BLOCKCHAIN LAB 1

Akash Kumar Roy
Person Number: 50316991
Email ID: akashroy@buffalo.edu

Title of the Idea: Vote for New Cryptocurrency

Relevant Links:

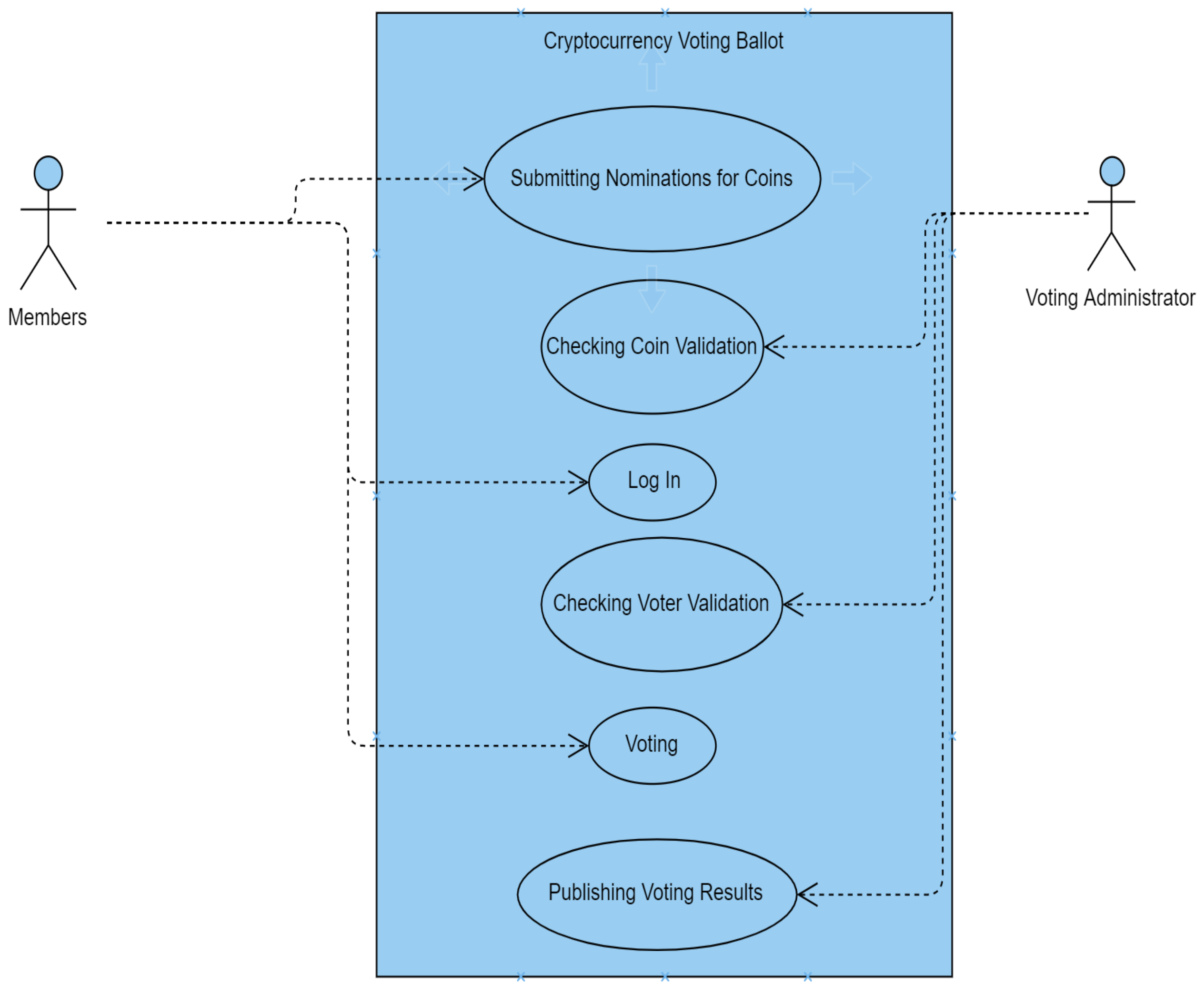
- https://coindeal.com/vote
- https://static.coindeal.com/voting_rules.pdf?1569508398&_ga=2.88138845.758097875.1569544721-1474087859.1569544721

