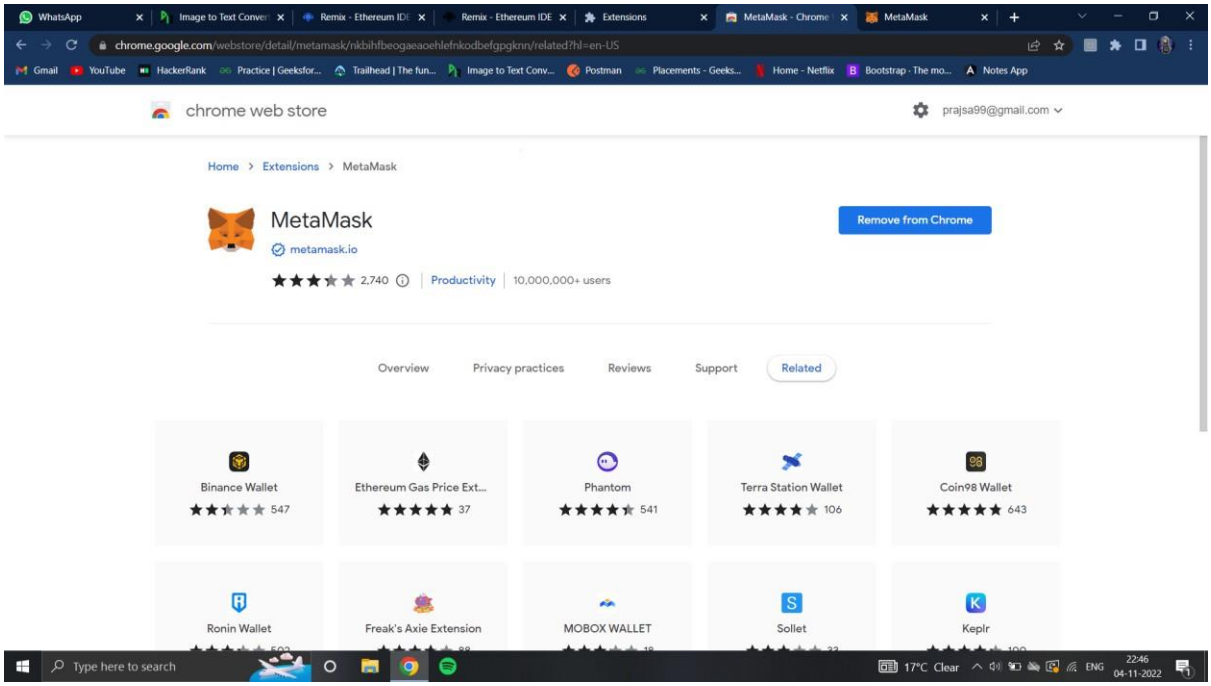
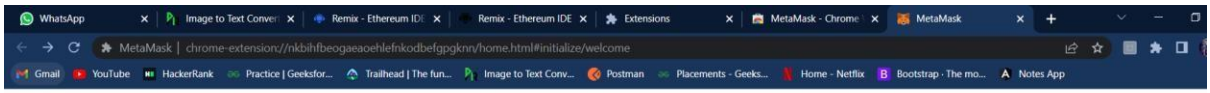
Name: Joel Neharu Gavit
Roll no:20121048

Assignment.1 : install metamask

OUTPUT:

Steps to install Metamask extension



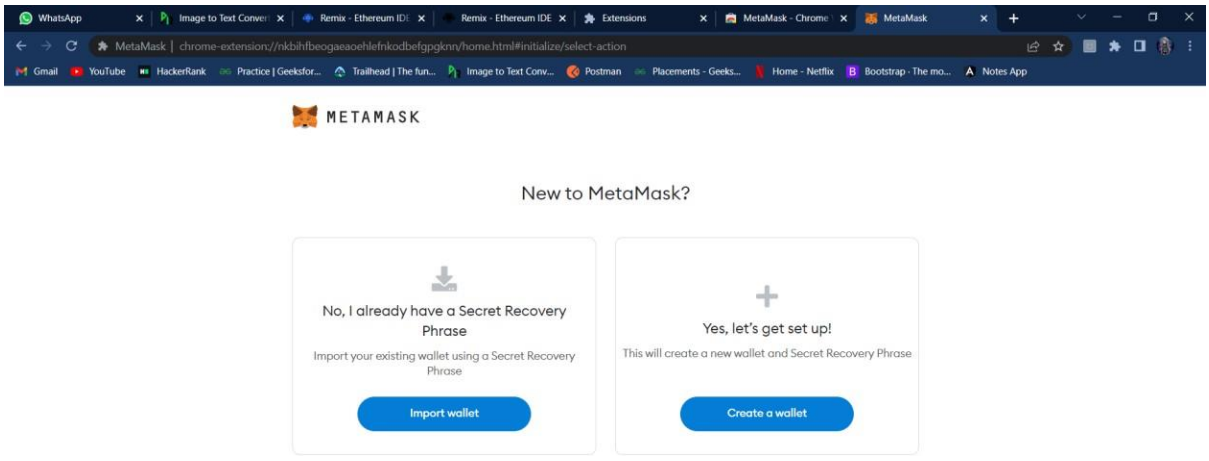


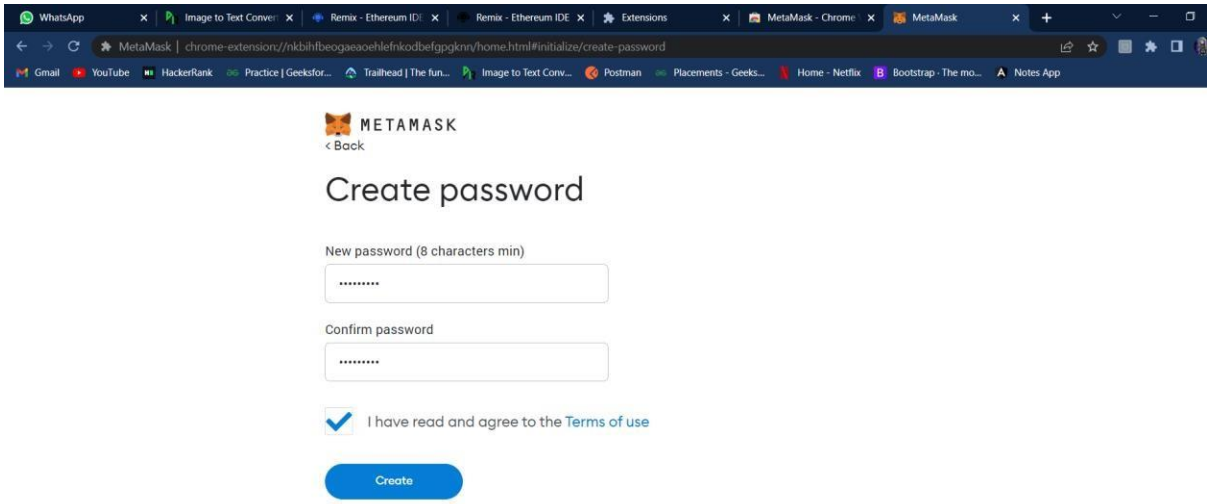
Welcome to MetaMask

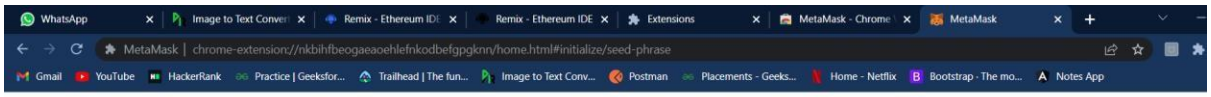
Connecting you to Ethereum and the Decentralized Web.

We're happy to see you.

Get started


WhatsApp
Image to Text Conver
Remix - Ethereum IDE
Remix - Ethereum IDE
Extensions
MetaMask - Chrome
MetaMask
MetaMask | chrome-extension://nkbihfbeogaeaoehlefnkodbefgpgknn/home.html#initialize/select-action
Gmail
YouTube
HackerRank
Practice | Geeksfor...
Trailhead | The fun...
Image to Text Conv...
Postman
Placements - Geeks...
Home - Netflix
Bootstrap · The mo...
Notes App
METAMASK
New to MetaMask?
No, I already have a Secret Recovery Phrase
Import your existing wallet using a Secret Recovery Phrase
Import wallet
Yes, let's get set up!
This will create a new wallet and Secret Recovery Phrase
Create a wallet


WhatsApp
Image to Text Conver
Remix - Ethereum IDE
Remix - Ethereum IDE
Extensions
MetaMask - Chrome
MetaMask
MetaMask | chrome-extension://nkbihfbeogaeaoehlefnkodbefgpgknn/home.html#initialize/create-password
Gmail
YouTube
HackerRank
Practice | Geeksfor...
Trailhead | The fun...
Image to Text Conv...
Postman
Placements - Geeks...
Home - Netflix
Bootstrap · The mo...
Notes App
METAMASK
< Back
Create password
New password (8 characters min)
Confirm password
I have read and agree to the Terms of use
Create



METAMASK

‹ Back

Secret Recovery Phrase

Your Secret Recovery Phrase makes it easy to back up and restore your account.

WARNING: Never disclose your Secret Recovery Phrase. Anyone with this phrase can take your Ether forever.

CLICK HERE TO REVEAL SECRET WORDS

Remind me later | Next

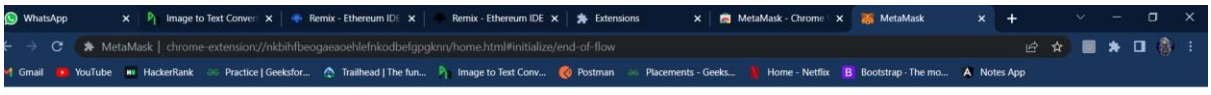
Tips:

Store this phrase in a password manager like 1Password.

Write this phrase on a piece of paper and store in a secure location. If you want even more security, write it down on multiple pieces of paper and store each in 2 - 3 different locations.

