ELECTRONIC VOTING SYSTEM

ABSTRACT:

Building a secure electronic voting system that offers the fairness and privacy of current voting schemes, while providing the transparency and flexibility offered by electronic systems has been a challenge for a long time. In this work-in-progress paper, we evaluate an application of blockchain as a service to implement distributed electronic voting systems. The paper proposes a novel electronic voting system based on blockchain that addresses some of the limitations in existing systems and evaluates some of the popular blockchain frameworks for the purpose of constructing a blockchain-based e-voting system. In particular, we evaluate the potential of distributed ledger technologies through the description of a case study; namely, the process of an election, and the implementation of a blockchain-based application, which improves the security and decreases the cost of hosting a nationwide election.

INTRODUCTION:

In each democracy, the protection of AN election may be a matter of national security. the pc security field has for a decade studied the probabilities of electronic choice systems, with the goal of minimizing the price of getting a national election, whereas fulfilling ANd increasing the protection conditions of an election. From the dawn of democratically electing candidates, the legal system has been supported pen and paper. commutation the normal pen and paper theme with a replacement election system is essential to limit fraud and having the choice method traceable and verifiable. Electronic choice machines are viewed as blemished, by the protection community, based totally on physical security considerations. Anyone with physical access to such machine will sabotage the machine, thereby moving all votes run up the said machine. Enter blockchain technology.

SOURCE CODE:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract VoteSystem{
    address public owner;

    constructor(){
        owner= msg.sender;
    }

 struct candidate {
```

```
    uint voterId;
    string name;
    uint age;
    uint voteCount;
}

mapping (uint => candidate) candidateMap;

struct voters {
    uint voterId;
    string name;
    uint age;
    bool votingState;
}

mapping (uint => voters) votersMap;
mapping (uint=>bool) registeredVoter;

modifier checkVoterVoted(uint _votersVoterId){
    require (votersMap[_votersVoterId].votingState == false);
    _;
}

modifier checkRegisteredVoter(uint _votersVoterId){
       require(registeredVoter[_votersVoterId]==true, "Voter is not Registered");
       _;
}
uint[] voterIdlist;
uint[] candidateIdList;


function enrollCandidate(uint _voterId,string memory _name,uint  _age )  public {

require (_age >= 25);
require (candidateMap[_voterId].voterId != _voterId);

   candidateMap[_voterId].voterId = _voterId;
   candidateMap[_voterId].name = _name;
   candidateMap[_voterId].age = _age;

   candidateIdList.push(_voterId);
}
```

```
 function enrollVoter(uint _voterId,string memory _name,uint _age)  public
returns(bool){

require (_age >= 18);
require (votersMap[_voterId].voterId != _voterId);

     votersMap[_voterId].voterId = _voterId;
     votersMap[_voterId].name = _name;
     votersMap[_voterId].age = _age;

     voterIdlist.push(_voterId);
    return registeredVoter[_voterId]=true;

 }

 function getCandidateDetails(uint _voterId) view public returns(uint,string
memory,uint,uint) {

     return
(candidateMap[_voterId].voterId,candidateMap[_voterId].name,candidateMap[_voterI
d].age,candidateMap[_voterId].voteCount);
 }

 function getVoterDetails(uint _voterId) view public returns (uint,string
memory,uint,bool){

     return
(votersMap[_voterId].voterId,votersMap[_voterId].name,votersMap[_voterId].age,vot
ersMap[_voterId].votingState);

 }

 function vote(uint _candidateVoterId,uint _votersVoterId) public
checkVoterVoted(_votersVoterId) checkRegisteredVoter(_votersVoterId) {
     candidateMap[_candidateVoterId].voteCount += 1;
     votersMap[_votersVoterId].votingState = true;
 }

 function getVotecountOf(uint _voterId) view public returns(uint){
        require(msg.sender== owner, "Only owner is allowed to Check Results");
     return candidateMap[_voterId].voteCount;
 }

 function getVoterList() view public returns (uint[] memory){
```

```
    return   voterIdlist;
    }

 function getCandidateList() view public returns(uint[] memory){


 return candidateIdList;
 }


}
```