Palestine Technical University (Kadoorie)

Faculty of Engineering and Technology

Department of Computer Systems Engineering

Voting System Solidity

By:

Afnan Jaber 201910560

Hala Shreem 201911474

Mays Najjar 201811598

Supervised:

Dr. Mahmoud Sawalheh.

The source of code.

Explain of our project:

This smart contract about voting system. It will allow us to list the candidates that will run in the election, and keep track of all the votes and voters. It will also govern all of the rules of the election, like enforcing accounts to only vote once.

The Code:

1. We to store multiple candidates, and store multiple attributes about each candidate. We want to initialize candidate's id, name, and voteCount is the number of votes that the candidate has received.

```
contract Election {
    // Model a Candidate
    struct Candidate {
        uint id;
        string name;
        uint voteCount;
    }
```

2. Now defines a mapping called voters that is used to store the addresses of accounts that have voted in the election. The mapping maps an address to a boolean value that indicates whether the account has voted or not.

```
// Store accounts that have voted
    mapping(address => bool) public voters;
```

3. Now defines a mapping called candidates that is used to store the candidates in the election. The mapping maps an integer (the candidate's id) to a Candidate struct.

```
// Store Candidates
```

```
    // Fetch Candidate
    mapping(uint => Candidate) public candidates;
```

4. Next, defines variable called candidatesCount that stores the number of candidates in the election.

```
// Store Candidates Count
    uint public candidatesCount;
```

5. Next, defines an event called votedEvent that is triggered whenever a vote is cast in the election. The event contains a single indexed parameter, _candidateId, which is the id of the candidate that received the vote.

```
    // voted event
    event votedEvent (
        uint indexed _candidateId
    );
```

6. Then defines a constructor function that is called when the contract is deployed. The constructor function adds two candidates to the election by calling the addCandidate function with the names "Candidate 1" and "Candidate 2".

```
constructor () public {
        addCandidate("Mays");
        addCandidate("Afnan");
        addCandidate("Hala");

    }
```

7. Next, define function which is used to add a new candidate to the election. The function takes a single parameter, _name, which is the name of the candidate to be added. The function increments candidatesCount, assigns a new id to the candidate, and stores the candidate in the candidates mapping.

```
  function addCandidate (string memory _name) private {
        candidatesCount ++;
        candidates[candidatesCount] = Candidate(candidatesCount, _name,
0);
    }
```

8. Create vote function, which is used to cast a vote in the election. The function takes a single parameter, _candidateId, which is the id of the candidate that the voter wishes to support.

```
function vote (uint _candidateId) public {
       // require that they haven't voted before
       require(!voters[msg.sender]);
```

9. The function checks that the voter has not already voted and that the _candidateId is valid (i.e., it corresponds to a candidate in the election). If these checks pass, the function records the vote by setting the voter's address in the voters mapping to true and incrementing the voteCount of the candidate specified by _candidateId. Finally, the function triggers the votedEvent event.

```
       // require a valid candidate
       require(_candidateId > 0 && _candidateId <= candidatesCount);

       // record that voter has voted
       voters[msg.sender] = true;

       // update candidate vote Count
       candidates[_candidateId].voteCount ++;
       // trigger voted event
      emit votedEvent(_candidateId);
       }

   }
```

- Screens of code:

Home | Election.sol

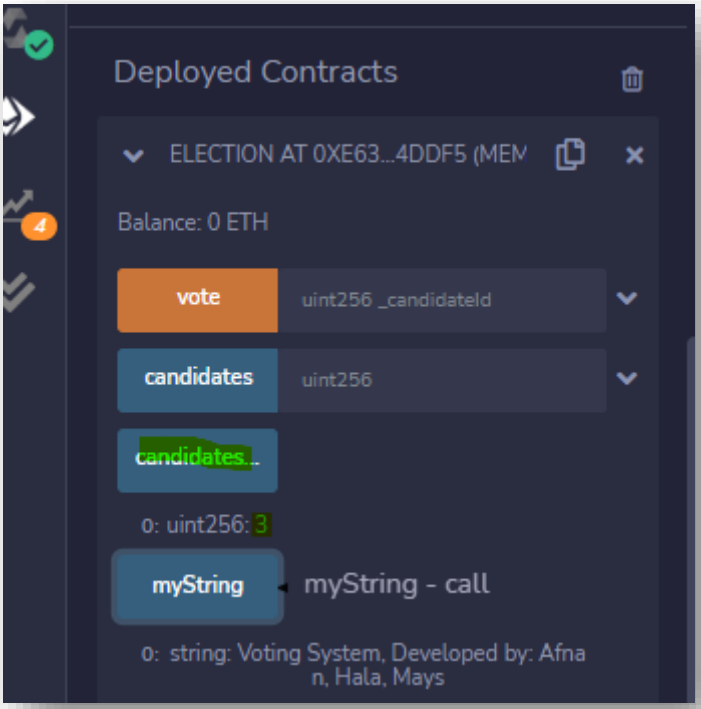
```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >= 0.5.5 ;
  // Developed by:
  // Afnan Jaber
  // Hala Shreem
  // Mays Najjar

contract Election {
    // the struct of our Candidate
    struct Candidate {
        uint id;
        string name;
        uint voteCount;
    }

     string public myString = "Voting System, Developed by: Afnan, Hala, Mays";
    // Store accounts that have voted
    mapping(address => bool) public voters;

    // Store Candidates
    // Fetch Candidate
    mapping(uint => Candidate) public candidates;

    // Store Candidates Count
    uint public candidatesCount;

    // voted event
    event votedEvent (
        uint indexed _candidateId
```

Home | Election.sol

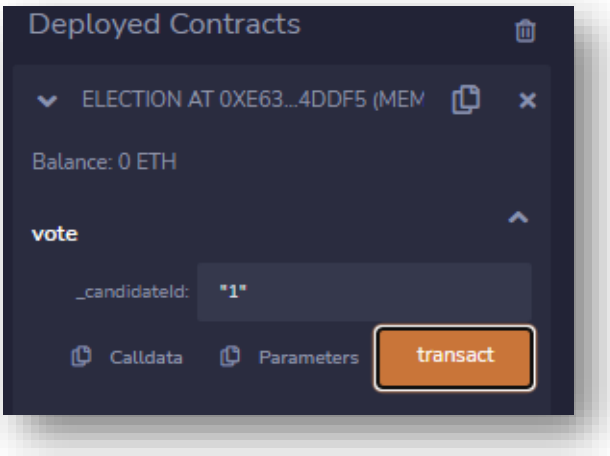
```
);

constructor () {
    addCandidate("Mays");
    addCandidate("Afnan");
    addCandidate("Hala");
}

function addCandidate (string memory _name) private {
    candidatesCount ++;
    candidates[candidatesCount] = Candidate(candidatesCount, _name, 0);
}

function vote (uint _candidateId) public {
    // require that they haven't voted before
    require(!voters[msg.sender]);

    // require a valid candidate
    require(_candidateId > 0 && _candidateId <= candidatesCount);

    // record that voter has voted
    voters[msg.sender] = true;

    // update candidate vote Count
    candidates[_candidateId].voteCount ++;

    // trigger voted event
    emit votedEvent(_candidateId);
```

Now Deploy the code:

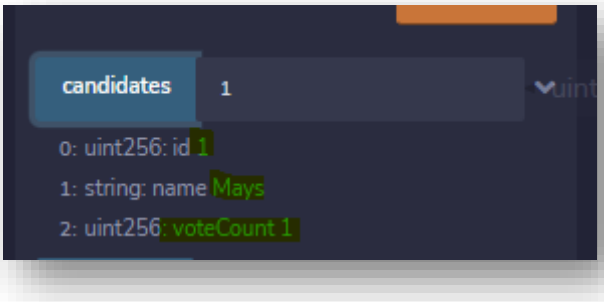
this is the number of candidates we have, according to the constructor that we implement. [3 candidates]