Abstract (100 Words):

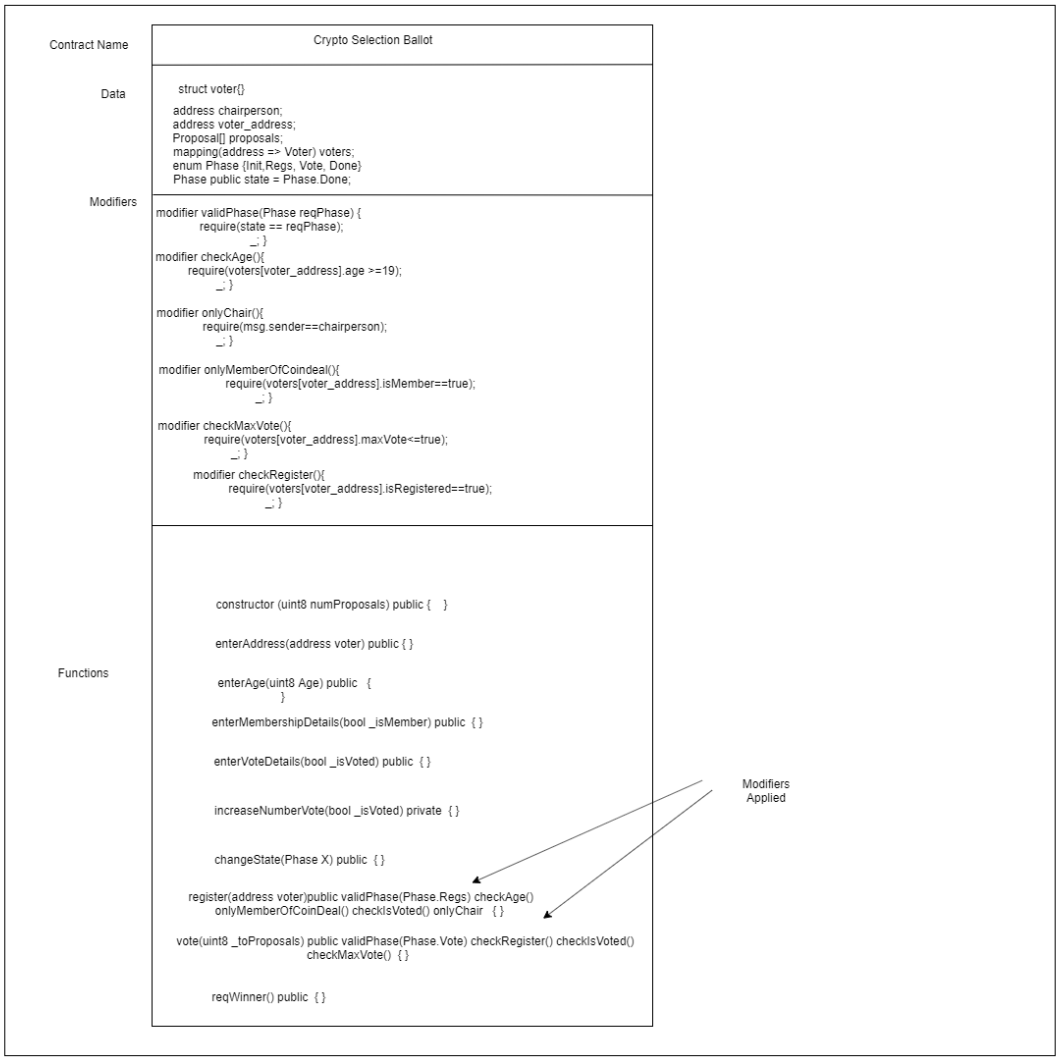
- The Voting system is for next upcoming Cryptocurrency. After 4 weeks of voting the Highest Voted Cryptocurrency will be implemented by coindeal.com
- To become a contestant in the voting the following criteria must be satisfied by the new coin:
    - Must provide the personal data of the creators of the Coin
    - The coin code must have a open repository
    - Must have a link indicating the full amount of coins in real time
    - Must have information about shares of the coins (how many creators and developers hold and how many were designated for other entities or for other purposes).
    - Must have a valid links to social media profiles, website
    - Must provide value of the Coin for the day of registration (cannot be lower than 0,00000005 BTC for 1 Coin/Token)
- General voting Rules:
    - Voter must have to be a part of the coindeal.com organization
    - Every voter during the Duration time can cast up to five votes a day, but can place only one vote for one Coin
    - The voting duration will be of 4 weeks.
    - If two or more Coins collect the same number of votes in the voting, Coin that collected greater amount of votes in previous all previous voting's shall be considered a winner.

