

PRACTICAL: 7

AIM: Create smart contracts for e-voting using solidity.

THEORY:

Online voting is a trend that is gaining momentum in modern society. It has great potential to reduce organizational costs and increase voter turnout. No more printing ballots or opening polling places. Voters can vote anywhere they have an internet connection. Despite these advantages, online voting solutions pose new threats and are therefore viewed with great caution. A single vulnerability could lead to large-scale manipulation of votes. Electronic voting systems must be legitimate, accurate, secure and convenient when used in elections. However, potential issues associated with electronic voting systems may limit their use. Blockchain technology was developed to overcome these problems, providing decentralized nodes for e-voting, and is primarily used to manufacture e-voting systems for its end-to-end verification benefits. . This technology is a great alternative to traditional electronic voting solutions with decentralized, undeniable secure protection. The following article provides an overview of electronic voting systems based on blockchain technology. The main purpose of this analysis was to examine the current state of blockchain-based election research and online voting systems and all associated difficulties in predicting future developments. This study provides a conceptual description of the intended blockchain-based e-voting application and an introduction to the basic structure and properties of blockchains relevant to e-voting. As a result of this research, it was discovered that blockchain systems could help solve some of the problems that currently plague voting systems. On the other hand, the most frequently mentioned issues in blockchain applications are data protection and transaction speed. For sustainable blockchain-based e-voting systems, remote participation security must be feasible, and transaction speed must be considered for scalability.

Due to these concerns, it was decided that existing frameworks needed to be improved for use in voting systems. Core Components of Ethereum The rest of this guide assumes that you are familiar with blockchain and Ethereum. Here's a great guide on how to do this. I also wrote a quick overview of the core components you might want to know. Smart contracts serve as backend logic and storage. A contract written in Solidity, a smart contract language, is a collection of code and data that resides at a specific address on the Ethereum blockchain. They are very similar to classes in object-oriented programming, containing functions and state variables. Smart contracts, along with blockchain, are the foundation of all decentralized applications. They are immutable and distributed like blockchains, so upgrading is a problem if they are already on the Ethereum network. Luckily, there are several ways to do this. The Ethereum Virtual Machine (EVM) manages the internal state and computations of the entire Ethereum network. Think of EVM as this large distributed computer containing "addresses" where you can run code, change data, and interact with each other. Web3.js is a JavaScript API that allows you to interact with the blockchain, including transactions and calling smart contracts. This API abstracts communication with Ethereum clients, allowing developers to focus on the content of their application.

To do this, you'll need to embed a web3 instance into your browser. Other tools to use Truffle is a popular test development framework for Ethereum. This includes a development blockchain, compile and migration scripts for deploying the contract to the blockchain, contract testing, etc. Easier to develop! Truffle Contracts is an abstraction of the Web3 JavaScript API that allows you to easily connect and interact with smart contracts. Metamask brings Ethereum to your browser. It's a browser extension that

provides a secure web3 instance tied to your Ethereum address, allowing you to use decentralized applications. This tutorial does not use Metamask, but this is how a user will interact with her DApp in production. Instead, inject your own web3 instance during development. See this link for more information.

