

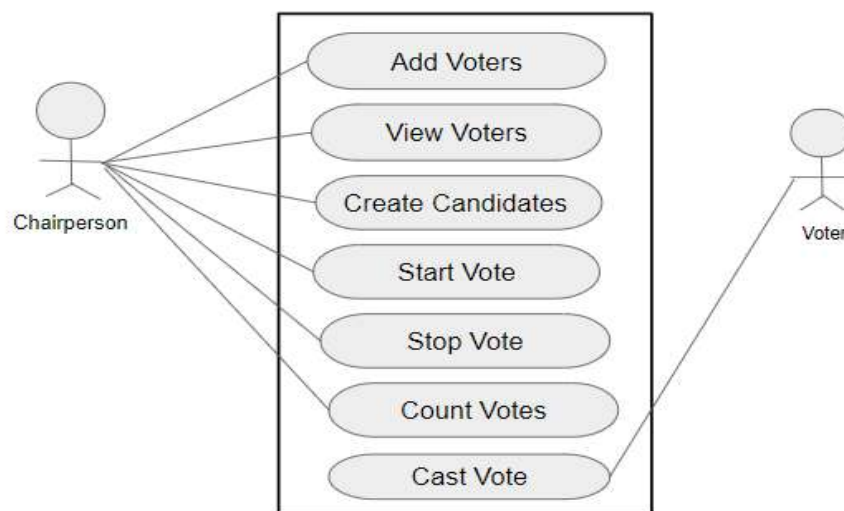
ASSIGNMENT:

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:

The screenshot displays the Solidity Compiler interface. On the left sidebar, the 'COMPILER' section shows version '0.8.7+commit.28d00a7' and options for 'Include nightly builds', 'Auto compile', and 'Hide warnings'. Below this, the 'CONTRACT' section shows 'Voteapp (assignments.sol)' selected. The main editor area displays the Solidity code for the 'Voteapp' contract, which includes a 'registeredCandidate' struct, a 'castVote' struct, a 'vote' struct, and mappings for 'votes' and 'voters'. The bottom status bar shows '0' transactions and a search bar for transaction hashes or addresses.

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

```
17 }
18 struct vote{
19     address voterAddress;
20     bool opt;
21 }
22
23 mapping(uint => vote) private votes;
24 mapping(address => registeredCandidate) public voters;
25
26 //mapping(address => Voter) public voters;
27 // Voters vs. VotesReceived
28 // ProposalNames--CandidateList
29 //mapping(string => uint256) private votesReceived;
30 uint private sum = 0;
31 uint public finalVotes = 0;
32 uint public numVoters = 0;
33 uint public totalVote = 0;
34
35
36
37
38 address public chair; //Chairperson/BallotOfficialName+BallotOfficialAddress
39 //Voter class
40 string public proposal; //voters=voterRegister
```

creation of Voteapp pending...

[vm] from: 0x5B3...eddC4 to: Voteapp.(constructor) value: 0 wei data: 0x608...00000 logs: 0 hash: 0xba9...7edF4

Activate Window
Go to Settings to a

DEPLOY & RUN TRANSACTIONS

Run transactions using the latest compilation result

Save **Run**

Deployed Contracts

VOTEAPP AT 0xD91...39138 (MEM)

Balance: 0 ETH

addVoter

_voterAddress: "0xAb8483F64d9C8d1EcF9b845"

_votename: "Mark"

transact

castBallot

_opt: true

transact

startVote

stopVote

chair

```
15     bool choice;
16 }
17
18 struct voter{
19     string voterName;
20     bool voted;
21 }
22
23 uint private countResult = 0;
24 uint public finalResult = 0;
25 uint public totalVoter = 0;
26 uint public totalVote = 0;
27 address public ballotOfficialAddress;
28 string public ballotOfficialName;
29 string public proposal;
30
31 mapping(uint => vote) private votes;
32 mapping(address => voter) public voterRegister;
33
34 enum State { Created, Voting, Ended }
35 State public state;
36
37 //creates a new ballot contract
38 constructor(
```

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

Activate Window
Go to Settings to a

DEPLOY & RUN TRANSACTIONS

stopVote

chair

chairName

checkpoint

finalVotes

numVotes

proposal

totalVote

voters

Low level interactions

Home

assignments.sol

Ballot.sol

ng.sol

Voting.sol

```
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
1
    checkpoint = Checkpoint.Ongoing;
    emit voteStarted();
}

function castBallot(bool _opt) public

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

        if (bytes(voters[msg.sender].votename).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;
        }
    }
}
```

