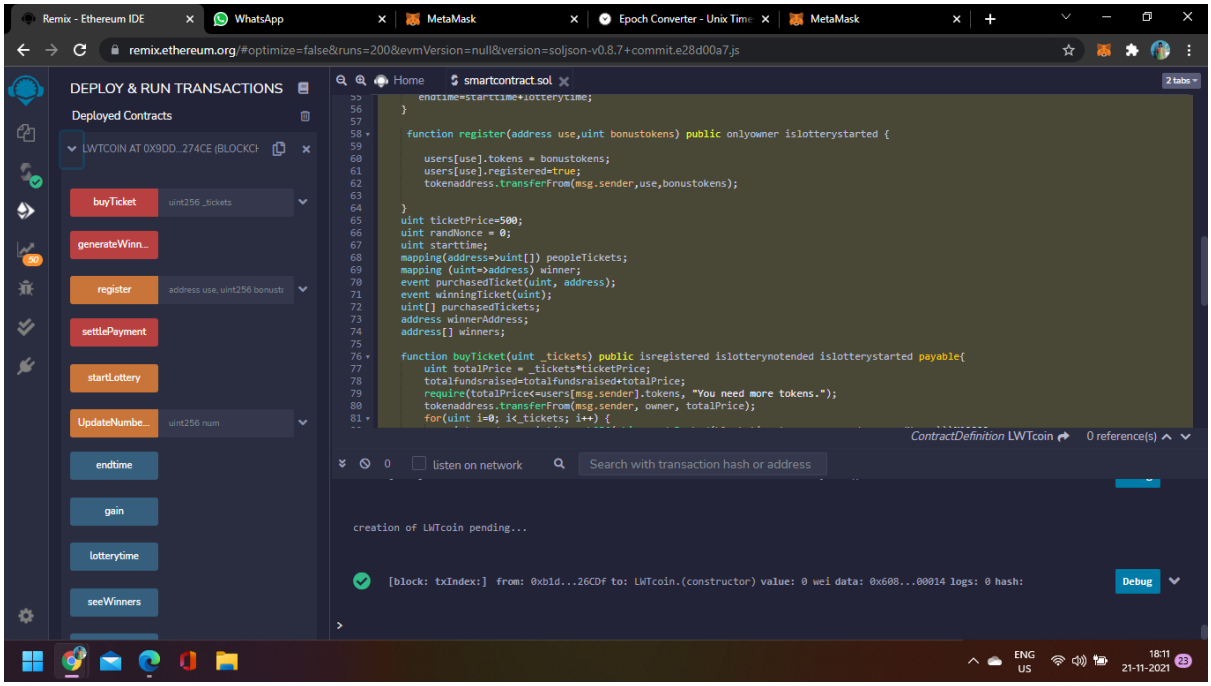
Figure : Sequence Diagram for Smart lottery system

Screenshots of working smart contract functions:

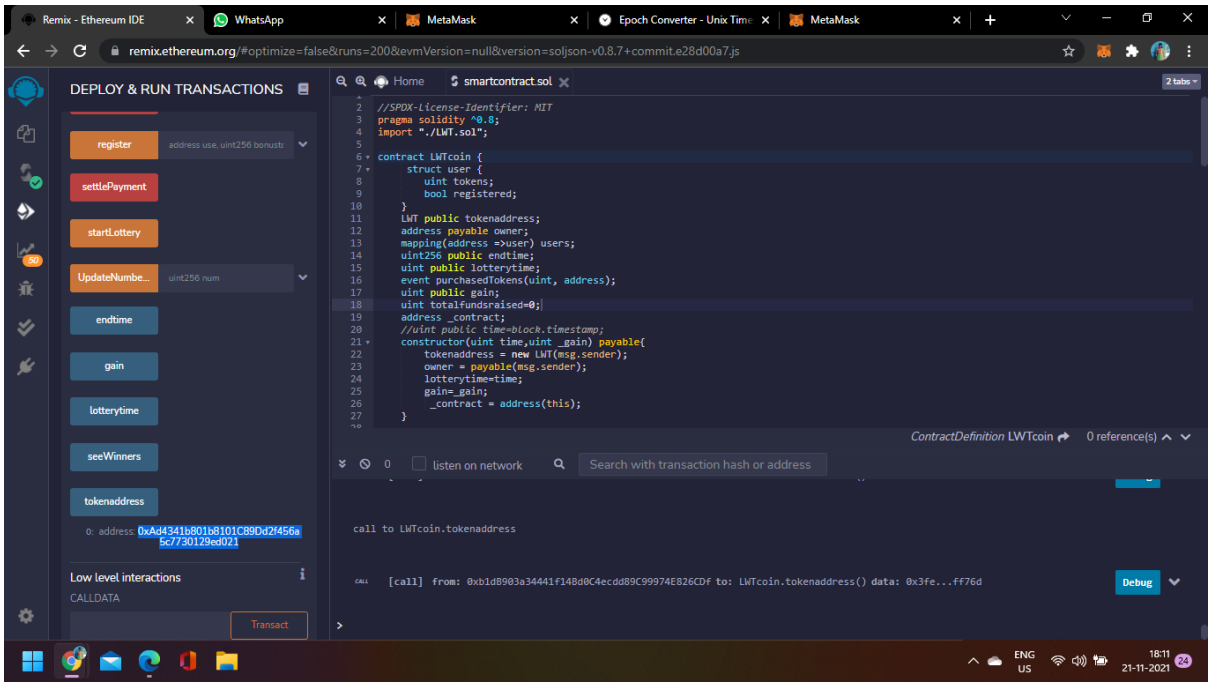
1.deploying smart contract(400 is the lotterytime and 20 is the gain% of the owner)

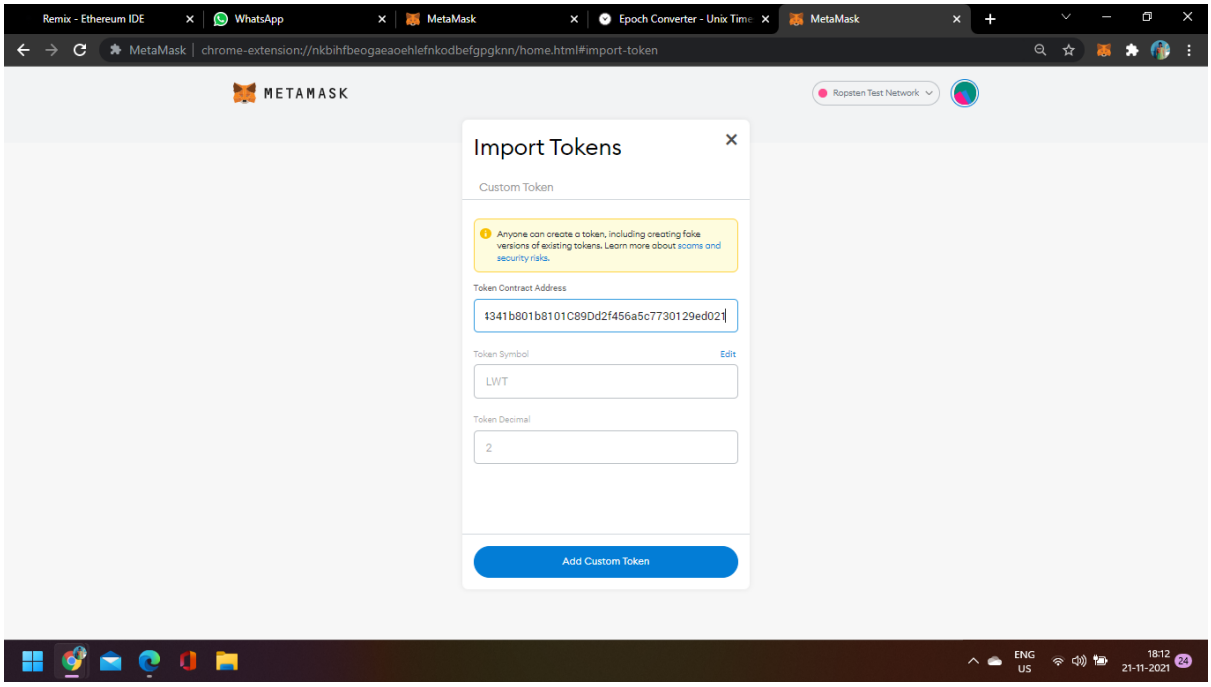


2.Functions in deployed smart contract

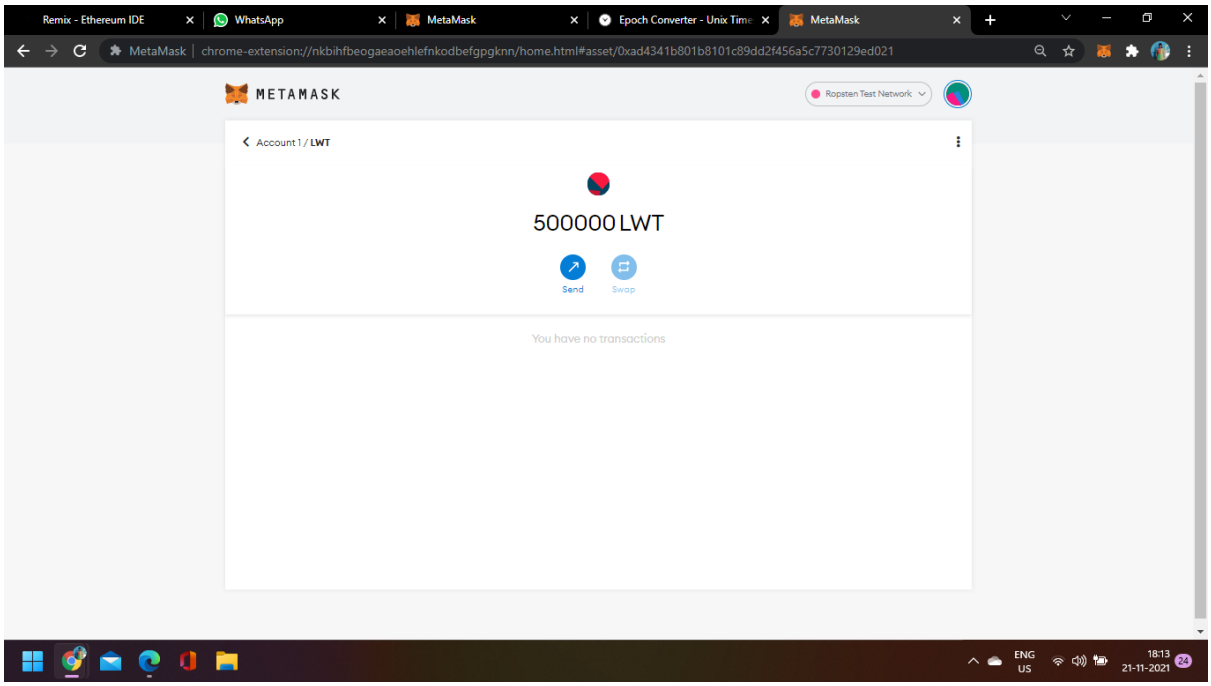


3.Importing LWT token using deployed smart contract

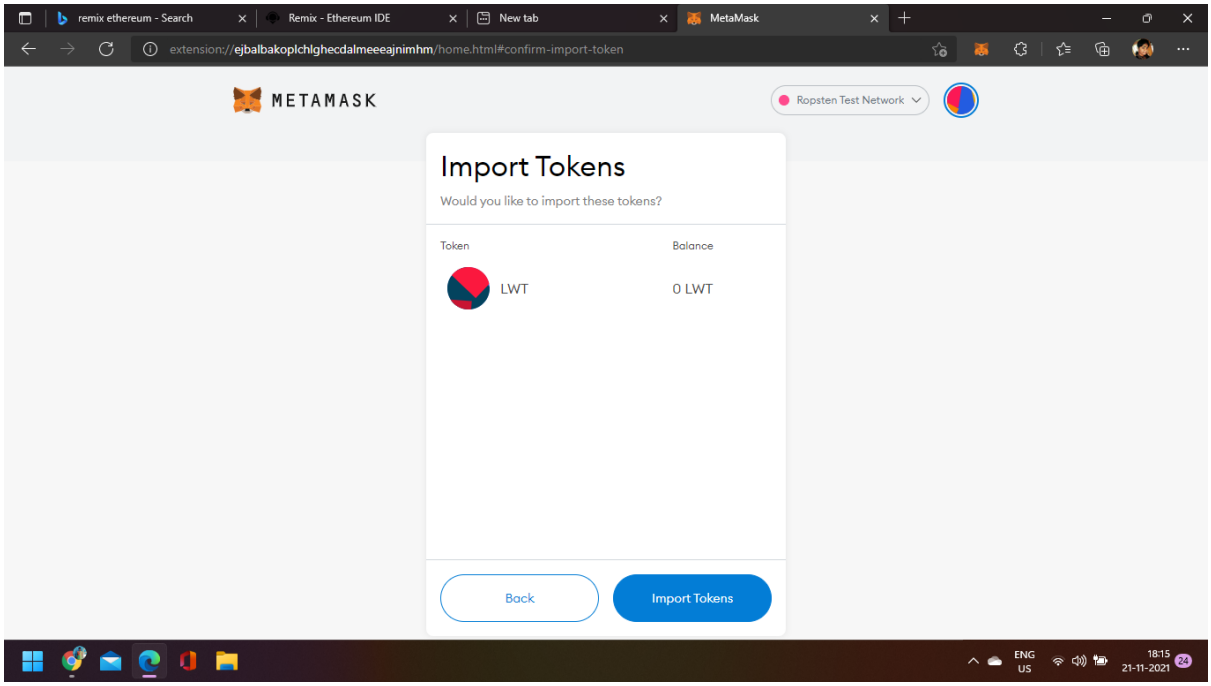




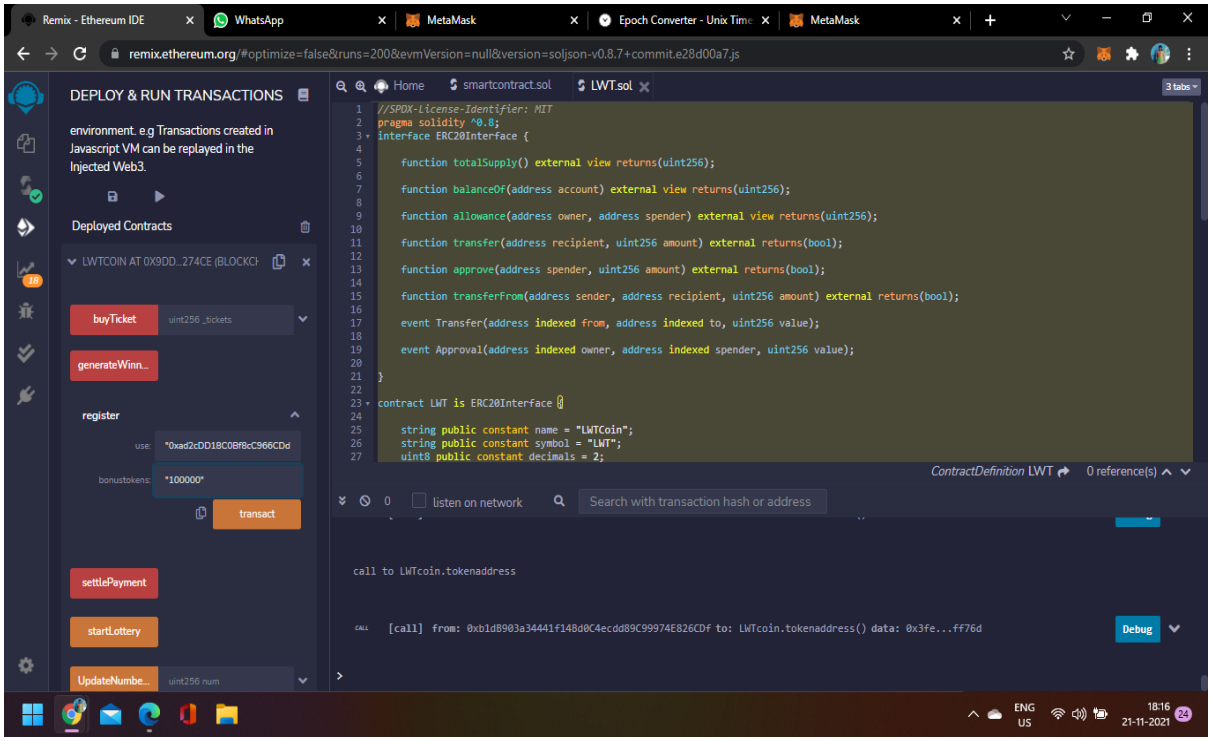
Importing tokens for Owner:

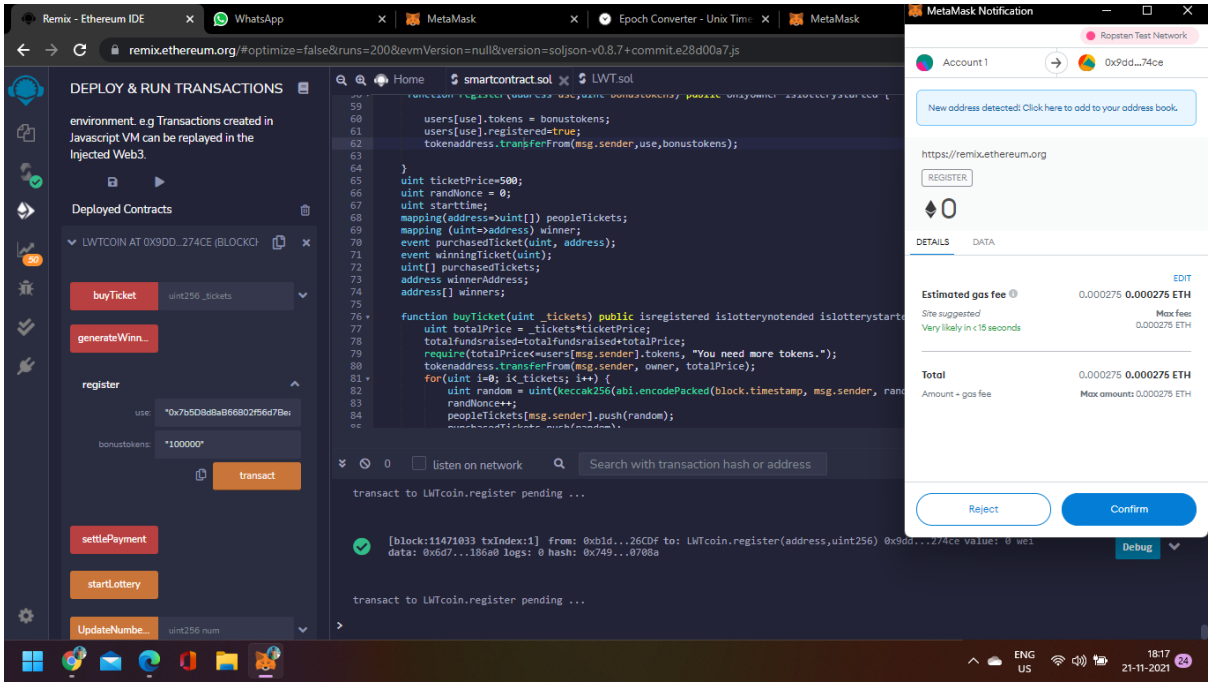


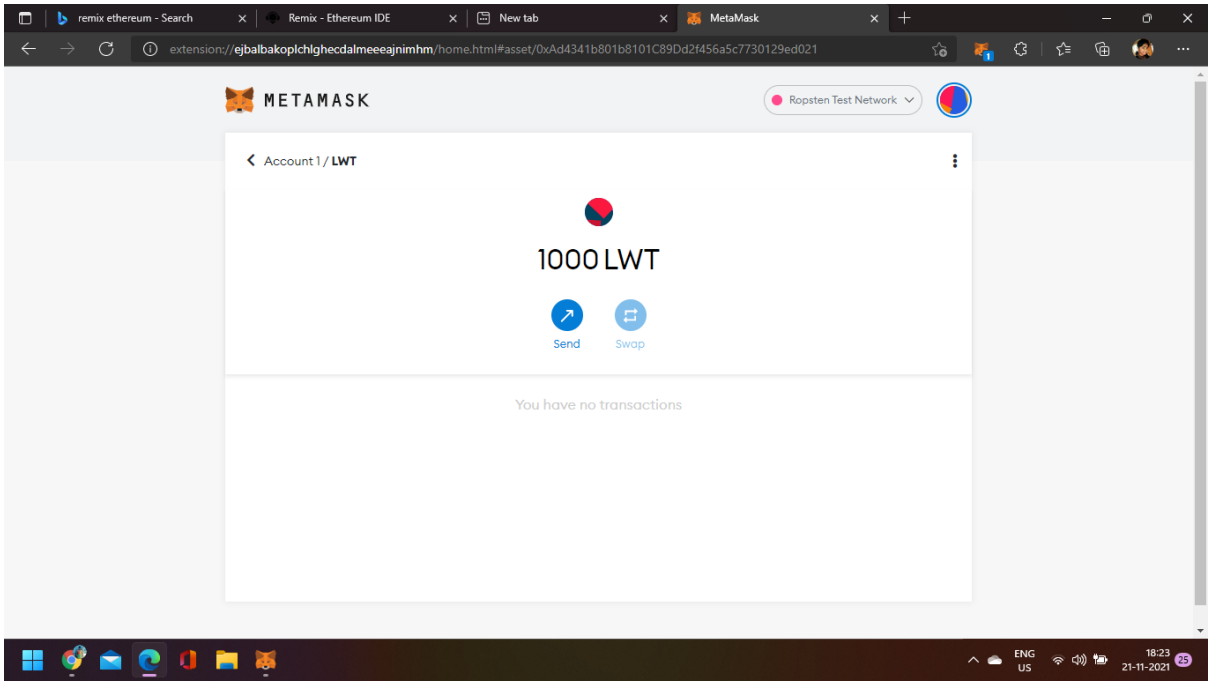
Importing tokens for user:



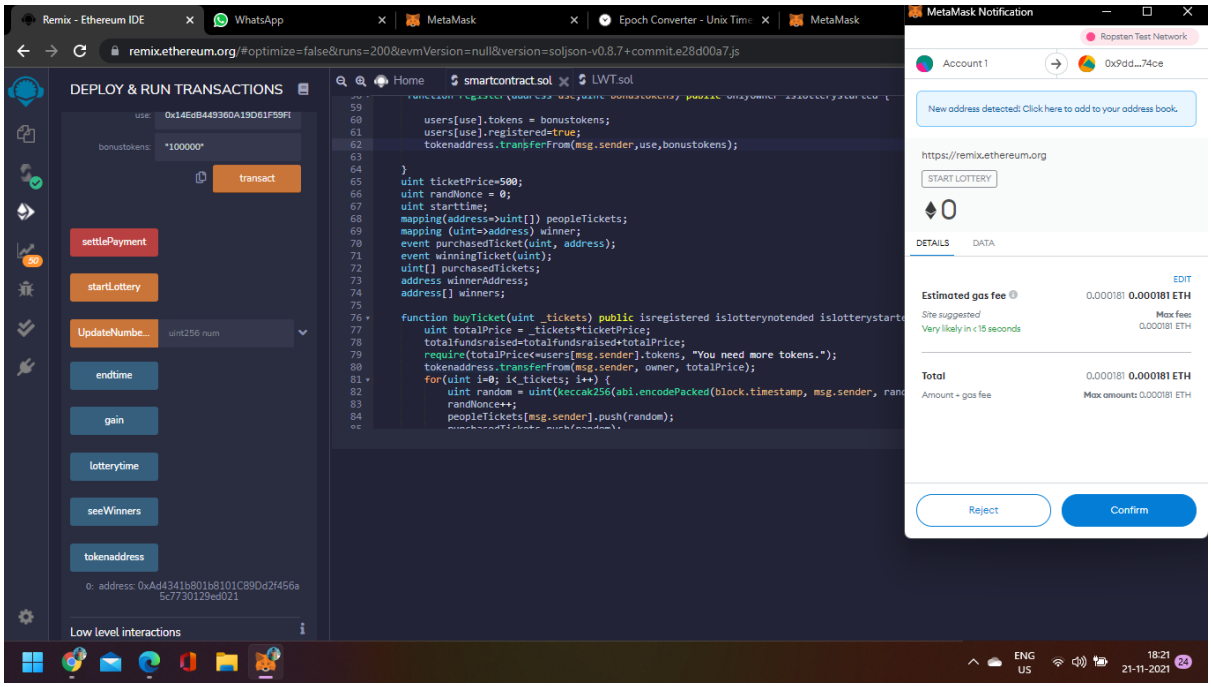
4.Owner Registering user: (Owner has to provide address of the user and bonus token he wants to give them)