CODE:

```
//SPDX-License-Identifier: MIT
pragma solidity ^0.8.10;

import "./Types.sol";

contract EVoting {
    Types.Candidate[] public candidates;
    mapping(uint256 => Types.Voter) voter;
    mapping(uint256 => Types.Candidate) public candidate;
    mapping(uint256 => uint256) internal votesCount;

    address public electionChief;
    uint256 public votingStartTime;
    uint256 public votingEndTime;

    constructor(uint256 startTime_, uint256 endTime_) {
        initializeCandidateDatabase_();
        initializeVoterDatabase_();
        votingStartTime = startTime_;
        votingEndTime = endTime_;
        electionChief = msg.sender;
    }

    function getCandidateList(uint256 voterAadharNumber)
        public
        view
        returns (Types.Candidate[] memory)
    {
        Types.Voter storage voter_ = voter[voterAadharNumber];
        uint256 _politicianOfMyConstituencyLength = 0;

        for (uint256 i = 0; i < candidates.length; i++) {
            if (
                voter_.stateCode == candidates[i].stateCode &&
                voter_.constituencyCode == candidates[i].constituencyCode
            ) _politicianOfMyConstituencyLength++;
        }
        Types.Candidate[] memory cc = new Types.Candidate[](
            _politicianOfMyConstituencyLength
        );

        uint256 _indx = 0;
        for (uint256 i = 0; i < candidates.length; i++) {
            if (
```

```

        voter_.stateCode == candidates[i].stateCode &&
        voter_.constituencyCode == candidates[i].constituencyCode
    ) {
        cc[_indx] = candidates[i];
        _indx++;
    }
}
return cc;
}

function isVoterEligible(uint256 voterAadharNumber)
    public
    view
    returns (bool voterEligible_)
{
    Types.Voter storage voter_ = voter[voterAadharNumber];
    if (voter_.age >= 18 && voter_.isAlive) voterEligible_ = true;
}

function didCurrentVoterVoted(uint256 voterAadharNumber)
    public
    view
    returns (bool userVoted_, Types.Candidate memory candidate_)
{
    userVoted_ = (voter[voterAadharNumber].votedTo != 0);
    if (userVoted_)
        candidate_ = candidate[voter[voterAadharNumber].votedTo];
}

function vote(
    uint256 nominationNumber,
    uint256 voterAadharNumber,
    uint256 currentTime_
)
    public
    votingLinesAreOpen(currentTime_)
    isEligibleVote(voterAadharNumber, nominationNumber)
{
    voter[voterAadharNumber].votedTo = nominationNumber;

    uint256 voteCount_ = votesCount[nominationNumber];
    votesCount[nominationNumber] = voteCount_ + 1;
}

function getVotingEndTime() public view returns (uint256 endTime_) {
    endTime_ = votingEndTime;
}

function updateVotingStartTime(uint256 startTime_, uint256 currentTime_)
    public
    isElectionChief

```

```

{
    require(votingStartTime > currentTime_);
    votingStartTime = startTime_;
}

function extendVotingTime(uint256 endTime_, uint256 currentTime_)
    public
    isElectionChief
{
    require(votingStartTime < currentTime_);
    require(votingEndTime > currentTime_);
    votingEndTime = endTime_;
}

function getResults(uint256 currentTime_)
    public
    view
    returns (Types.Results[] memory)
{
    require(votingEndTime < currentTime_);
    Types.Results[] memory resultsList_ = new Types.Results[](
        candidates.length
    );
    for (uint256 i = 0; i < candidates.length; i++) {
        resultsList_[i] = Types.Results({
            name: candidates[i].name,
            partyName: candidates[i].partyName,
            nominationNumber: candidates[i].nominationNumber,
            stateCode: candidates[i].stateCode,
            constituencyCode: candidates[i].constituencyCode,
            voteCount: votesCount[candidates[i].nominationNumber]
        });
    }
    return resultsList_;
}

modifier votingLinesAreOpen(uint256 currentTime_) {
    require(currentTime_ >= votingStartTime);
    require(currentTime_ <= votingEndTime);
    _;
}

modifier isEligibleVote(uint256 voterAadhar_, uint256 nominationNumber_) {
    Types.Voter memory voter_ = voter[voterAadhar_];
    Types.Candidate memory politician_ = candidate[nominationNumber_];
    require(voter_.age >= 18);
    require(voter_.isAlive);
    require(voter_.votedTo == 0);
    require(
        (politician_.stateCode == voter_.stateCode &&
        politician_.constituencyCode == voter_.constituencyCode)
    )
}

