

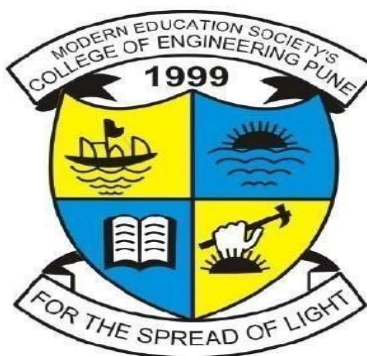
**Savitribai Phule Pune University**  
**Modern Education Society's College of Engineering, Pune**

19, Bund Garden, V.K. Joag Path, Pune – 411001.

**ACCREDITED BY NAAC**

**WITH "A++" GRADE (CGPA – 3.13)**

**DEPARTMENT OF COMPUTER ENGINEERING**



A REPORT

ON

**"BT Lab: Mini Project on  
e-Voting System"**

**B.E. (COMPUTER)**

*SUBMITTED BY*

Sahil Naik  
Sohan Chidrawar  
Supreet Birajdar

*UNDER THE GUIDANCE OF*

**Prof. Anand Dhawale**

**Problem Statement:**

Develop a Blockchain based application dApp (de-centralized app) for e-voting system.

**Hardware Requirements:**

2 GB RAM, 500 GB Hard Disk, PC / Laptop

**Learning Objective:**

To develop a blockchain based application for e-voting system using solidity programming language.

**Outcome:**

Developed a blockchain based application for e-voting system. A person can be registered as a candidate and others can vote for the person.

**Introduction:**

Blockchain is a technology that is rapidly gaining momentum in era of industry 4.0. With high security and transparency provisions, it is being widely used in supply chain management systems, healthcare, payments, business, IoT, voting systems, etc.

Why do we need it?

Current voting systems like ballot box voting or electronic voting suffer from various security threats such as DDoS attacks, polling booth capturing, vote alteration and manipulation, malware attacks, etc, and also require huge amounts of paperwork, human resources, and time. This creates a sense of distrust among existing systems.

**Some of the disadvantages are:**

- Long Queues during elections.
- Security Breaches like data leaks, vote tampering.
- Lot of paperwork involved, hence less eco-friendly and time-consuming. Difficult for differently-abled voters to reach polling booth.
- Cost of expenditure on elections is high.

Using blockchain, voting process can be made more secure, transparent, immutable, and reliable. How? Let's take an example.

Suppose you are an eligible voter who goes to polling booth and cast vote using EVM (Electronic Voting Machine). But since it's a circuitry after all and if someone tampers with microchip, you may never know that did your vote reach to person for whom you voted or was diverted into another candidate's account?

Since there's no tracing back of your vote. But if you use blockchain- it stores everything as a transaction that will be explained soon below; and hence gives you a receipt of your vote (in a form of a transaction ID) and you can use it to ensure that your vote has been counted securely.

Now suppose a digital voting system (website/app) has been launched to digitize process and all confidential data is stored on a single admin server/machine, if someone tries to hack it or snoop over it, he/she can change candidate's vote count- from 2 to 22! You may never know that hacker installs malware or performs clickjacking attacks to steal or negate your vote or simply attacks central server.

To avoid this, if system is integrated with blockchain- a special property called immutability protects system. Consider SQL, PHP, or any other traditional database systems. You can insert, update, or delete votes. But in a blockchain you can just insert data but cannot update or delete. Hence when you insert something, it stays there forever and no one can manipulate it- Thus name immutable ledger.

But building a blockchain system is not enough. It should be decentralized i.e. if one server goes down or something happens on a particular node, other nodes can function normally and do not have to wait for victim node's recovery.

**So, a gist of advantages are listed below:**

You can vote anytime/anywhere (During Pandemics like COVID-19 where it's impossible to hold elections physically)

- Secure
- Immutable
- Faster
- Transparent

All data is then encrypted and stored as a transaction. This transaction is then broadcasted to every node in network, which in turn is then verified. If network approves transaction, it is stored in a block and added to chain. Note that once a block is added into chain, it stays there forever and cannot be updated. Users can now see results and trace back transaction if they want.

### Smart contract:

A smart contract is a self-imposed contract that is embedded in a blockchain managed computer code. This code includes a set of rules governing the communication and decision on the contract between the parties, the contract will be enforced automatically once the already defined rules are met. Smart contract gives a framework for efficient control between two or more parties of tokenizes assets and access rights [1]. Fig 1 [1]; shows the working principle of smart contract. Blockchain is just a database that cannot be altered, without smart contract, which expands and leverages blockchain.

