MAXWELL TETTEH
AMSCU3CDS20052
S5 BCA-DATA SCIENCE
20CSA349-BLOCKCHAIN TECHOLOGIES

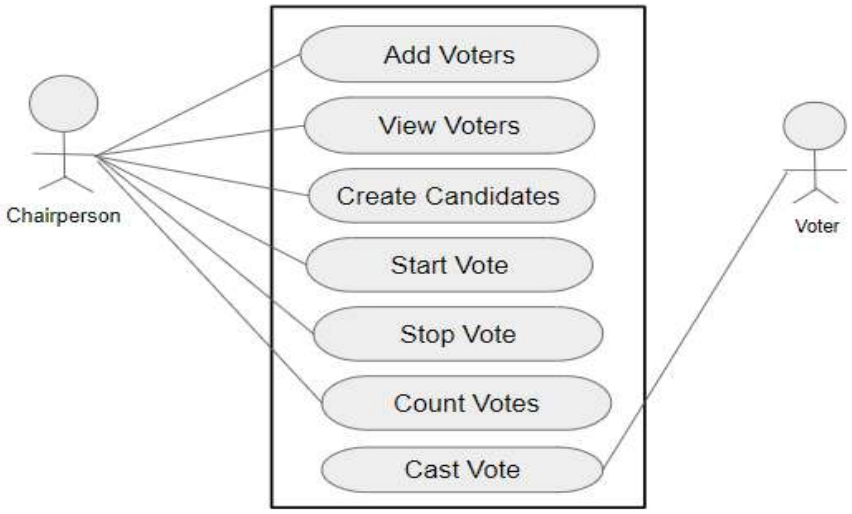
ASSIGNMENT:

QUESTION 1:

Ballot.sol

1. Ballot.sol is a smart contract to conduct voting with a specific set of candidates. Modify this contract with the following functionalities
   - Chairperson can specify the voters (hint: take a set of valid EOA as voters list)
   - Chairperson can stop the voting

Use-Case Diagram:



Ballot Use-Case Diagram

Code:

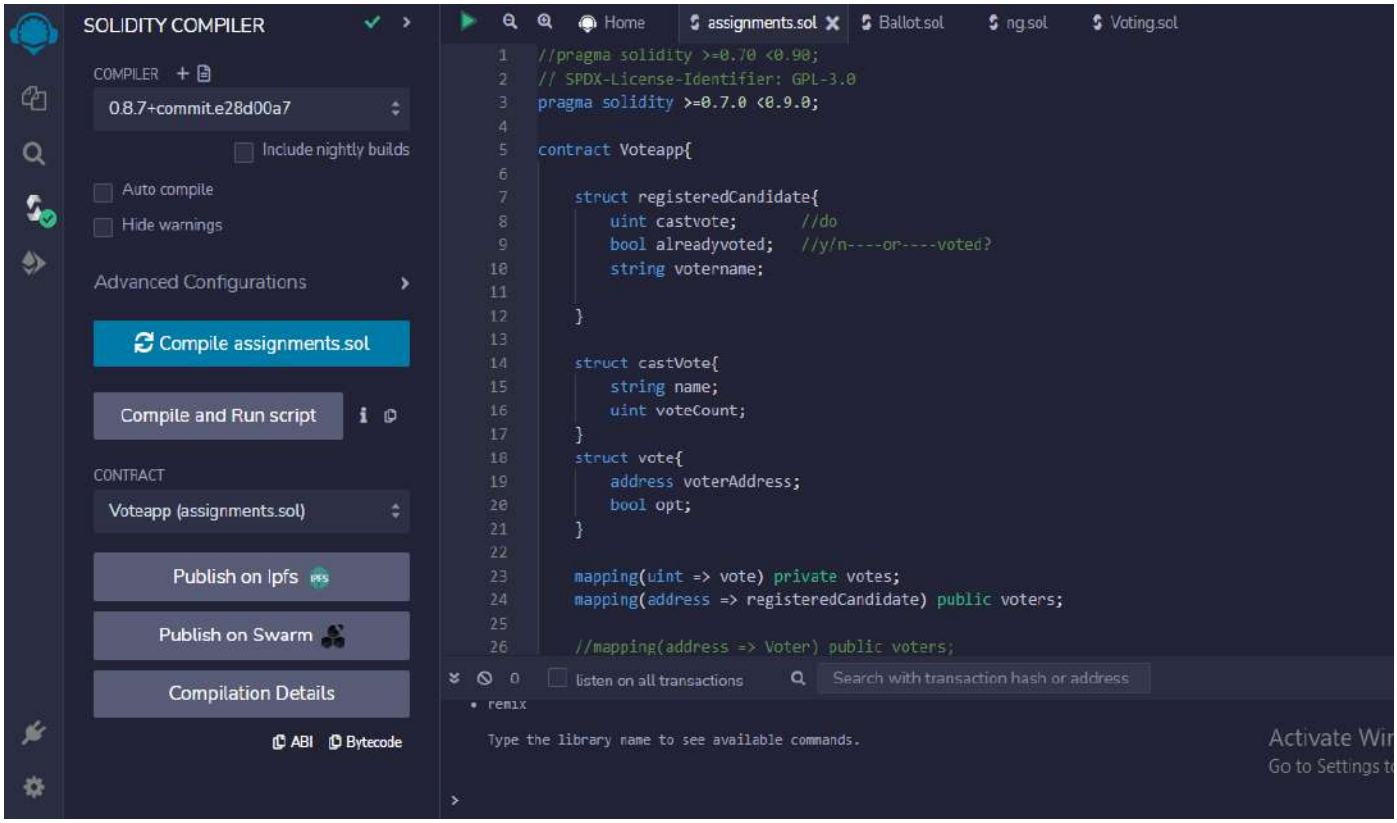
```
//pragma solidity >=0.70 <0.90;
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.7.0 <0.9.0;

contract Voteapp{

    struct registeredCandidate{
        uint castvote;       //do
        bool alreadyvoted;   //y/n----or----voted?
```