```

```

    );
    -;
}

modifier isElectionChief() {
    require(msg.sender == electionChief);
    -;
}

function initializeCandidateDatabase_() internal {
    Types.Candidate[] memory candidates_ = new Types.Candidate[](4);

    // Gujarat
    candidates_[0] = Types.Candidate({
        name: "Chandra Babu Naidu",
        partyName: "TDP",
        nominationNumber: uint256(727477314982),
        stateCode: uint8(10),
        constituencyCode: uint8(1)
    });
    candidates_[1] = Types.Candidate({
        name: "Jagan Mohan Reddy",
        partyName: "YSRCP",
        nominationNumber: uint256(835343722350),
        stateCode: uint8(10),
        constituencyCode: uint8(1)
    });
    candidates_[2] = Types.Candidate({
        name: "G V Anjaneyulu",
        partyName: "TDP",
        nominationNumber: uint256(969039304119),
        stateCode: uint8(10),
        constituencyCode: uint8(2)
    });
    candidates_[3] = Types.Candidate({
        name: "Anil Kumar Yadav",
        partyName: "YSRCP",
        nominationNumber: uint256(429300763874),
        stateCode: uint8(10),
        constituencyCode: uint8(2)
    });

    for (uint256 i = 0; i < candidates_.length; i++) {
        candidate[candidates_[i].nominationNumber] = candidates_[i];
        candidates.push(candidates_[i]);
    }
}

function initializeVoterDatabase_() internal {
    // Gujarat
    voter[uint256(482253918244)] = Types.Voter({

```

```
        name: "Suresh",
        aadharNumber: uint256(482253918244),
        age: uint8(21),
        stateCode: uint8(10),
        constituencyCode: uint8(1),
        isAlive: true,
        votedTo: uint256(0)
    });
    voter[uint256(532122269467)] = Types.Voter({
        name: "Ramesh",
        aadharNumber: uint256(532122269467),
        age: uint8(37),
        stateCode: uint8(10),
        constituencyCode: uint8(1),
        isAlive: false,
        votedTo: uint256(0)
    });
    voter[uint256(468065932286)] = Types.Voter({
        name: "Mahesh",
        aadharNumber: uint256(468065932286),
        age: uint8(26),
        stateCode: uint8(10),
        constituencyCode: uint8(1),
        isAlive: true,
        votedTo: uint256(0)
    });
    voter[uint256(809961147437)] = Types.Voter({
        name: "Krishna",
        aadharNumber: uint256(809961147437),
        age: uint8(19),
        stateCode: uint8(10),
        constituencyCode: uint8(2),
        isAlive: true,
        votedTo: uint256(0)
    });
    voter[uint256(908623597782)] = Types.Voter({
        name: "Narendra",
        aadharNumber: uint256(908623597782),
        age: uint8(36),
        stateCode: uint8(10),
        constituencyCode: uint8(2),
        isAlive: true,
        votedTo: uint256(0)
    });

    voter[uint256(760344621247)] = Types.Voter({
        name: "Raghu",
        aadharNumber: uint256(760344621247),
        age: uint8(42),
        stateCode: uint8(10),
        constituencyCode: uint8(2),
```

```
        isAlive: true,  
        votedTo: uint256(0)  
    });  
    }  
}
```

Types.sol

// SPDX-License-Identifier: MIT
pragma solidity ^0.8.10;

```
library Types {  
    struct Voter {  
        uint256 aadharNumber; // voter unique ID  
        string name;  
        uint8 age;  
        uint8 stateCode;  
        uint8 constituencyCode;  
        bool isAlive;  
        uint256 votedTo; // aadhar number of the candidate  
    }  
  
    struct Candidate {  
        string name;  
        string partyName;  
        uint256 nominationNumber; // unique ID of candidate  
        uint8 stateCode;  
        uint8 constituencyCode;  
    }  
  
    struct Results {  
        string name;  
        string partyName;  
        uint256 voteCount; // number of accumulated votes  
        uint256 nominationNumber; // unique ID of candidate  
        uint8 stateCode;  
        uint8 constituencyCode;  
    }  
}
```

