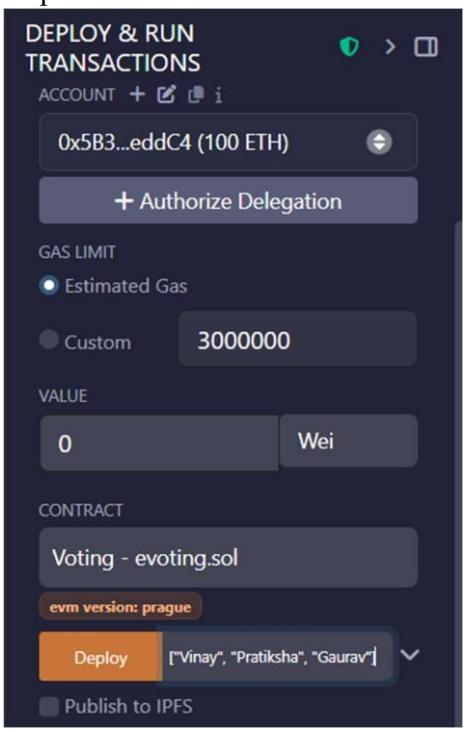
Name: Darshan Bele
Roll No: 14110
Class: BE – A – A1

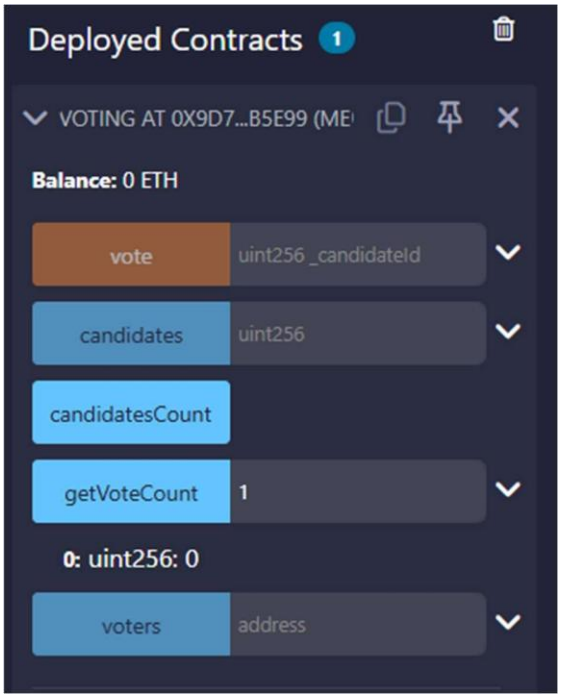
```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

/**
* @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;
    }
}
```

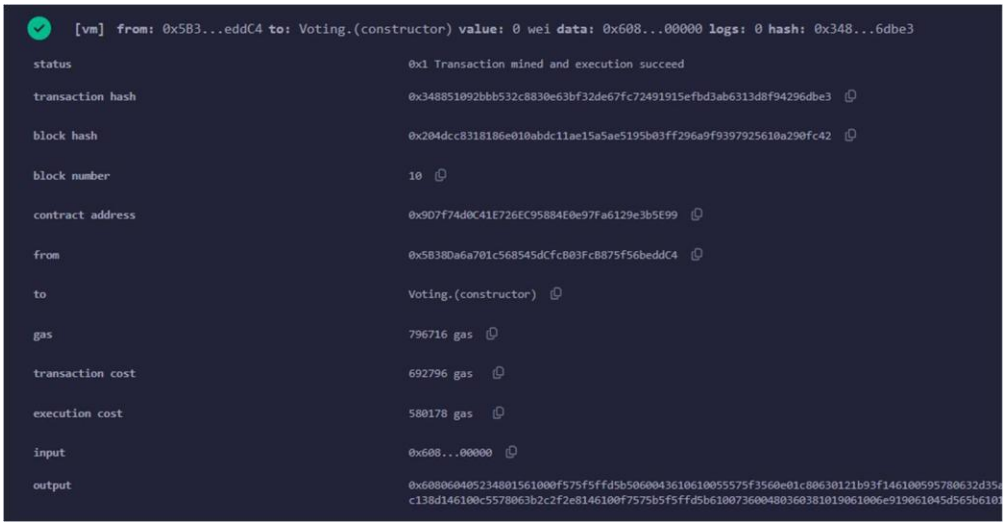
Output:



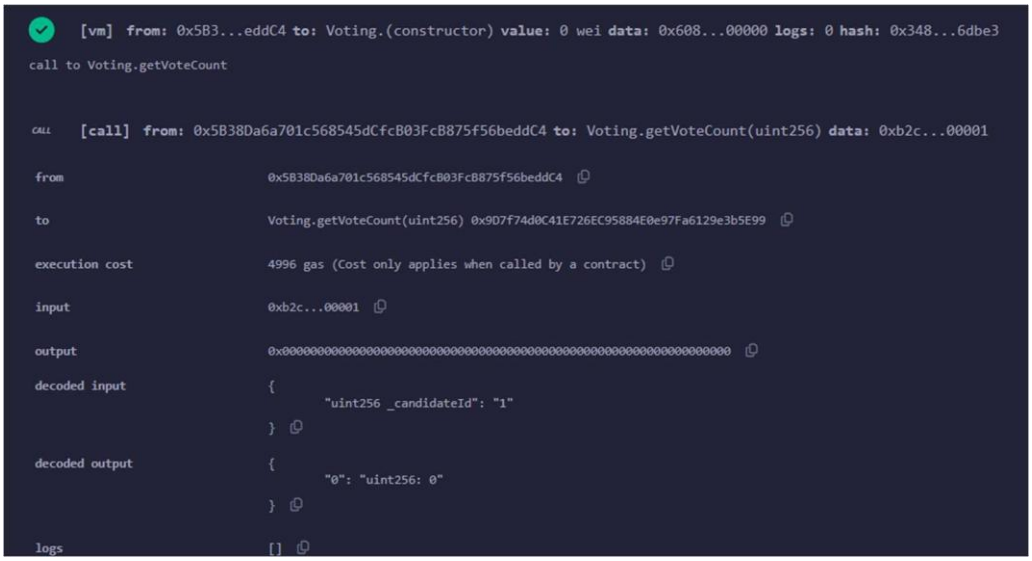
Initial Candidate



Initial Votes

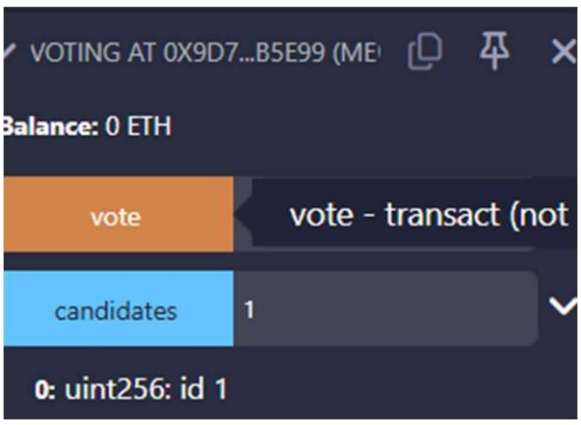


Added Candidates Transaction 1.1

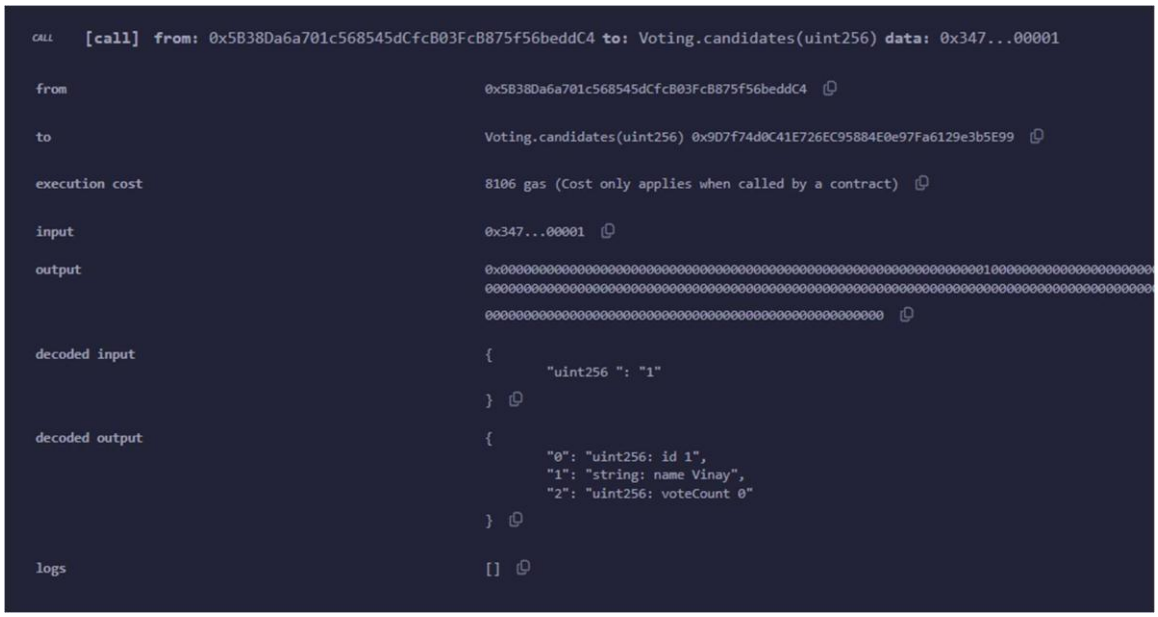


Initial Votes Transaction 1.2

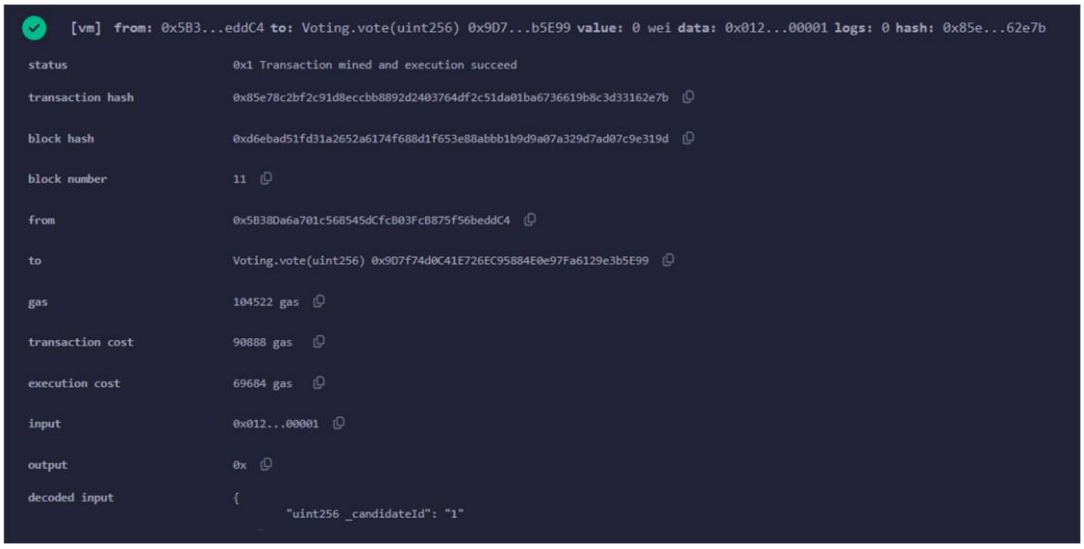
First Vote