0

☐ listen on all transactions

Search with transaction hash or address

[call] from: 0x48209938c481177ec7E8F571ceCaE8A9e22C02db to: Voteapp.totalVote() data: 0xf1c...ea4c7

call to Voteapp.chairname

[call] from: 0x48209938c481177ec7E8F571ceCaE8A9e22C02db to: Voteapp.chairName() data: 0xb1...05e02

Debug

Debug

DEPLOY & RUN TRANSACTIONS

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

Call data

Parameters

transact

☐ Publish to IPFS

OR

At Address

Load contract from Address

Transactions recorded 1

Home

assignments.sol

Ballot.sol

ng.sol

Voting.sol

```
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
}
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
```

0

☐ 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...0000 logs: 0 hash: 0xb9...7ed44

Debug

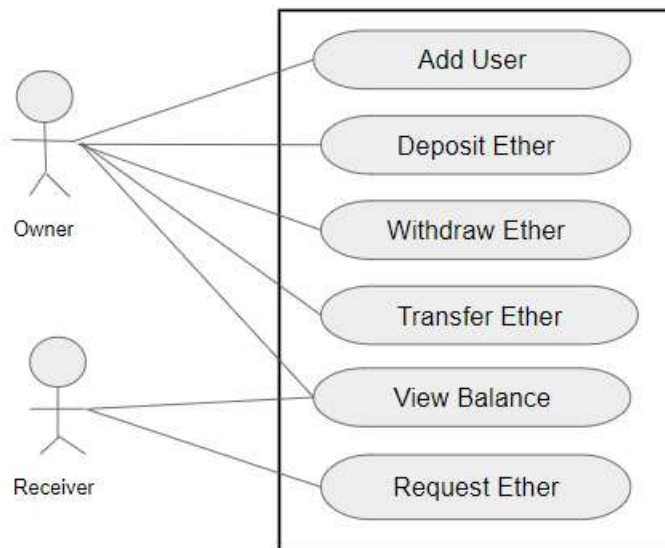
Changing the Address to a different one other than the Chairperson/Admin reverts the transaction when we try and access functions that are restricted for only the Chairperson to call/view.

The screenshot displays the Remix IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' panel shows a list of deployed contracts, including 'VOTEAPP AT 0XA13...EAD95 (MEM)'. Below this, a 'Balance: 0 ETH' is shown, and several buttons are available: 'addVoter', 'castBallot', 'startVote', 'stopVote', 'chair', 'chairName', 'checkpoint', and 'finalVotes'. The 'stopVote' button is highlighted, and a tooltip indicates 'stopVote - transact (not payable)'. The main editor shows the Solidity code for 'Voting.sol', with the 'stopVote' function visible. The bottom panel shows a transaction error: '[vm] from: 0xA8b...35cb2 to: Voteapp.stopVote() 0xA13...eAD95 value: 0 wei data: 0x249...5c0ce logs: 0 hash: 0x021...f4e13'. The error message states: 'transact to voteapp.stopvote errored: VM error: revert.' and 'revert: The transaction has been reverted to the initial state. Note: The called function should be payable if you send value and the value you send should be less than your current balance. Debug the transaction to get more information.'

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-
exists.
    }
}
```



```

    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 X ng.sol Wallets.sol

30 return ('Ether has been Deposited to wallet');
31 //return createdUser[msg.sender];
32 }
33
34 function etherWithdraw(uint request) public payable returns(string memory){
35 require(ownersWallet[msg.sender]>request, 'insufficeint balance');
36 require(ownersWallet[msg.sender]>1, 'cannot withdraw below minimum balance limit');
37 require(request>0);
38 require(createdUser[msg.sender]==true);
39
40 ownersWallet[msg.sender]=ownersWallet[msg.sender]-request;
41 payable(msg.sender).transfer(request);
42 return 'Ether withdrawal completed';
43 }
44
45 function etherTransfer(address payable receiver, uint request) public returns(string memory){
46 require(ownersWallet[msg.sender]>request); //ensure there is enough ether in wallet for the tran
47 require(createdUser[msg.sender]==true);
48 require(createdUser[receiver]==true, 'recipient does not exist');
49 require(request>0);
50 ownersWallet[msg.sender]=ownersWallet[msg.sender]-request;
51 ownersWallet[receiver]=ownersWallet[receiver]+request; //receiver=receiver's address
52 return 'Ether has been transferred to Recipient Wallet';
53 //return ownersWallet[msg.sender];
54 }
55 }

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...39133 (MEM)