Evoting.test.sol

```
let EVoting = artifacts.require("EVoting");  
  
contract(  
    "EVoting",  
    (async = (accounts) => {  
        let chief = accounts[0];
```

```
it("Contract Deployed", async () => {
  let evc = await EVoting.deployed();
  let electionChief = await evc.electionChief();
  let electionStart = await evc.votingStartTime();
  let electionEnd = await evc.votingEndTime();

  assert.notEqual(evc.address, 0x0, "Has Contract Address");

  assert.equal(electionChief, chief, "Not Chief");

  assert.equal(
    electionStart.toNumber(),
    1662440960,
    "Start Time not: (1662440960) Tuesday, September 6, 2022 10:39:20"
  );

  assert.equal(
    electionEnd.toNumber(),
    1667711360,
    "End Time not: (1667711360) Sunday, November 6, 2022 10:39:20"
  );
});

it("candidate", async () => {

  let evc = await EVoting.deployed();
  let voter = accounts[1];

  let actualCandidate = [
    ["Chandra Babu Naidu", "TDP", "727477314982", "10", "1"],
    ["Jagan Mohan Reddy", "YSRCP", "835343722350", "10", "1"],
  ];

  let candidateList = await evc.getCandidateList(532122269467, {
    from: voter,
  });

  assert.equal(
    candidateList.length,
    actualCandidate.length,
    "Candidate List not empty"
  );
});

it("Voter Tests", async () => {

  let evc = await EVoting.deployed();
  let voter = accounts[1];

  let isEligible = await evc.isVoterEligible(727938171119, { from: voter });
  assert.equal(isEligible, false, "Should not be eligible");
});
```



```
let isAlive = await evc.isVoterEligible(756623869645, { from: voter });
assert.equal(isAlive, false, "Should be eligible");

let didVoted = await evc.didCurrentVoterVoted(532122269467, {
  from: voter,
});
// console.log(didVoted);
assert.equal(didVoted.userVoted_, false, "Should not have voted");

await evc.vote(727477314982, 468065932286, 1666365324, {
  from: voter,
});

didVoted = await evc.didCurrentVoterVoted(468065932286, {
  from: voter,
});
assert.equal(didVoted.userVoted_, true, "Should have voted");
});

it("Result", async () => {
  let evc = await EVoting.deployed();

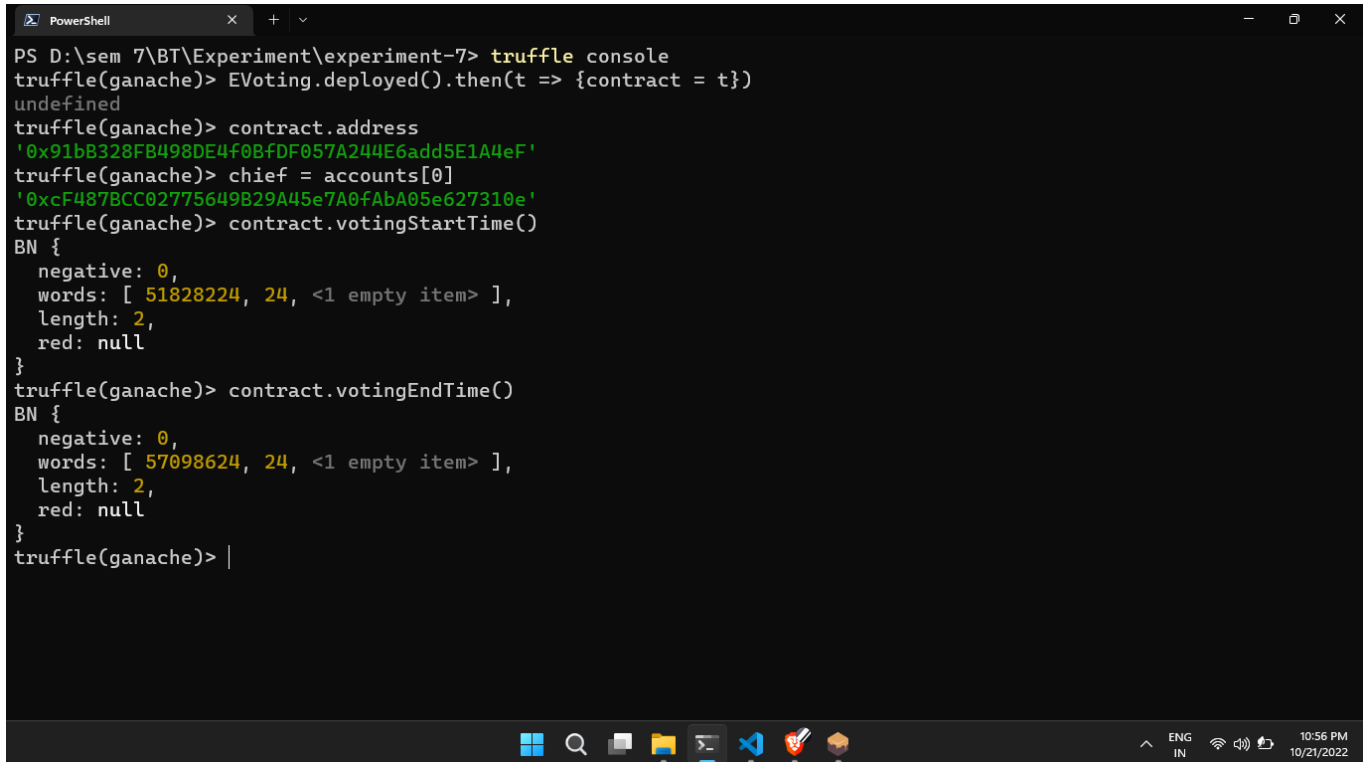
  // constituency: 1
  await evc.vote(835343722350, 482253918244, 1666365324, {
    from: accounts[3],
  });

  // constituency: 2
  await evc.vote(969039304119, 760344621247, 1666365324, {
    from: accounts[4],
  });

  await evc.vote(429300763874, 908623597782, 1666365324, {
    from: accounts[5],
  });

  await evc.vote(429300763874, 809961147437, 1666365324, {
    from: accounts[6],
  });

  let res = await evc.getResults(1767711360, { from: accounts[1] });
  // console.log(res);
  assert.equal(res.length, 4, "Should have 4 results");
});
});
```