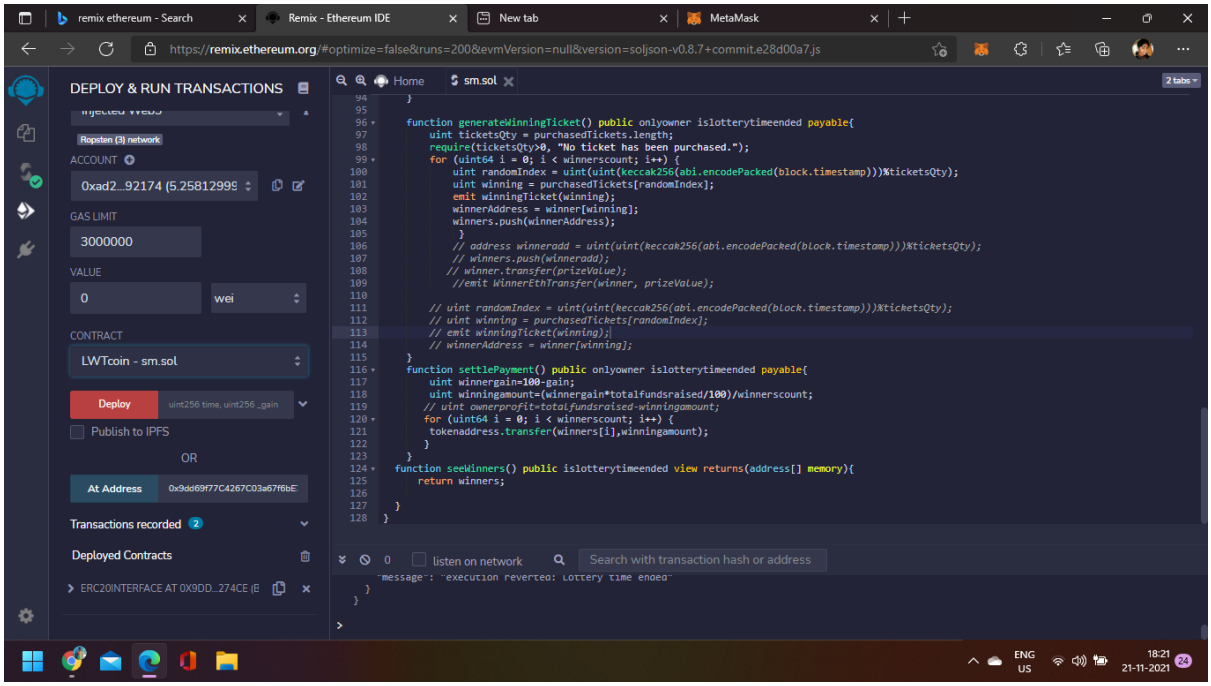


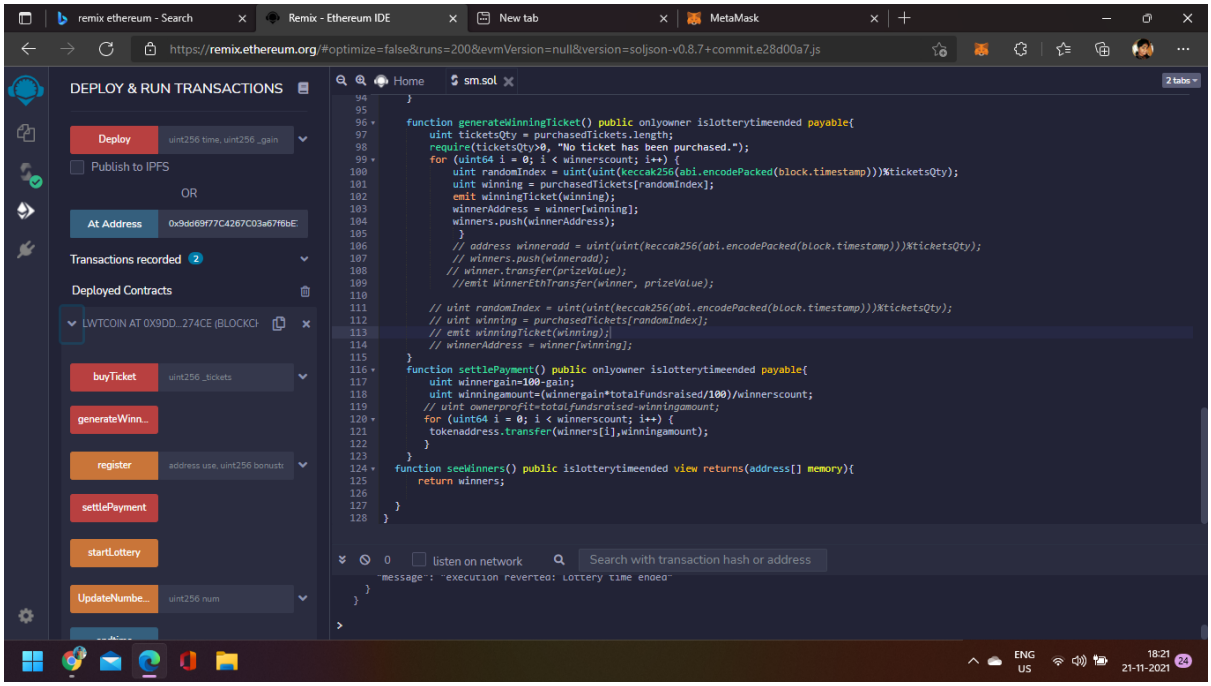


5. Starting lottery

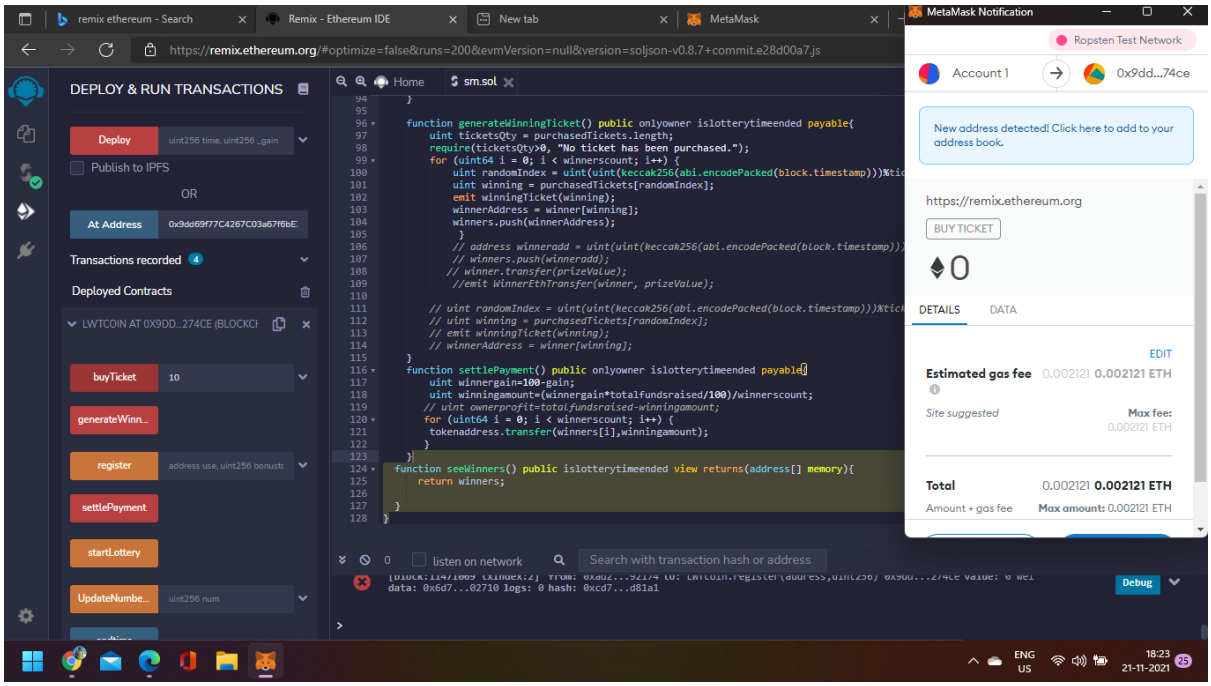


6. Using smart contract address deployed and placing it in “At Address” and accessing the functions by user:

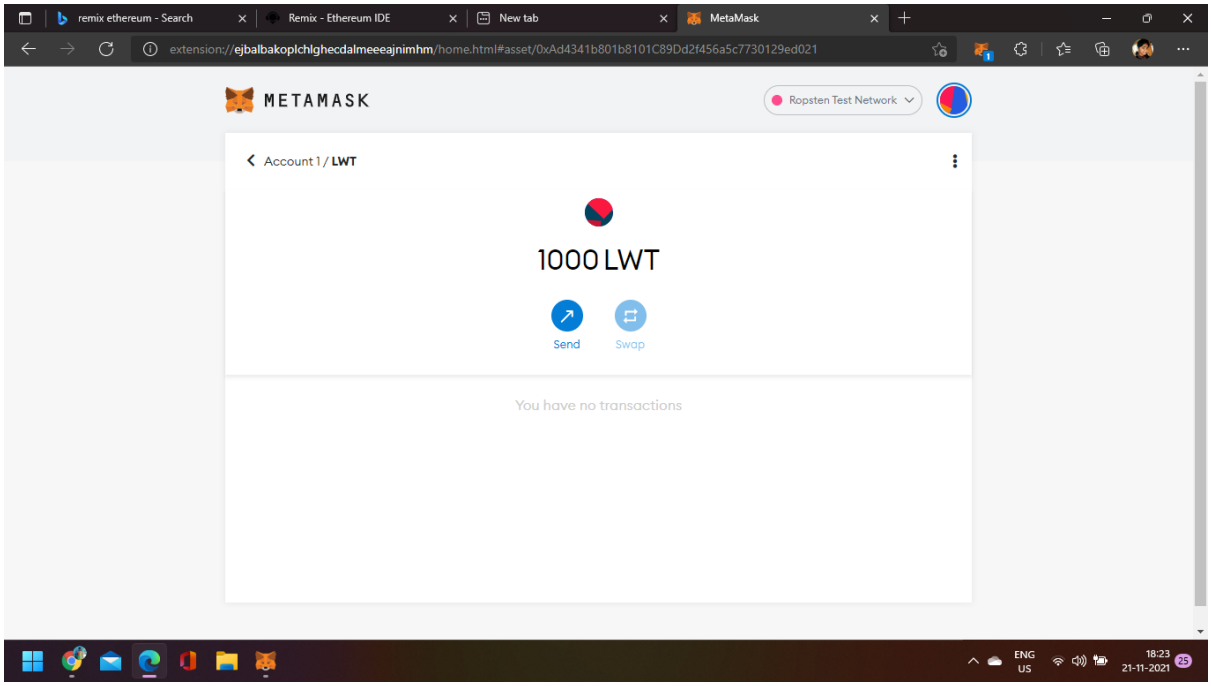




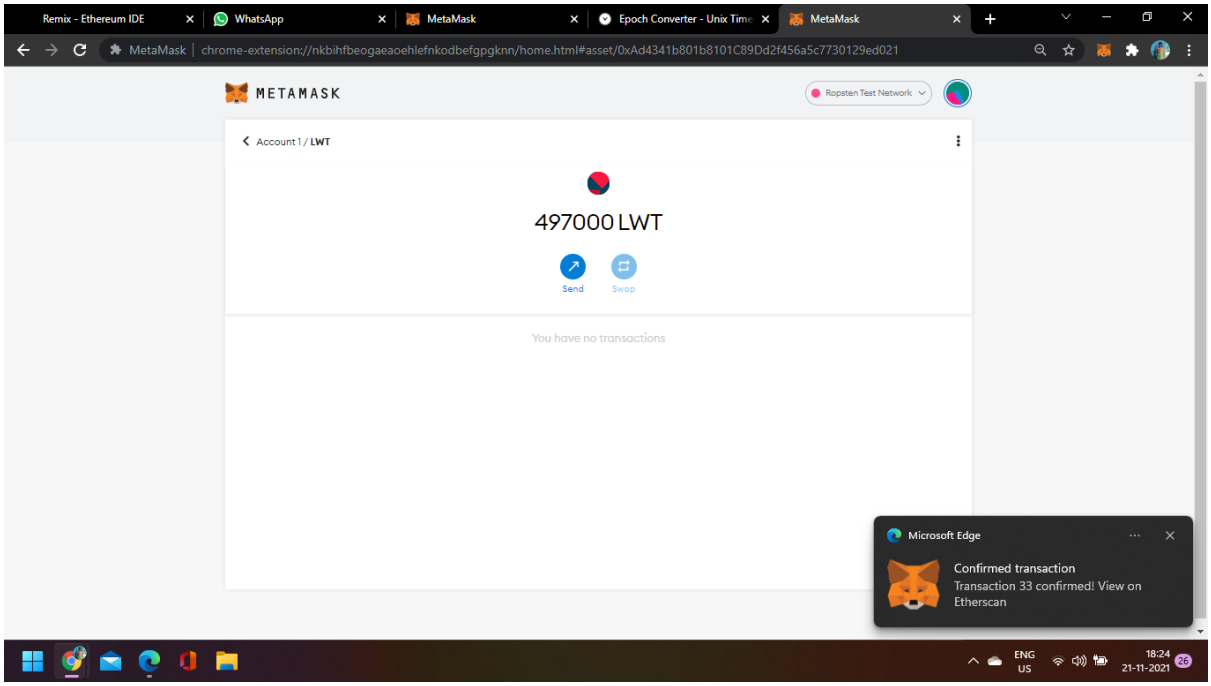
7. Buying ticket for lottery on participant side



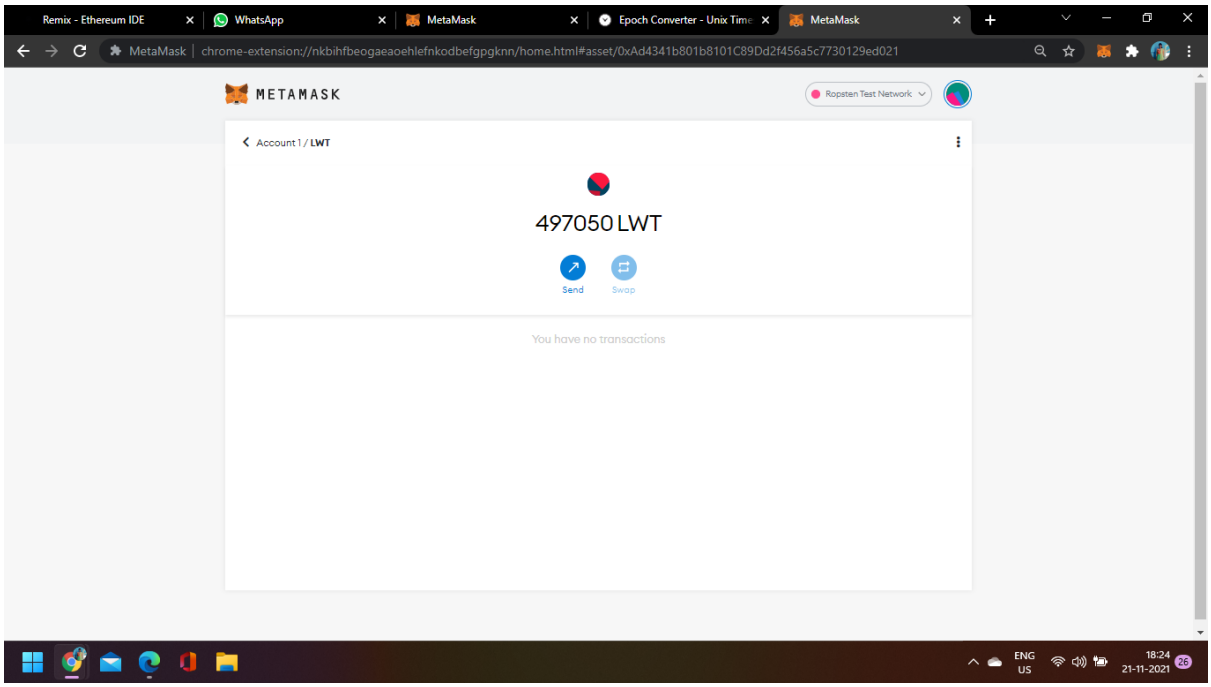
8. Bonus given to registered users in lottery by owner



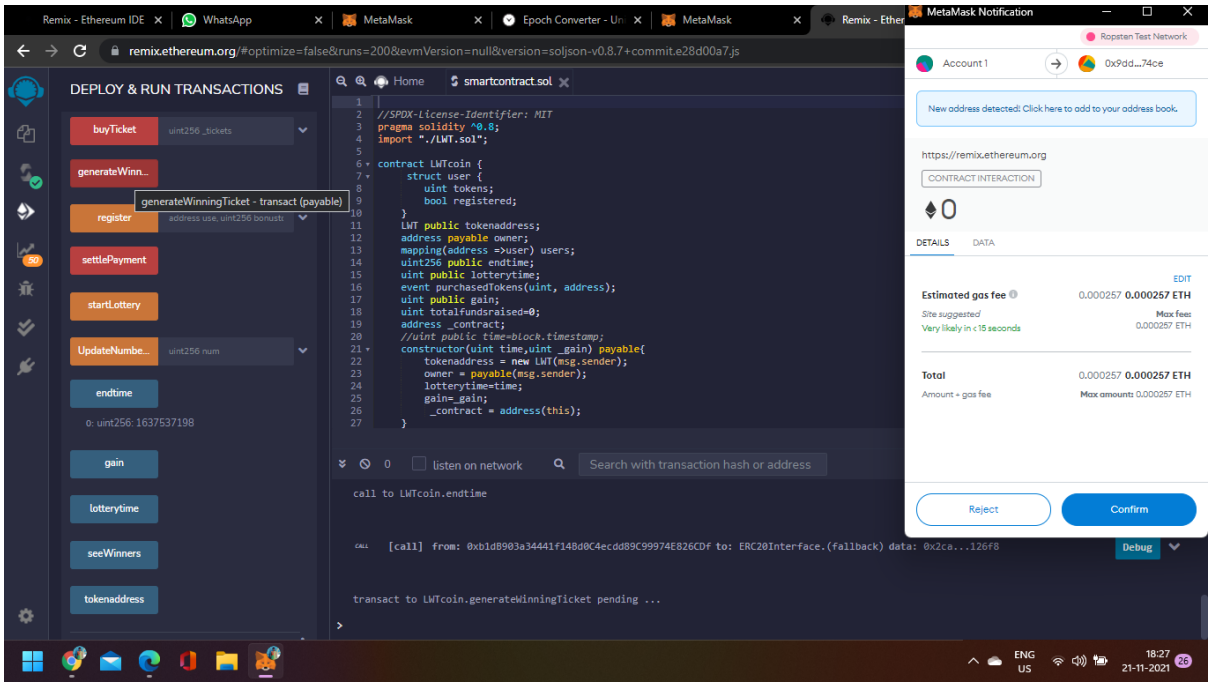
9. Tokens on owner side after registering 3 participants and giving them bonus tokens



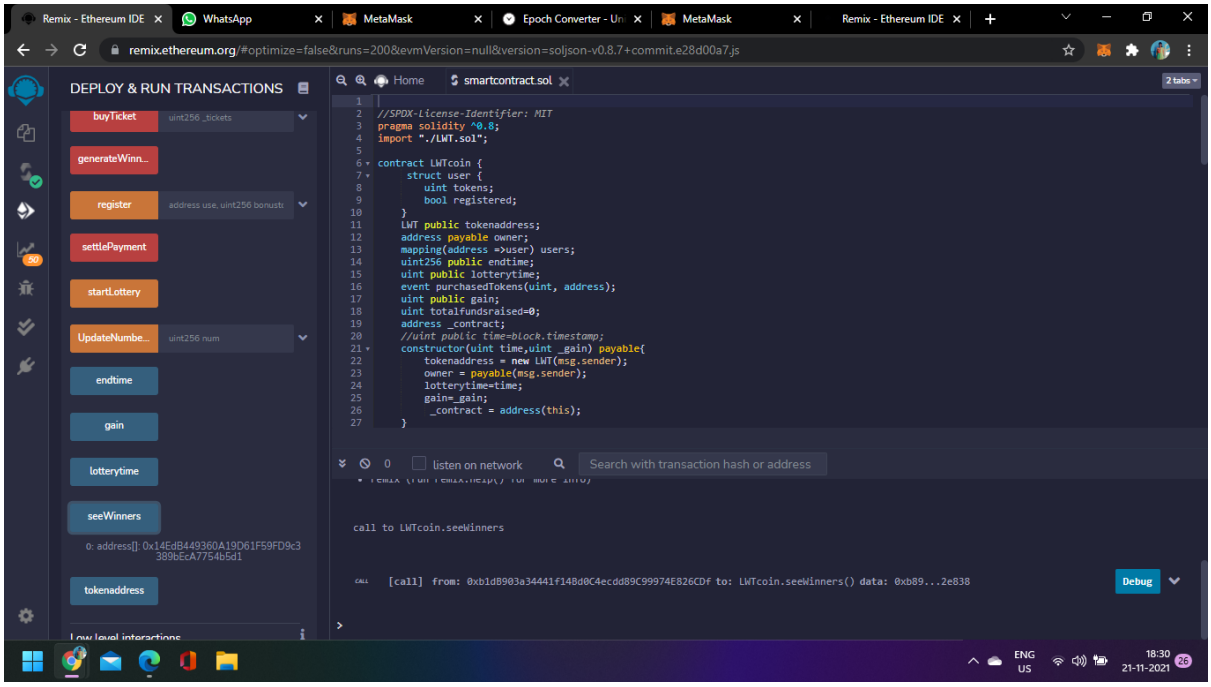
10.After buying tickets by participants the tokens on owner side



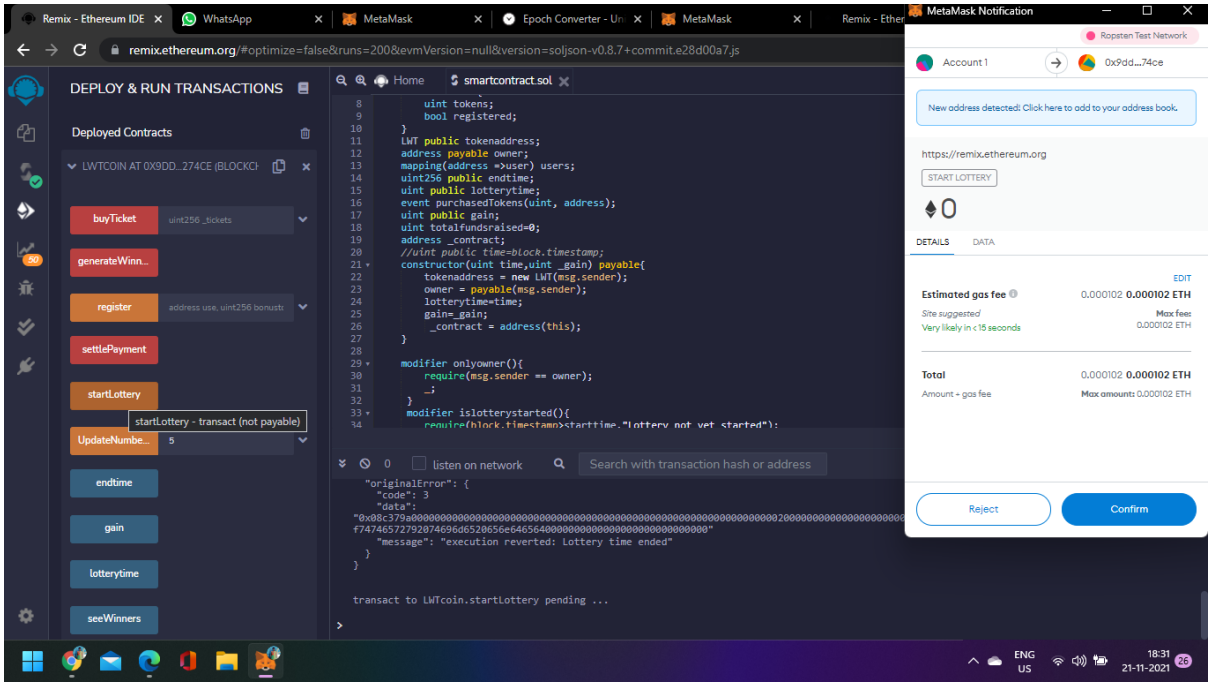
11.Generating winners in the lottery on owner side



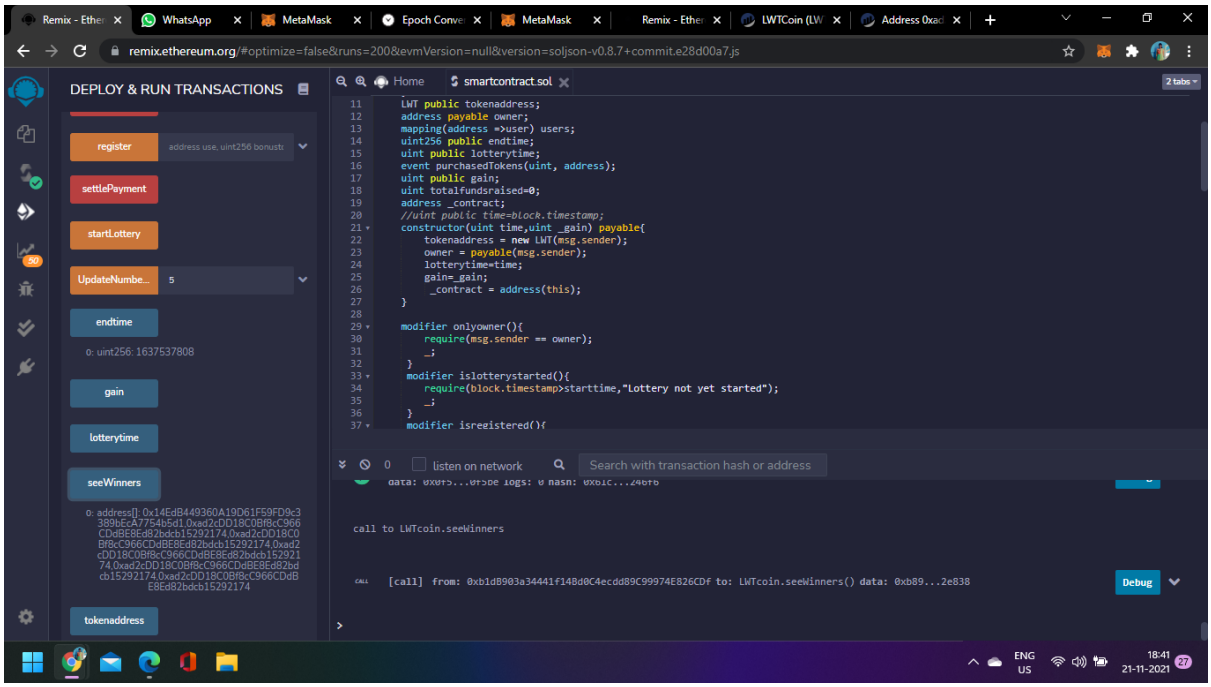
a) Single winner in lottery(see winners will give the address of the winners)



12.Similarly after starting a new lottery with 5 winners in the lottery:

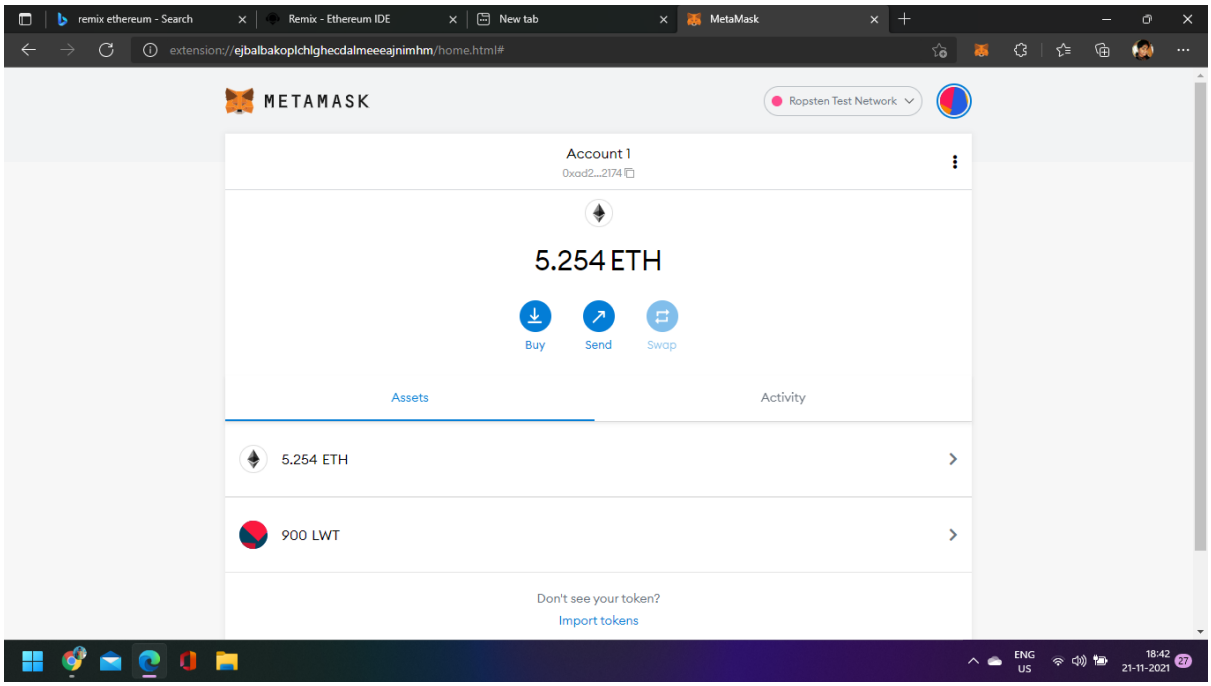


a) Winner addresses for the updated lottery system

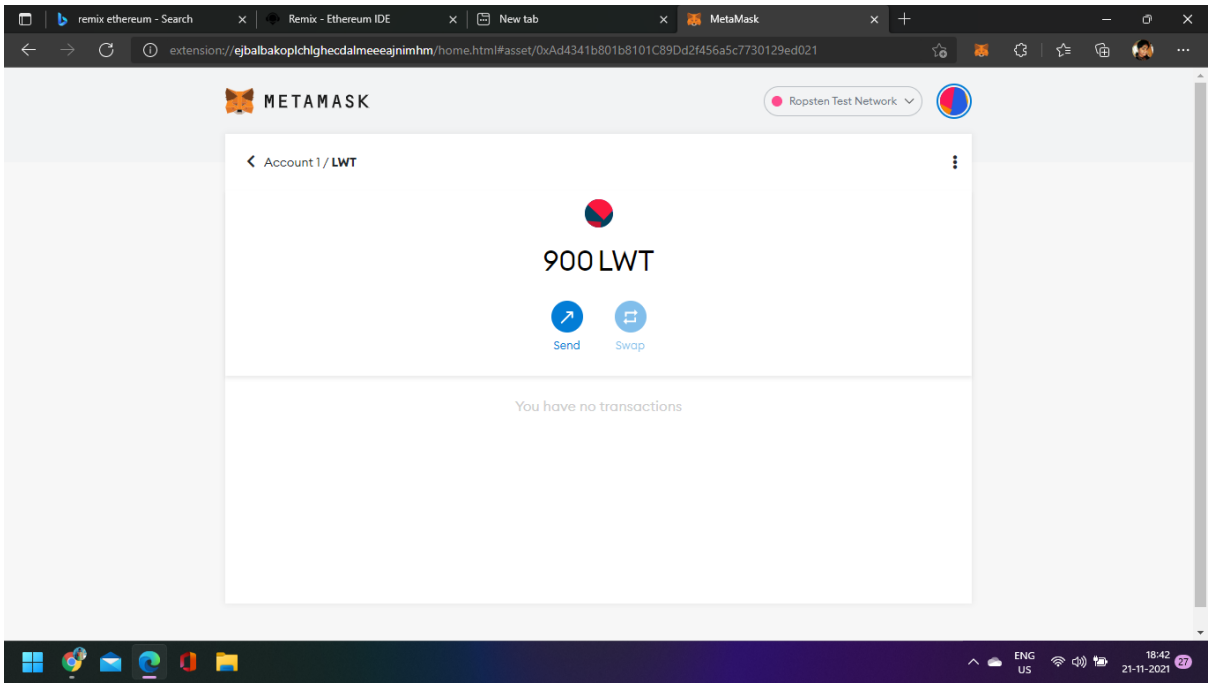


13.Settle payment for both owner and winners

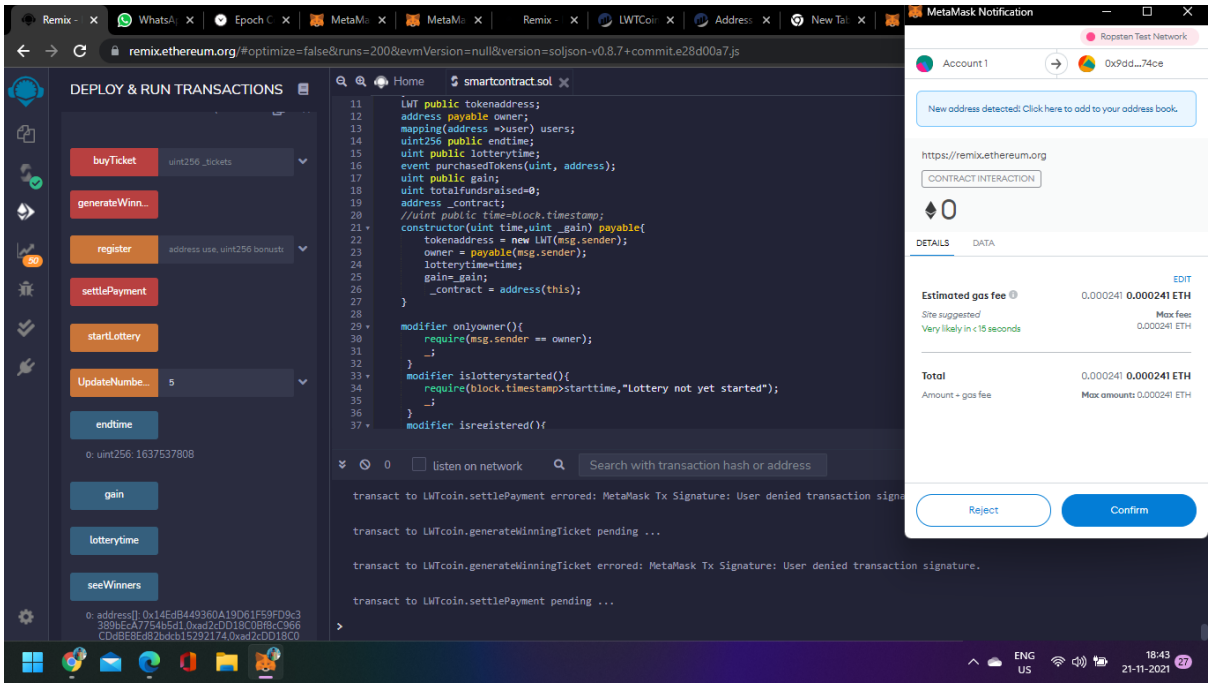
a) Initial tokens on first winner address wallet



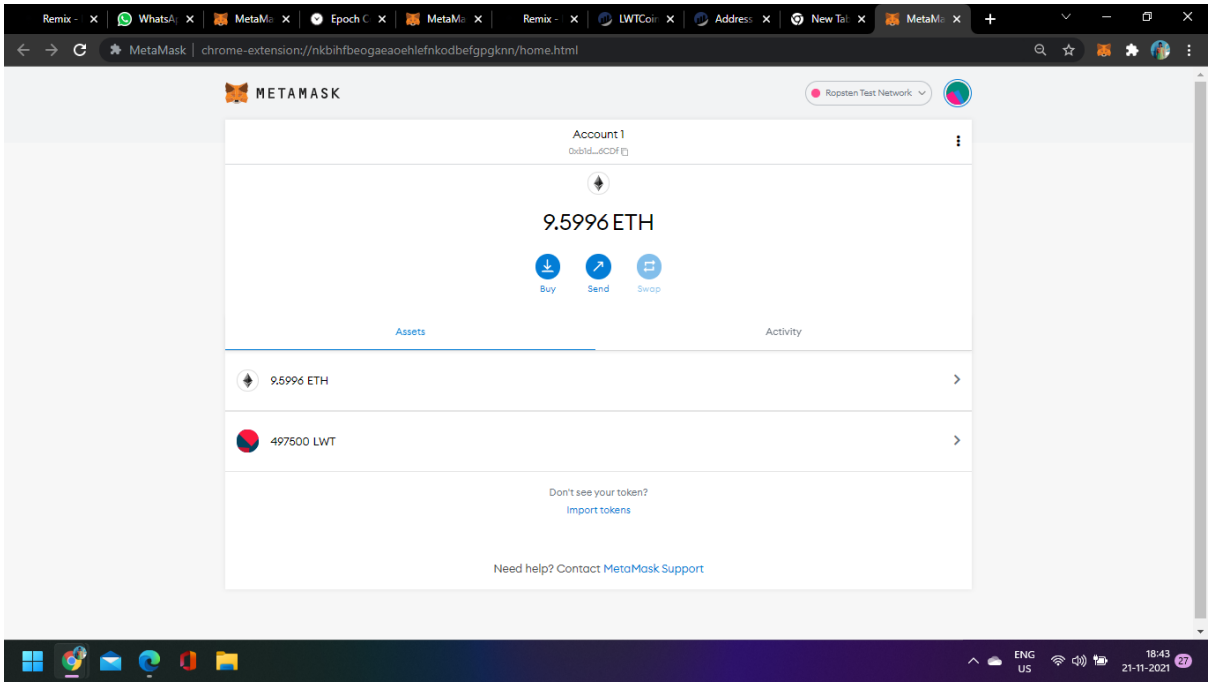
Initial tokens on the second winner before payment settlement:



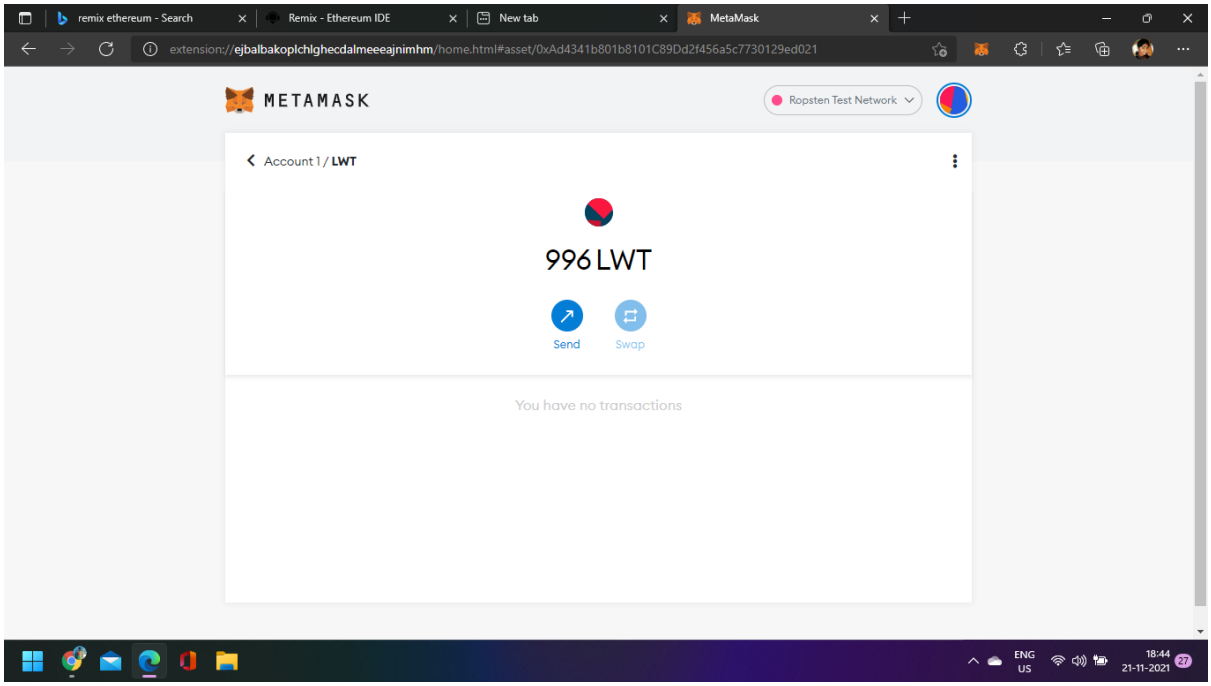
b) Settling payment by owner



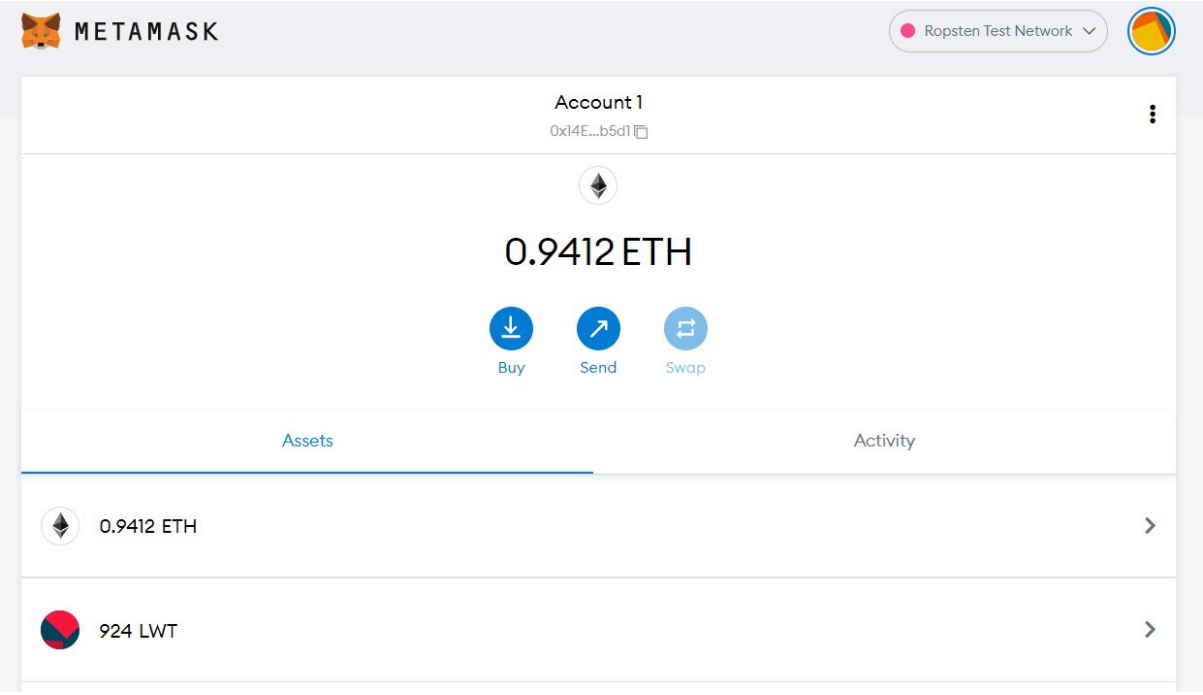
c) Tokens on owner side after payment is settled



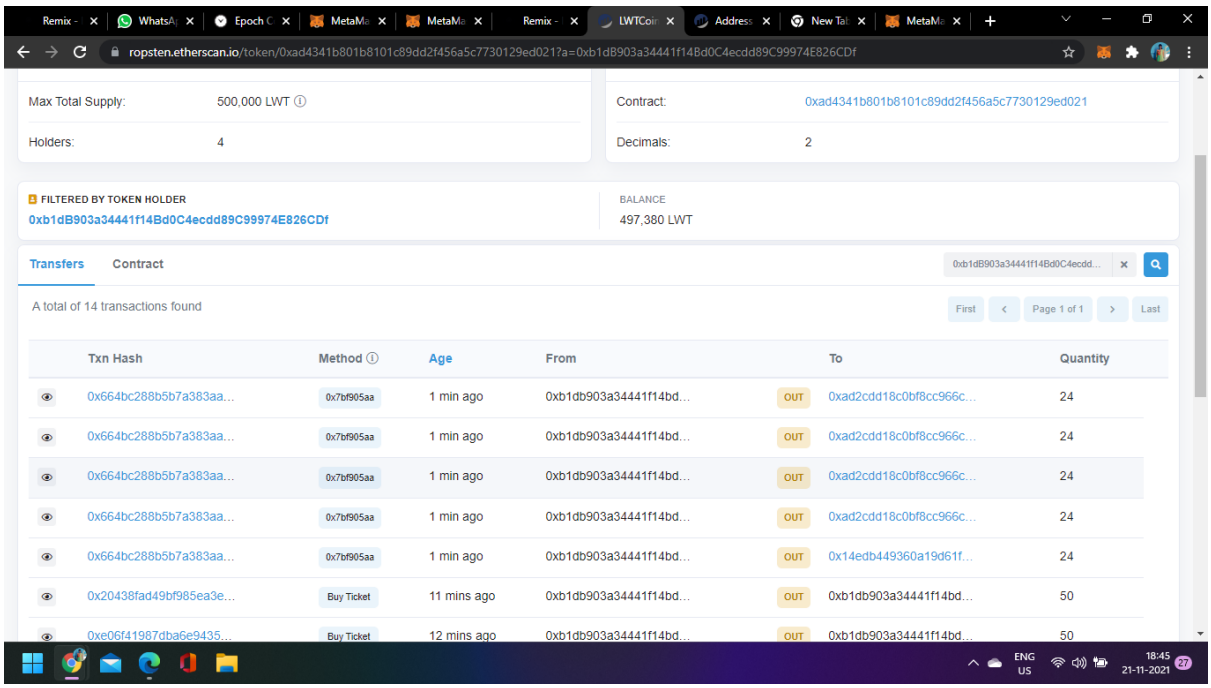
d) Tokens on winner side after payment settlement



e) Tokens on second winner side after payment is settled



14. Transactions on etherscan done by owner of lottery(the winning amount is distributed equally among the 5 winners)



Phase -4 DAPP for Smart Lottery System

- A decentralized application(DAPP) is an application built on a decentralized network that combines a smart contract and a frontend user interface.
- On Ethereum, smart contracts are accessible and transparent like open APIs so Dapp can even include a smart contract that someone else has written.
- A dapp has its backend code running on a decentralized peer to peer network. Contrast this with a normal app running on centralized servers.
- A dapp can have frontend code and user interfaces written in any language to make calls to the backend.
- For our smart lottery system dapp,
    - We used solidity code for smartcontract .
    - We used bootstrap, Javascript for the smartlottery dapp .
    - We used web3 for connecting the dapp to smartcontract.
    - We deployed it on Ganache network.
    - We used Express framework for server purpose.