There are other rules but I think these are the most important ones so I have implemented these as a part of the project.

USE CASE DIAGRAM:

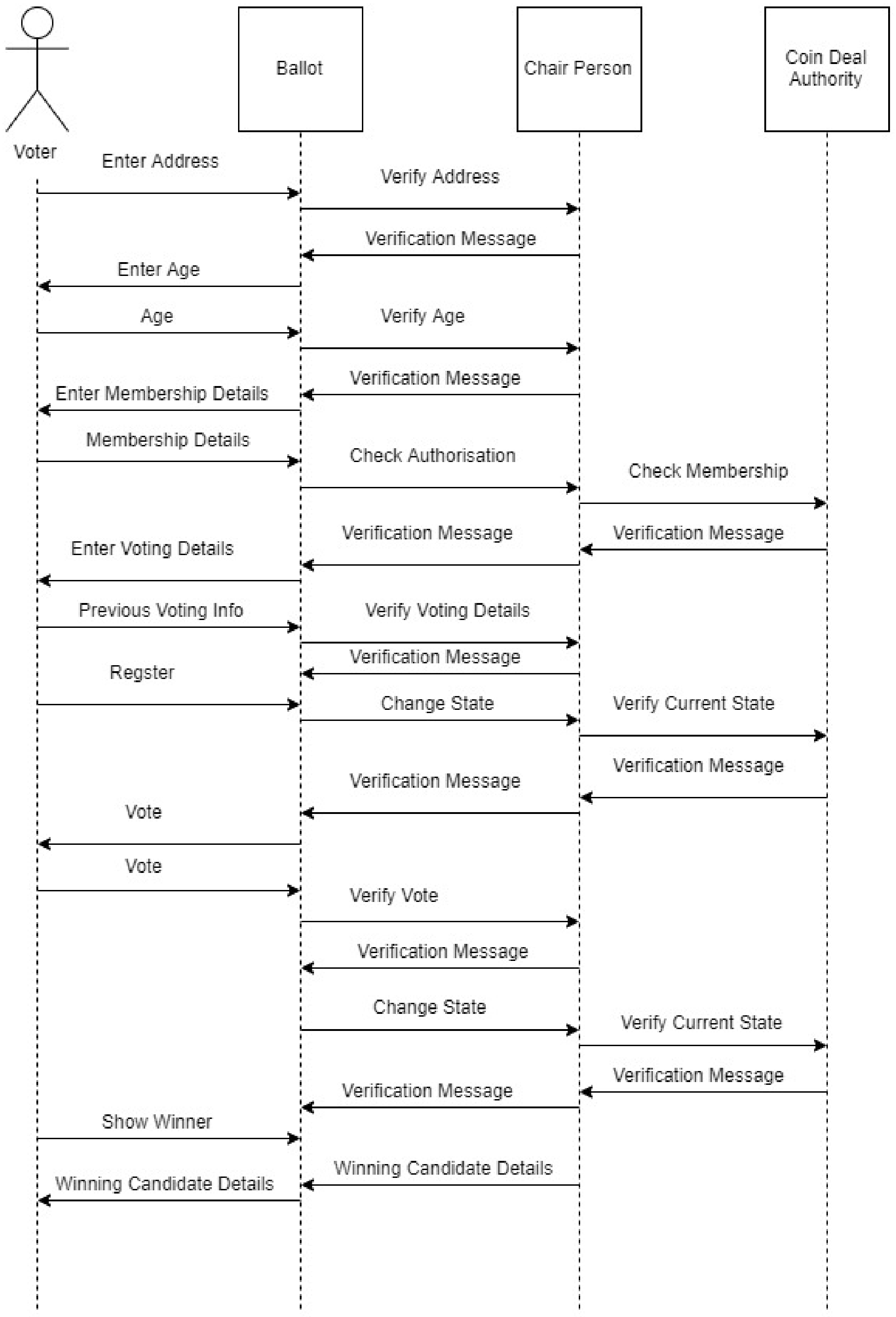

Cryptocurrency Voting Ballot
Members
Voting Administrator
Submitting Nominations for Coins
Checking Coin Validation
Log In
Checking Voter Validation
Voting
Publishing Voting Results