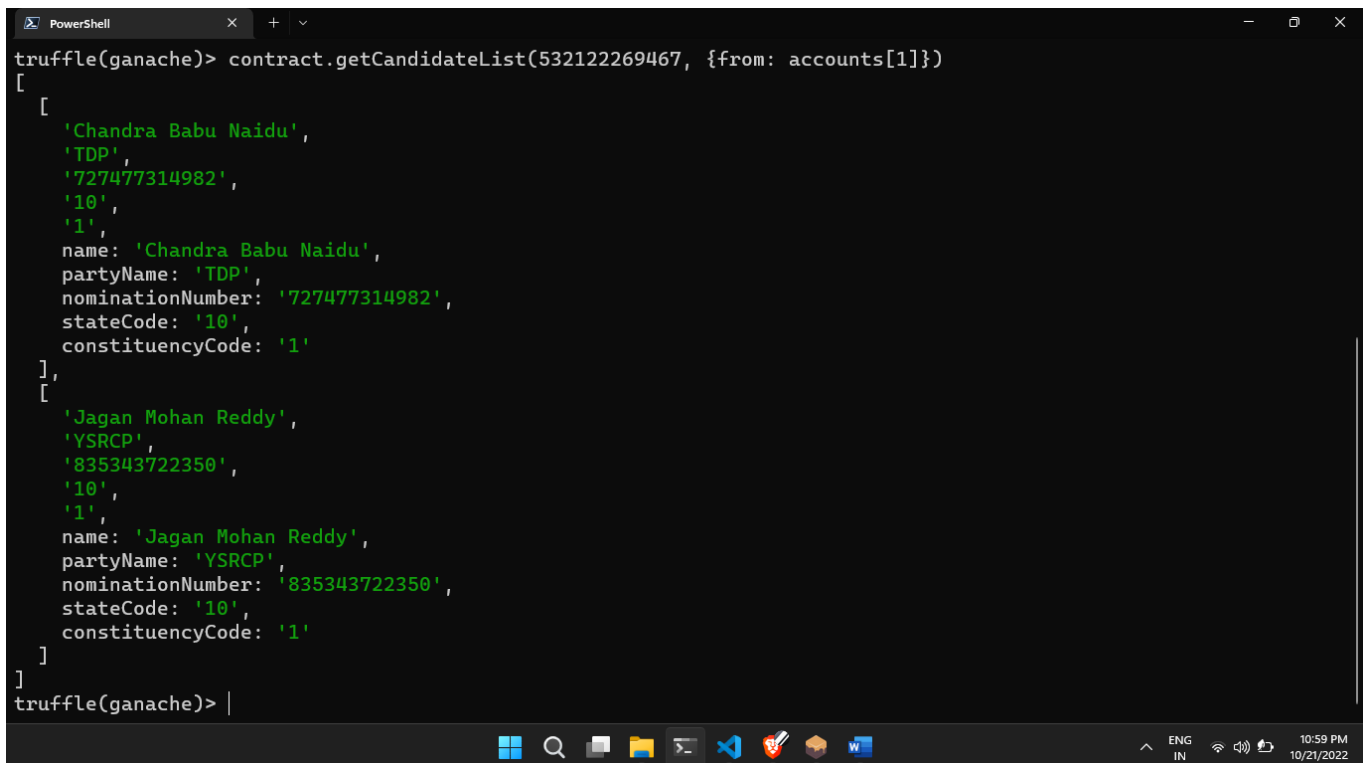
OUTPUT:


```

PS D:\sem 7\BT\Experiment\experiment-7> truffle console
truffle(ganache)> EVoting.deployed().then(t => {contract = t})
undefined
truffle(ganache)> contract.address
'0x91b8328FB498DE4f0BfDF057A244E6add5E1A4eF'
truffle(ganache)> chief = accounts[0]
'0xcF4878CC02775649B29A45e7A0fAbA05e627310e'
truffle(ganache)> contract.votingStartTime()
BN {
  negative: 0,
  words: [ 51828224, 24, <1 empty item> ],
  length: 2,
  red: null
}
truffle(ganache)> contract.votingEndTime()
BN {
  negative: 0,
  words: [ 57098624, 24, <1 empty item> ],
  length: 2,
  red: null
}
truffle(ganache)> |