Addresses used by us during testing:

- Smart contract address:
- Token address:
- Owner address:
- Players addresses:

Dapp working Procedure and screenshots before and after running:

1. Smart contract is deployed by admin using his decentralized address.

2. Contract address obtained by migration is used to deploy smartcontract along with admin address.

Contract address: **0x39115f1dB9250AA71dEAE9E4D7ee96570e238dAA**

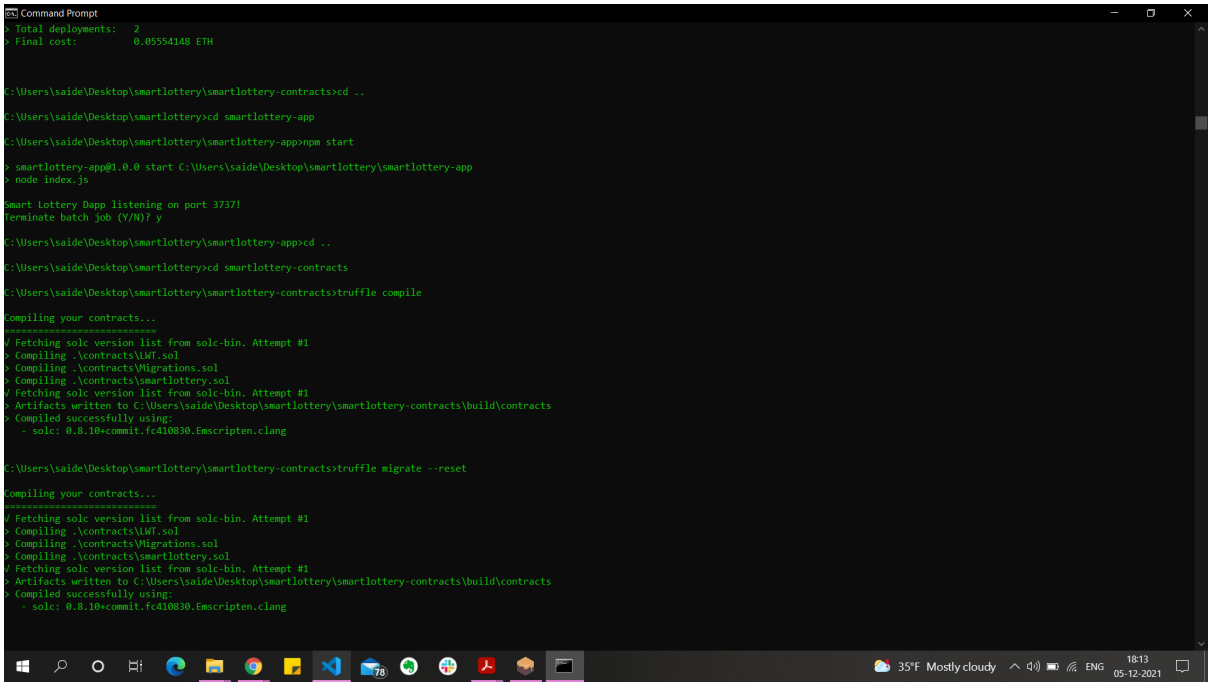


Figure: The above figure shows usage of truffle compile and migrate

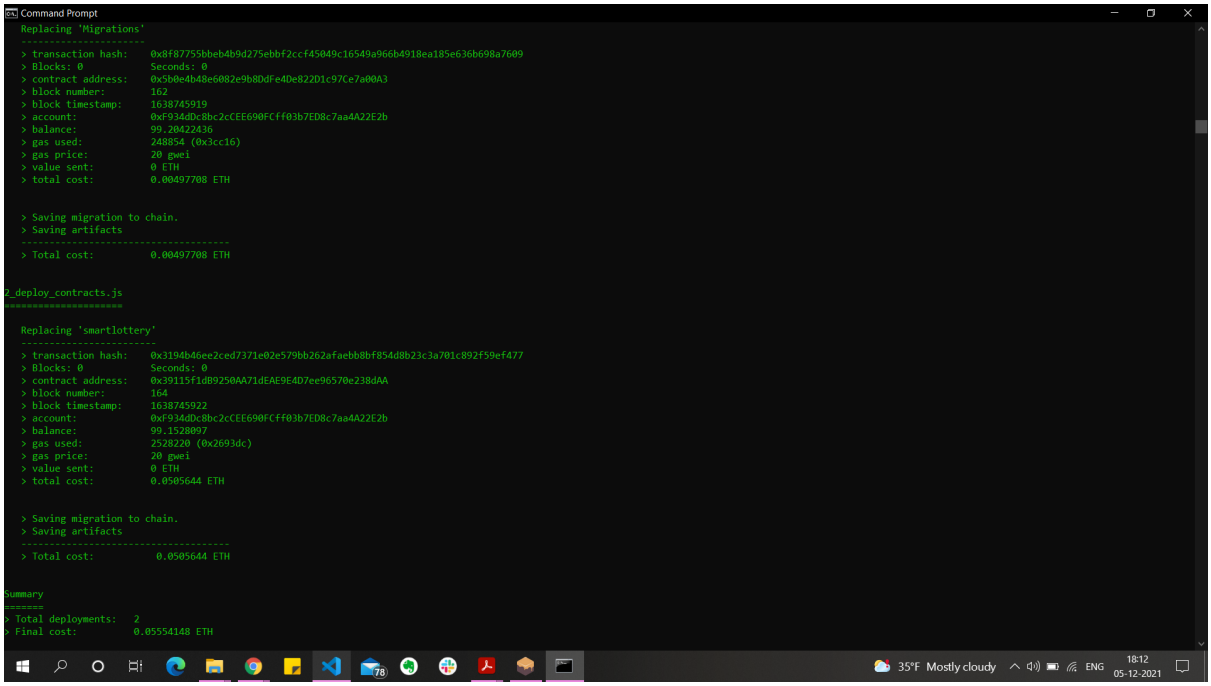
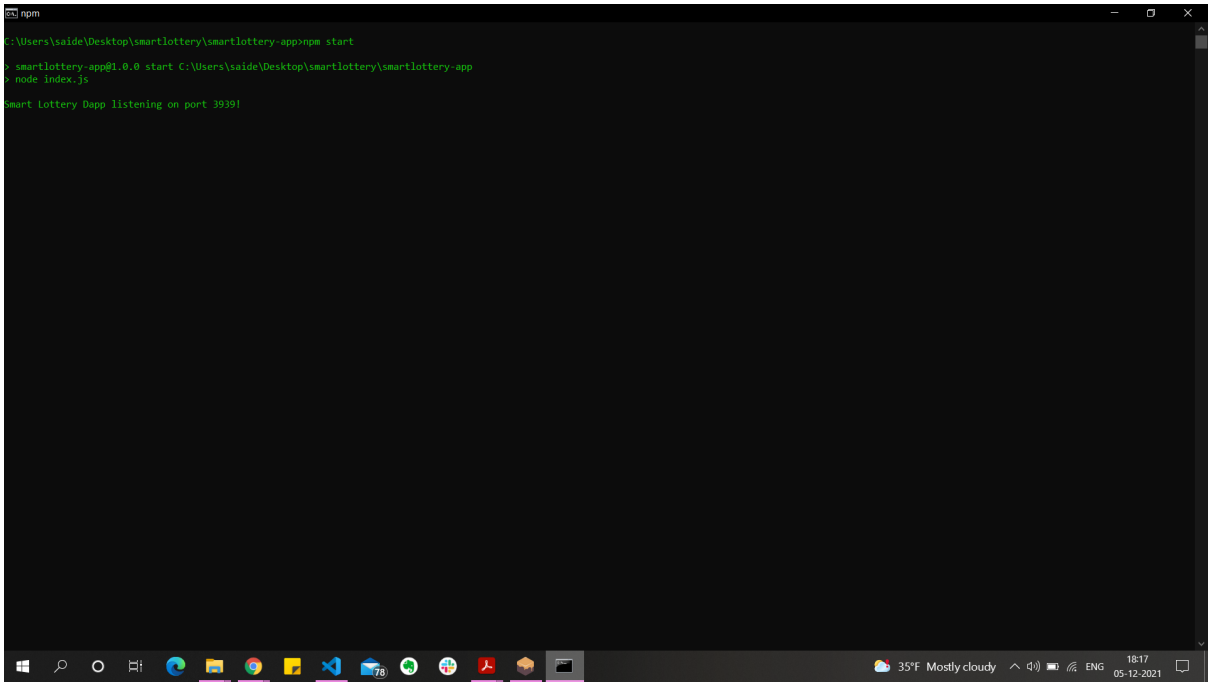
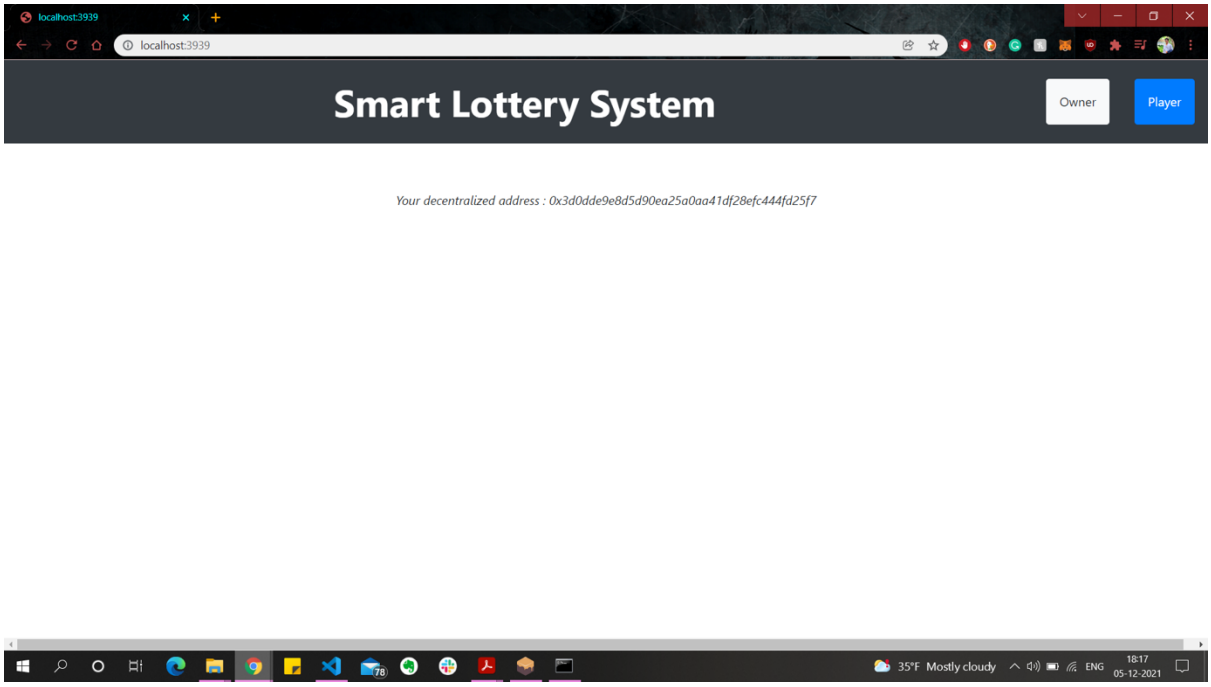


Figure: The above figure shows the result from “truffle migrate --reset”

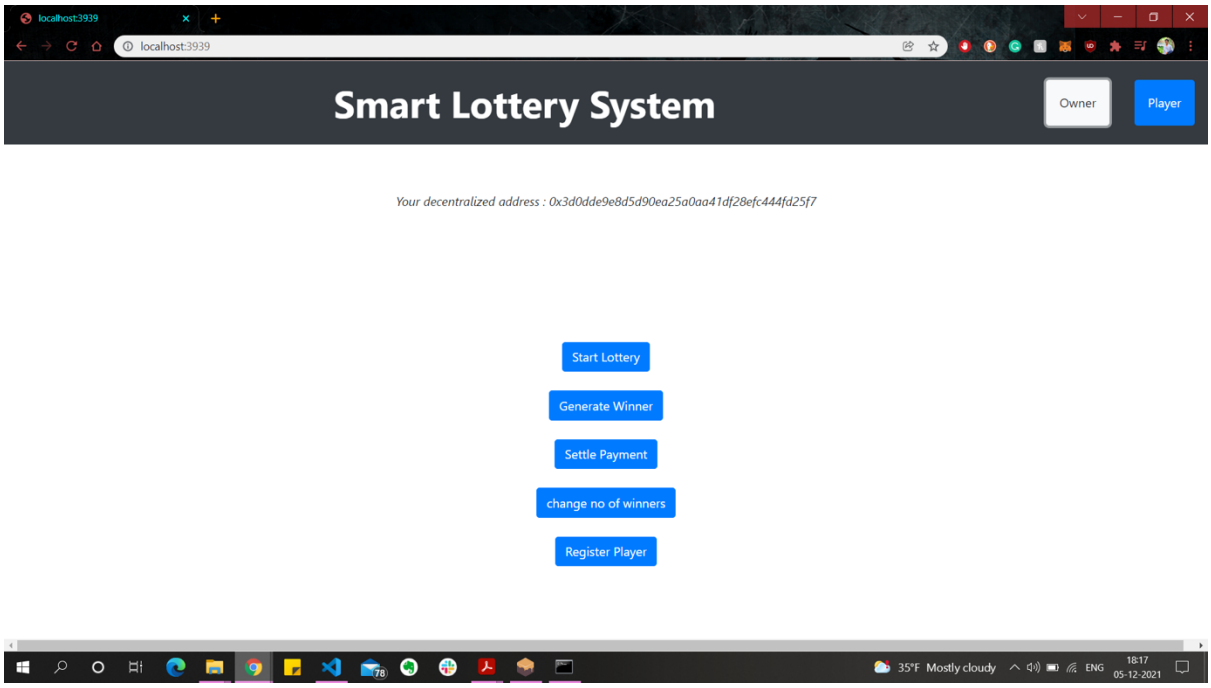
3. After successful migration, the dapp is started using “npm start” and UI is hosted on port specified “3939”, using the smartcontract address.

