AISSMS
INSTITUTE OF INFORMATION TECHNOLOGY
(IOIT)
ADDING VALUE TO ENGINEERING
An Autonomous Institute Affiliated to Savitribai Phule Pune University
Approved by AICTE, New Delhi and Recognised by Govt. of Maharashtra
Accredited by NAAC with "A+" Grade | NBA - 5 UG Programmes

2022-2023

Department of Computer Engineering
MINI PROJECT REPORT ON

"Blockchain based application dApp (decentralised app) for e voting system"

Submitted By

76	Atharva Mohan Tirkhunde	72145918J

Guided by
Ms. Shilpa Pimpalkar

Project Aim

To Develop a Blockchain based application dApp (decentralised app) for e-voting system

Problem Statement

Develop a Blockchain based application dApp (decentralised app) for e-voting system

Objectives:

- Develop a secure and transparent e-voting system using blockchain technology.
- Enable voters to cast their ballots from anywhere in the world with an internet connection.
- Reduce the risk of voter fraud and ensure the integrity of the voting process.
- Increase voter turnout and make democracy more accessible.

Project Scope:

The project scope will include the development of the following components:

- A blockchain-based e-voting platform that allows voters to cast their ballots securely and anonymously.
- A web application that allows voters to register for elections, view their ballots, and cast their votes.
- A mobile application that allows voters to cast their votes from their smartphones or tablets.
- A smart contract that will be deployed on the blockchain to manage the voting process and ensure the integrity of the results.

System Requirements:

- The blockchain platform must be secure, scalable, and tamper-proof.
- The web and mobile applications must be user-friendly and easy to navigate.
- The smart contract must be carefully designed to ensure the integrity of the voting process.
- The system must be able to handle a large number of concurrent users.
- The system must be able to generate accurate and timely voting results.
- The system must be accessible to voters with disabilities.

Theory

Blockchain

Blockchain is a revolutionary and decentralised ledger technology that has disrupted traditional data management systems by offering an immutable, transparent, and secure platform for recording transactions. It operates as a chain of blocks, each containing a set of data and a cryptographic link to the previous block, creating a chronological and tamper-proof record.

DApp

A Decentralised Application (DApp) is a groundbreaking software application that leverages blockchain technology to function as a decentralised, trustless, and transparent platform. DApps are designed to operate autonomously, without the need for intermediaries or central authorities, ensuring data integrity and user control. DApps have diverse use cases, from financial services like decentralised finance (DeFi) and non-fungible tokens (NFTs) to supply chain management, gaming, and voting systems. They embody the principles of blockchain, offering users a secure and open ecosystem where transactions are publicly verifiable, resistant to censorship, and free from a single point of control, making them a pivotal innovation in the digital age.

Smart Contracts

Smart Contracts represent a transformative concept within the realm of blockchain technology, enabling self-executing and tamper-proof agreements between parties without the need for intermediaries. These contracts are essentially lines of code that automate, enforce, and facilitate the terms of a predefined agreement. Smart contracts are designed to trigger actions or transfer assets when specific conditions are met, revolutionising the way agreements are made and executed, ultimately reducing the need for intermediaries and enhancing efficiency, security, and accountability in various business processes.

Features of Blockchain

Several features of blockchain technology address critical requirements for secure, efficient, and transparent electronic voting systems. Here are some key features:

1. Immutability: The immutability of blockchain records ensures that once votes are recorded, they cannot be altered or deleted. This feature guarantees the integrity of the voting process, making it highly resistant to fraud and manipulation.
2. Transparency: Blockchain's transparent ledger allows all voters to independently verify the authenticity of their votes and the overall election process. This transparency fosters trust and confidence in the electoral system.
3. Decentralisation: The decentralised nature of blockchain reduces the risk of single points of failure and vulnerabilities associated with centralised systems. It also minimises the potential for external interference or manipulation in the voting process.
4. Data Integrity: Blockchain ensures that data, including voter registrations and ballots, is stored securely and cannot be altered without consensus from the network participants. This guarantees data integrity and maintains the sanctity of the vote.
5. Smart Contracts: The use of smart contracts automates the entire voting process, from voter registration to result verification, reducing the need for intermediaries and enhancing efficiency. They enforce the rules and conditions of the election, preventing unauthorised changes.
6. Tamper Resistance: Once a vote is recorded on the blockchain, it becomes practically impossible to alter or delete it without the consensus of the network. This tamper resistance is crucial for ensuring the accuracy and fairness of the election.
7. Accessibility: Blockchain technology allows for remote voting, enabling citizens to participate in the electoral process from various locations and devices. This feature enhances accessibility and inclusivity in the voting system.
8. Auditability: The transparent and auditable nature of blockchain facilitates post-election audits, allowing authorities and independent auditors to verify the results and the integrity of the process.
9. Trust and Verification: Blockchain's features instil trust in the electoral process, as voters can independently verify that their votes have been correctly recorded and counted, reducing scepticism and disputes.

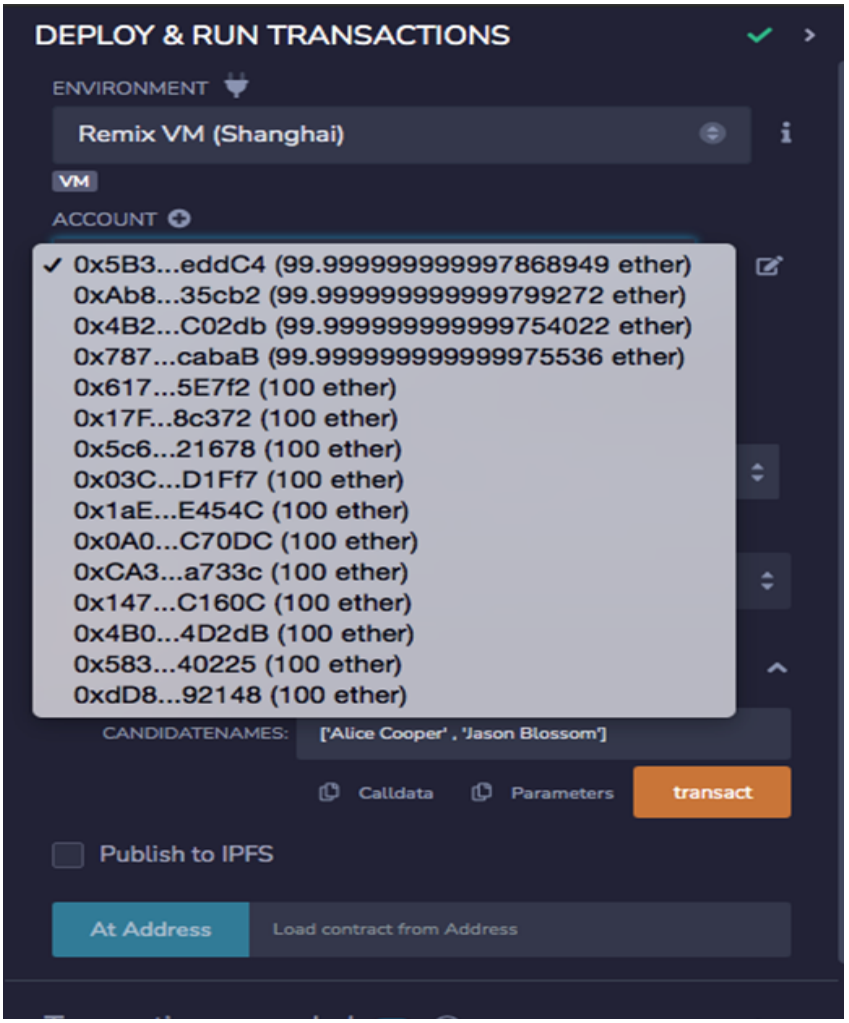
Code:

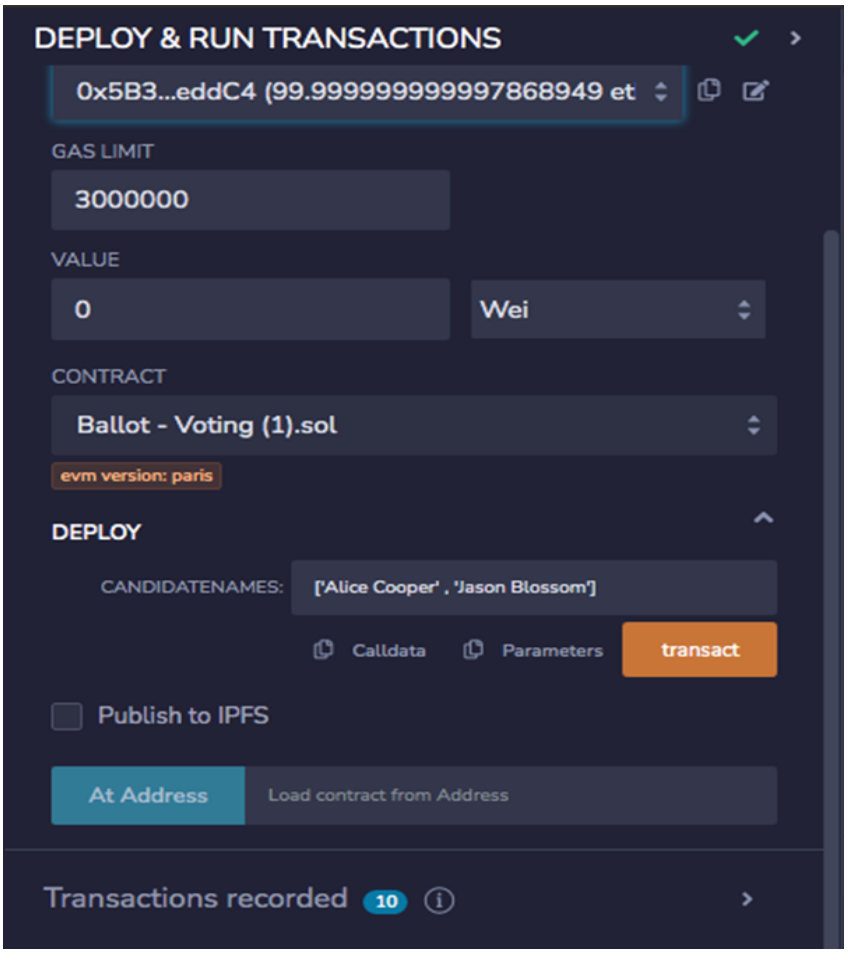
```
//SPDX-License-Identifier:MIT
pragma solidity ^0.8.13;

contract Vote{
    address electionComision;
    address public winner;

    struct Voter{
        string name;
        uint age;
        uint voterId;
        string gender;
        uint voteCandidateId;
        address voterAddress;
    }
    struct Candidate{
        string name;
        string party;
        uint age;
        string gender;
        uint candidateId;
        address candidateAddress;
        uint votes;
    }
    uint nextVoterId=1;// starting voterid with 1
    uint nextCandidateId=1;// starting candidateid with 1
    uint startTime;//startime of election
    uint endTime;//end time of election
    mapping(uint=>Voter) voterDetails;//details of voters
    mapping(uint=>Candidate) candidateDetails;///details of candidates
    bool stopVoting;//This  is used to stop the voting process in emergency situation
    constructor(){
            electionComision=msg.sender;//assigning  the  deployer  of  contract  as  election
commision
    }
    modifier isVotingOver(){
        require(block.timestamp >endTime ||stopVoting ==true,"voting is not over");
        _;
    }
    modifier onlyCommisioner(){
        require(electionComision== msg.sender,"You are not election commisioner");
        _;
    }
    function candidateRegister(//registration for the candidate
        string calldata _name,
        string calldata _party,
        uint _age,
        string calldata _gender
    )external {
```