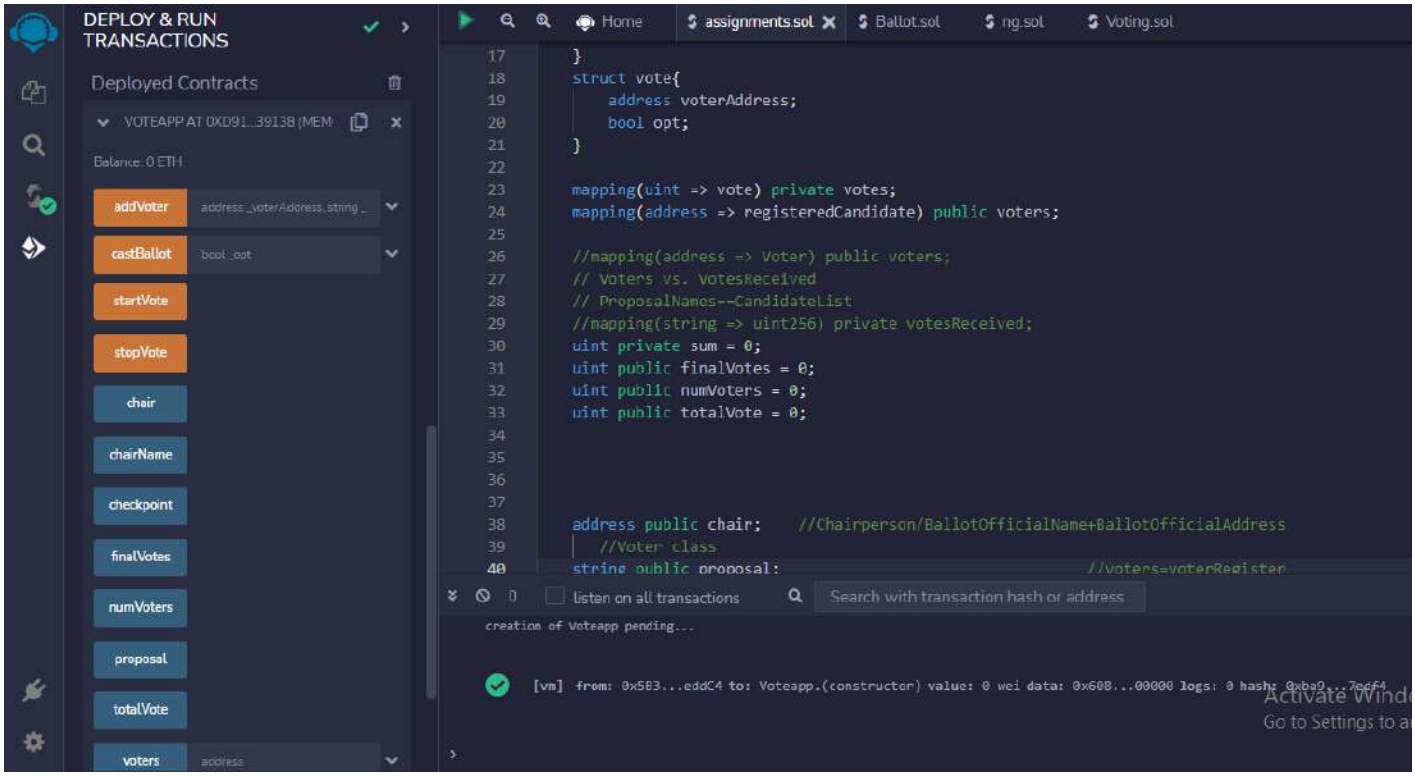
```
        string votername;

    }

    struct castVote{
        string name;
        uint voteCount;
    }
    struct vote{
        address voterAddress;
        bool opt;
    }

    mapping(uint => vote) private votes;
    mapping(address => registeredCandidate) public voters;

    //(registeredCandidate => voters);

    //mapping(string => uint256) private votesReceived;
    uint private sum = 0;
    uint public finalVotes = 0;
    uint public numVoters = 0;
    uint public totalVote = 0;




    address public chair;    //Chairperson
   //Voter class
    string public proposal;
    string public chairName;


    enum Checkpoint{Start, Ongoing, Stop}
        Checkpoint public checkpoint;
        constructor(string memory _chairName,string memory _proposal) {
            chair=msg.sender;
            chairName=_chairName;
            proposal=_proposal;

            checkpoint=Checkpoint.Start;
    }

        modifier condition(bool _condition) {
            require(_condition);
            _;
        }

        modifier restrictedChair() {
            require(msg.sender == chair);
```