Contract Diagram:

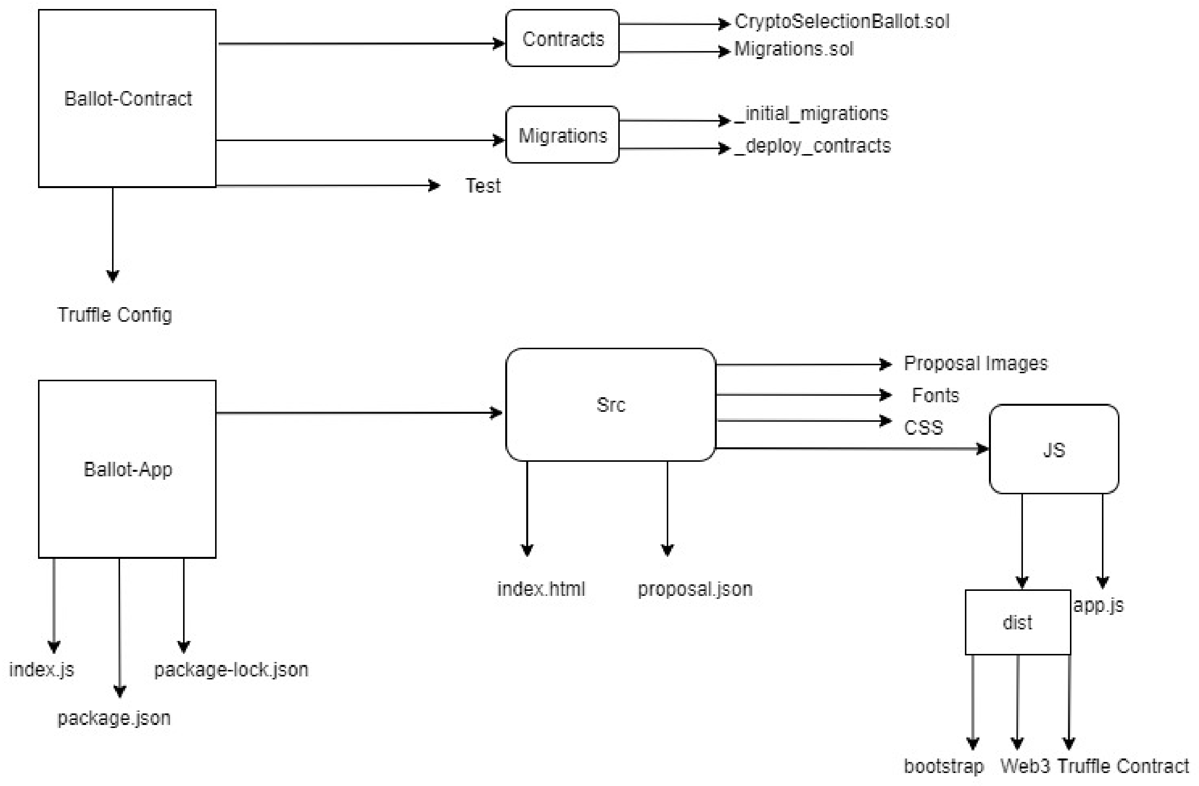


```
Crypto Selection Ballot

  struct voter{}
 address chairperson;
 address voter_address;
 Proposal[] proposals;
 mapping(address => Voter) voters;
 enum Phase {Init,Regs, Vote, Done}
 Phase public state = Phase.Done;

modifier validPhase(Phase reqPhase) {
         require(state == reqPhase);
                  _; }
modifier checkAge(){
       require(voters[voter_address].age >=19);
             _; }

modifier onlyChair(){
          require(msg.sender==chairperson);
             _; }

 modifier onlyMemberOfCoindeal(){
              require(voters[voter_address].isMember==true);
                   _; }

 modifier checkMaxVote(){
          require(voters[voter_address].maxVote<=true);
               _; }
       modifier checkRegister(){
               require(voters[voter_address].isRegistered==true);
                     _; }

            constructor (uint8 numProposals) public {    }

            enterAddress(address voter) public { }

             enterAge(uint8 Age) public   {
                          }
           enterMembershipDetails(bool _isMember) public  { }

            enterVoteDetails(bool _isVoted) public  { }

            increaseNumberVote(bool _isVoted) private  { }

            changeState(Phase X) public  { }

       register(address voter)public validPhase(Phase.Regs) checkAge()
            onlyMemberOfCoinDeal() checkIsVoted() onlyChair   { }

    vote(uint8 _toProposals) public validPhase(Phase.Vote) checkRegister() checkIsVoted()
                               checkMaxVote()  { }

           reqWinner() public  { }
```