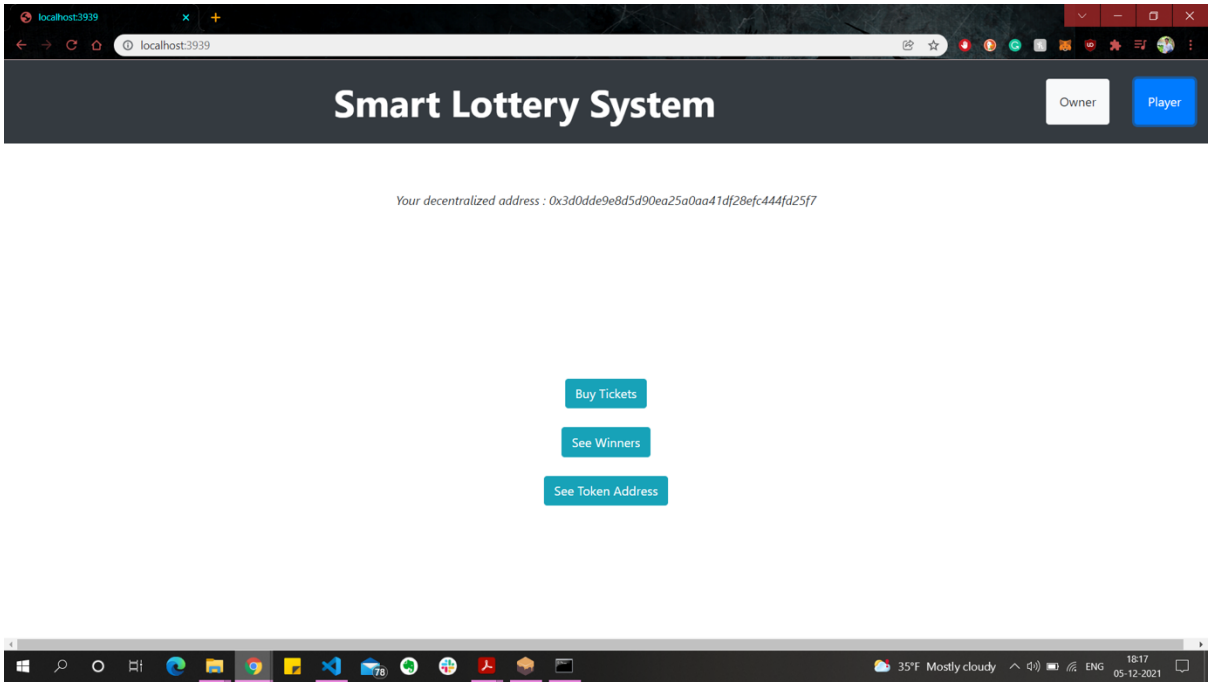


4. Each player or owner can access their functions through the UI by using the buttons specified.

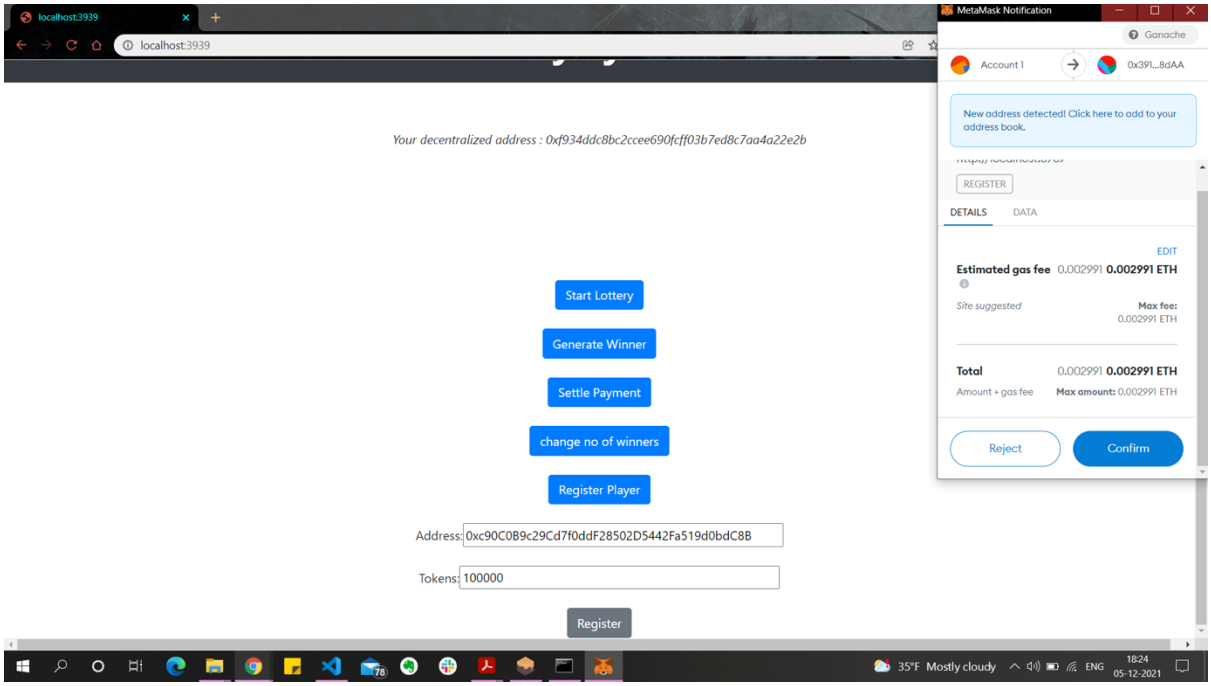


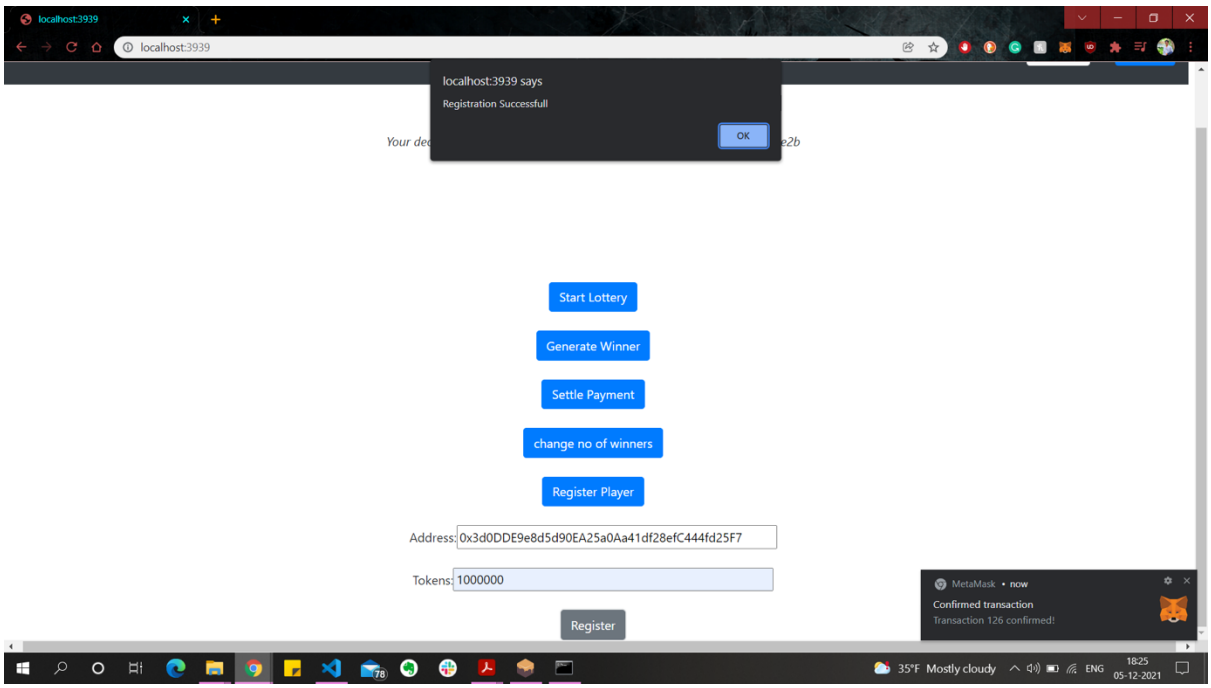
5.Below are the options for owner and player in each image.



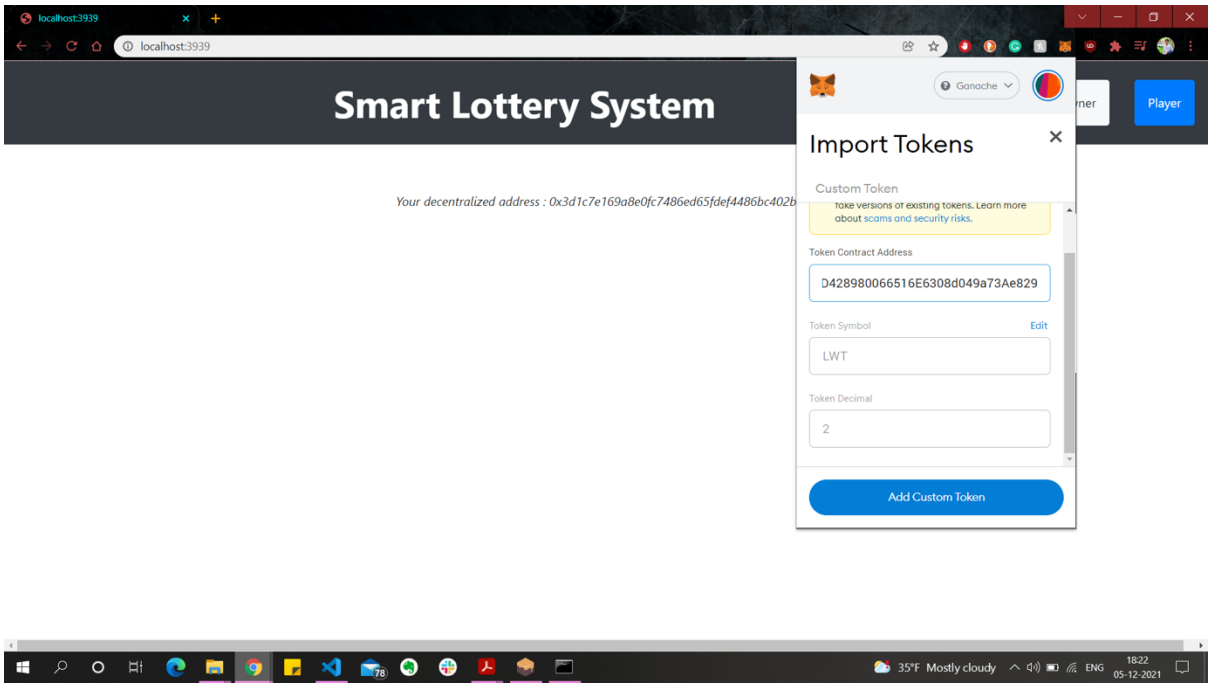


6. Only owner can register players to participate in lottery. The images clearly depict the information.





7. Each player must import our ERC-20 token “LWT”.



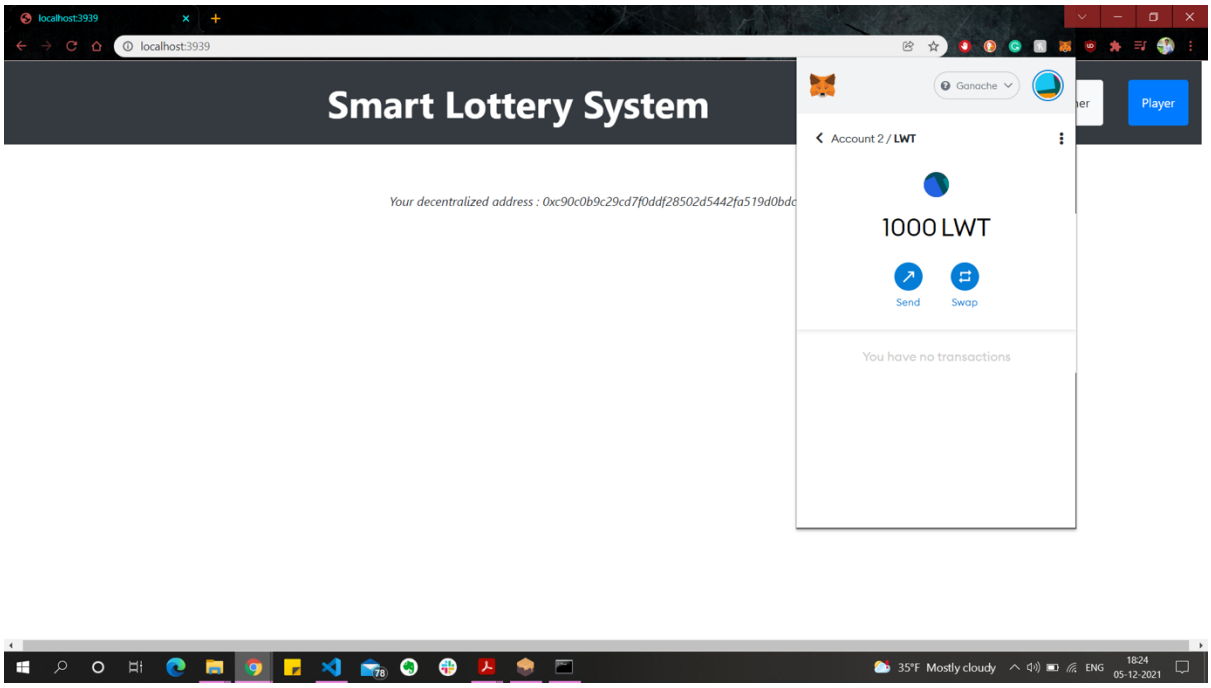
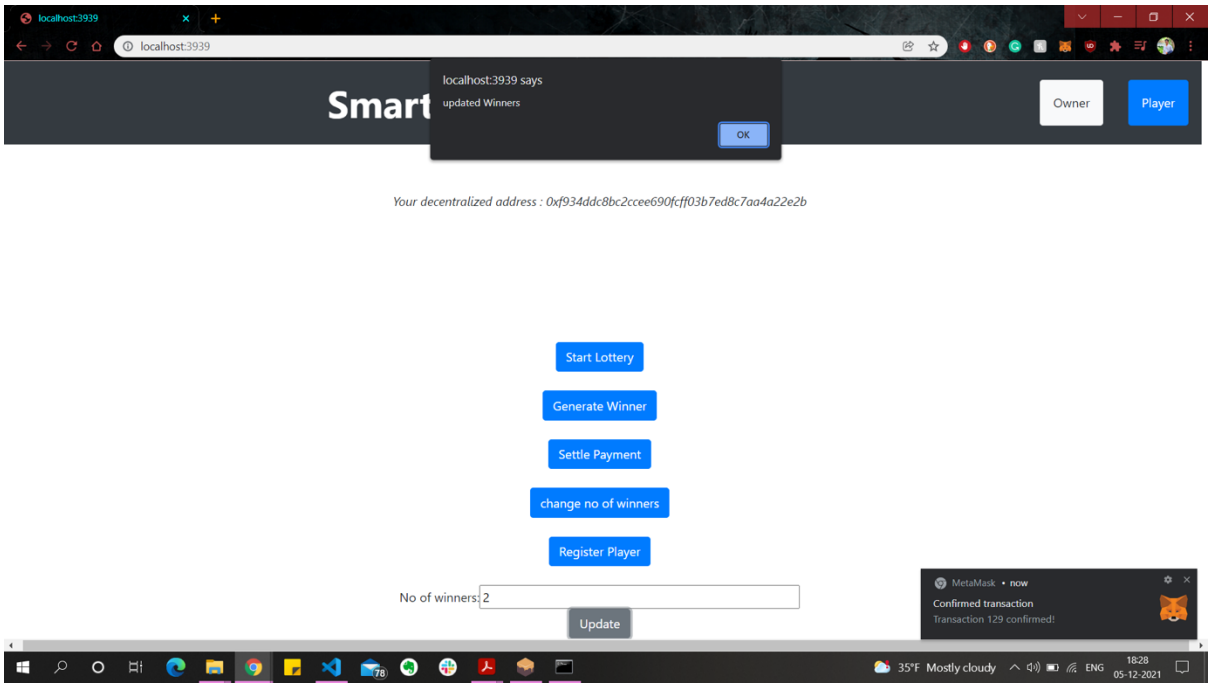
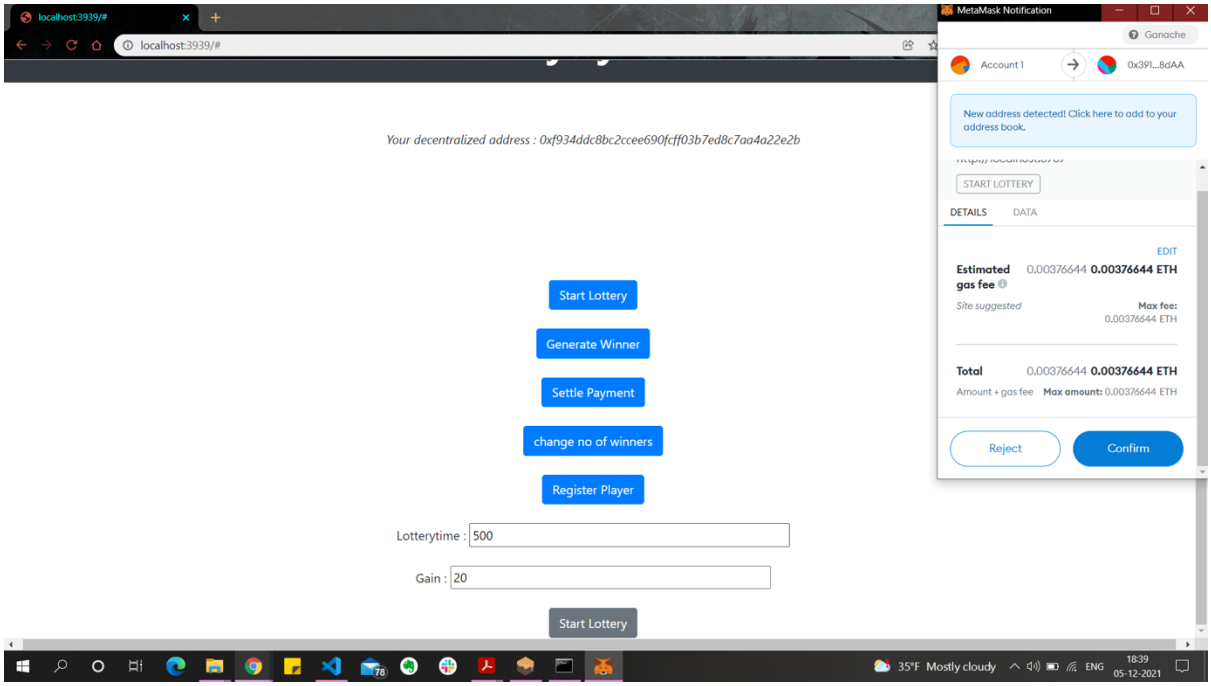


Figure: Tokens obtained by player after owner registers the player.

8.Owner can also update the number of winners for lottery.



9.Only owner can start the lottery as well with the time of lottery and gain to specify.