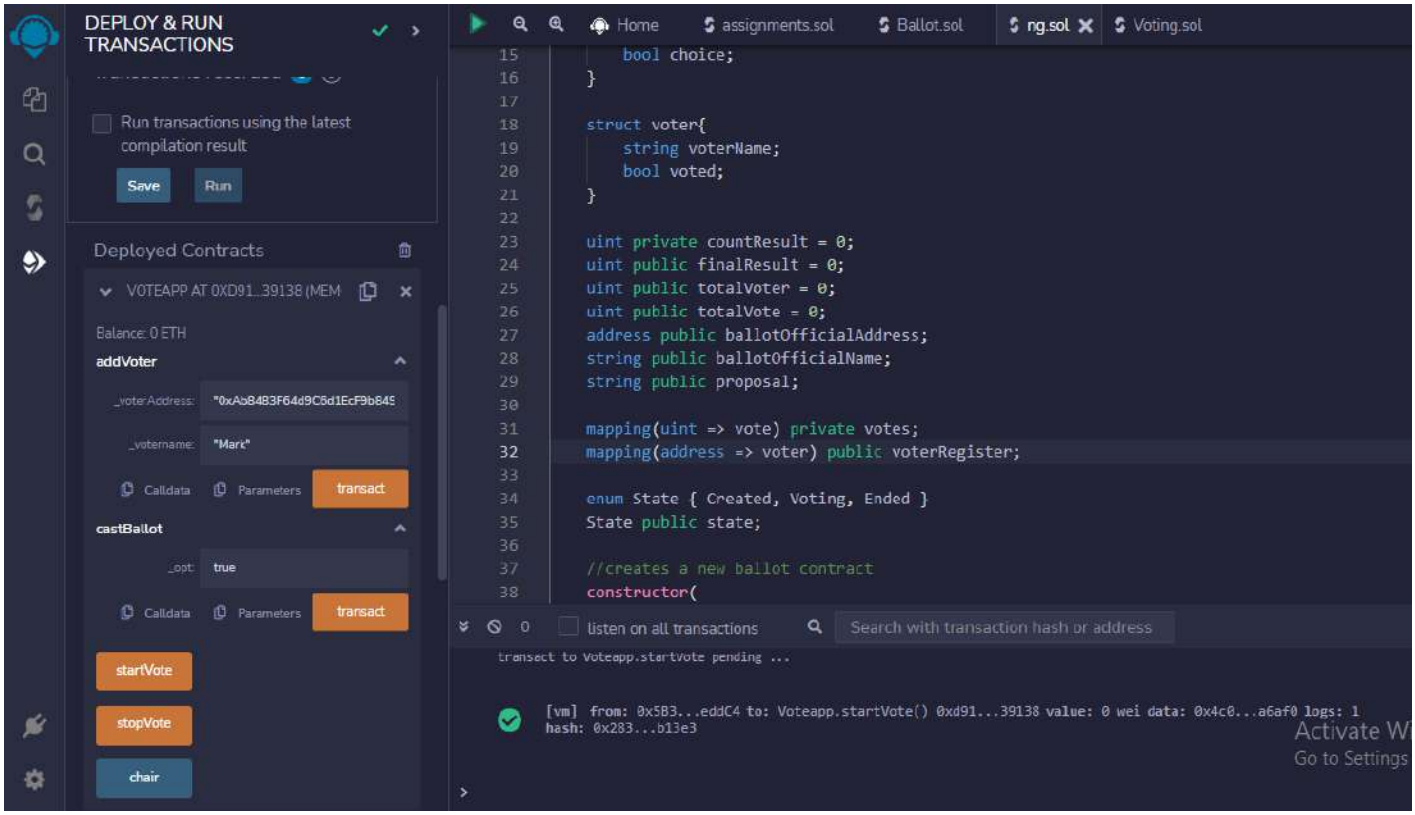
```
        _;
    }

    modifier currentState(Checkpoint _checkpoint) {
        require(checkpoint == _checkpoint);
        _;
    }


    event voterAdded(address registeredCandidate);
    event voteDone(address registeredCandidate);
    event voteStarted();
    event voteStopped(uint finalVotes);

    function addVoter(address _voterAddress, string memory _votername)
        public
        currentState(Checkpoint.Start)
        restrictedChair
    {
        registeredCandidate memory poll;    //poll
        poll.votername = _votername;
        poll.alreadyvoted = false;
        voters[_voterAddress] = poll;
        numVoters++;
    emit voterAdded(_voterAddress);
    }

    function startVote() public

        currentState(Checkpoint.Start)
        restrictedChair
    {
        checkpoint = Checkpoint.Ongoing;
        emit voteStarted();
    }

    function castBallot(bool _opt) public

        currentState(Checkpoint.Ongoing)
        returns (bool alreadyvoted)
        {
            bool found=false;

            if (bytes(voters[msg.sender].votername).length != 0
                && !voters[msg.sender].alreadyvoted){
                voters[msg.sender].alreadyvoted = true;
                vote memory poll;
                poll.voterAddress = msg.sender;
                poll.opt = _opt;
                if (_opt){
```

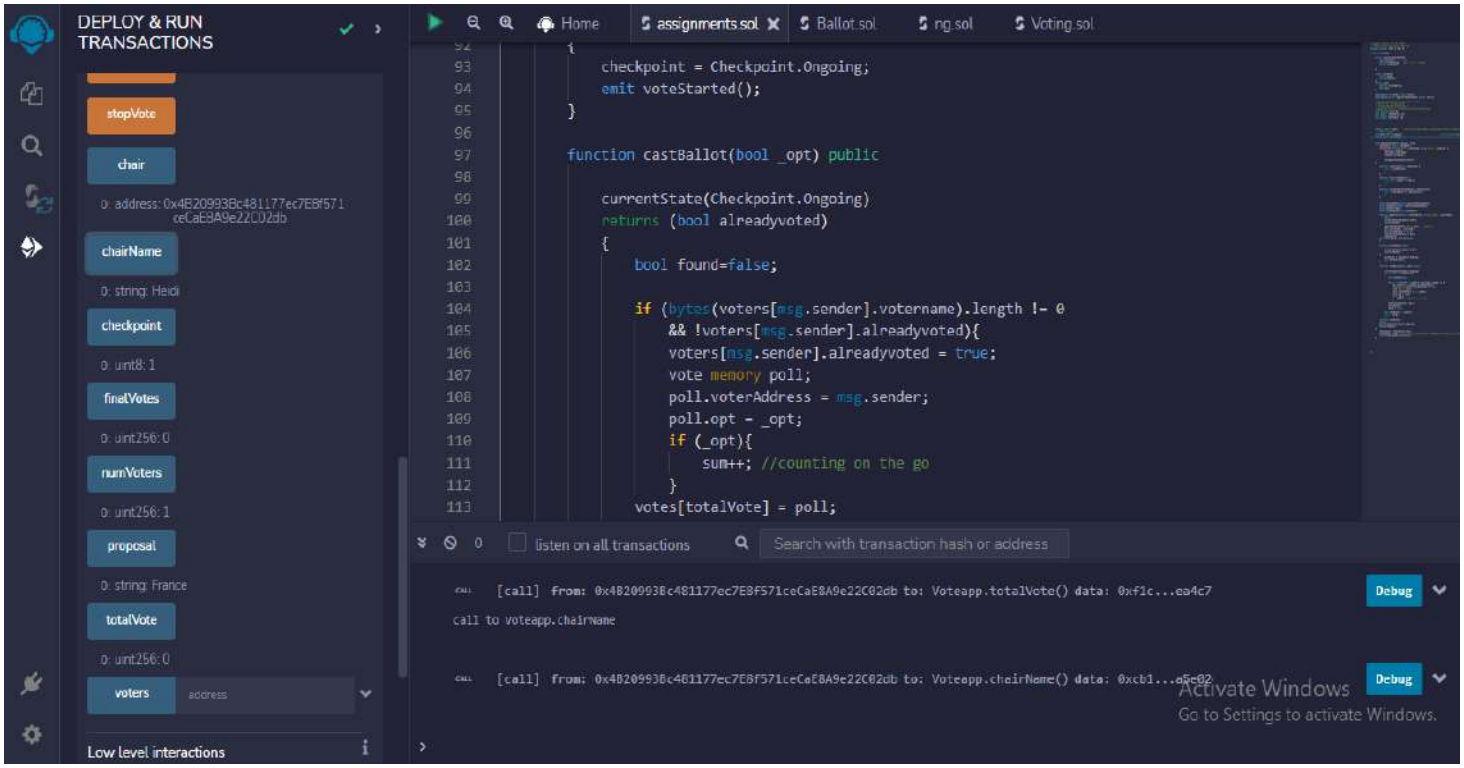
```
                        sum++;
                    }
                votes[totalVote] = poll;
                totalVote++;
                found = true;
            }
            emit voteDone(msg.sender);
            return found;
            }
        function stopVote()
        public
        currentState(Checkpoint.Ongoing)
        restrictedChair
    {
        checkpoint = Checkpoint.Stop;
        finalVotes = sum;
        emit voteStopped(finalVotes);
    }

}
```