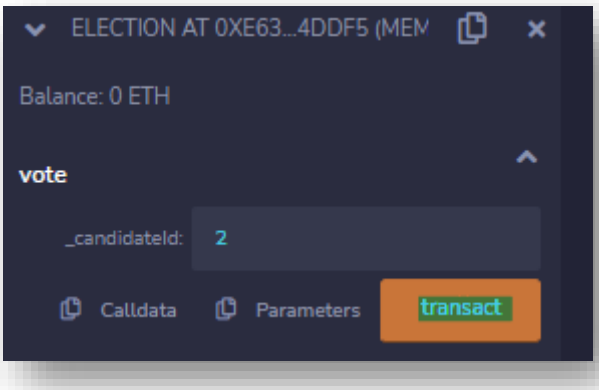
Now we want to vote to the first candidate [Mays], click the transact to vote to the first one.



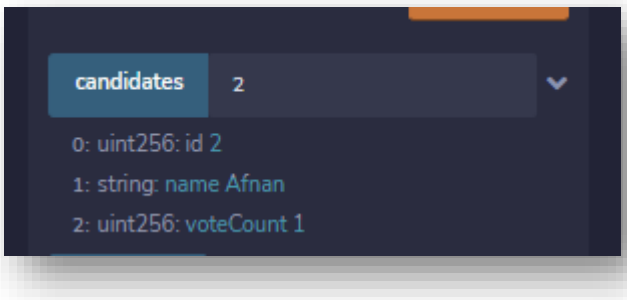
After we click, we want to see the update of number of votCount



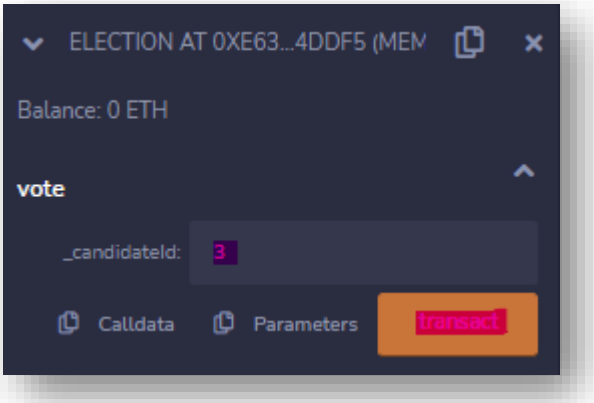
Now we want to vote to the second candidate [Afnan], click the transact to vote to the second one.



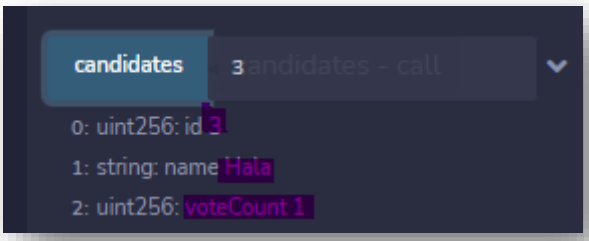
Now Show the update



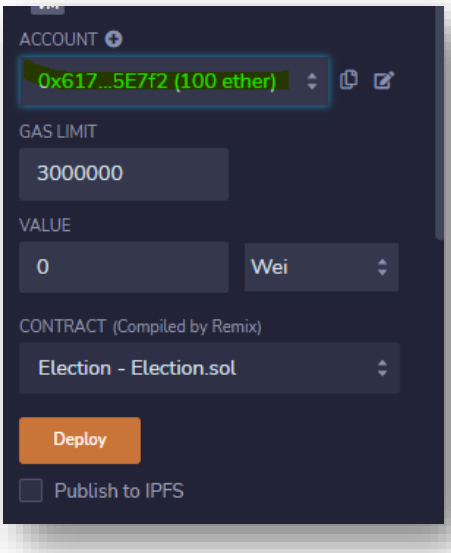
Now we want to vote to the third candidate [Hala], click the transact to vote to the third one.



Now Show the update



But in our system, we noticed that we need to change the ether after voting, like this



If we want to increase our candidates, we add them in the constructor.

Now we want to vote to the first candidate [Mays] another time, click the transact to vote to the first one.