

Name: Darshan Bele

Roll No: 14110

Class: BE – A – A1

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.20;

/**

* @title E-Voting System

* @dev A simple smart contract for conducting an election.

*/

contract Voting {

// Structure to represent a candidate struct

Candidate {

uint id;

string name;

uint voteCount;

}

// Structure to represent a voter

struct Voter { bool

isRegistered; bool hasVoted;

uint votedFor; // The ID of the candidate they voted for

}

// Mapping to store candidates by their ID

mapping(uint => Candidate) public candidates;

// Mapping to store voters by their address

mapping(address => Voter) public voters;

// Counter to keep track of the total number of candidates uint

public candidatesCount;

/**

* @dev Constructor to initialize the election with a list of candidate names. * The person who
 deploys the contract is the administrator.

*/

constructor(string[] memory _candidateNames) {

for (uint i = 0; i < _candidateNames.length; i++) {

addCandidate(_candidateNames[i]);

}

}

/**

* @dev Private function to add a new candidate.

*/

function addCandidate(string memory _name) private {

candidatesCount++;

candidates[candidatesCount] = Candidate(candidatesCount, _name, 0);

```

    }

    /**
    * @dev Allows a voter to cast their vote.
    * Requires the voter has not already voted.
    */
    function vote(uint _candidateId) public {
// Check if the voter has already voted
        require(!voters[msg.sender].hasVoted, "You have already voted.");

        // Check if the candidate ID is valid
        require(_candidateId > 0 && _candidateId <= candidatesCount, "Invalid candidate ID.");

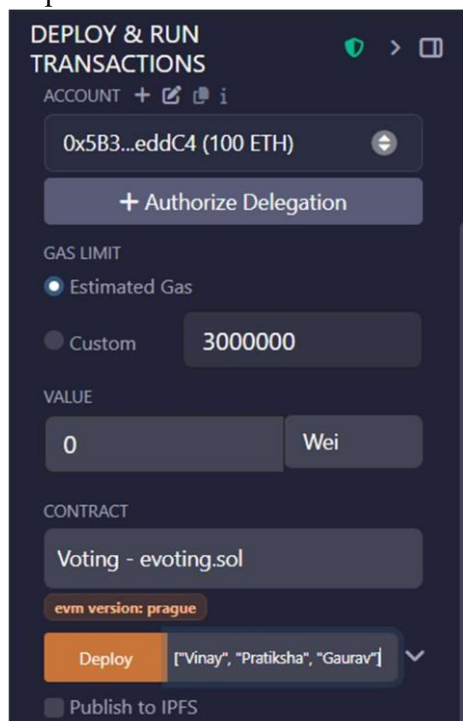
        // Mark the voter as having voted
        voters[msg.sender].hasVoted = true;
        voters[msg.sender].votedFor = _candidateId;

        // Increment the candidate's vote count
        candidates[_candidateId].voteCount++;
    }

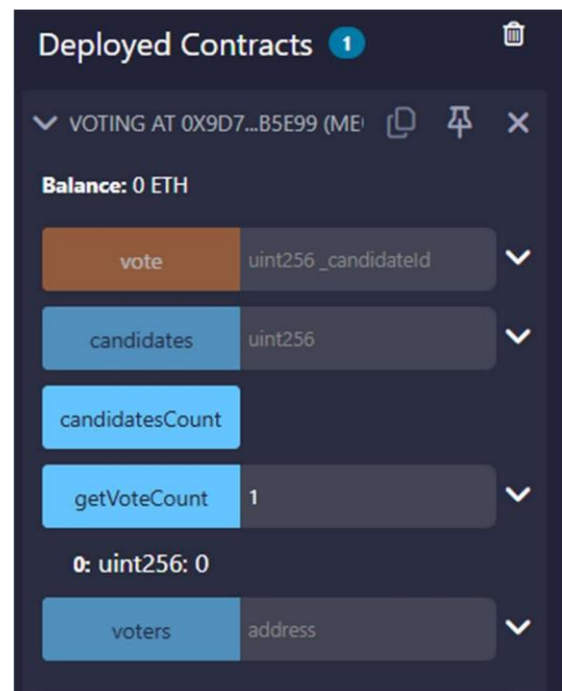
    /**
    * @dev Retrieves the total vote count for a specific candidate.
    */
    function getVoteCount(uint _candidateId) public view returns (uint) {
        require(_candidateId > 0 && _candidateId <= candidatesCount, "Invalid candidate ID.");
        return candidates[_candidateId].voteCount;
    }
}