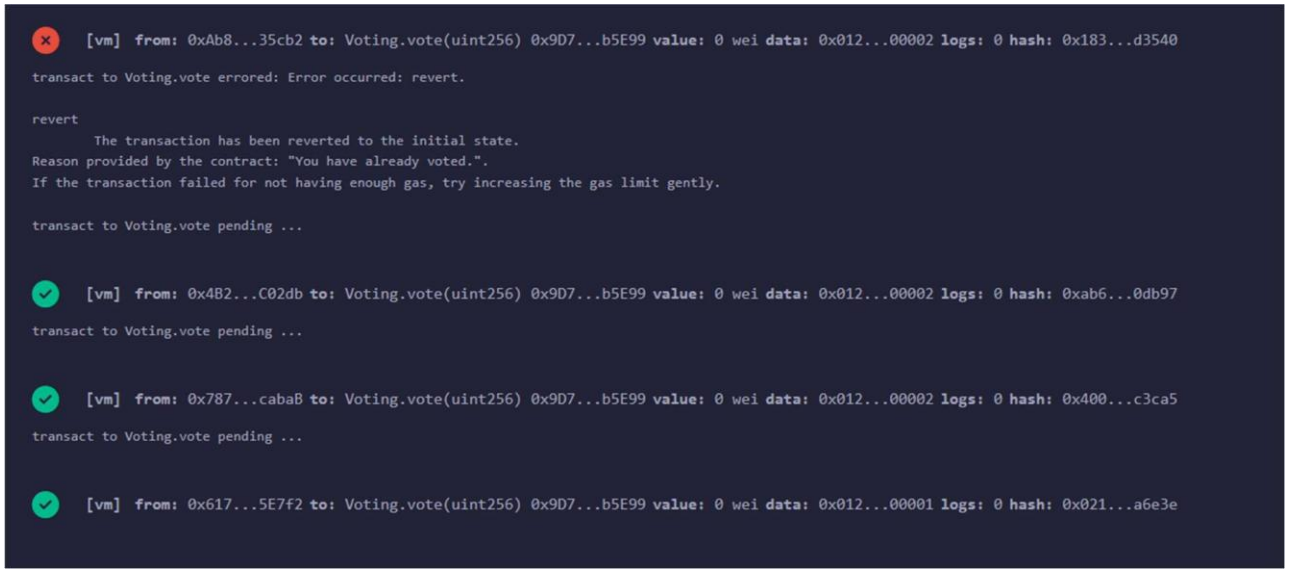
First Vote Count



First Vote Transaction 2.1



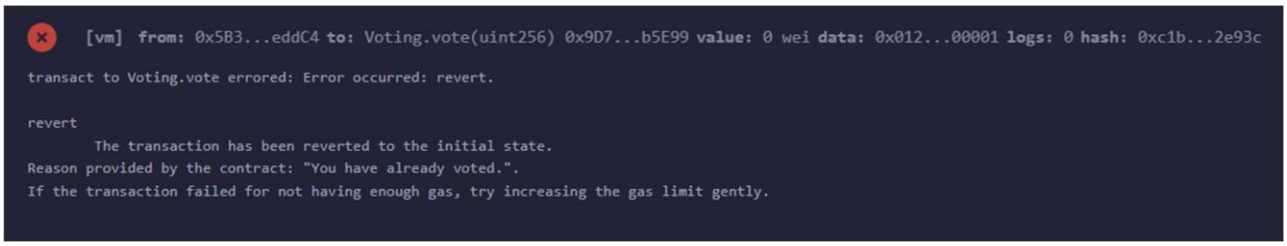
Valid Vote Transaction 2.2



Subsequent Votes Transaction 3.1

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

Winner Transaction 3.2



Already Voted Transaction 3.3