### Code:

```
pragma solidity ^0.4.25;

contract ElectionFact {

    struct ElectionDet {
        address deployedAddress;
        string el_n;
        string el_d;
    }

    mapping(string=>ElectionDet) companyEmail;

    function createElection(string memory email,string memory election_name,
string memory election_description) public{
        address newElection = new Election(msg.sender ,
election_name,election_description);
        companyEmail[email].deployedAddress = newElection;
        companyEmail[email].el_n = election_name;
        companyEmail[email].el_d = election_description;
    }

    function getDeployedElection(string memory email) public view
returns(address,string,string) {
        address val = companyEmail[email].deployedAddress;
        if(val == 0)
```

```

        return (0,"","Create an election.");
    else
        return(companyEmail[email].deployedAddress,companyEmail[email].el_n,
companyEmail[email].el_d);
    }
}

contract Election {
    //election_authority's address
    address election_authority;
    string election_name;
    string election_description;
    bool status;

    //election_authority's address taken when it deploys the contract
    constructor(address authority , string name, string description) public {
        election_authority = authority;
        election_name = name;
        election_description = description;
        status = true;
    }

    //Only election_authority can call this function
    modifier owner() {require(msg.sender == election_authority, "Error: Access
Denied.");
    _;
    }
    //candidate election_description
    struct Candidate {
        string candidate_name;
        string candidate_description;
        string imgHash;
        uint8 voteCount;
        string email;
    }
    //candidate mapping
    mapping(uint8=>Candidate) public candidates;
    //voter election_description
    struct Voter {
        uint8 candidate_id_voted;
        bool voted;
    }

    //voter mapping
    mapping(string=>Voter) voters;
    //counter of number of candidates
    uint8 numCandidates;
    //counter of number of voters

```

```

uint8 numVoters;
//function to add candidate to mapping
function addCandidate(string memory candidate_name, string memory
candidate_description, string memory imgHash,string memory email) public owner {
    uint8 candidateID = numCandidates++; //assign id of the candidate
    candidates[candidateID] =
Candidate(candidate_name,candidate_description,imgHash,0,email); //add the
values to the mapping
}

//function to vote and check for double voting
function vote(uint8 candidateID,string e) public {
//if false the vote will be registered
require(!voters[e].voted, "Error:You cannot double vote");
voters[e] = Voter (candidateID,true); //add the values to the mapping
numVoters++;
candidates[candidateID].voteCount++; //increment vote counter of
candidate
}

//function to get count of candidates
function getNumOfCandidates() public view returns(uint8) {
    return numCandidates;
}

//function to get count of voters
function getNumOfVoters() public view returns(uint8) {
    return numVoters;
}

//function to get candidate information
function getCandidate(uint8 candidateID) public view returns (string memory,
string memory, string memory, uint8,string memory) {
    return
(candidates[candidateID].candidate_name,candidates[candidateID].candidate_descri
ption, candidates[candidateID].imgHash,candidates[candidateID].voteCount,
candidates[candidateID].email);
}

//function to return winner candidate information
function winnerCandidate() public view owner returns (uint8) {
    uint8 largestVotes = candidates[0].voteCount;
    uint8 candidateID;
    for(uint8 i = 1;i<numCandidates;i++) {
        if(largestVotes < candidates[i].voteCount) {
            largestVotes = candidates[i].voteCount;
            candidateID = i;
        }
    }
}