Sequence Diagram:



Architecture Diagram:

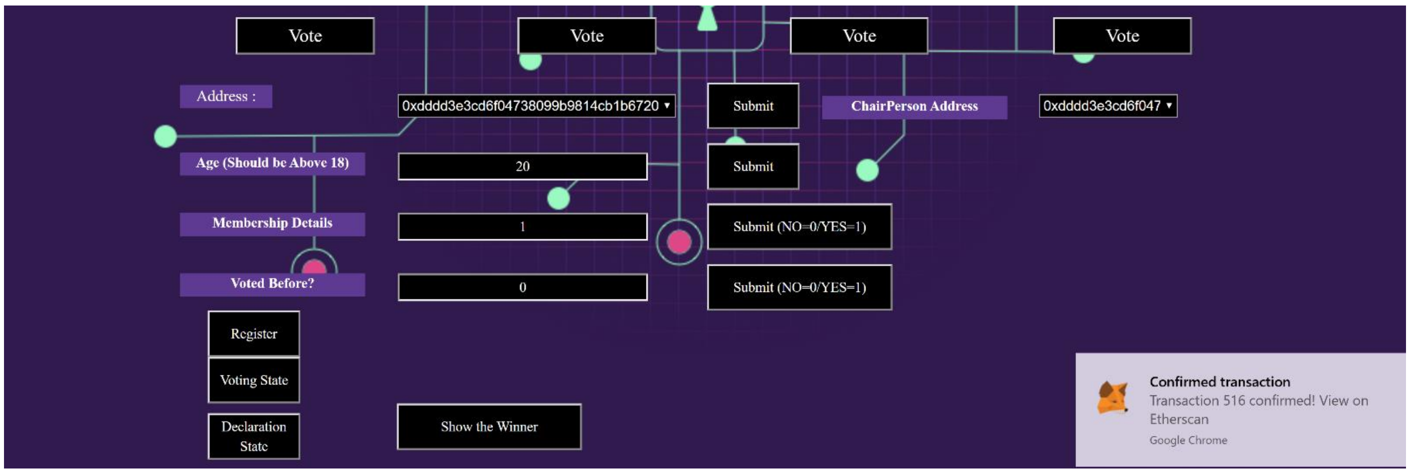


Work-Flow Instructions:

- **Compile the Contracts:**
  - Cd to ballot contract Folder
  - Compile the contracts using **truffle migrate –reset** command
- **Open Ganache and Set Up Metamask**
- **Compiling Front-End**
  - Cd to ballot-up
  - Use the command NPM Install
  - Use the command NPM Run
- **Go to** http://localhost:3000/ to open the web page
- **Now How to Use the Contract**