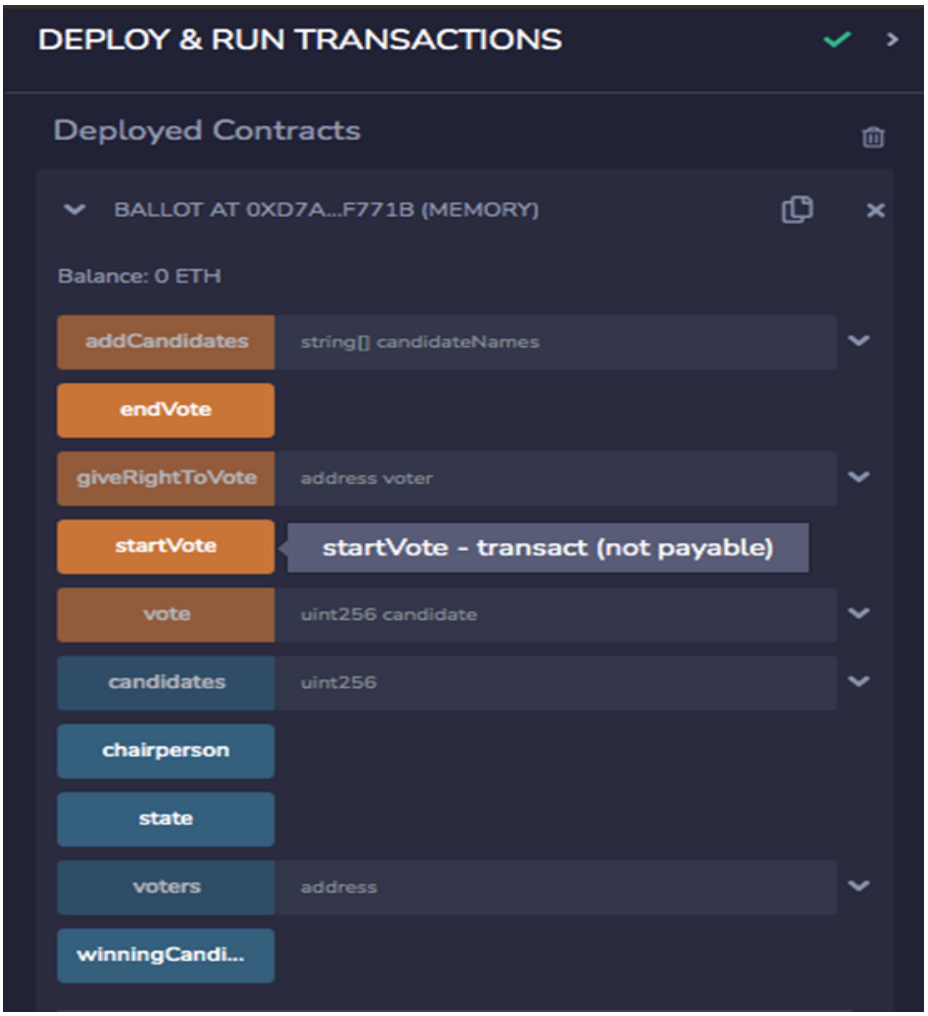
```
        require(msg.sender!=electionComision,"Election Comision not eligible for candidate registration");
        require(candidateVerification(msg.sender),"Candidate Already Registered");
        require(_age>=18,"You are not eligible");
        require(nextCandidateId<3,"Candidate registration");
                                                candidateDetails[nextCandidateId]=Candidate(_name, _party,_age,_gender,nextCandidateId,msg.sender,0);
        nextCandidateId++;
    }
    function candidateVerification(address _person) internal view returns(bool){
        for(uint i=1; i<=nextCandidateId; i++){
            if(candidateDetails[i].candidateAddress==_person){
                return false;
            }
        }
        return true;
    }
    function candidateList() public view returns(Candidate[] memory){
        Candidate[] memory array=new Candidate[](nextCandidateId-1);
        for(uint i=1;i<nextCandidateId;i++){
            array[i-1]=candidateDetails[i];
        }
        return array;
    }
    function voterRegister(string calldata _name, uint _age, string calldata _gender)external {
        require(voterVerification(msg.sender),"voter Already Registered");
        require(_age>=18,"You are not eligible");
        voterDetails[nextVoterId]=Voter(_name,_age,nextVoterId,_gender,0,msg.sender);
        nextVoterId++;
    }
    function voterVerification(address _person) internal view returns (bool){
        for(uint i=1;i<nextVoterId;i++){
            if(voterDetails[i].voterAddress== _person){
                return false;
            }
        }
        return true;
    }
    function voterList() public view returns (Voter[] memory){
        Voter[] memory array= new Voter[](nextVoterId-1);
        for(uint i=1;i<nextVoterId;i++){
            array[i-1]=voterDetails[i];
        }
        return array;
    }
    function vote(uint _voterId, uint _id)external {
        require(voterDetails[_voterId].voteCandidateId==0,"Voting already done");
        require(voterDetails[_voterId].voterAddress==msg.sender,"You are not registered");
        require(startTime!=0,"voting has not started");
        require(nextCandidateId==3,"Candidate has not registered");
        voterDetails[_voterId].voteCandidateId=_id;
```