```

Figure 1: Accessing the deployed contract through truffle console and accessing the two methods for getting start and end time for voting

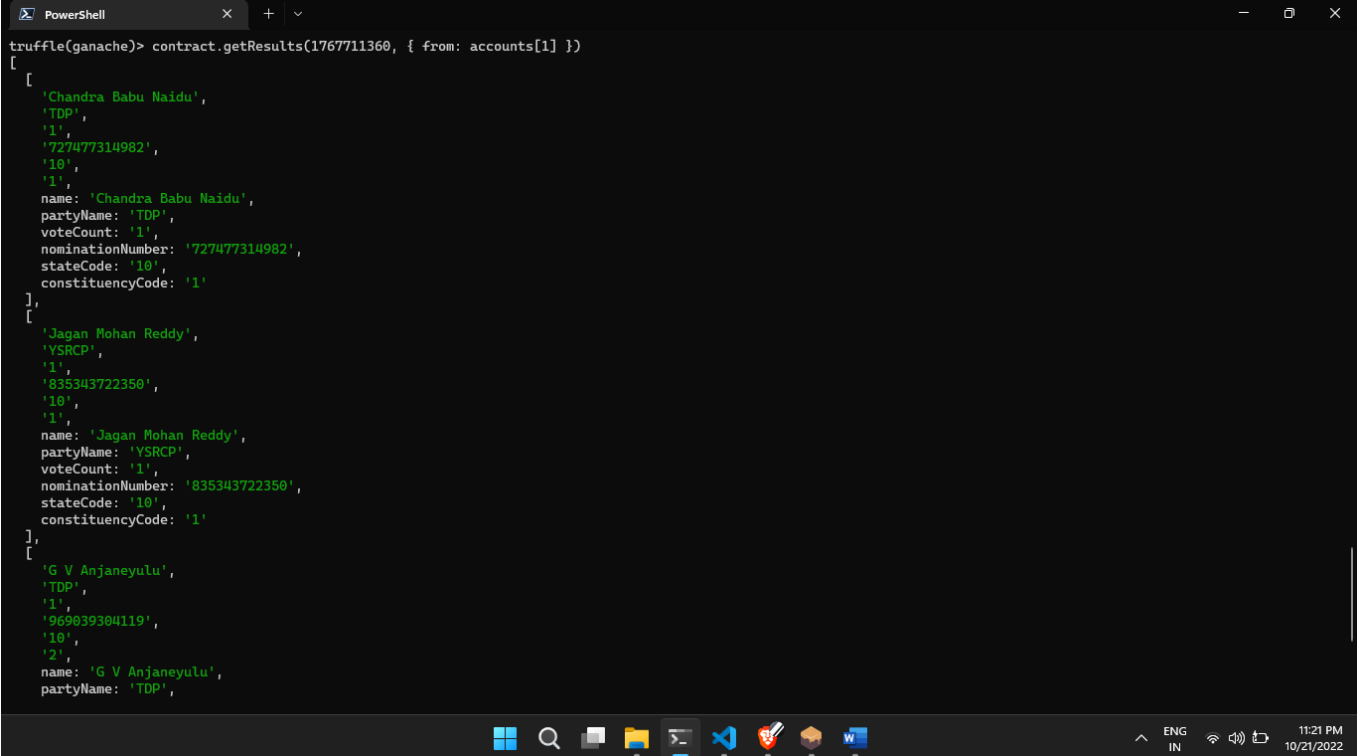


```

truffle(ganache)> contract.getCandidateList(532122269467, {from: accounts[1]})
[
  [
    'Chandra Babu Naidu',
    'TDP',
    '727477314982',
    '10',
    '1',
    name: 'Chandra Babu Naidu',
    partyName: 'TDP',
    nominationNumber: '727477314982',
    stateCode: '10',
    constituencyCode: '1'
  ],
  [
    'Jagan Mohan Reddy',
    'YSRCP',
    '835343722350',
    '10',
    '1',
    name: 'Jagan Mohan Reddy',
    partyName: 'YSRCP',
    nominationNumber: '835343722350',
    stateCode: '10',
    constituencyCode: '1'
  ]
]
truffle(ganache)> |