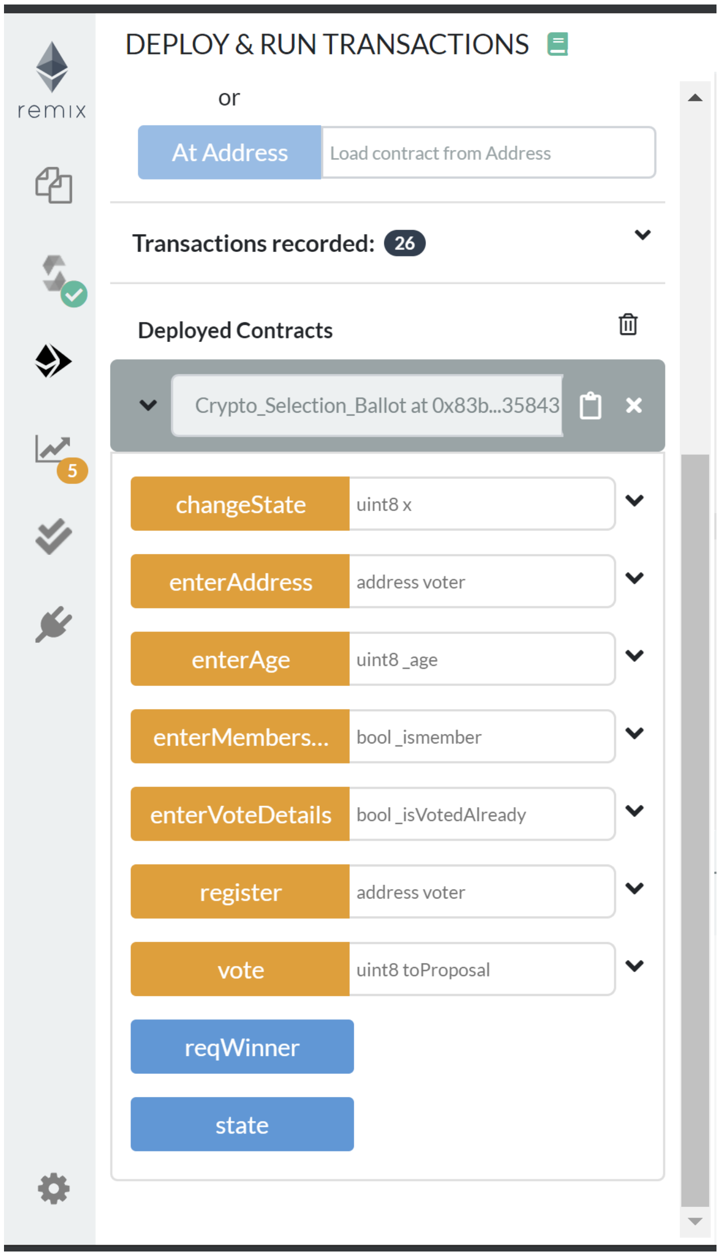
a. First enter the address of the voter to verify the correct address
b. Enter the age (Should be above 18 otherwise contract will throw an error)
c. Enter Membership Details (Should be member of coindeal.com otherwise contract will throw an error) (Input should be 0 or 1)
d. Enter if you have Voted Before (A member can Vote Maximum 4 times)
e. Register Yourself as a voter
f. Change the State to Voting State
g. Vote for a Particular Cryptocurrency
h. Remember one voter can vote maximum 4 times
i. Once you are done change the state to declaration
j. Click the "Show the winner button" to see the result.