```

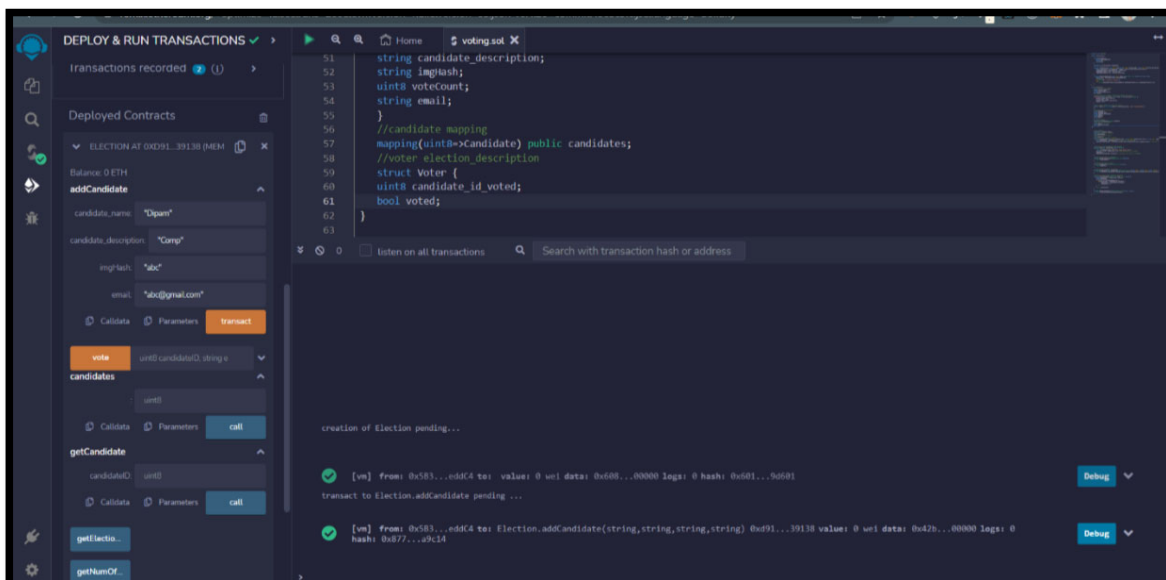
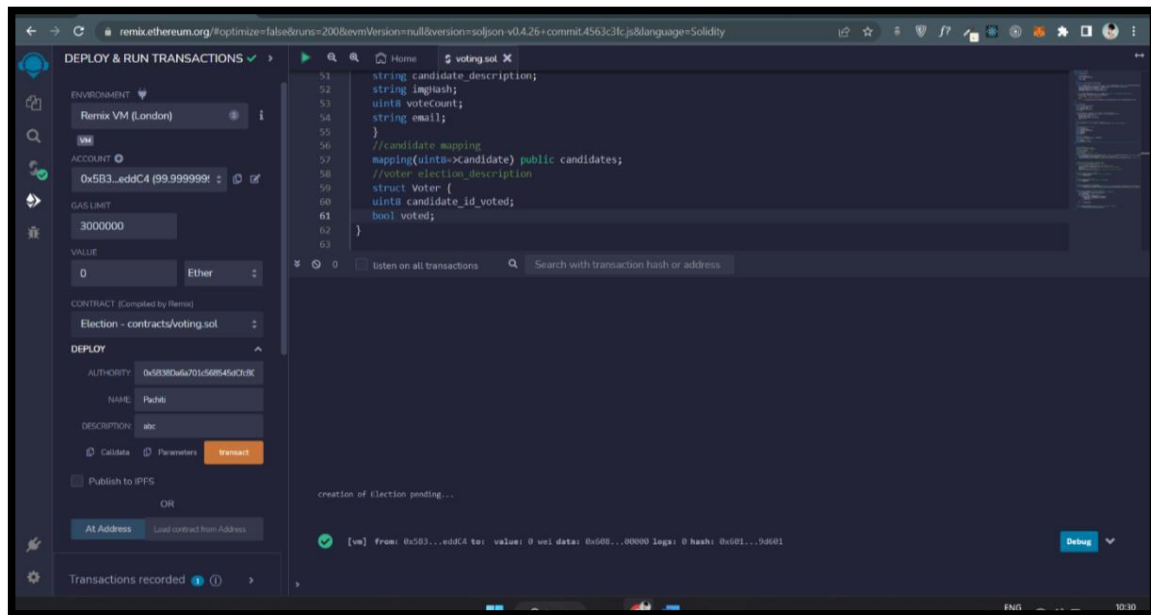
```

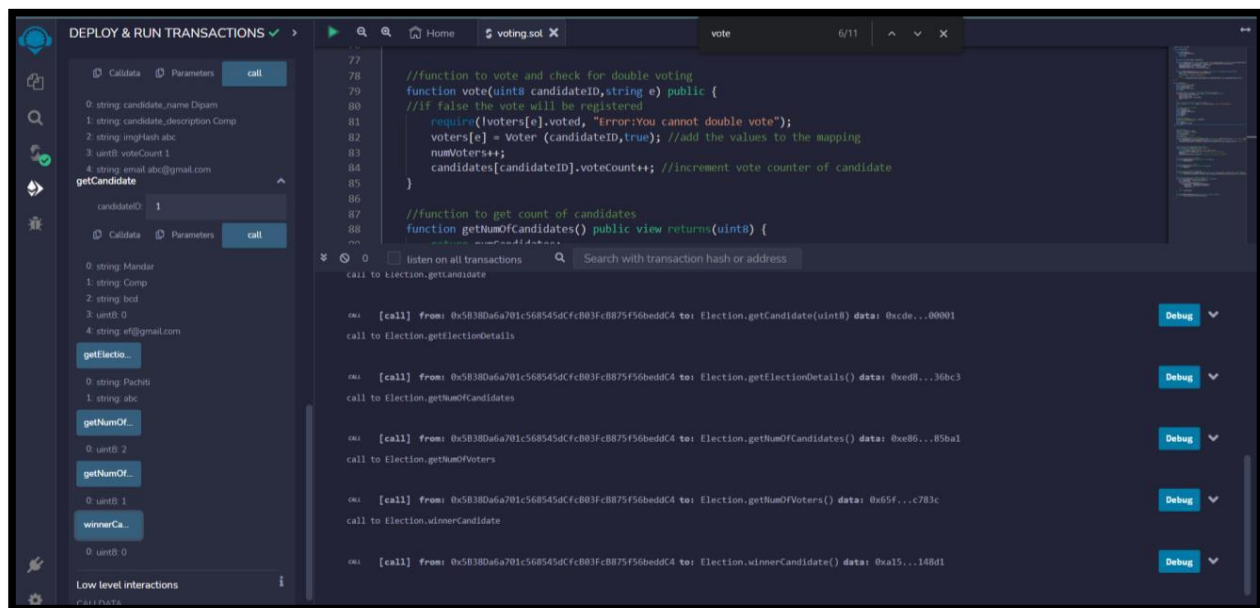
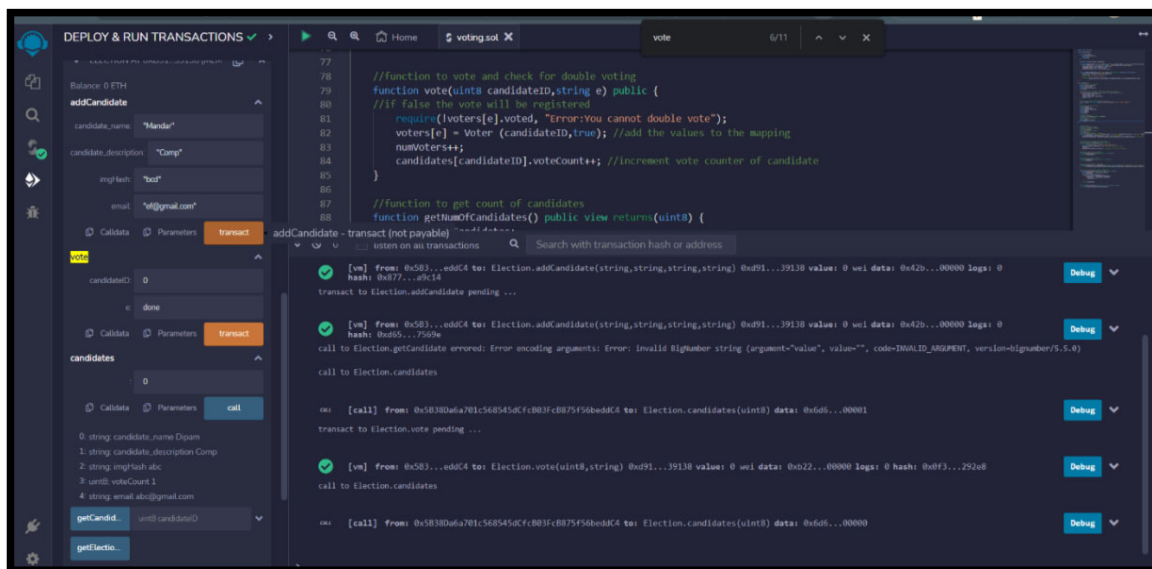
    }
    return (candidateID);
}

function getElectionDetails() public view returns(string, string) {
    return (election_name,election_description);
}
}

```

## Results:





## CONCLUSION:

Successfully developed a blockchain application for e-voting system.