```

Figure 2: Getting the candidate list for particular voter based on the stateCode and constituencyCode

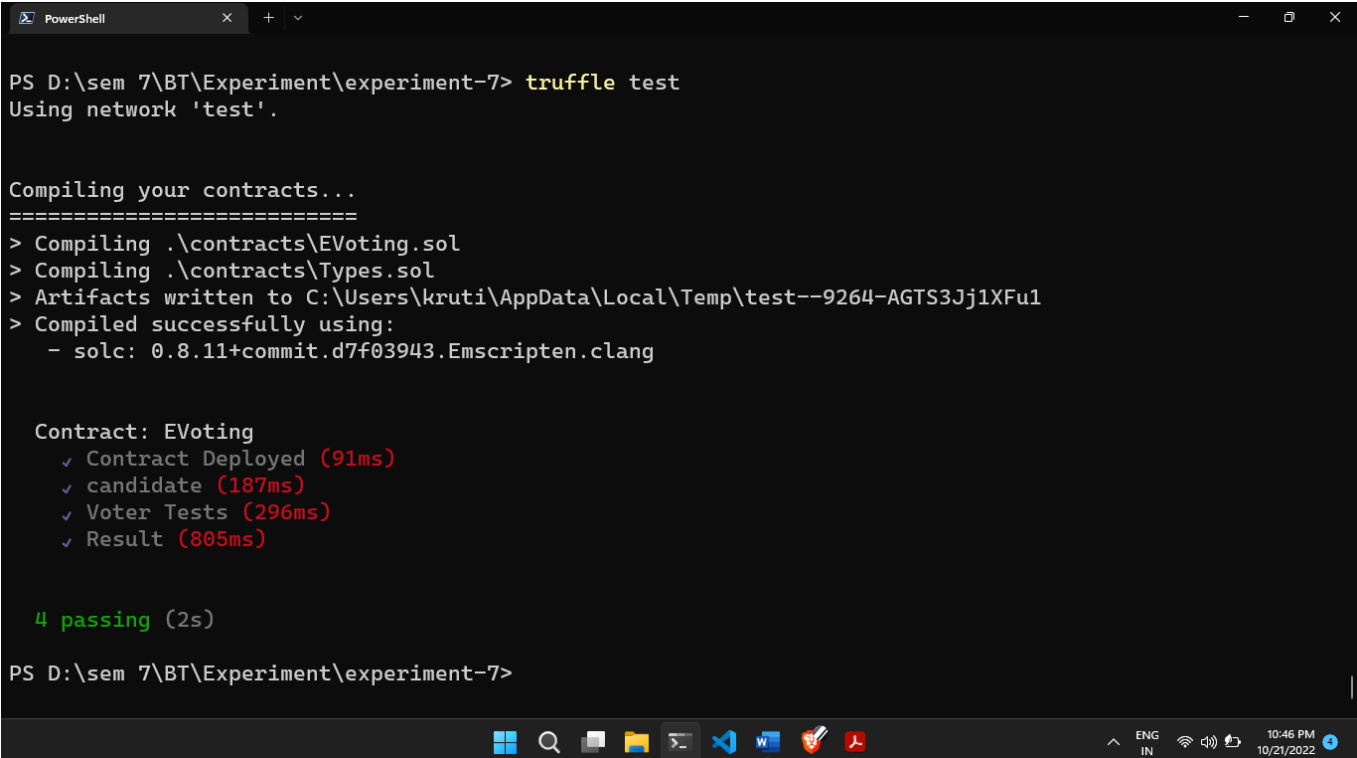


```

PowerShell
truffle(ganache)> contract.getResults(1767711360, { from: accounts[1] })
[
  [
    'Chandra Babu Naidu',
    'TDP',
    '1',
    '727477314982',
    '10',
    '1',
    name: 'Chandra Babu Naidu',
    partyName: 'TDP',
    voteCount: '1',
    nominationNumber: '727477314982',
    stateCode: '10',
    constituencyCode: '1'
  ],
  [
    'Jagan Mohan Reddy',
    'YSRCP',
    '1',
    '835343722350',
    '10',
    '1',
    name: 'Jagan Mohan Reddy',
    partyName: 'YSRCP',
    voteCount: '1',
    nominationNumber: '835343722350',
    stateCode: '10',
    constituencyCode: '1'
  ],
  [
    'G V Anjaneyulu',
    'TDP',
    '1',
    '969039304119',
    '10',
    '2',
    name: 'G V Anjaneyulu',
    partyName: 'TDP'
  ]
]