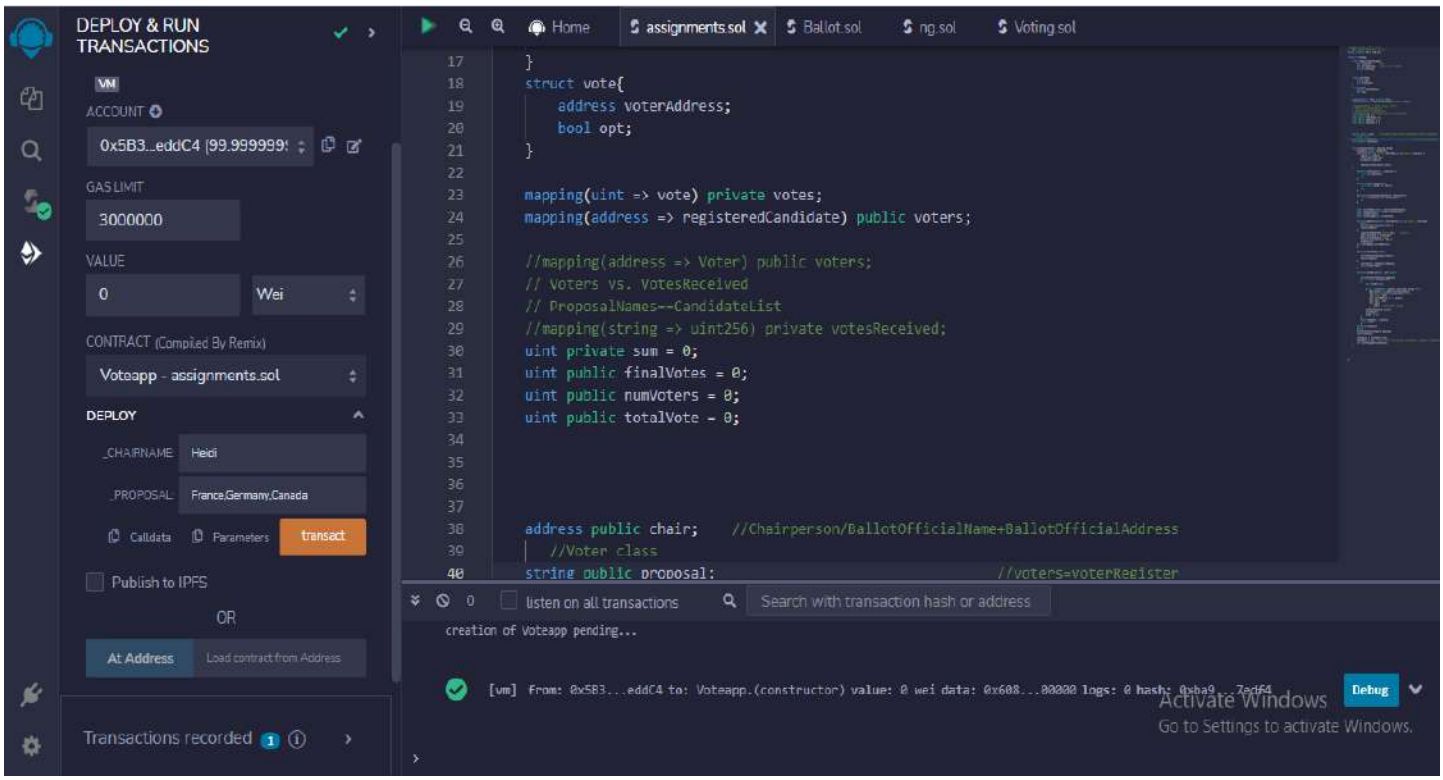
Execution Screenshots:



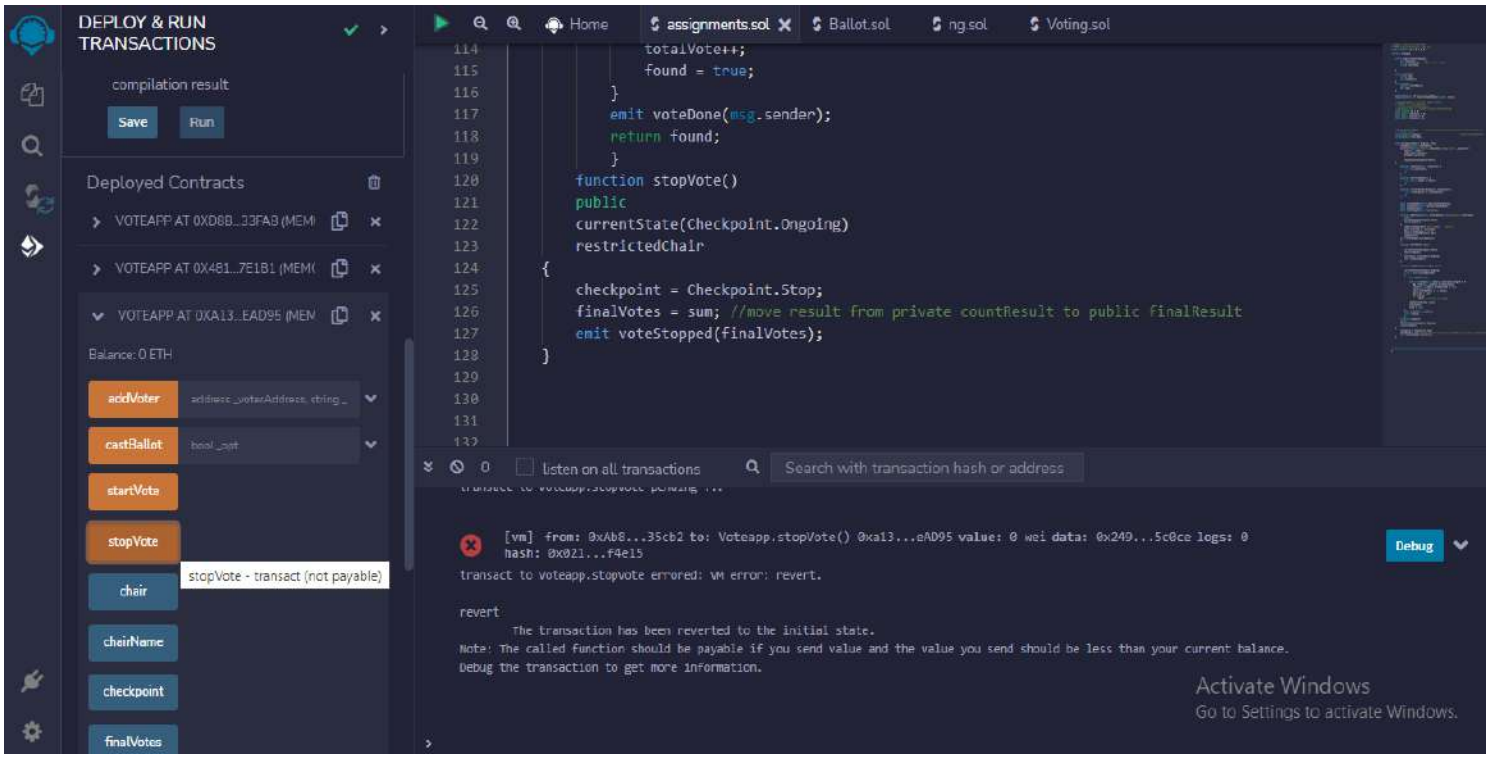

DEPLOY & RUN TRANSACTIONS
Deployed Contracts
VOTEAPP AT 0XD91...39138 (MEM
Balance: 0 ETH
addVoter
address _voterAddress, string _
castBallot
bool _opt
startVote
stopVote
chair
chairName
checkpoint
finalVotes
numVoters
proposal
totalVote
voters
address
Home
assignments.sol
Ballot.sol
ng.sol
Voting.sol
}
struct vote{
address voterAddress;
bool opt;
}
mapping(uint => vote) private votes;
mapping(address => registeredCandidate) public voters;
//mapping(address => Voter) public voters;
// Voters vs. VotesReceived
// ProposalNames--CandidateList
//mapping(string => uint256) private votesReceived;
uint private sum = 0;
uint public finalVotes = 0;
uint public numVoters = 0;
uint public totalVote = 0;
address public chair; //Chairperson/BallotOfficialName+BallotOfficialAddress
//Voter class
string public proposal; //voters=voterRegister
listen on all transactions
Search with transaction hash or address
creation of Voteapp pending...
[vm] from: 0x5B3...eddC4 to: Voteapp.(constructor) value: 0 wei data: 0x608...00000 logs: 0 hash: 0xba9...7ecf4


DEPLOY & RUN TRANSACTIONS
Run transactions using the latest compilation result
Save
Run
Deployed Contracts
VOTEAPP AT 0XD91...39138 (MEM
Balance: 0 ETH
addVoter
_voterAddress: "0xAb8483F64d9C6d1EcF9b849
_votername: "Mark"
Calldata
Parameters
transact
castBallot
_opt: true
Calldata
Parameters
transact
startVote
stopVote
chair
Home
assignments.sol
Ballot.sol
ng.sol
Voting.sol
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 ballotOfficialName;
string public proposal;
mapping(uint => vote) private votes;
mapping(address => voter) public voterRegister;
enum State { Created, Voting, Ended }
State public state;
//creates a new ballot contract
constructor(
listen on all transactions
Search with transaction hash or address
transact to Voteapp.startVote pending ...
[vm] from: 0x5B3...eddC4 to: Voteapp.startVote() 0xd91...39138 value: 0 wei data: 0x4c0...a6af0 logs: 1
hash: 0x283...b13e3


DEPLOY & RUN TRANSACTIONS
stopVote
chair
0: address: 0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db
chairName
0: string: Heidi
checkpoint
0: uint8: 1
finalVotes
0: uint256: 0
numVoters
0: uint256: 1
proposal
0: string: France
totalVote
0: uint256: 0
voters
address
Low level interactions
Home
assignments.sol
Ballot.sol
ng.sol
Voting.sol
checkpoint = Checkpoint.Ongoing;
emit voteStarted();
}
function castBallot(bool _opt) public
currentState(Checkpoint.Ongoing)
returns (bool alreadyvoted)
{
bool found=false;
if (bytes(voters[msg.sender].votername).length != 0
&& !voters[msg.sender].alreadyvoted){
voters[msg.sender].alreadyvoted = true;
vote memory poll;
poll.voterAddress = msg.sender;
poll.opt = _opt;
if (_opt){
sum++; //counting on the go
}
votes[totalVote] = poll;
listen on all transactions
Search with transaction hash or address
[call] from: 0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db to: Voteapp.totalVote() data: 0xf1c...ea4c7
Debug
call to Voteapp.chairName
[call] from: 0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db to: Voteapp.chairName() data: 0xcb1...e5e02
Debug
Activate Windows
Go to Settings to activate Windows.