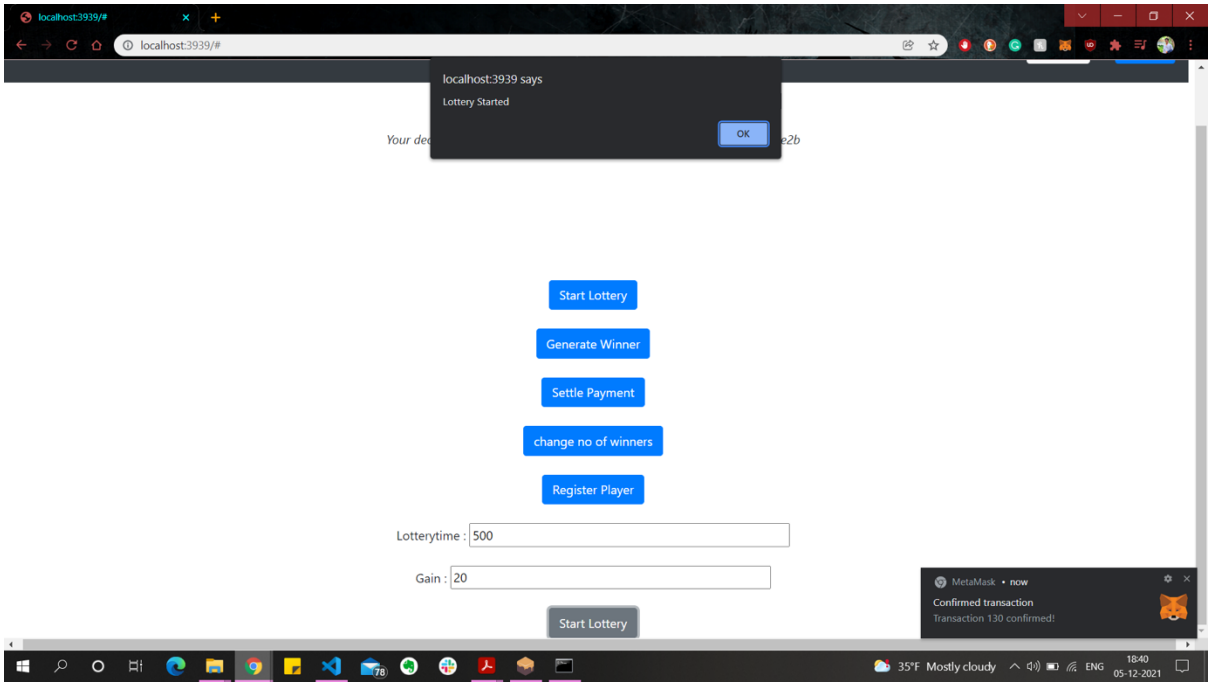
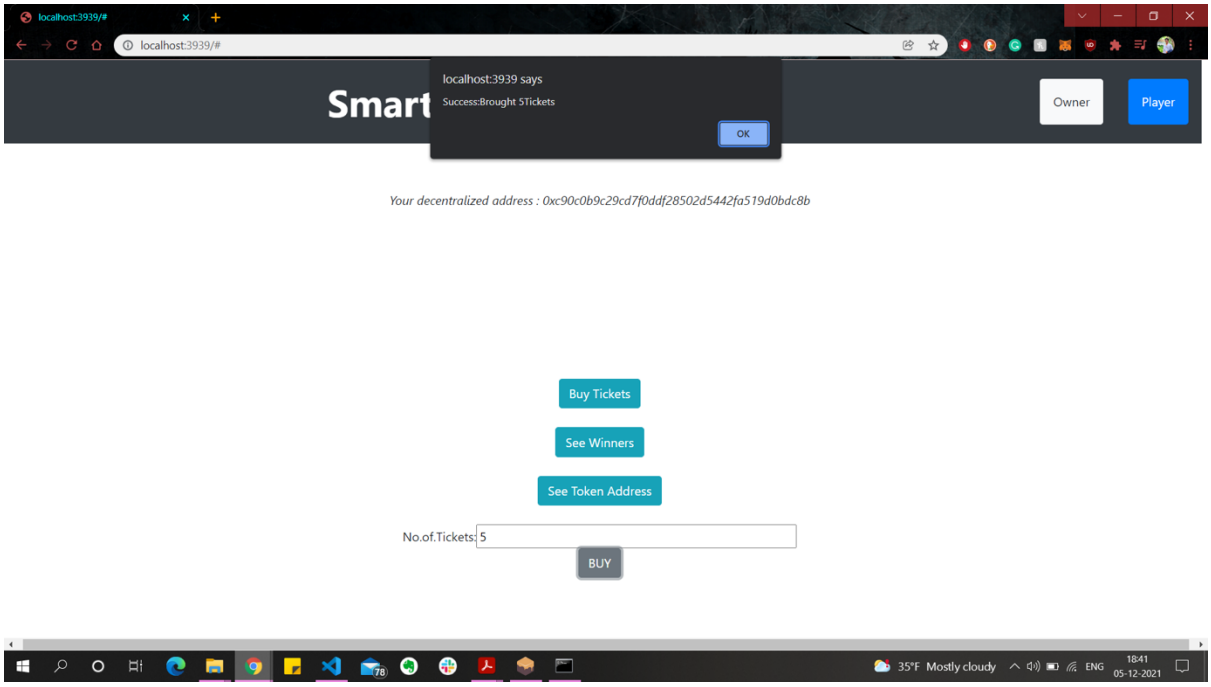
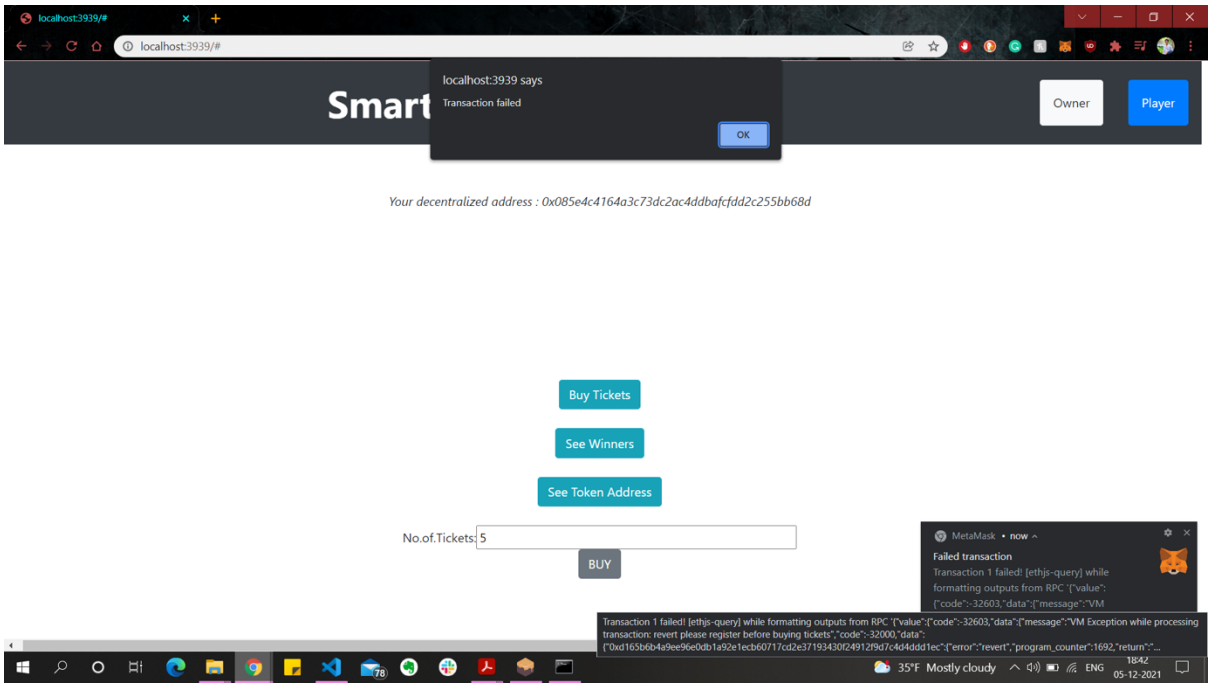


Figure: After the lottery is started the message is displayed using alert box.

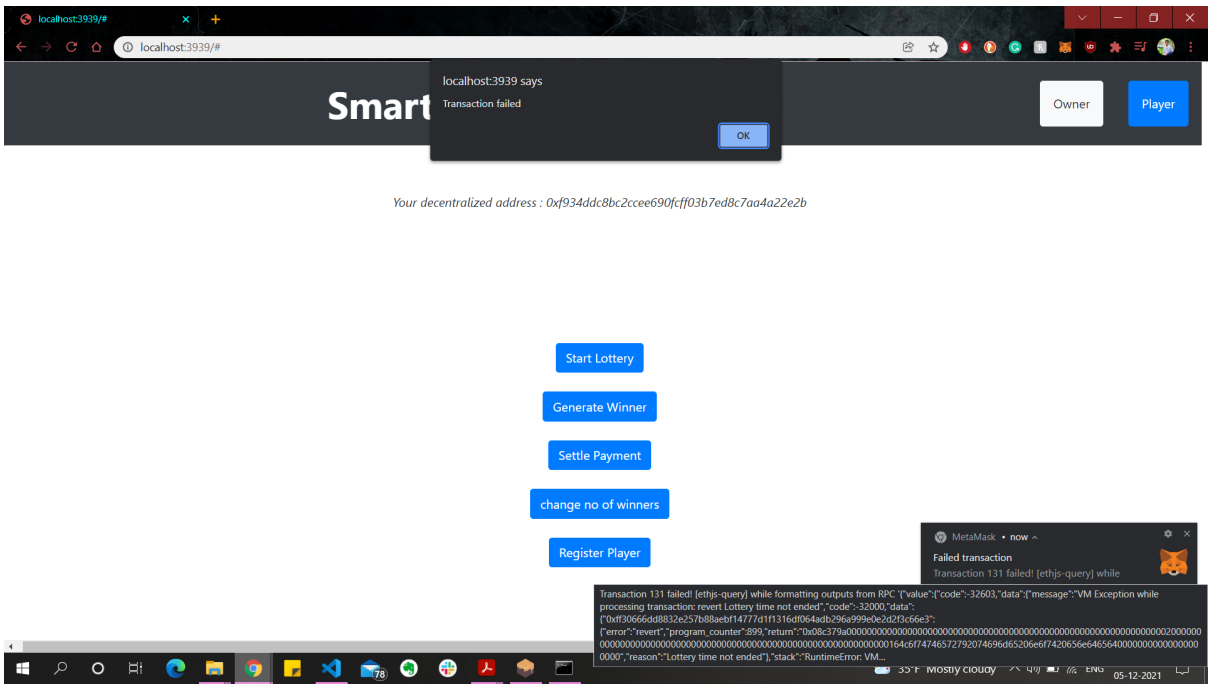
10. Players can buy tickets and owner also gives each player some bonus tokens.



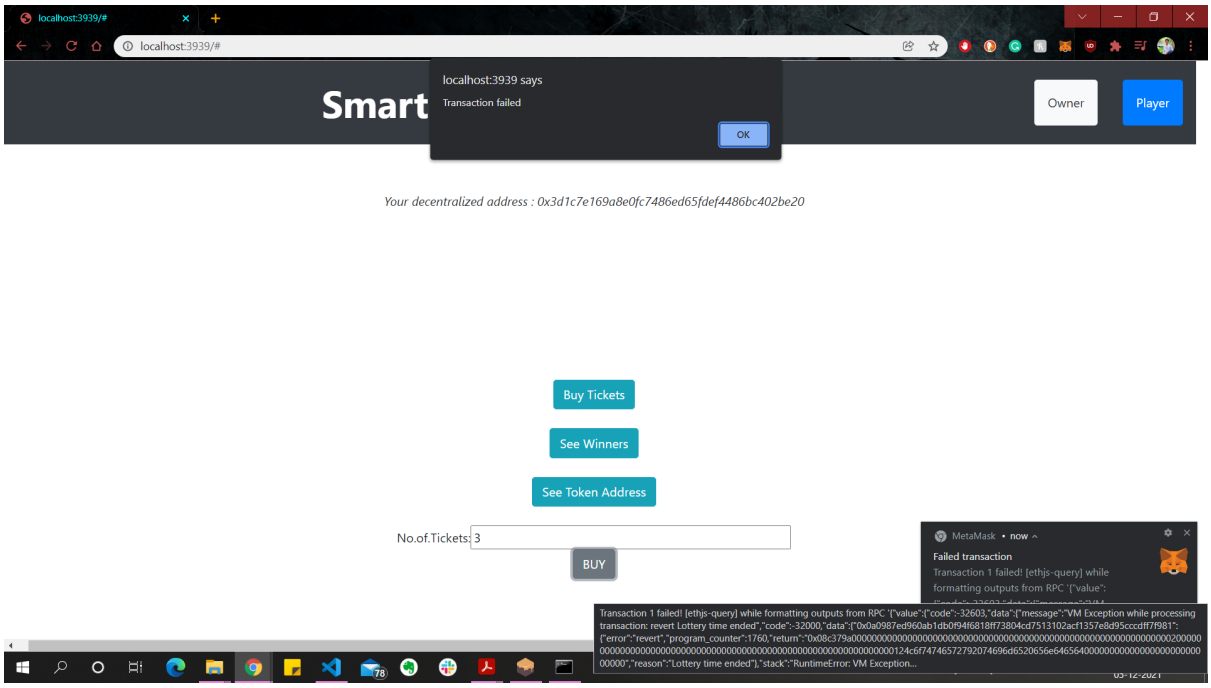
11. Trying to buy tickets before owner registering player gives an error as below.



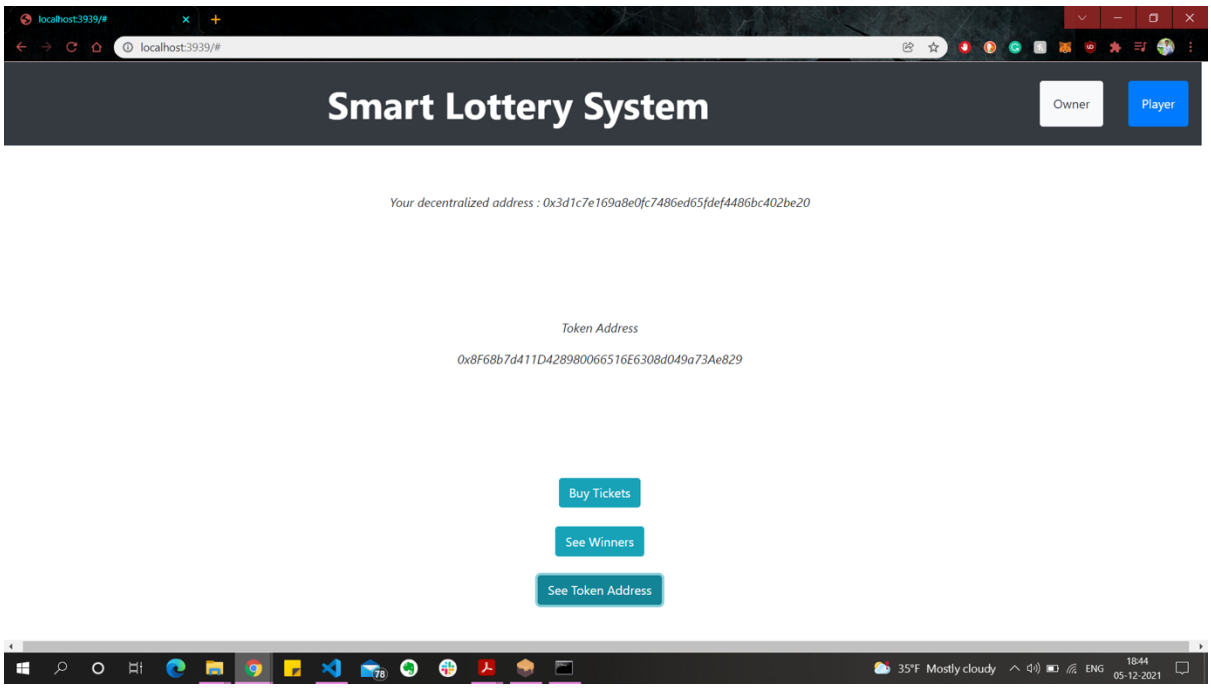
12. Requesting winners without completion of lottery gives an error as below.