Home Voting.sol Gateway.sol X ng.sol Wallets.sol

1 // SPDX-License-Identifier: GPL-3.0
2 pragma solidity >=0.7.0 <0.9.0;
3
4
5 contract Gateway {
6
7 mapping(address=>uint) public ownersWallet; //wallet balance
8 mapping(address=>bool) public createdUser; //createdUser--checks if a User/address exists
9
10 function addAccount() public payable returns(string memory)
11 {
12
13 require(createdUser[msg.sender]==false);
14 if(msg.value==0){
15 ownersWallet[msg.sender]=0;
16 createdUser[msg.sender]=true;
17 return 'New account has been created with balance 0';
18 }
19
20 require(createdUser[msg.sender]==false);
21 ownersWallet[msg.sender] = msg.value;
22 createdUser[msg.sender] = true;
23 return 'New account created account created';
24 }
25 }

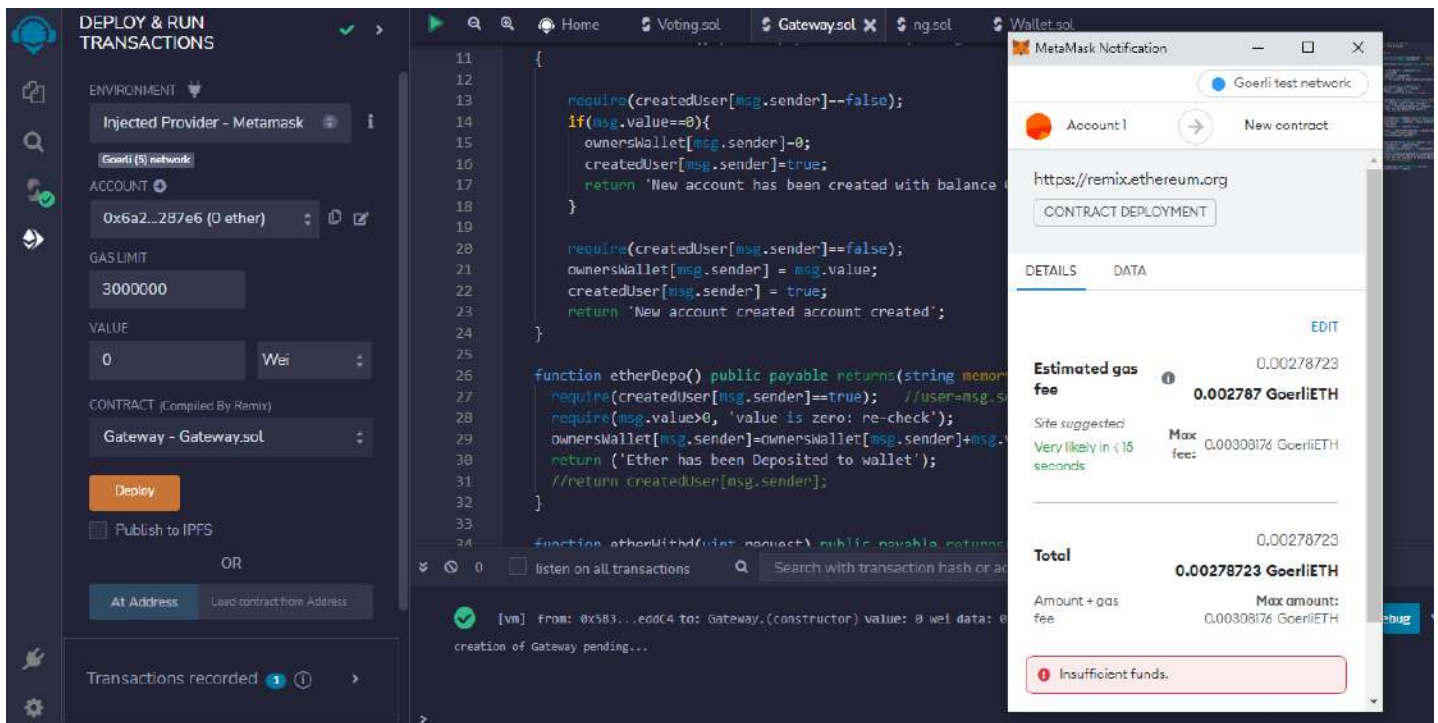
listen on all transactions

Search with transaction hash or address

creation of Gateway pending...

[vm] From: 0x5B3...eddC4 to: Gateway.(constructor) value: 0 wei data: 0x608...76033 logs: 0 hash: 0x940...cb7a6

Debug



With GoerliETH Wallet(MetaMask):