DEPLOY & RUN TRANSACTIONS
VM
ACCOUNT
0x5B3...eddC4 (99.999999
GAS LIMIT
3000000
VALUE
0
Wei
CONTRACT (Compiled By Remix)
Voteapp - assignments.sol
DEPLOY
_CHAIRNAME
Heidi
_PROPOSAL
France,Germany,Canada
Calldata
Parameters
transact
Publish to IPFS
OR
At Address
Load contract from Address
Transactions recorded 1
Home
assignments.sol
Ballot.sol
ng.sol
Voting.sol
}
struct vote{
address voterAddress;
bool opt;
}
mapping(uint => vote) private votes;
mapping(address => registeredCandidate) public voters;
//mapping(address => Voter) public voters;
// Voters vs. VotesReceived
// ProposalNames--CandidateList
//mapping(string => uint256) private votesReceived;
uint private sum = 0;
uint public finalVotes = 0;
uint public numVoters = 0;
uint public totalVote = 0;
address public chair; //Chairperson/BallotOfficialName+BallotOfficialAddress
//Voter class
string public proposal; //voters=voterRegister
listen on all transactions
Search with transaction hash or address
creation of Voteapp pending...
[vm] from: 0x5B3...eddC4 to: Voteapp.(constructor) value: 0 wei data: 0x608...00000 logs: 0 hash: 0xba9...7edf4
Debug
Activate Windows
Go to Settings to activate Windows.

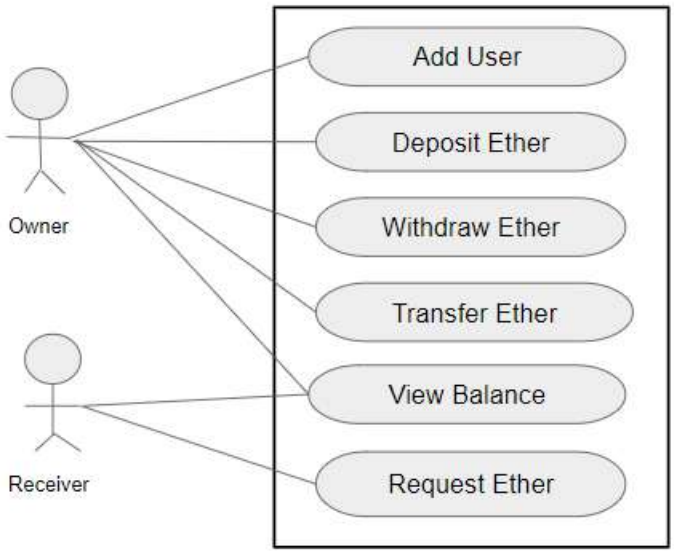
Changing the Address to a different one other than the Chairperson/Admin reverts the transaction when we try and access functions that are restricted for only the Chairperson to call/view.



QUESTION 2:

1. Implement a simple bank application using a smart contract with the following functionalities
    a. Deposit the money
    b. Allow the withdrawal by keeping a minimum balance (set minimum balance as 1 ETH)
    c. Transfer money between two valid accounts

Use-Case Diagram:

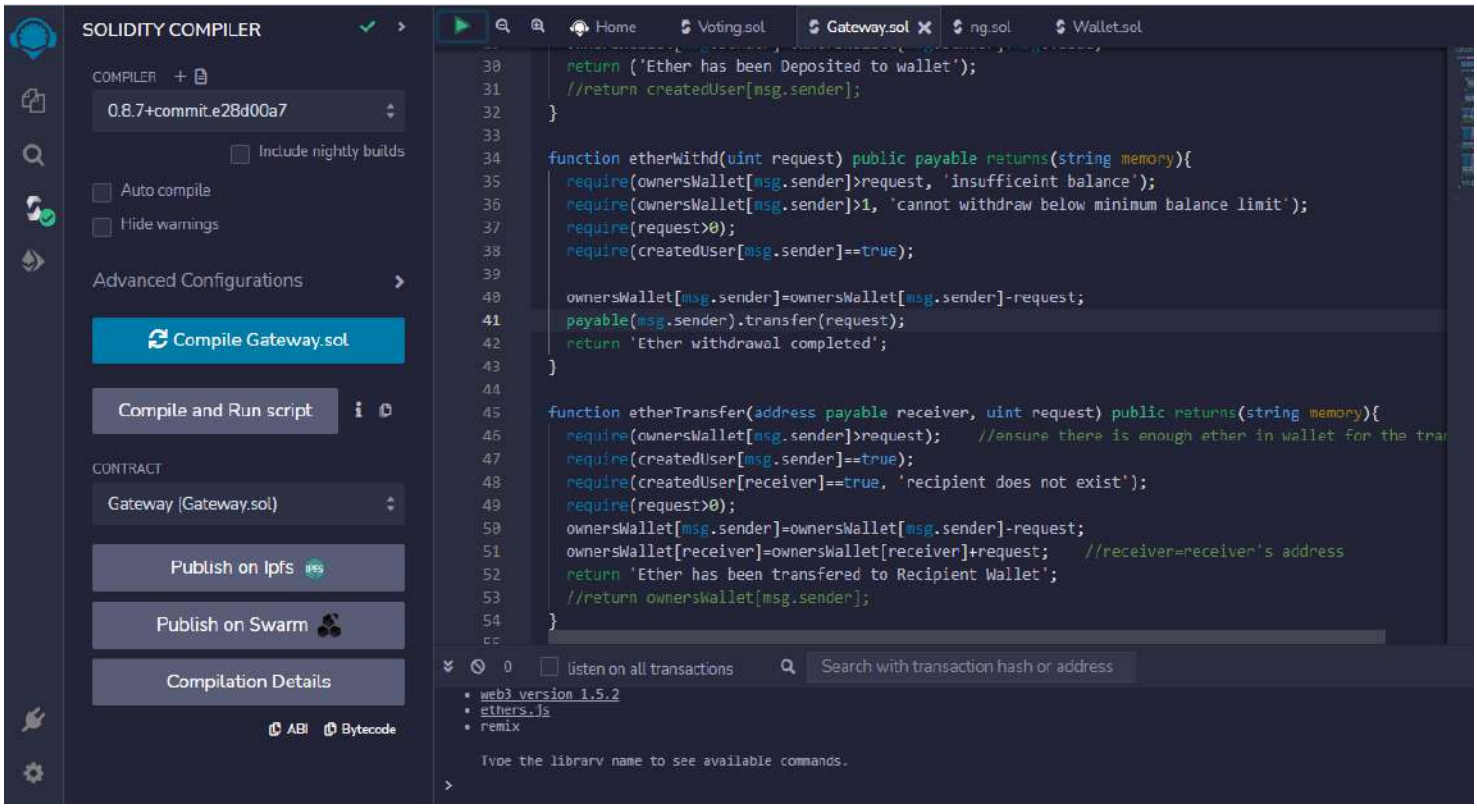


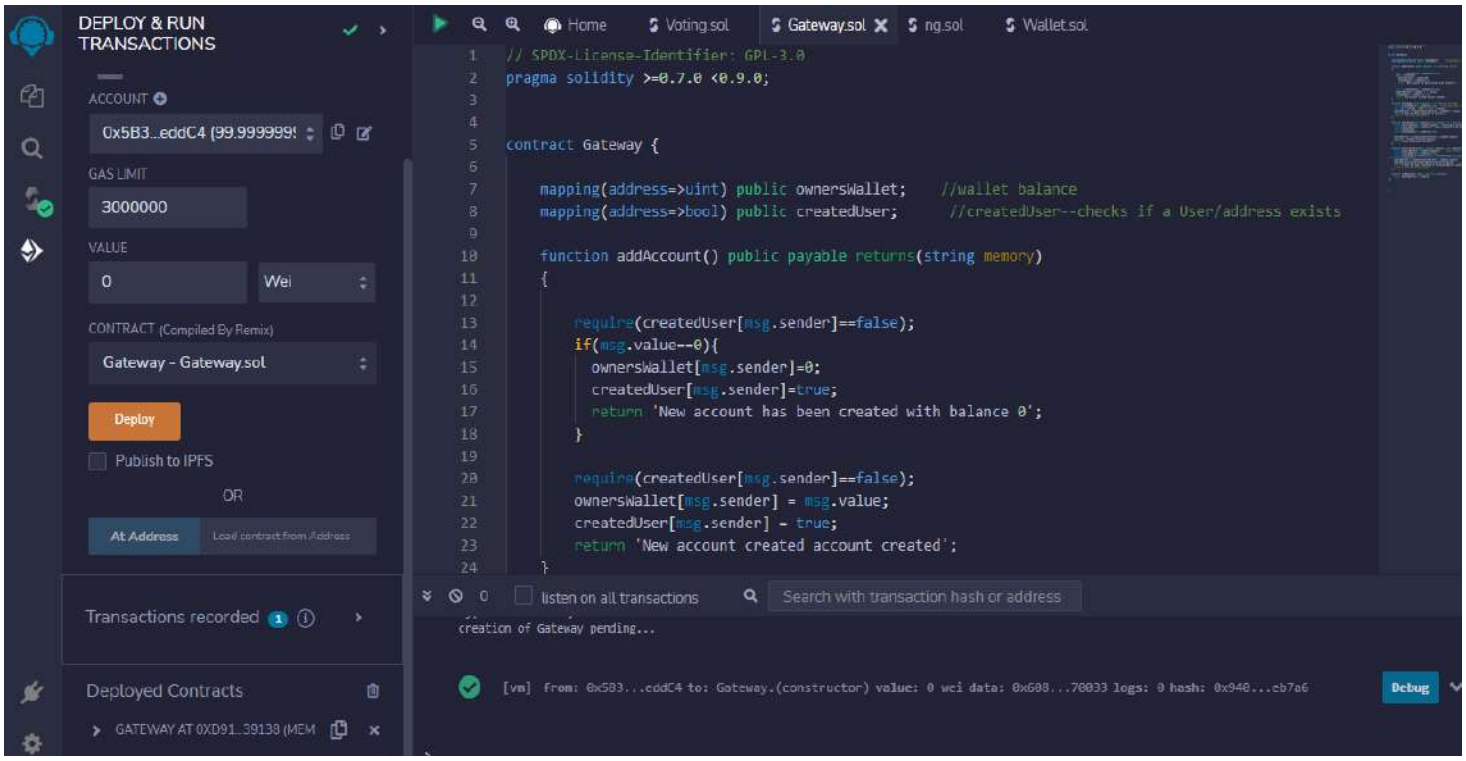
Use-case for Banking App

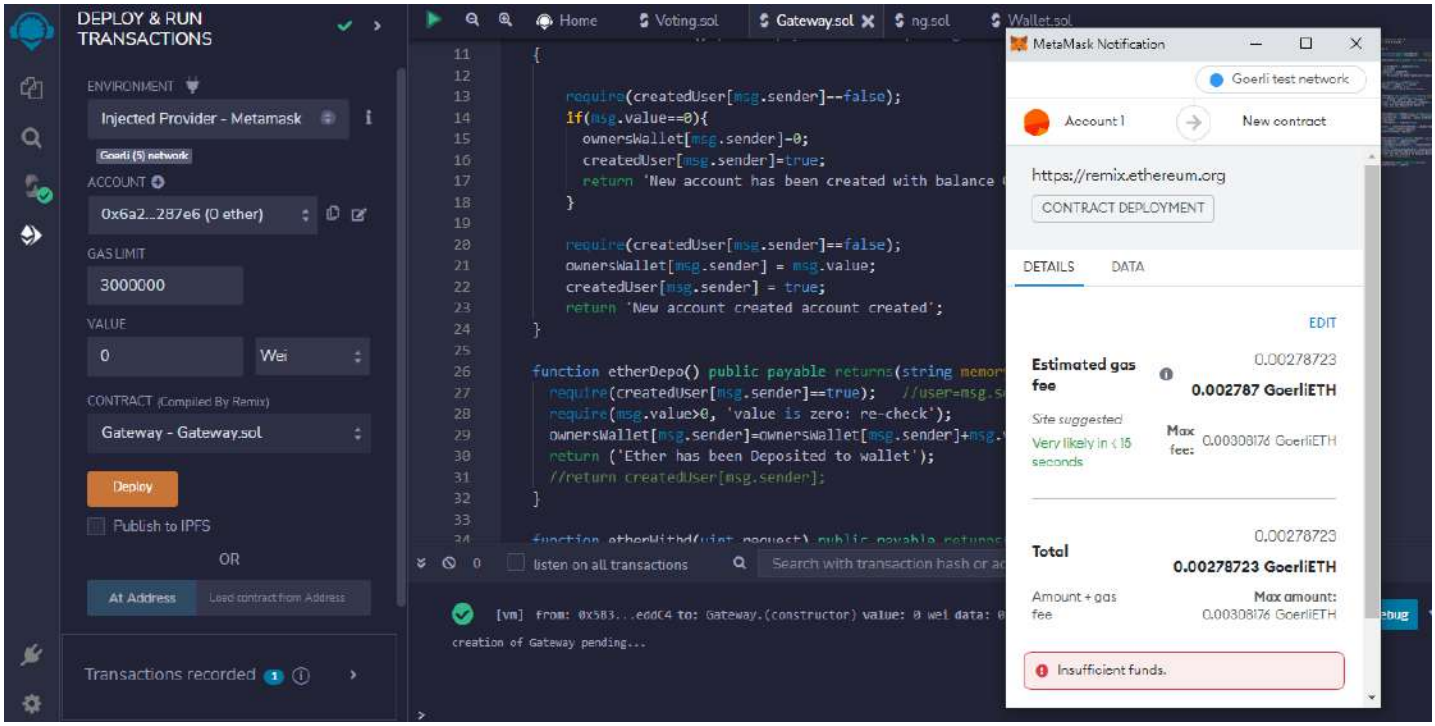
Code:

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.7.0 <0.9.0;


contract Gateway {

    mapping(address=>uint) public ownersWallet;    //wallet balance
    mapping(address=>bool) public createdUser;      //createdUser--checks if a User/address
exists

    function addAccount() public payable returns(string memory)
    {

        require(createdUser[msg.sender]==false);
        if(msg.value==0){
          ownersWallet[msg.sender]=0;
          createdUser[msg.sender]=true;
          return 'New account has been created with balance 0';
        }

        require(createdUser[msg.sender]==false);
        ownersWallet[msg.sender] = msg.value;
        createdUser[msg.sender] = true;
        return 'New account created account created';
    }

    function etherDepo() public payable returns(string memory){
      require(createdUser[msg.sender]==true);   //user=msg.sender else account is non-
existent.
```

```
      require(msg.value>0, 'value is zero: re-check');
      ownersWallet[msg.sender]=ownersWallet[msg.sender]+msg.value;
      return ('Ether has been Deposited to wallet');
      //return createdUser[msg.sender];
    }

    function etherWithd(uint request) public payable returns(string memory){
      require(ownersWallet[msg.sender]>request, 'insufficeint balance');
      require(ownersWallet[msg.sender]>1, 'cannot withdraw below minimum balance limit');
      require(request>0);
      require(createdUser[msg.sender]==true);

      ownersWallet[msg.sender]=ownersWallet[msg.sender]-request;
      payable(msg.sender).transfer(request);
      return 'Ether withdrawal completed';
    }

    function etherTransfer(address payable receiver, uint request) public returns(string
memory,uint){
      require(ownersWallet[msg.sender]>request);    //ensure there is enough ether in wallet
for the transfer
      require(createdUser[msg.sender]==true);
      require(createdUser[receiver]==true, 'recipient does not exist');
      require(request>0);
      ownersWallet[msg.sender]=ownersWallet[msg.sender]-request;
      ownersWallet[receiver]=ownersWallet[receiver]+request;    //receiver=receiver's
address
      return ('Ether has been transfered to Recipient Wallet',ownersWallet[msg.sender]);
      //return ownersWallet[msg.sender];
    }

    function viewBalance() public view returns(uint){
      return ownersWallet[msg.sender];
  }




}
```


SOLIDITY COMPILER
COMPILER
0.8.7+commit.e28d00a7
Include nightly builds
Auto compile
Hide warnings
Advanced Configurations
Compile Gateway.sol
Compile and Run script
CONTRACT
Gateway (Gateway.sol)
Publish on Ipfs
Publish on Swarm
Compilation Details
ABI
Bytecode
Home
Voting.sol
Gateway.sol
ng.sol
Wallet.sol
return ('Ether has been Deposited to wallet');
//return createdUser[msg.sender];
}
function etherWithd(uint request) public payable returns(string memory){
require(ownersWallet[msg.sender]>request, 'insufficeint balance');
require(ownersWallet[msg.sender]>1, 'cannot withdraw below minimum balance limit');
require(request>0);
require(createdUser[msg.sender]==true);
ownersWallet[msg.sender]=ownersWallet[msg.sender]-request;
payable(msg.sender).transfer(request);
return 'Ether withdrawal completed';
}
function etherTransfer(address payable receiver, uint request) public returns(string memory){
require(ownersWallet[msg.sender]>request);
require(createdUser[msg.sender]==true);
require(createdUser[receiver]==true, 'recipient does not exist');
require(request>0);
ownersWallet[msg.sender]=ownersWallet[msg.sender]-request;
ownersWallet[receiver]=ownersWallet[receiver]+request;
//receiver=receiver's address
return 'Ether has been transfered to Recipient Wallet';
//return ownersWallet[msg.sender];
}
0
listen on all transactions
Search with transaction hash or address
web3 version 1.5.2
ethers.js
remix
Type the library name to see available commands.


DEPLOY & RUN TRANSACTIONS
ACCOUNT
0x5B3...eddC4 (99.999999
GAS LIMIT
3000000
VALUE
0
Wei
CONTRACT (Compiled By Remix)
Gateway - Gateway.sol
Deploy
Publish to IPFS
OR
At Address
Load contract from Address
Transactions recorded 1
Deployed Contracts
GATEWAY AT 0XD91...39138 (MEM
Home
Voting.sol
Gateway.sol
ng.sol
Wallet.sol
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.7.0 <0.9.0;
contract Gateway {
mapping(address=>uint) public ownersWallet;
//wallet balance
mapping(address=>bool) public createdUser;
//createdUser--checks if a User/address exists
function addAccount() public payable returns(string memory)
{
require(createdUser[msg.sender]==false);
if(msg.value==0){
ownersWallet[msg.sender]=0;
createdUser[msg.sender]=true;
return 'New account has been created with balance 0';
}
require(createdUser[msg.sender]==false);
ownersWallet[msg.sender] = msg.value;
createdUser[msg.sender] = true;
return 'New account created account created';
}
0
listen on all transactions
Search with transaction hash or address
creation of Gateway pending...
[vm] from: 0x583...eddC4 to: Gateway.(constructor) value: 0 wei data: 0x608...70033 logs: 0 hash: 0x940...cb7a6
Debug



With GoerLiETH Wallet(MetaMask):