Memorize this phrase.

Download this Secret Recovery Phrase and keep it stored safely on an external encrypted hard drive or storage medium.



METAMASK

Congratulations

You passed the test - keep your Secret Recovery Phrase safe, it's your responsibility!

Tips on storing it safely

- Save a backup in multiple places.
- Never share the phrase with anyone.
- Be careful of phishing! MetaMask will never spontaneously ask for your Secret Recovery Phrase.
- If you need to back up your Secret Recovery Phrase again, you can find it in Settings > Security.
- If you ever have questions or see something fishy, contact our support here.

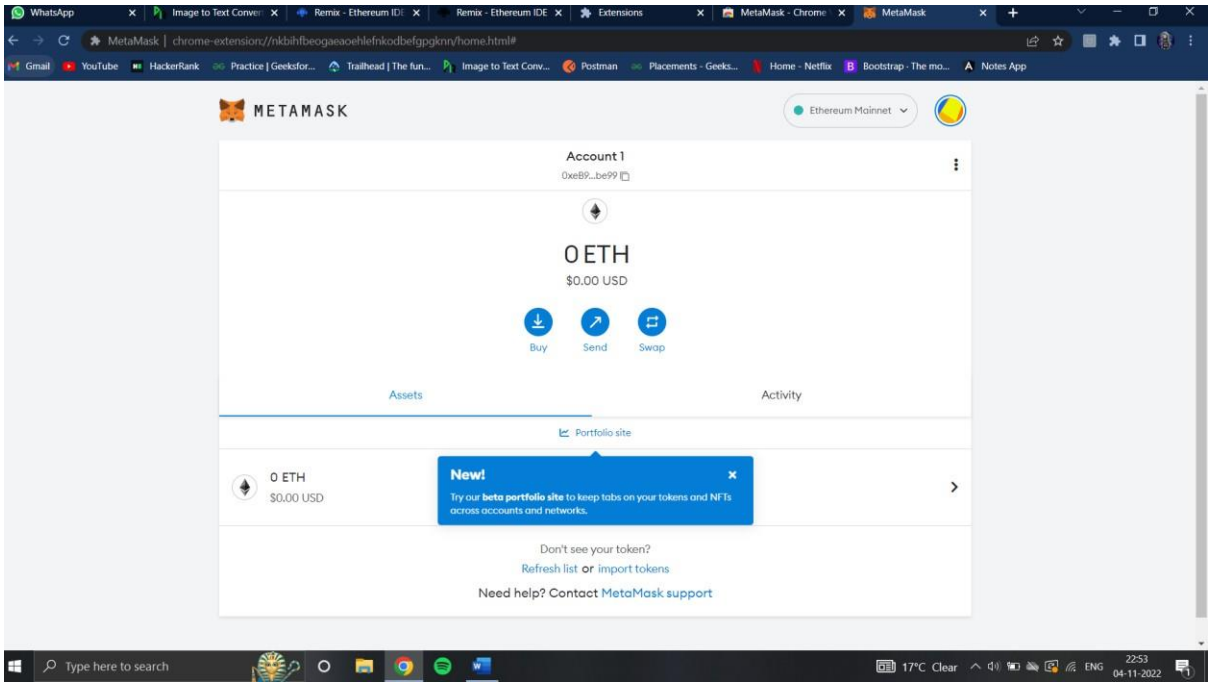
*MetaMask cannot recover your Secret Recovery Phrase. Learn more.

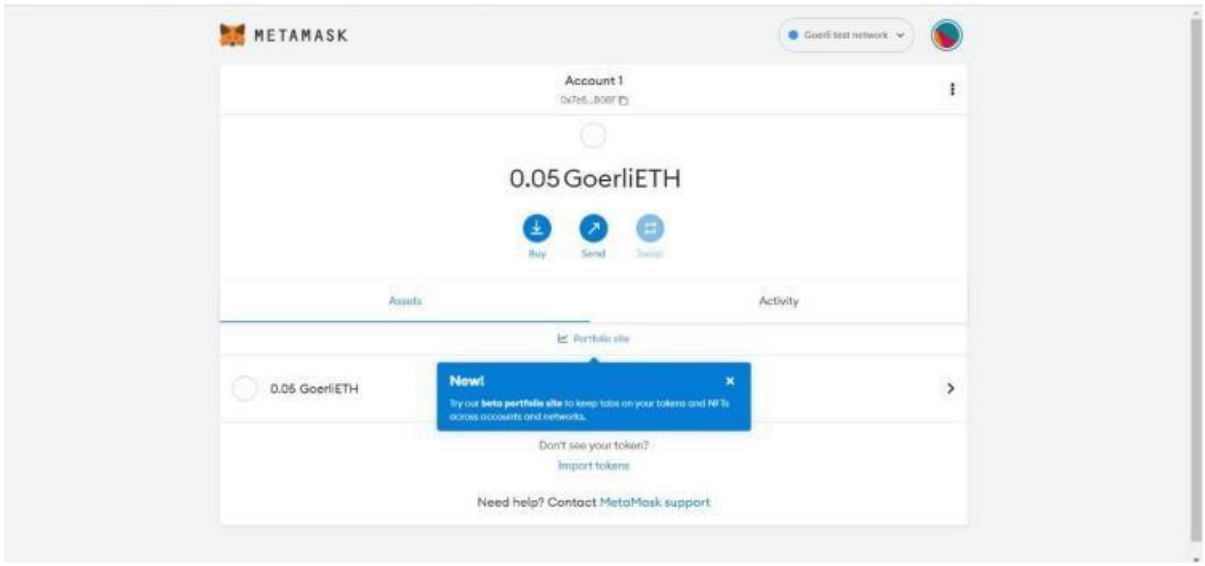
All done

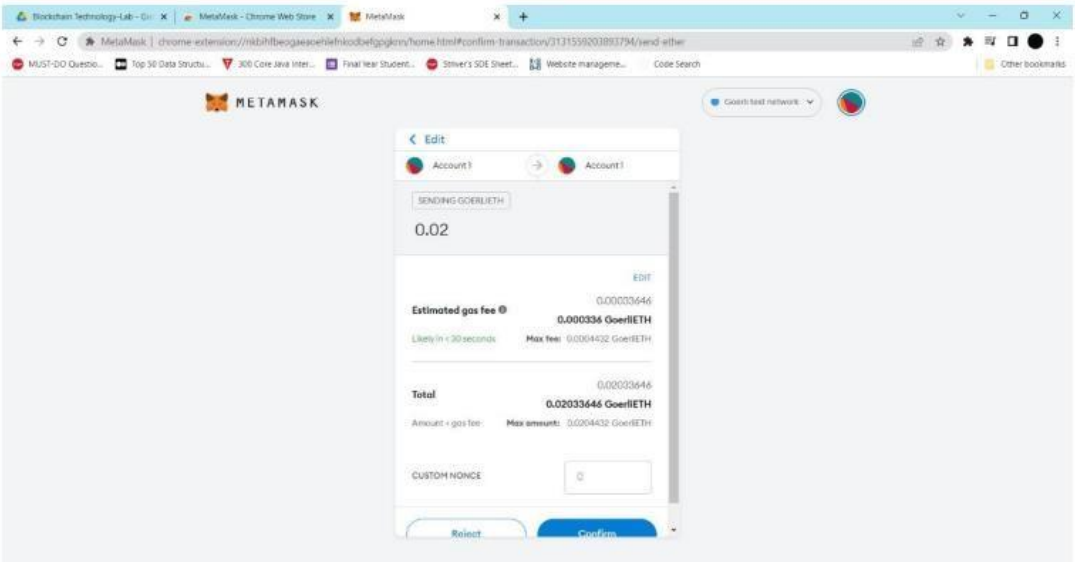
Assignment.2: perfrom transaction on test network

OUTPUT:

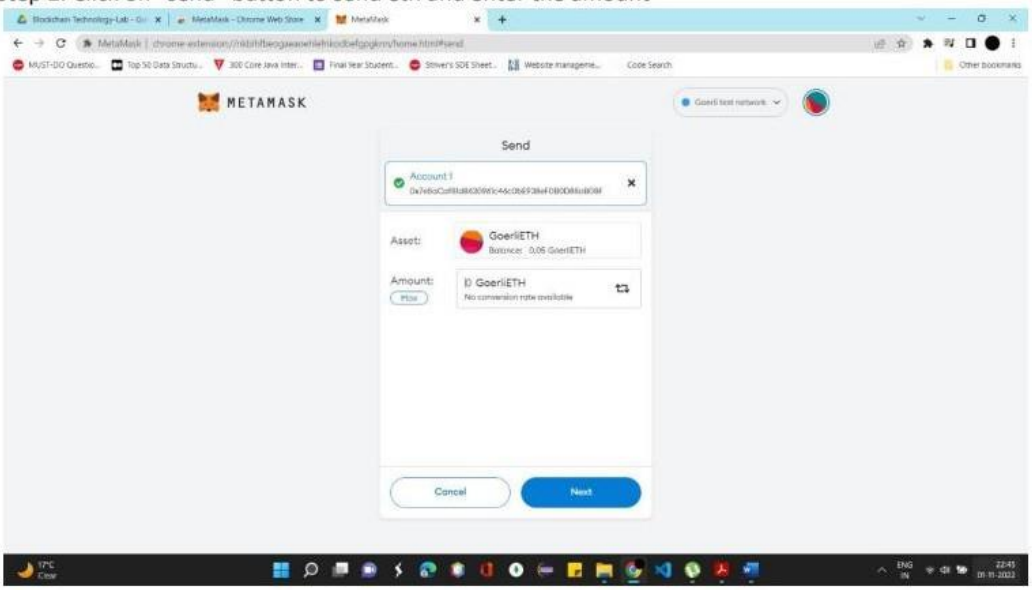
Transaction

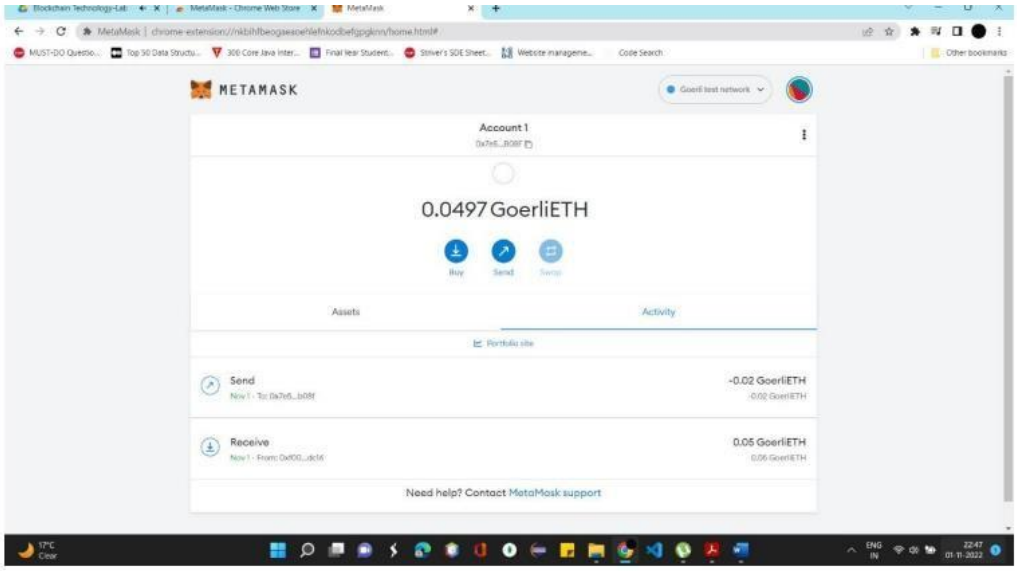


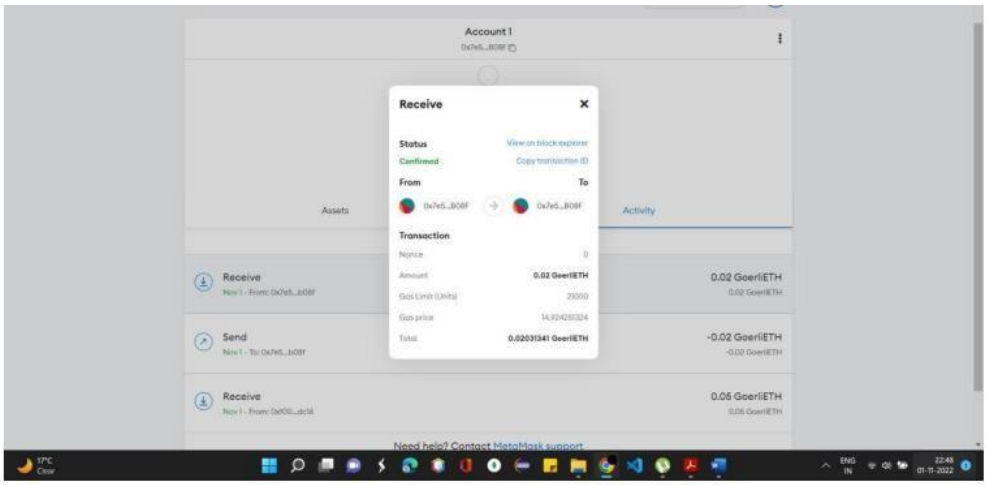




Step 2: Click on “send” button to send eth and enter the amount




METAMASK
Account 1
0.0497 GoerliETH
Buy
Send
Swap
Assets
Activity
Send
-0.02 GoerliETH
Receive
0.05 GoerliETH
Need help? Contact MetaMask support


Account 1
Receive
Status
Confirmed
From
To
Transaction
Amount
0.02 GoerliETH
Gas Limit (Units)
21000
Total
0.02031341 GoerliETH
Assets
Activity
Receive
0.02 GoerliETH
Send
-0.02 GoerliETH
Receive
0.05 GoerliETH
Need help? Contact MetaMask support

Assignment.3: smart contract for bank account

CODE and OUTPUT:

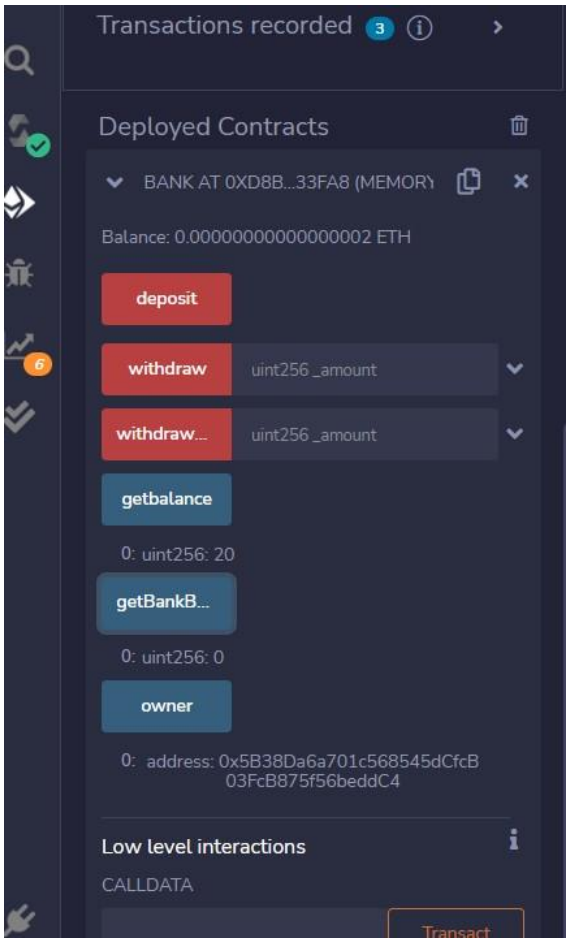
```
// SPDX-License-Identifier: MIT
pragma solidity >=0.7.0 <0.9.0;

contract Bank {

    address public owner;

    mapping(address =>uint256) private userbalance;

    constructor() {
        owner = msg.sender;
    }

    modifier onlyOwner(){
        require (msg.sender==owner, 'You are not the owner of this contract');
    _;
    }

    function deposit() public payable returns(bool) {
        require(msg.value >10 wei, 'Please deposit at least 10 wei');
        userbalance[msg.sender] +=msg.value;
        return true;
    }

    function withdraw(uint256 _amount) public payable returns (bool) {
        require(_amount <=userbalance[msg.sender], 'You dont have sufficient funds');
        userbalance[msg.sender] -=_amount;
        payable(msg.sender).transfer(_amount);
```

```
        return true;
    }

    function getbalance() public view returns(uint256){
        return userbalance[msg.sender];
    }

    function getBankBalance() public view onlyOwner returns(uint256){
        return address(this).balance;
    }

    function withdrawBankBalance (uint256 _amount) public payable onlyOwner returns (bool){
        payable(owner).transfer (_amount);
    return true;
    }

}
```

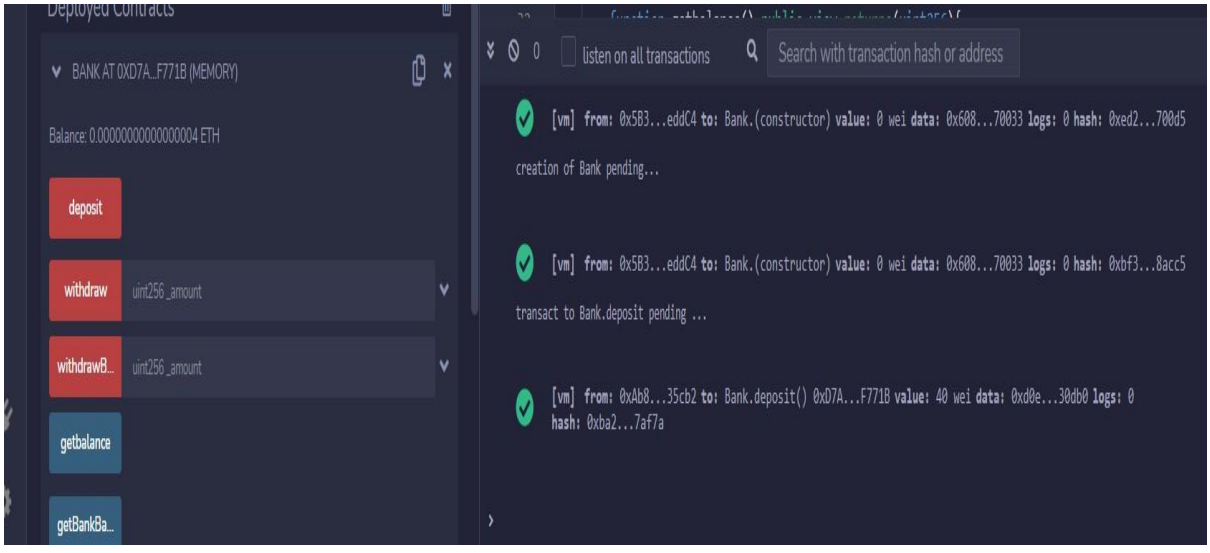
Bank contract with id (will be bank account): 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4