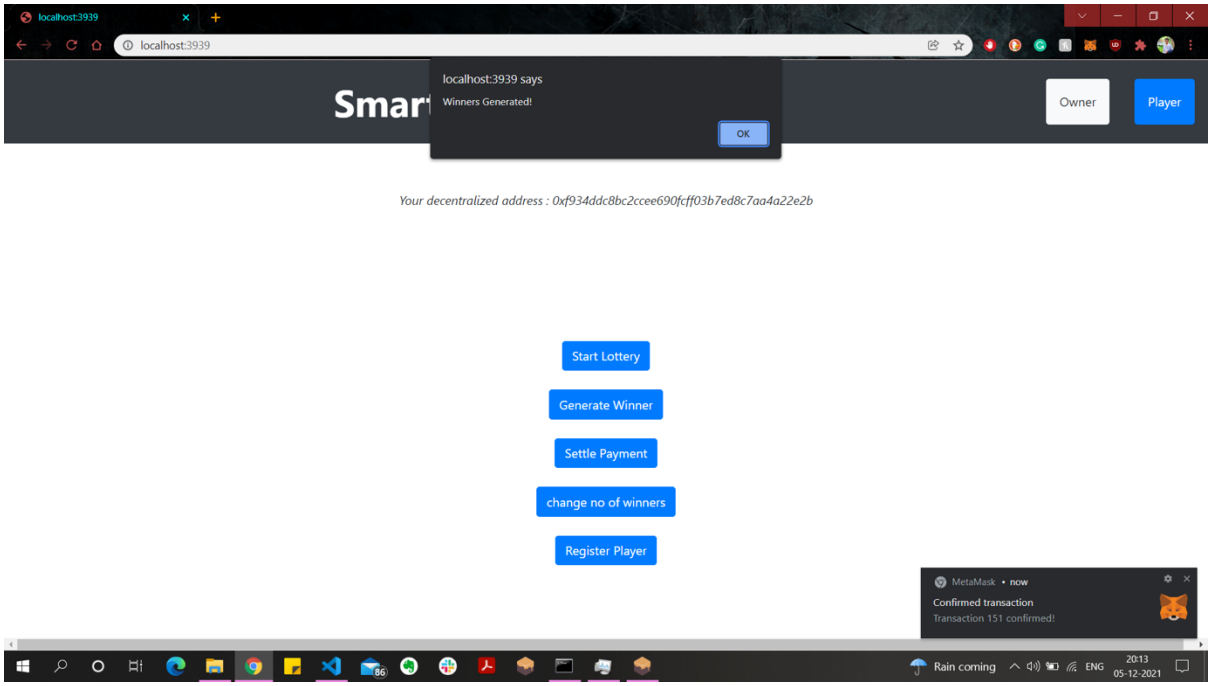


13. Trying to generate winners before lottery is ended throws an error as below.

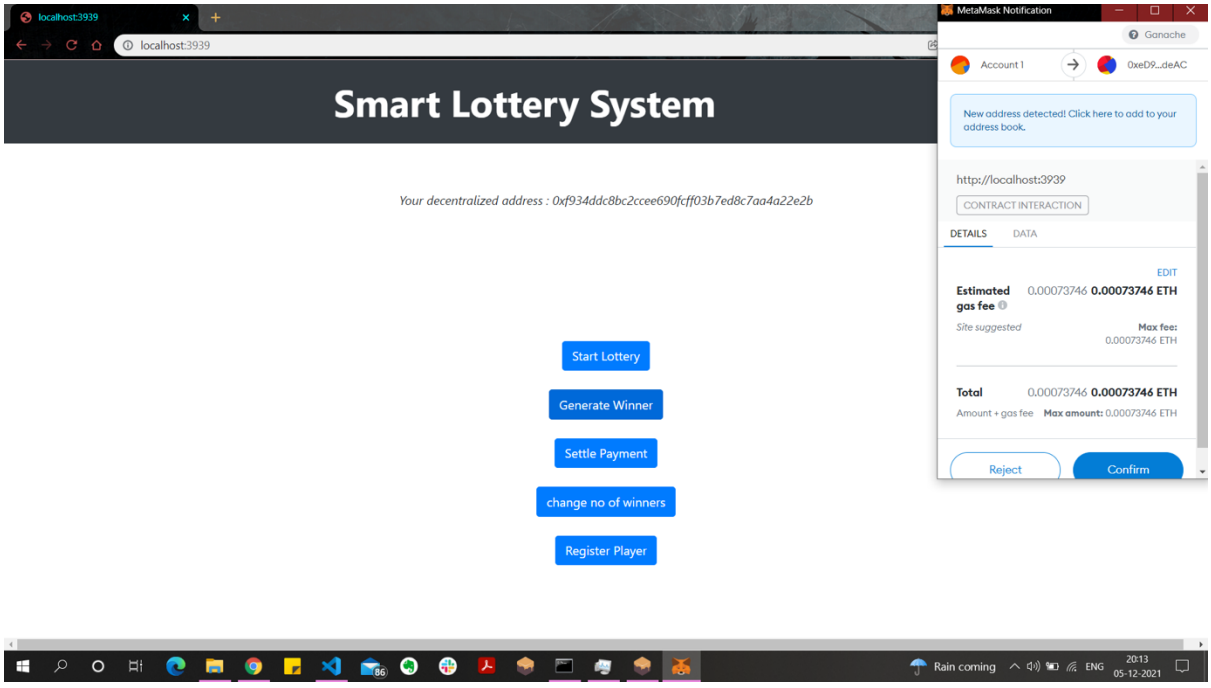


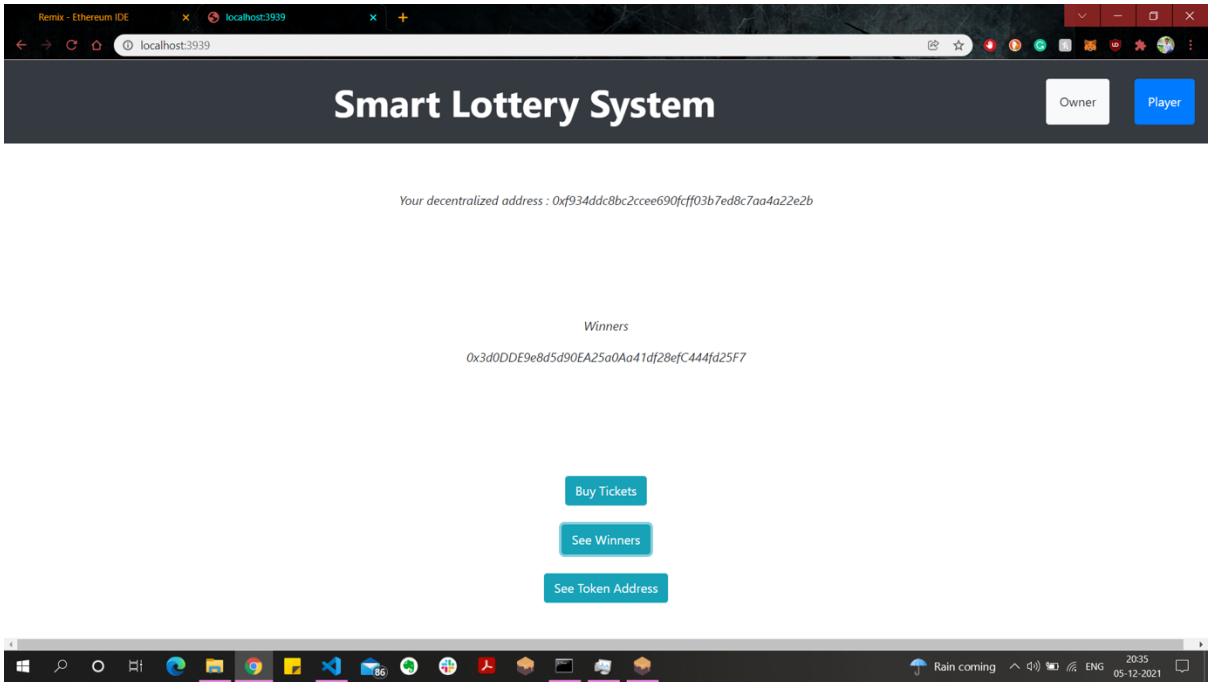
14. We can view current token address by using see token address function.



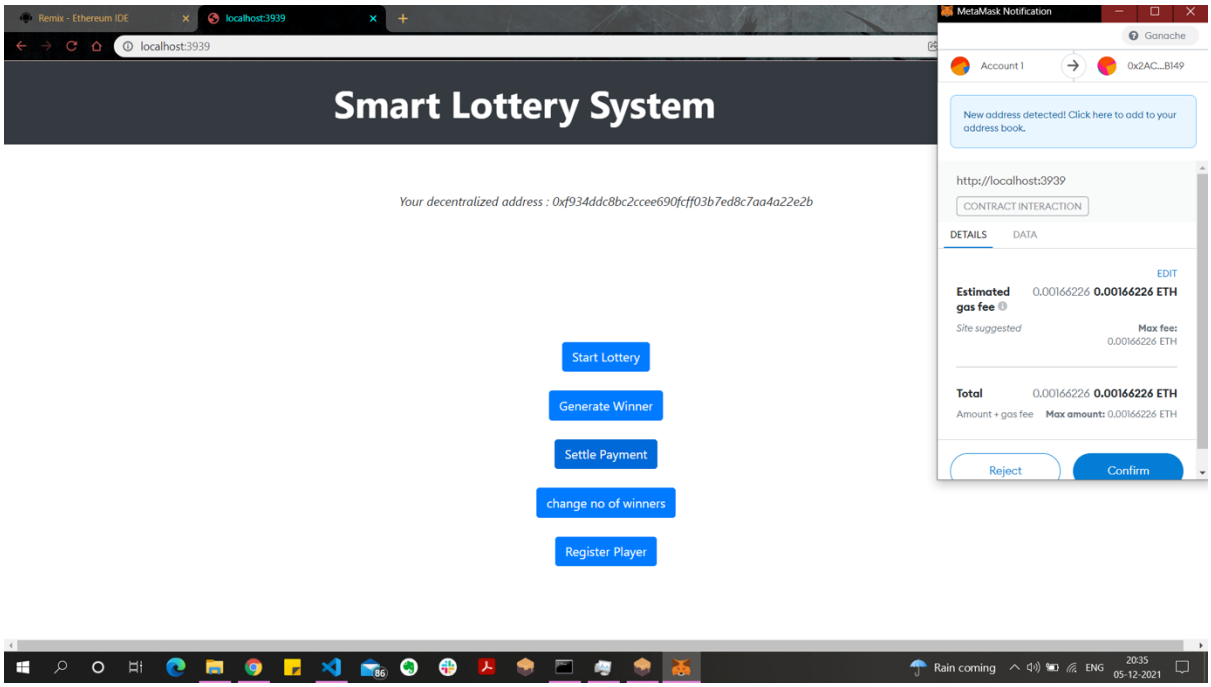


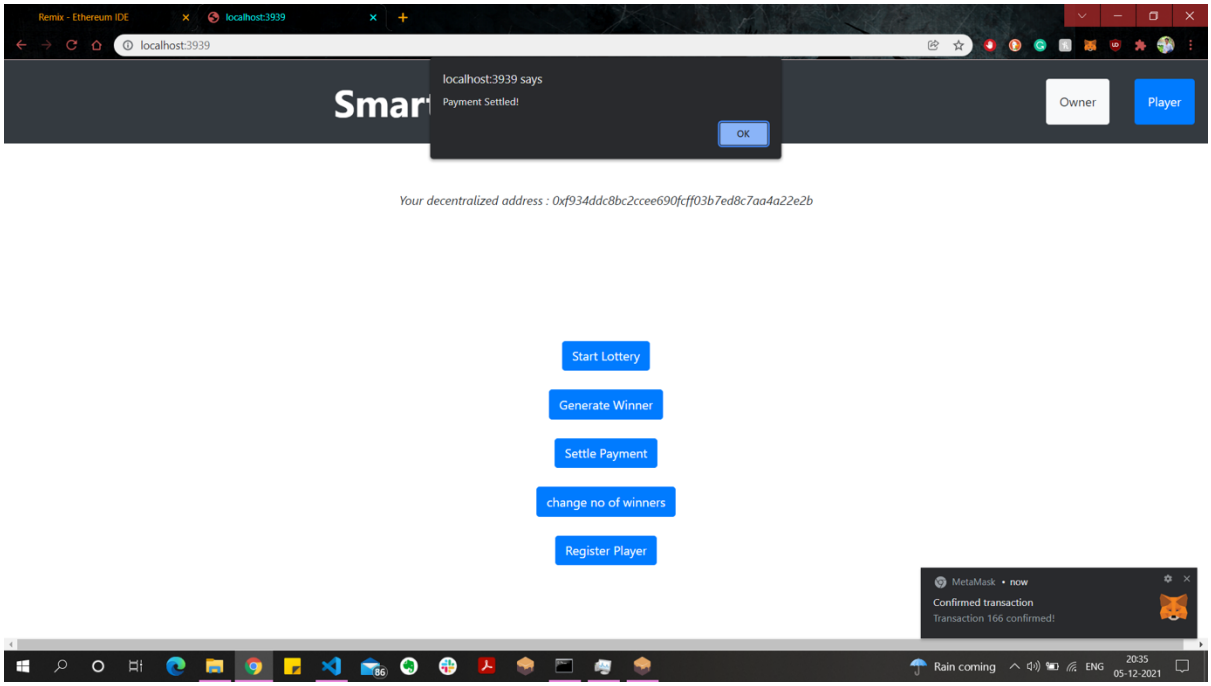
15. Winners are generated after lottery is ended and displayed as below.



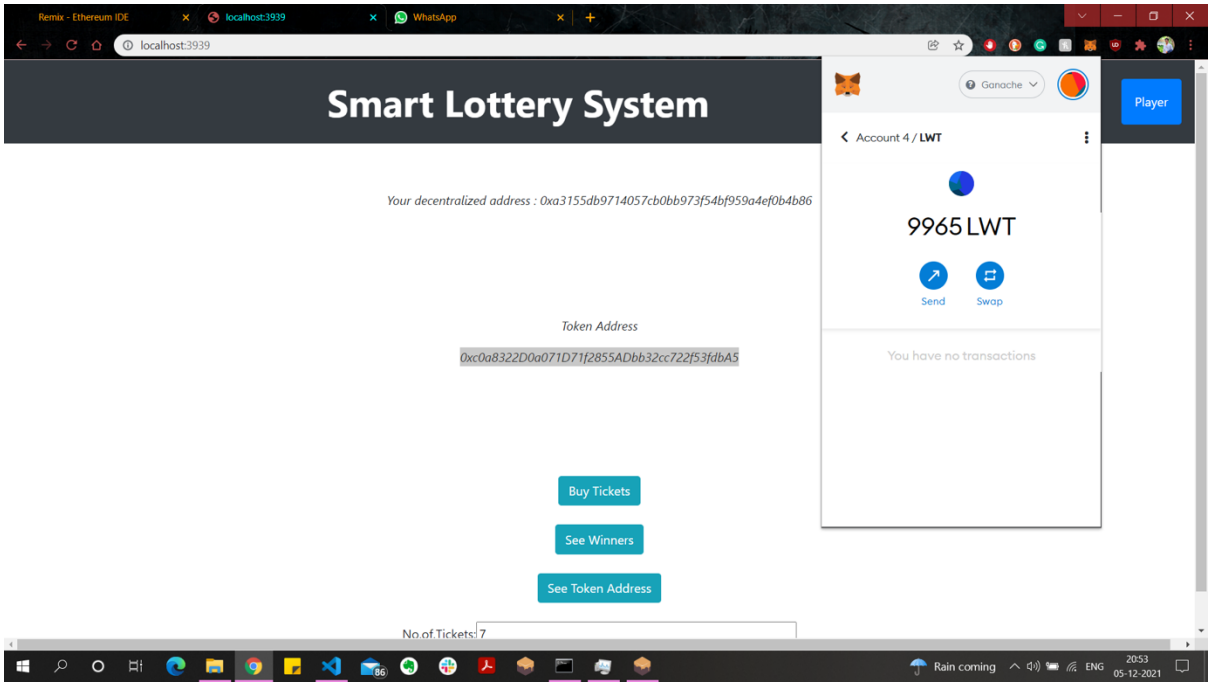


16. Payment is settled after the lottery is ended and owner uses the button settle payment.





17. Tokens in Wallets of winner before and after lottery ended .



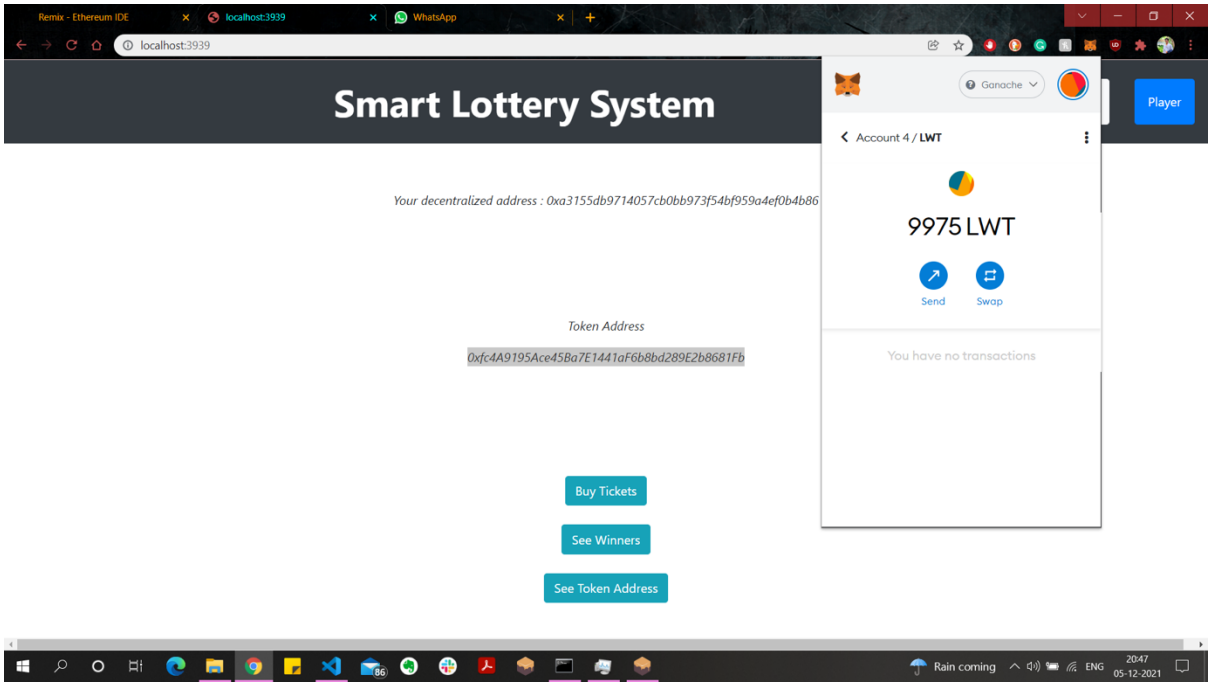


Figure: After addition of prize amount to wallet.