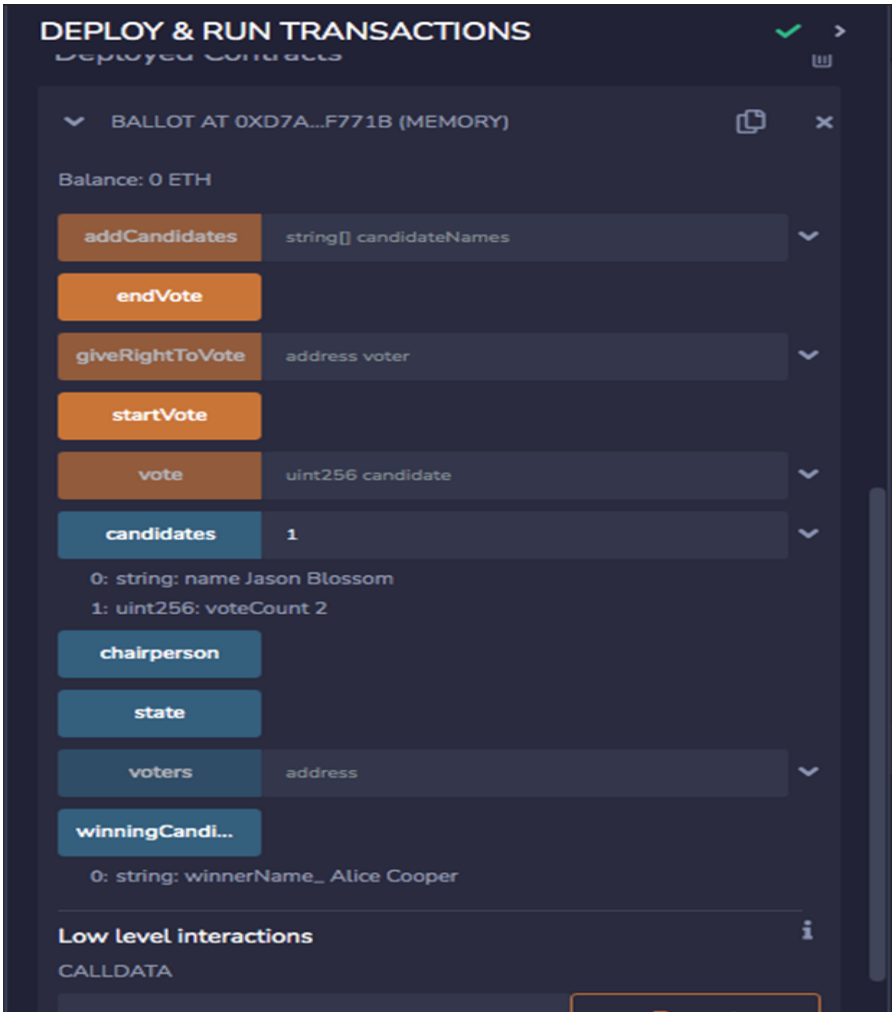
```
        candidateDetails[_id].votes++;
    }
    function voteTime(uint _startTime,uint _endTime)external onlyCommisioner(){
        startTime=_startTime+ block.timestamp;
        endTime=startTime+_endTime;
    }
    function votingStatus() public view returns(string memory){
        if(startTime==0){
            return "Voting has not started";
        }
        else if((endTime>block.timestamp) && stopVoting==false){
            return "Voting is in progress";
        }
        else{
            return "Voting Ended";
        }
    }
    function result() external onlyCommisioner(){
        uint max;
        for (uint i=1;i<nextCandidateId;i++){
            if(candidateDetails[i].votes>max){
                max=candidateDetails[i].votes;
                winner=candidateDetails[i].candidateAddress;
            }
        }
    }
    function emergency() public onlyCommisioner(){
        stopVoting =true;
    }
}
```