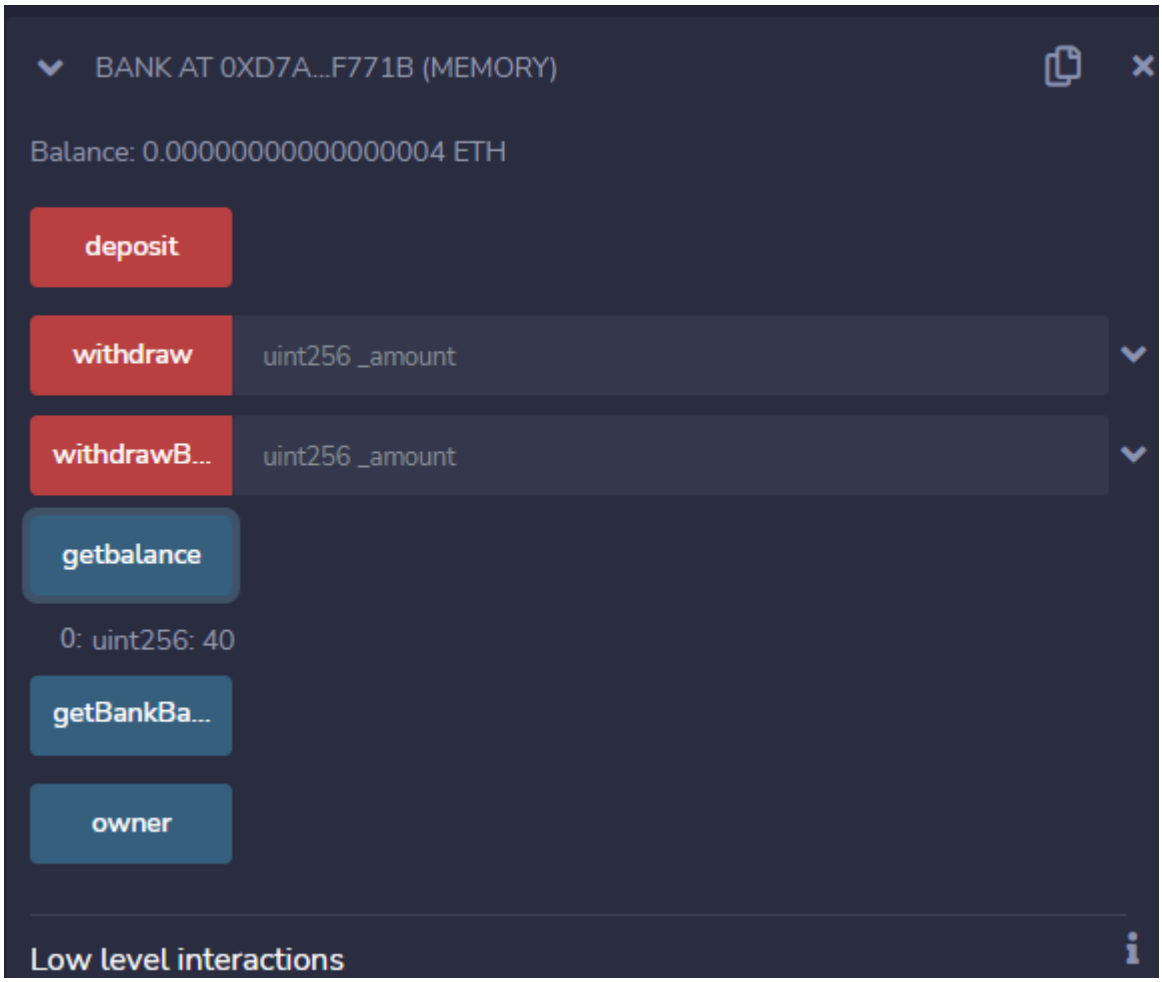
Bank customer with id : 0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2