```

Output:



Initial Candidate



Initial Votes

```
[vm] from: 0x5B3...eddC4 to: Voting.(constructor) value: 0 wei data: 0x608...00000 logs: 0 hash: 0x348...6dbe3

status                                0x1 Transaction mined and execution succeed

transaction hash                       0x348851892bb532c830e63bf32de67fc72491915efbd3ab6313d8f94296dbe3 ⓘ

block hash                            0x204dcc8318186e010abdc11ae15a5ae5195b03ff296a9f9397925610a290fc42 ⓘ

block number                          10 ⓘ

contract address                      0x9D7f74d0C41E726EC95884E0e97Fa6129e3b5E99 ⓘ

from                                  0x5B38Da6a701c568545dCfc803Fc875f56beddC4 ⓘ

to                                    Voting.(constructor) ⓘ

gas                                    796716 gas ⓘ

transaction cost                       692796 gas ⓘ

execution cost                         580178 gas ⓘ

input                                  0x608...00000 ⓘ

output                                0x608060405234801561000f575f5ffdb506004361061005557f3560e01c80630121b93f146100595780632d35d
c138d146100c5578063b2c2f2e8146100f7575b5f5ffdb50610073600480360381019061006e919061045d565b6101
```

Added Candidates Transaction 1.1

```
[vm] from: 0x5B3...eddC4 to: Voting.(constructor) value: 0 wei data: 0x608...00000 logs: 0 hash: 0x348...6dbe3
call to Voting.getVoteCount

CALL [call] from: 0x5B38Da6a701c568545dCfc803Fc875f56beddC4 to: Voting.getVoteCount(uint256) data: 0xb2c...00001

from                                0x5B38Da6a701c568545dCfc803Fc875f56beddC4 ⓘ

to                                  Voting.getVoteCount(uint256) 0x9D7f74d0C41E726EC95884E0e97Fa6129e3b5E99 ⓘ

execution cost                      4996 gas (Cost only applies when called by a contract) ⓘ

input                               0xb2c...00001 ⓘ

output                              0x0000000000000000000000000000000000000000000000000000000000000000 ⓘ

decoded input                       {
  "uint256 _candidateId": "1"
} ⓘ

decoded output                      {
  "0": "uint256: 0"
} ⓘ

logs                                [] ⓘ
```

Initial Votes Transaction 1.2

VOTING AT 0X9D7...B5E99 (ME)

Balance: 0 ETH

vote

1

candidates

1

0: uint256: id 1

1: string: name Darshan

2: uint256: voteCount 0

candidatesCount

0: uint256: 3

getVoteCount

3

2: uint256: 3

voters

address

First Vote

VOTING AT 0X9D7...B5E99 (ME)

Balance: 0 ETH

vote

vote - transact (not

candidates

1

0: uint256: id 1

First Vote Count

✓ VOTING AT 0X9D7...B5E99 (ME) [🔗] [🔧] [✕]

Balance: 0 ETH

vote 1 ▼

candidates 1 ▼

0: uint256: id 1
1: string: name Darshan
2: uint256: voteCount 0

candidatesCount

0: uint256: 3

getVoteCount 3 ▼

2: uint256: 3

voters address ▼

Final Vote Count (Winner ID-1 “Darshan”)

```
CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Voting.getVoteCount(uint256) data: 0xb2c...00001

from 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 [🔗]
to Voting.getVoteCount(uint256) 0x9D7f74d0C41E726EC95884E0e97Fa6129e3b5E99 [🔗]
execution cost 4996 gas (Cost only applies when called by a contract) [🔗]
input 0xb2c...00001 [🔗]
output 0x0000000000000000000000000000000000000000000000000000000000000001 [🔗]
decoded input {
  "uint256 _candidateId": "1"
} [🔗]
decoded output {
  "0": "uint256: 1"
} [🔗]
logs [] [🔗]
raw logs [] [🔗]
```

Winner Transaction 3.2

```
✗ [vm] from: 0x5B3...eddC4 to: Voting.vote(uint256) 0x9D7...b5E99 value: 0 wei data: 0x012...00001 logs: 0 hash: 0xc1b...2e93c
transact to Voting.vote errored: Error occurred: revert.

revert
The transaction has been reverted to the initial state.
Reason provided by the contract: "You have already voted.".
If the transaction failed for not having enough gas, try increasing the gas limit gently.
```

Already Voted Transaction 3.3