THIS IS HOW THE BUTTONS LOOK IN THE UI. PLEASE FOLLOW SEQUENTIALLY AS STATED ABOVE



Confirmed transaction
Transaction 516 confirmed! View on Etherscan
Google Chrome

Remix ScreenShot



DEPLOY & RUN TRANSACTIONS

or

At Address | Load contract from Address

Transactions recorded: 26

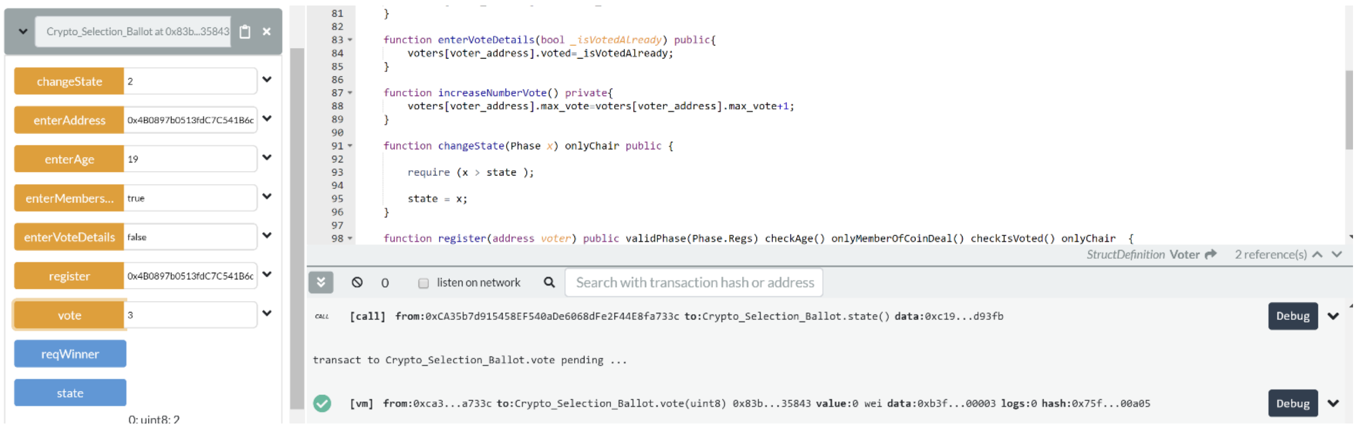
Deployed Contracts

Crypto_Selection_Ballot at 0x83b...35843

changeState	uint8 x
enterAddress	address voter
enterAge	uint8 _age
enterMembers...	bool _ismember
enterVoteDetails	bool _isVotedAlready
register	address voter
vote	uint8 toProposal
reqWinner	
state	

Positive Testcase :

While all the modifiers are correct it will execute as a positive result

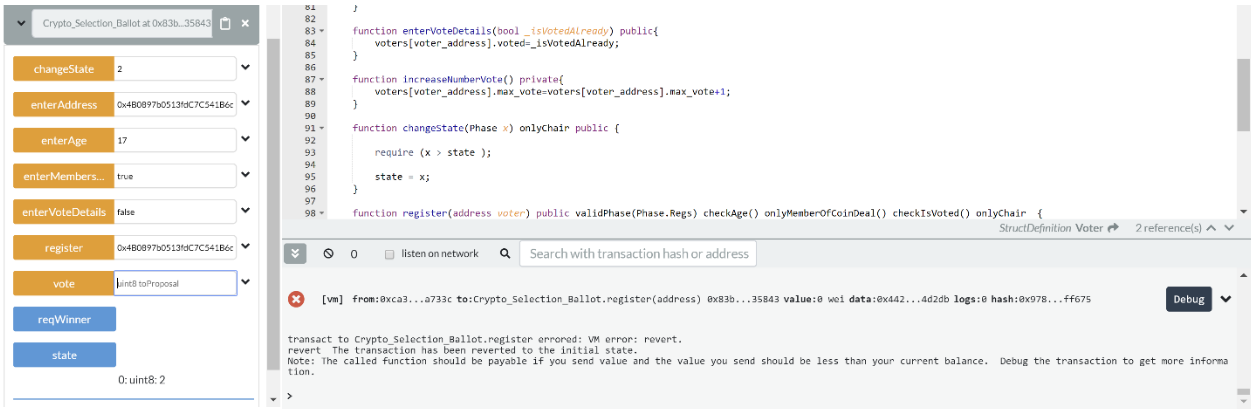


```
}

function enterVoteDetails(bool _isVotedAlready) public{
    voters[voter_address].voted=_isVotedAlready;
}

function increaseNumberVote() private{
    voters[voter_address].max_vote=voters[voter_address].max_vote+1;
}

function changeState(Phase x) onlyChair public {

    require (x > state );

    state = x;
}

function register(address voter) public validPhase(Phase.Regs) checkAge() onlyMemberOfCoinDeal() checkIsVoted() onlyChair  {
```

```
[call] from:0xCA35b7d915458EF540aDe6068dFe2F44E8fa733c to:Crypto_Selection_Ballot.state() data:0xc19...d93fb

transact to Crypto_Selection_Ballot.vote pending ...

[vm] from:0xca3...a733c to:Crypto_Selection_Ballot.vote(uint8) 0x83b...35843 value:0 wei data:0xb3f...00003 logs:0 hash:0x75f...00a05
```

Negative Testcase:

While one of the modifiers does not satisfy the condition . For example age needs to be 18 or above to be eligible for voting . If I put age as 17 it will fail



```
}

function enterVoteDetails(bool _isVotedAlready) public{
    voters[voter_address].voted=_isVotedAlready;
}

function increaseNumberVote() private{
    voters[voter_address].max_vote=voters[voter_address].max_vote+1;
}

function changeState(Phase x) onlyChair public {

    require (x > state );

    state = x;
}

function register(address voter) public validPhase(Phase.Regs) checkAge() onlyMemberOfCoinDeal() checkIsVoted() onlyChair  {
```

```
[vm] from:0xca3...a733c to:Crypto_Selection_Ballot.register(address) 0x83b...35843 value:0 wei data:0x442...4d2db logs:0 hash:0x978...ff675

transact to Crypto_Selection_Ballot.register errored: VM error: revert.
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 informa
tion.
>
```