1. **Contract signed**

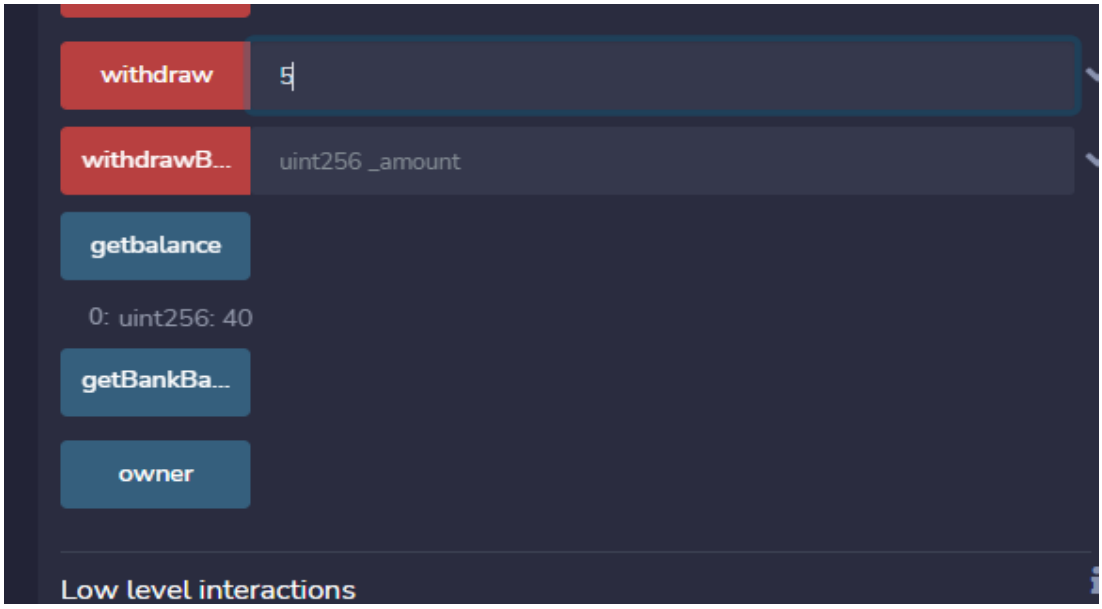


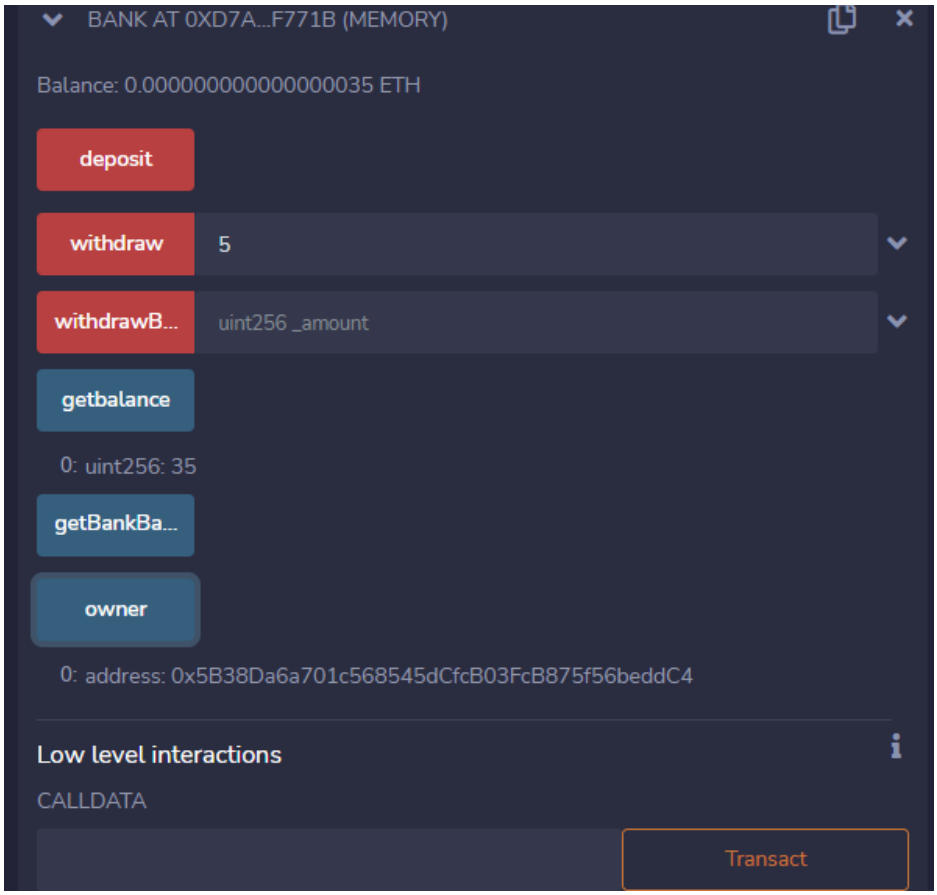
2 User deposits 40



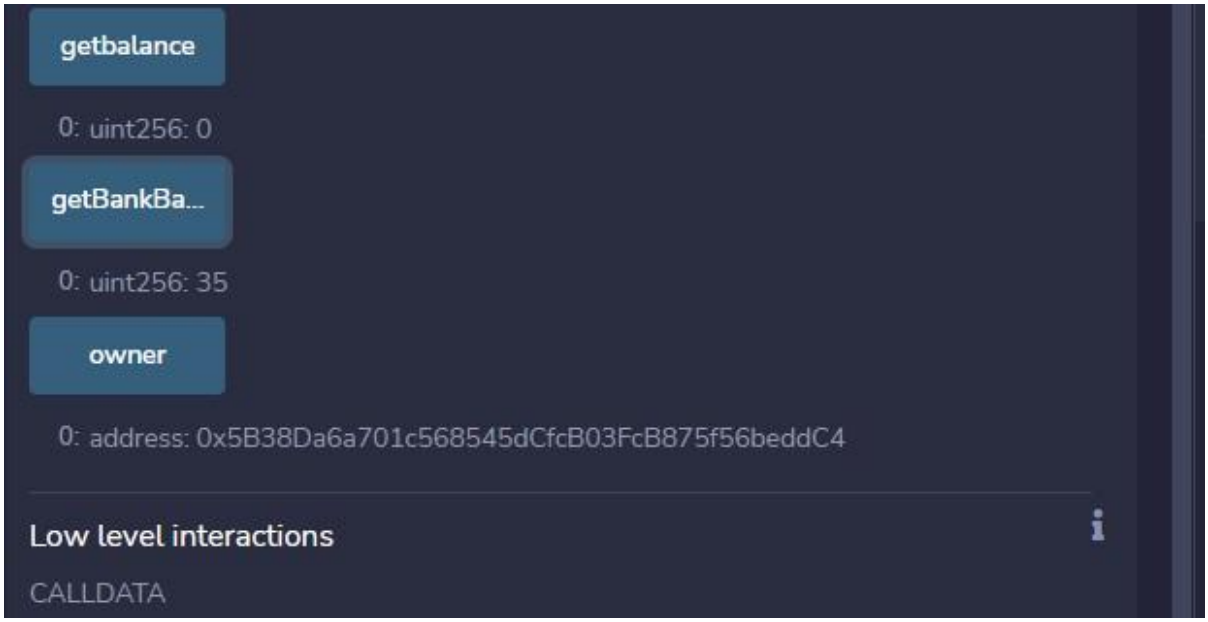


3 **User withdraw**





4 .<u>Bank balance becomes 35 (as 40 - 5)</u>



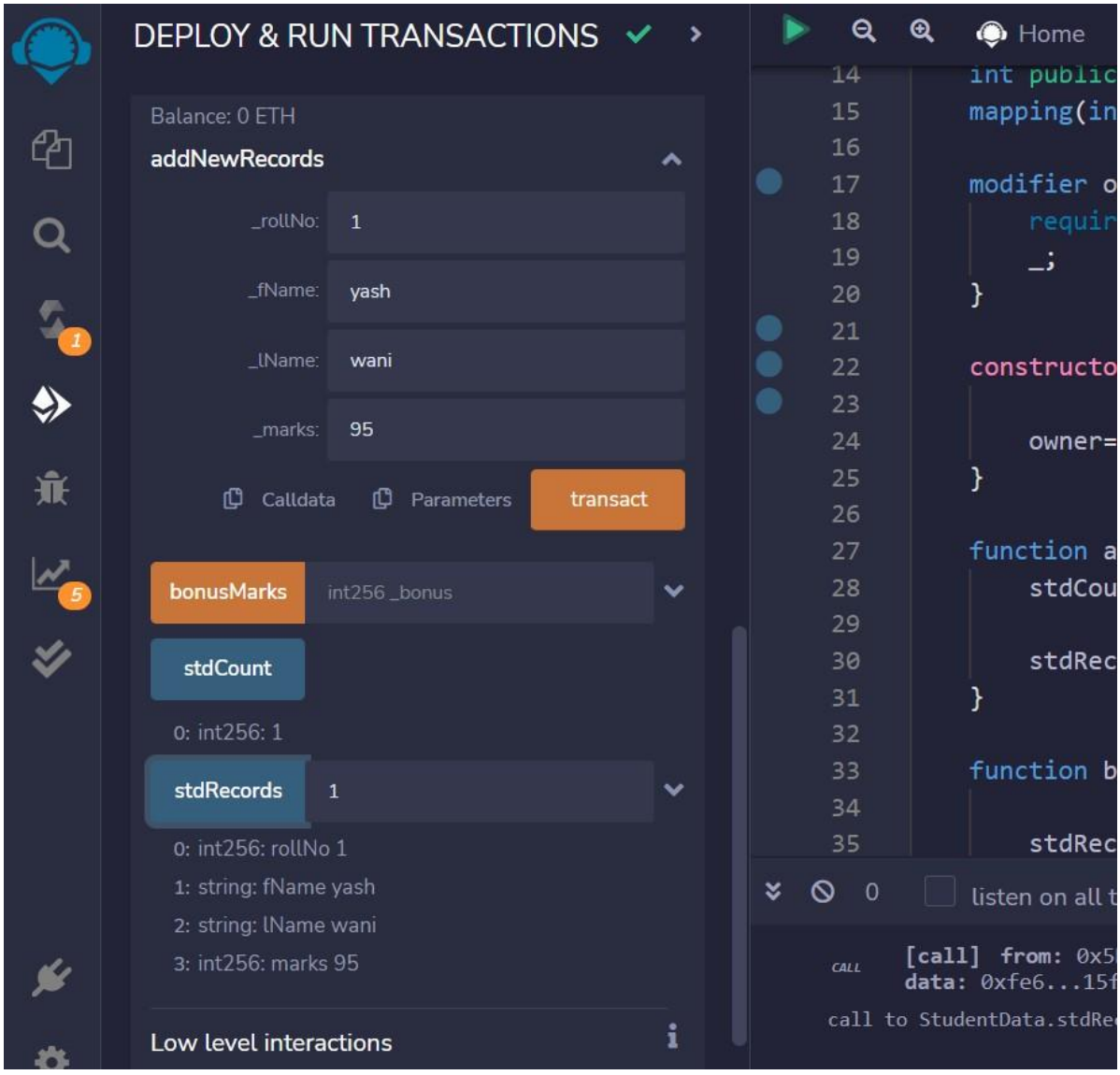
Assignment.4: program in solidity to create studentdata

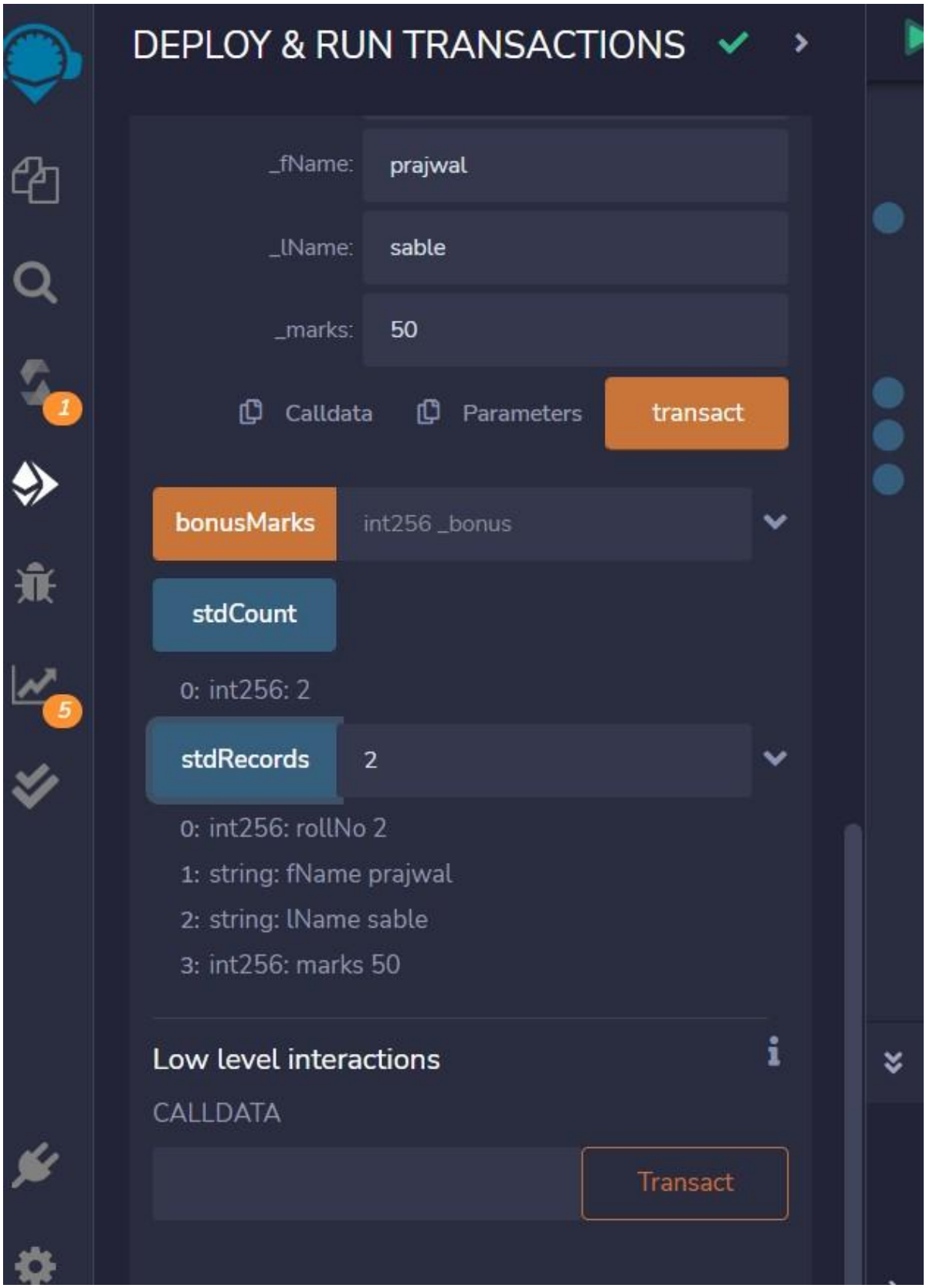
CODE/OUTPUT:

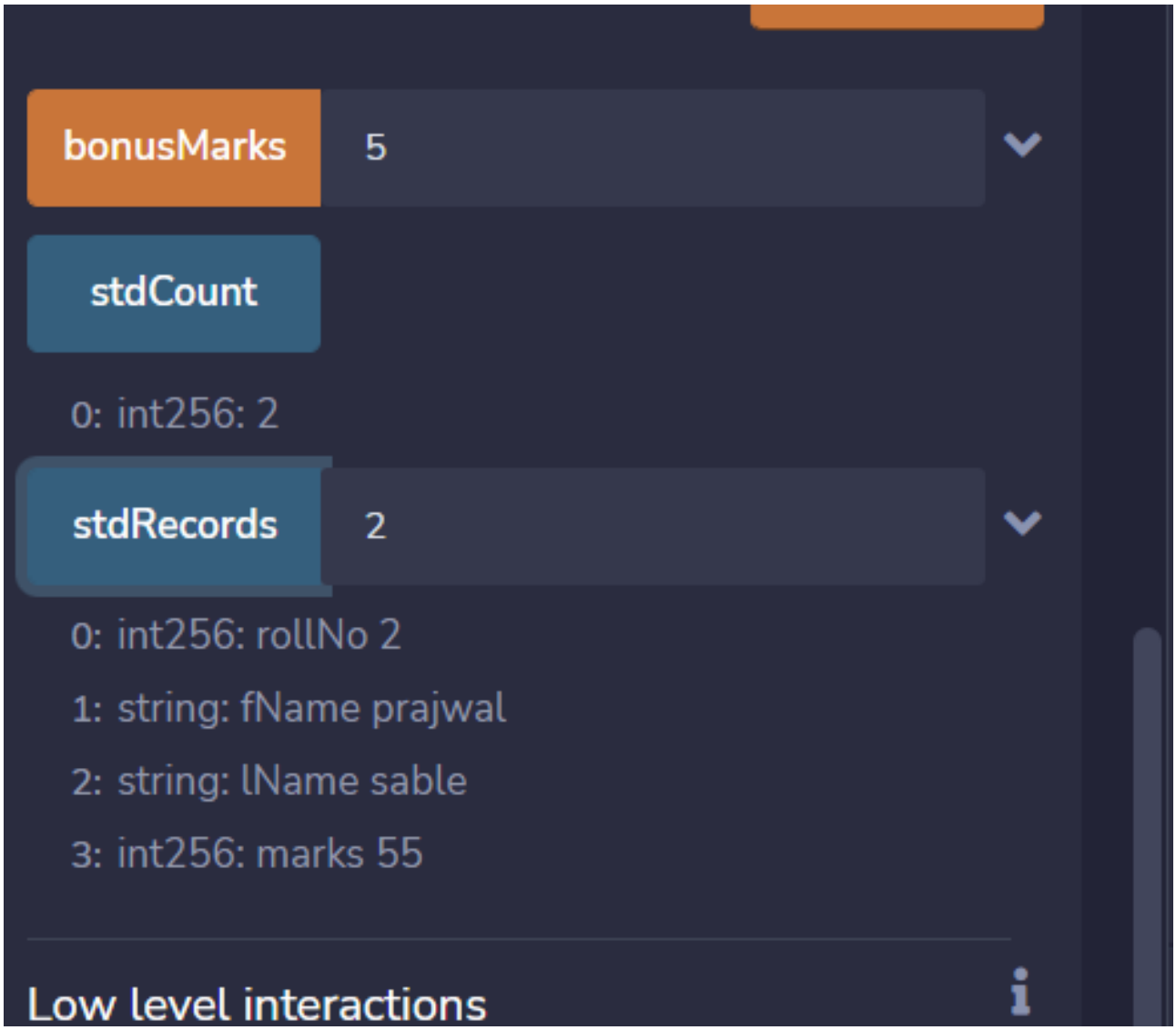
```
// SPDX-License-Identifier: MIT
pragma solidity >=0.7.0 <0.9.0;

contract StudentData{

    struct Student{
        int rollNo;
        string fName;
        string lName;
        int marks;
    }

    address owner;
    int public stdCount =0;
    mapping(int => Student) public stdRecords;

    modifier onlyOwner{
        require (owner == msg.sender);
        _;
    }

    constructor(){

        owner=msg.sender;
    }
    function addNewRecords (int _rollNo, string memory _fName, string memory _lName, int
_marks) public onlyOwner{
        stdCount=stdCount+1;
```

```
stdRecords [stdCount]=Student (_rollNo,_fName,_lName,_marks);
}
function bonusMarks(int _bonus) public onlyOwner{
    stdRecords[stdCount].marks=stdRecords[stdCount].marks + _bonus;
}
fallback () external payable{
}
}
```

OUTPUT




DEPLOY & RUN TRANSACTIONS
_fName:
prajwal
_lName:
sable
_marks:
50
Calldata
Parameters
transact
bonusMarks
int256 _bonus
stdCount
0: int256: 2
stdRecords
2
0: int256: rollNo 2
1: string: fName prajwal
2: string: lName sable
3: int256: marks 50
Low level interactions
CALLDATA
Transact


bonusMarks
5
stdCount
0: int256: 2
stdRecords
2
0: int256: rollNo 2
1: string: fName prajwal
2: string: lName sable
3: int256: marks 55
Low level interactions

MINI PROJECT :

Decnetralized e-voting system using solidity:

CODE:

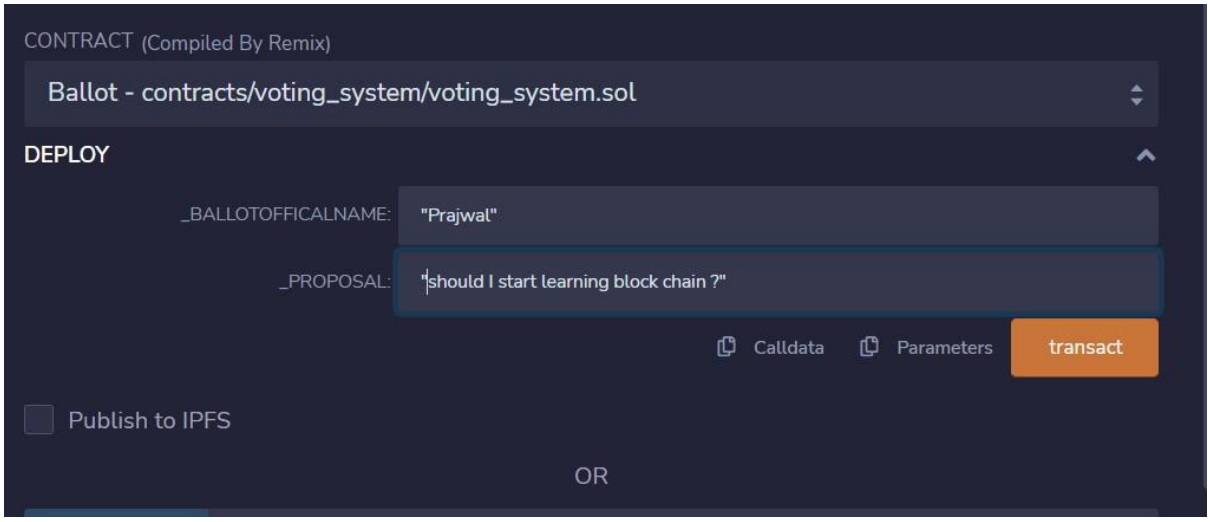
```
// SPDX-License-Identifier: MIT
pragma solidity >= 0.7.0 <0.8.0;


contract Ballot {
   // VARIBLES
   struct vote {
      address voterAddresss;
      bool choice;
   }
   struct voter {
      string voterName;
      bool voted;
   }
   uint private countResult = 0;
   uint public finalResult = 0;
   uint public totalVoter = 0;
   uint public totalVote = 0;

   address public ballotOfficialAddress;
   string public ballotOfficalName;
   string public proposal;

   mapping(uint => vote) private votes;
   mapping(address => voter) public voterRegister;

   enum State { Created, Voting, Ended }
   State public state;
```

```
// MODIFIER
modifier condition(bool _condition) {
    require(_condition);
    _;
}

modifier onlyOfficial() {
    require(msg.sender == ballotOfficialAddress);
    _;
}

modifier inState(State _state) {
    require(state == _state);
    _;
}


// FUNCTION
constructor(
    string memory _ballotofficalName,
    string memory _proposal
)  {
    ballotOfficialAddress = msg.sender;
    ballotOfficalName = _ballotofficalName;
    proposal = _proposal;
    state = State.Created;
}
```

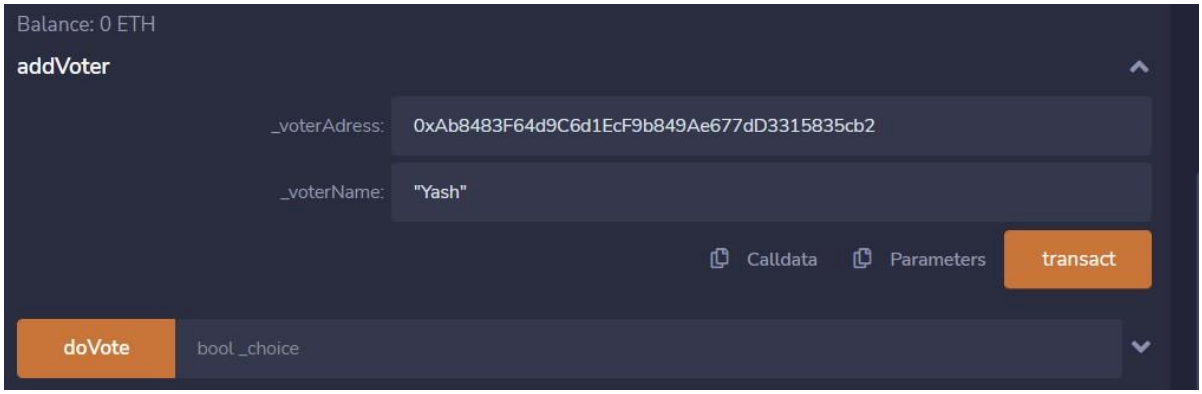
```
function addVoter(
    address _voterAdress,
    string memory _voterName
) public
    inState(State.Created)
    onlyOfficial
{
    voter memory v;
    v.voterName = _voterName;
    v.voted = false;
    voterRegister[_voterAdress] = v;
    totalVoter++;
}


function startVote()
    public
    inState(State.Created)
    onlyOfficial
{
    state = State.Voting;
}



function doVote(bool _choice)
    public
    inState(State.Voting)
    returns (bool voted)
{
    bool isFound = false;
```