```

Figure 5: After voting for other voter we can get the results from this function



```

PowerShell
PS D:\sem 7\BT\Experiment\experiment-7> truffle test
Using network 'test'.

Compiling your contracts...
=====
> Compiling .\contracts\EVoting.sol
> Compiling .\contracts\Types.sol
> Artifacts written to C:\Users\kruti\AppData\Local\Temp\test--9264-AGTS3Jj1XFu1
> Compiled successfully using:
   - solc: 0.8.11+commit.d7f03943.Emscripten.clang

Contract: EVoting
  ✓ Contract Deployed (91ms)
  ✓ candidate (187ms)
  ✓ Voter Tests (296ms)
  ✓ Result (805ms)

4 passing (2s)

PS D:\sem 7\BT\Experiment\experiment-7>

```

Figure 6: The test that I ran for the contract

LATEST APPLICATIONS:

- **Immutability:** Any blockchain documents cannot be changed or deleted;
- **Provenance:** It refers to the fact that every transaction can be tracked in the blockchain ledger;
- **Decentralization:** The entire distributed database may be accessible by all members of the blockchain network. A consensus algorithm allows control of the system, as shown in the core process;
- **Anonymity:** A blockchain network participant has generated an address rather than a user identification. It maintains anonymity, especially in a blockchain public system;
- **Transparency:** unable to manipulate the blockchain network

LEARNING OUTCOME:

Performing this Practical learned about mapping how to create mapping variable and also learned about modifiers how we can create custom modifiers and use it with functions, learned about modularizing the code how we can import other solidity file in another file for code readability.

REFERENCE:

1. <https://remix.ethereum.org>
2. <https://codeburst.io/build-your-first-ethereum-smart-contract-with-solidity-tutorial-94171d6b1c4b>
3. <https://techblog.geekyants.com/e-voting-via-blockchain-a-case-study>
4. <https://github.com/GeekyAnts/sample-e-voting-system-ethereum>