Solidity Code :

pragma solidity ^0.5.2;

contract CryptoSelectionBallot {

struct Voter {

uint weight;

uint8 age;

uint8 isMember;

uint8 voted;

uint8 vote;

```
    bool isRegistered;
    uint max_vote;
}

struct Proposal {
    uint voteCount;
}

address chairperson;
address voter_address;
Proposal[] proposals;
mapping(address => Voter) voters;
uint8 state=0;

//modifiers
modifier validPhase(uint8 reqPhase)
{
    require(state == reqPhase);
    _;
}

modifier checkAge()
{
    require(voters[voter_address].age >=19);
    _;
}
modifier onlyChair()
 {
     require(msg.sender == chairperson);
     _;
 }

 modifier onlyMemberOfCoinDeal()
 {
     require(voters[voter_address].isMember==1);
     _;
 }

 modifier checkIsVoted()
 {
     require(voters[voter_address].voted==0);
     _;
 }

 modifier checkMaxVote()
 {
     require(voters[voter_address].max_vote<=4);
     _;
 }

modifier checkRegister()
{
    require(voters[voter_address].isRegistered==true);
     _;
}
constructor (uint8 numProposals) public  {
     chairperson = msg.sender;
     proposals.length = numProposals;
     voters[chairperson].weight = 4;
     state = 1;
```

```
}


function enterAddress(address voter) public{
    voter_address=voter;
}
function enterAge(uint8 _age) public {
    voters[voter_address].age=_age;
}


function enterMembershipDetails(uint8 _ismember) public{


    voters[voter_address].isMember=_ismember;


}


function enterVoteDetails(uint8 _isVotedAlready) public{
    voters[voter_address].voted=_isVotedAlready;
}


function registerAddress(address voter) checkAge onlyMemberOfCoinDeal checkAge public{
    voters[voter].isRegistered=true;
    voters[voter].max_vote=0;
}


function increaseNumberVote() private{
    voters[voter_address].max_vote=voters[voter_address].max_vote+1;
}


function change(uint8 x)  public {
        state = x;
    }

    function vote(uint8 toProposal)  public checkRegister validPhase(2) checkMaxVote{

        Voter memory sender = voters[voter_address];

        require (toProposal < proposals.length);
        enterVoteDetails(1);
        increaseNumberVote();
        voters[voter_address].vote = toProposal;
        proposals[toProposal].voteCount += 1;
    }

    function reqWinner() public validPhase(3) view returns (uint8 winningProposal) {

        uint256 winningVoteCount = 0;
        for (uint8 prop = 0; prop < proposals.length; prop++)
            if (proposals[prop].voteCount > winningVoteCount) {
                winningVoteCount = proposals[prop].voteCount;
                winningProposal = prop;
            }
       assert(winningVoteCount>=1);
    }

}
    function register(address voter) public validPhase(Phase.Regs) checkAge()
onlyMemberOfCoinDeal() checkIsVoted() onlyChair  {

        voters[voter].isRegistered=true;
        voters[voter].max_vote=0;

        voters[voter].weight = 1;

    }


    function vote(uint8 toProposal) public validPhase(Phase.Vote) checkRegister() checkIsVoted()
checkMaxVote() {


        Voter memory sender = voters[msg.sender];


        require (toProposal < proposals.length);

        enterVoteDetails(true);

        increaseNumberVote();

        sender.vote = toProposal;

        proposals[toProposal].voteCount += sender.weight;

    }


    function reqWinner() public validPhase(Phase.Done) view returns (uint8 winningProposal) {


        uint256 winningVoteCount = 0;

        for (uint8 prop = 0; prop < proposals.length; prop++)

            if (proposals[prop].voteCount > winningVoteCount) {

                winningVoteCount = proposals[prop].voteCount;

                winningProposal = prop;

            }

       assert(winningVoteCount>=1);

    }


}
```