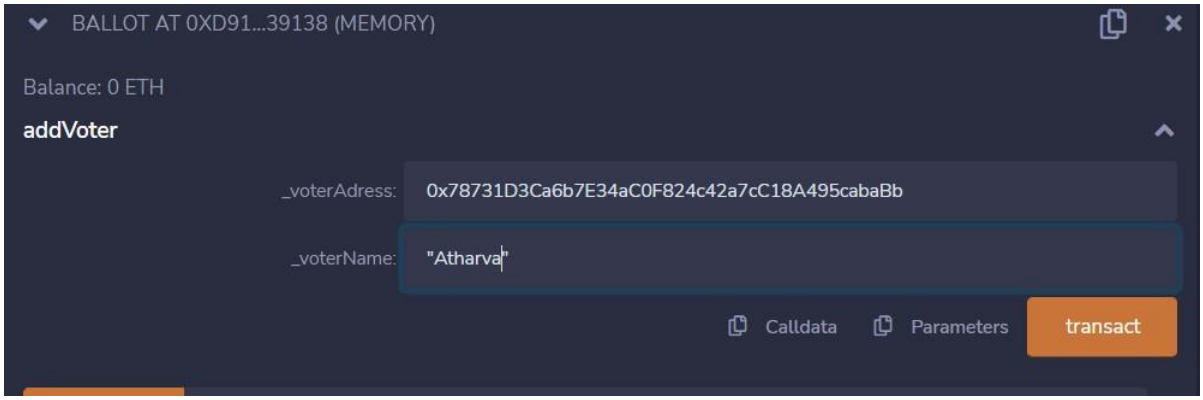
```
        if(bytes(voterRegister[msg.sender].voterName).length != 0
            && voterRegister[msg.sender].voted == false )
        {
            voterRegister[msg.sender].voted = true;
            vote memory v;
            v.voterAddresss = msg.sender;
            v.choice = _choice;
            if(_choice) {
                countResult++;
            }
            votes[totalVote] = v;
            totalVote++;
            isFound = true;
        }
        return isFound;
    }
    function endVote()
        public
        inState(State.Voting)
        onlyOfficial
    {
        state = State.Ended;
        finalResult = countResult;
    }

}
```

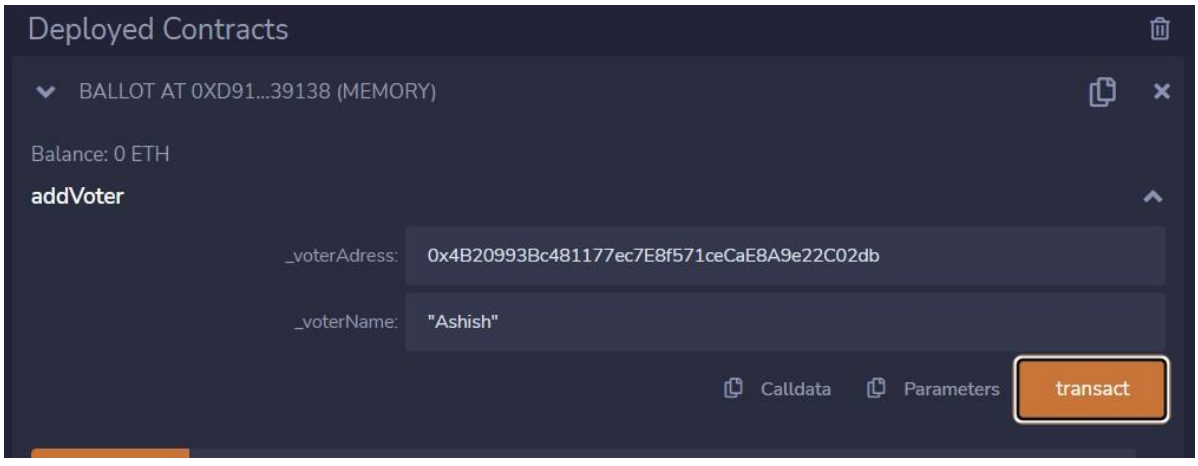
1. **Deploy contract for election**



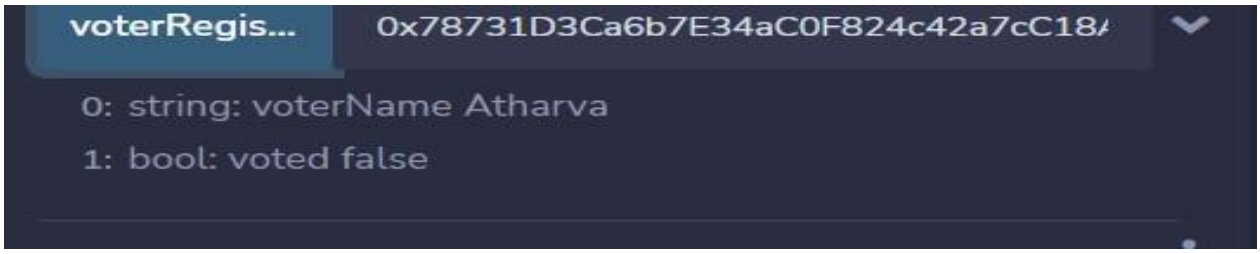
2. **Add voters**


Balance: 0 ETH
addVoter
_voterAdress:
0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2
_voterName:
"Yash"
Calldata
Parameters
transact
doVote
bool _choice


BALLOT AT 0XD91...39138 (MEMORY)
Balance: 0 ETH
addVoter
_voterAdress:
0x78731D3Ca6b7E34aC0F824c42a7cC18A495cabaBb
_voterName:
"Atharva"
Calldata
Parameters
transact


Deployed Contracts
BALLOT AT 0XD91...39138 (MEMORY)
Balance: 0 ETH
addVoter
_voterAdress:
0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db
_voterName:
"Ashish"
Calldata
Parameters
transact

3. **Start election**

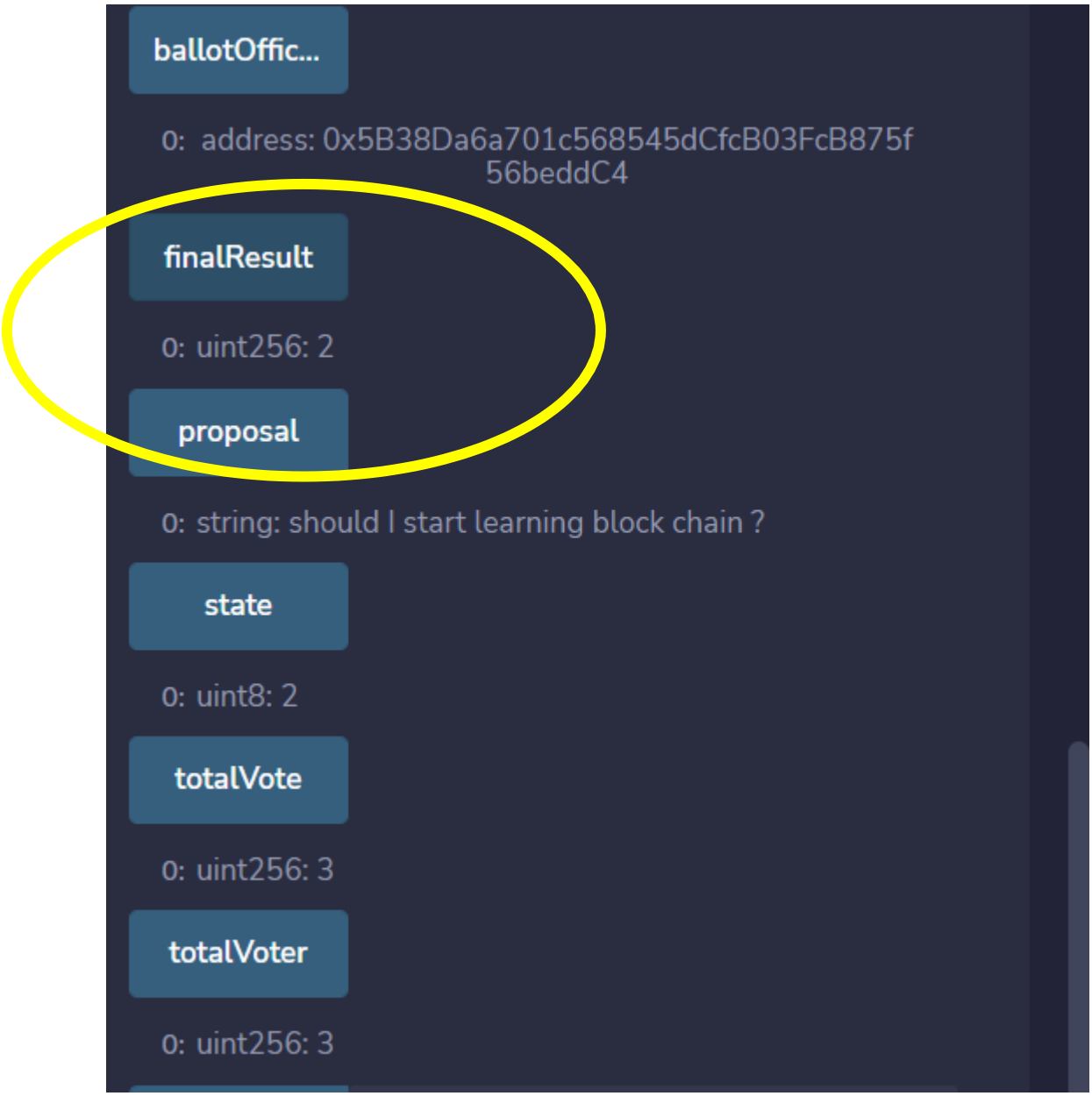


4 . results

2 votes :yes

1 vote :no

2/3



MINI PROJECT :

Decnetralized e-voting system using solidity:

CODE:

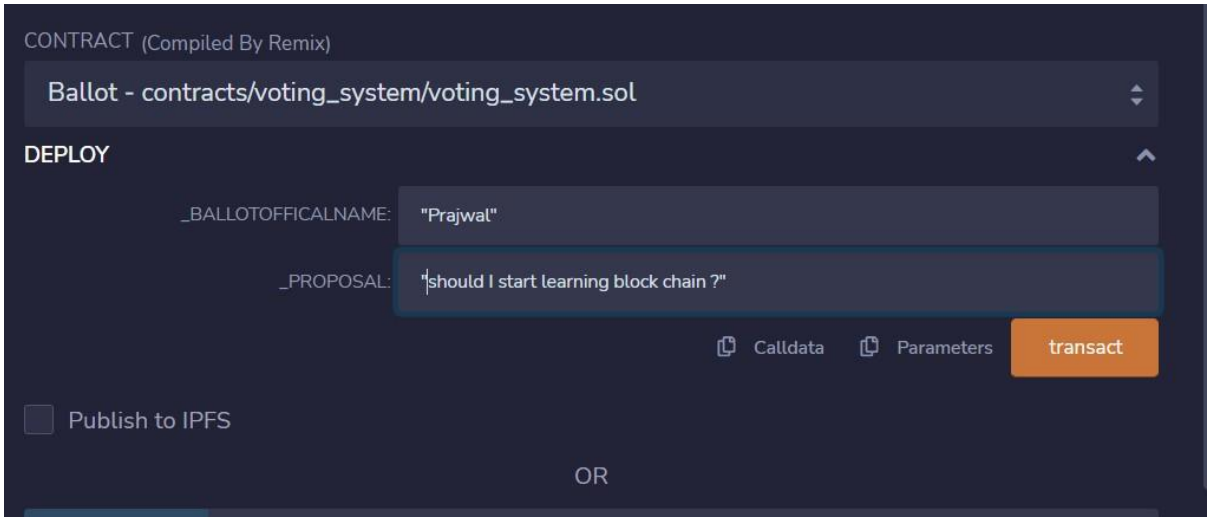
```
// SPDX-License-Identifier: MIT
pragma solidity >= 0.7.0 <0.8.0;


contract Ballot {
   // VARIBLES
   struct vote {
      address voterAddresss;
      bool choice;
   }
   struct voter {
      string voterName;
      bool voted;
   }
   uint private countResult = 0;
   uint public finalResult = 0;
   uint public totalVoter = 0;
   uint public totalVote = 0;

   address public ballotOfficialAddress;
   string public ballotOfficalName;
   string public proposal;

   mapping(uint => vote) private votes;
   mapping(address => voter) public voterRegister;

   enum State { Created, Voting, Ended }
   State public state;
```

```
// MODIFIER
modifier condition(bool _condition) {
    require(_condition);
    _;
}

modifier onlyOfficial() {
    require(msg.sender == ballotOfficialAddress);
    _;
}

modifier inState(State _state) {
    require(state == _state);
    _;
}


// FUNCTION
constructor(
    string memory _ballotofficalName,
    string memory _proposal
)  {
    ballotOfficialAddress = msg.sender;
    ballotOfficalName = _ballotofficalName;
    proposal = _proposal;
    state = State.Created;
}
```

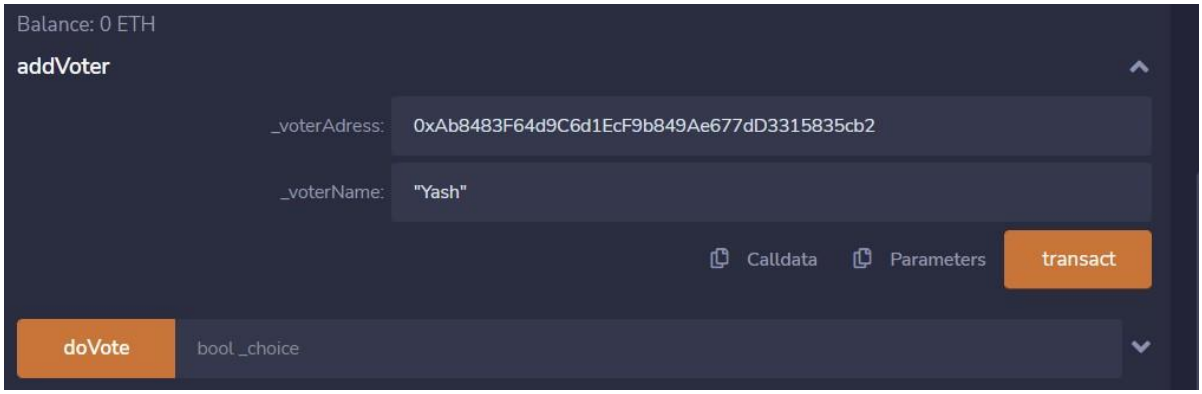
```
function addVoter(
    address _voterAdress,
    string memory _voterName
) public
    inState(State.Created)
    onlyOfficial
{
    voter memory v;
    v.voterName = _voterName;
    v.voted = false;
    voterRegister[_voterAdress] = v;
    totalVoter++;
}


function startVote()
    public
    inState(State.Created)
    onlyOfficial
{
    state = State.Voting;
}



function doVote(bool _choice)
    public
    inState(State.Voting)
    returns (bool voted)
{
    bool isFound = false;
```

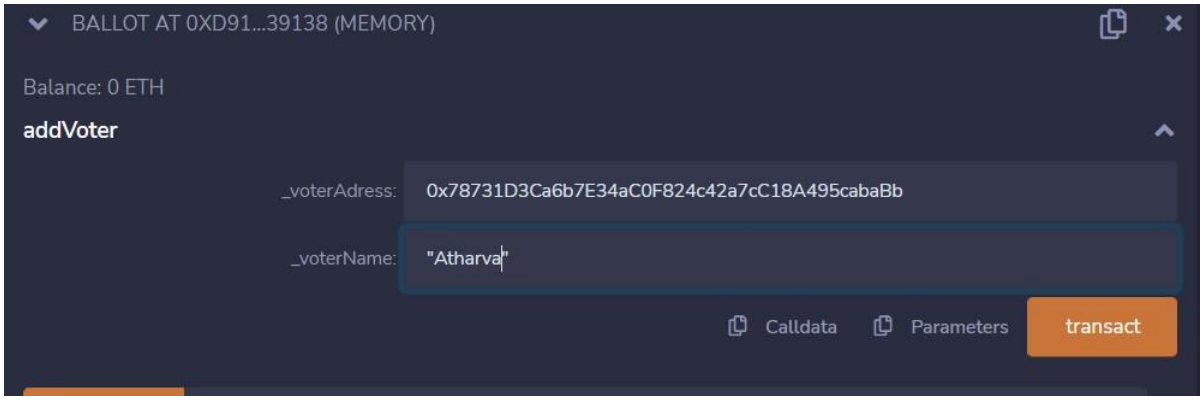
```
        if(bytes(voterRegister[msg.sender].voterName).length != 0
            && voterRegister[msg.sender].voted == false )
        {
            voterRegister[msg.sender].voted = true;
            vote memory v;
            v.voterAddresss = msg.sender;
            v.choice = _choice;
            if(_choice) {
                countResult++;
            }
            votes[totalVote] = v;
            totalVote++;
            isFound = true;
        }
        return isFound;
    }
    function endVote()
        public
        inState(State.Voting)
        onlyOfficial
    {
        state = State.Ended;
        finalResult = countResult;
    }

}
```

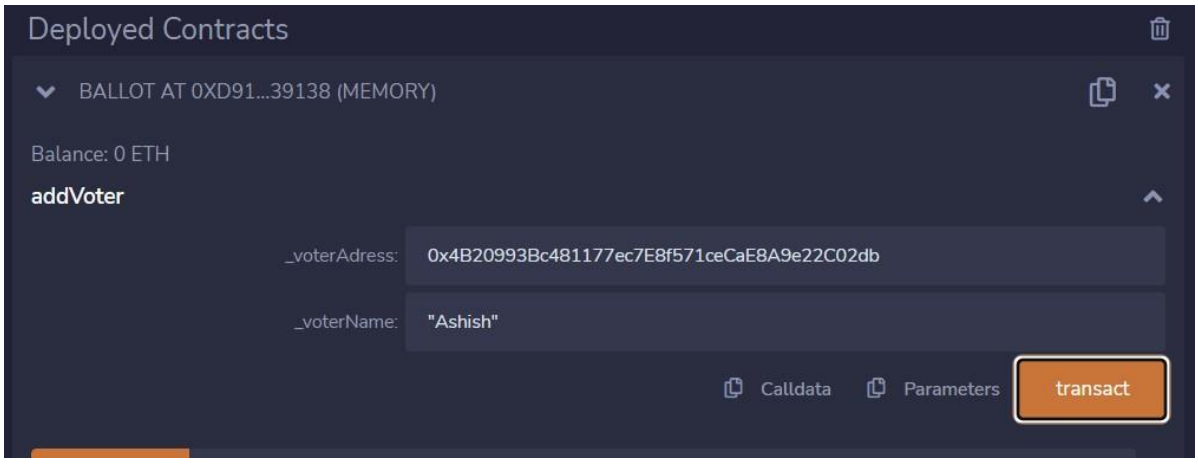
1. **Deploy contract for election**



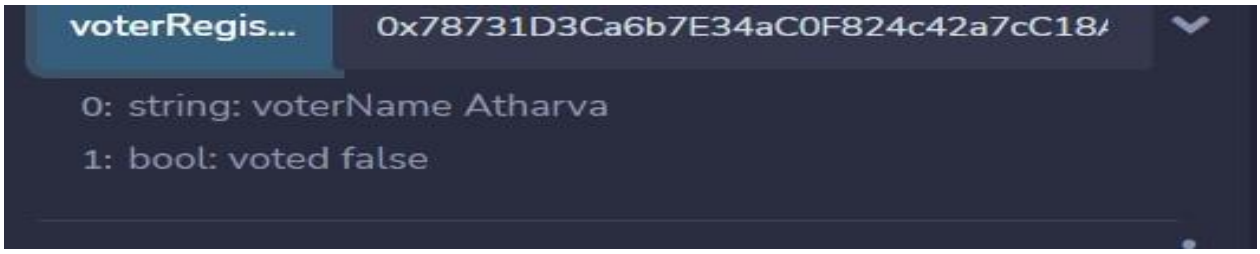
2. **Add voters**


Balance: 0 ETH
addVoter
_voterAdress:
0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2
_voterName:
"Yash"
Calldata
Parameters
transact
doVote
bool _choice


BALLOT AT 0XD91...39138 (MEMORY)
Balance: 0 ETH
addVoter
_voterAdress:
0x78731D3Ca6b7E34aC0F824c42a7cC18A495cabaBb
_voterName:
"Atharva"
Calldata
Parameters
transact


Deployed Contracts
BALLOT AT 0XD91...39138 (MEMORY)
Balance: 0 ETH
addVoter
_voterAdress:
0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db
_voterName:
"Ashish"
Calldata
Parameters
transact

3. **Start election**



4 . results

2 votes :yes

1 vote :no

2/3