Result:


DEPLOY & RUN TRANSACTIONS
ENVIRONMENT
Remix VM (Shanghai)
VM
ACCOUNT
0x5B3...eddC4 (99.999999999997868949 ether)
0xAb8...35cb2 (99.99999999999799272 ether)
0x4B2...C02db (99.99999999999754022 ether)
0x787...cabaB (99.99999999999975536 ether)
0x617...5E7f2 (100 ether)
0x17F...8c372 (100 ether)
0x5c6...21678 (100 ether)
0x03C...D1Ff7 (100 ether)
0x1aE...E454C (100 ether)
0x0A0...C70DC (100 ether)
0xCA3...a733c (100 ether)
0x147...C160C (100 ether)
0x4B0...4D2dB (100 ether)
0x583...40225 (100 ether)
0xdD8...92148 (100 ether)
CANDIDATENAMES:
['Alice Cooper' , 'Jason Blossom']
Calldata
Parameters
transact
Publish to IPFS
At Address
Load contract from Address


DEPLOY & RUN TRANSACTIONS
0x5B3...eddC4 (99.999999999997868949 et
GAS LIMIT
3000000
VALUE
0
Wei
CONTRACT
Ballot - Voting (1).sol
evm version: paris
DEPLOY
CANDIDATENAMES:
['Alice Cooper' , 'Jason Blossom']
Calldata
Parameters
transact
Publish to IPFS
At Address
Load contract from Address
Transactions recorded
10


DEPLOY & RUN TRANSACTIONS
Deployed Contracts
BALLOT AT 0XD7A...F771B (MEMORY)
Balance: 0 ETH
addCandidates
string[] candidateNames
endVote
giveRightToVote
address voter
startVote
startVote - transact (not payable)
vote
uint256 candidate
candidates
uint256
chairperson
state
voters
address
winningCandi...


DEPLOY & RUN TRANSACTIONS
BALLOT AT 0XD7A...F771B (MEMORY)
Balance: 0 ETH
addCandidates
string[] candidateNames
endVote
giveRightToVote
address voter
startVote
vote
uint256 candidate
candidates
1
0: string: name Jason Blossom
1: uint256: voteCount 2
chairperson
state
voters
address
winningCandi...
0: string: winnerName_ Alice Cooper
Low level interactions
CALLDATA

Conclusion

The development of a Blockchain-based E-Voting Decentralised Application (DApp) represents a significant leap forward in the quest to modernise and fortify electoral processes. The immutable and tamper-resistant nature of blockchain ensures the integrity and authenticity of every vote, rendering traditional vulnerabilities such as electoral fraud and data manipulation obsolete. Smart contracts streamline the voting process, from registration to result verification, reducing the need for intermediaries and enhancing efficiency. Furthermore, blockchain technology promotes accessibility through remote voting, all while maintaining the highest standards of security.
By eliminating doubts and disputes surrounding the electoral process, a Blockchain-based E-Voting DApp has the potential to redefine the democratic landscape, fostering public trust and participation while safeguarding the sanctity of the ballot. As we look to the future, this innovative application of blockchain stands as a beacon of hope for the evolution of democracy, one that is secure, efficient, and truly representative of the will of the people.

References

1. https://ieeexplore.ieee.org/abstract/document/8457919
2. https://www.sciencedirect.com/science/article/pii/S1877050918318271
3. https://www.mdpi.com/2073-8994/12/8/1328
4. https://remix-ide.readthedocs.io/en/latest/
5. https://docs.soliditylang.org/en/v0.8.21/