Name : Pratik Tiruwa
Seat No : 31031523035

C. Create a DApp to implement elections.

1. index.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>DApp-4</title>
</head>

<body>
    <h1>Blockchain Voting DApp</h1>
    <h2>Select Candidate to Vote</h2>
    <button onclick="vote('Can1')">Candidate 1</button>
    <button onclick="vote('Can2')">Candidate 2</button>
    <button onclick="vote('Can3')">Candidate 3</button>
    <br><br>
    <h2>Check Results:</h2>
    <button onclick="checkResult()">Check Result</button>
    <p>The Winner Is: <span id="result"></span></p>

    <script src="https://cdn.jsdelivr.net/npm/web3@latest/dist/web3.min.js"></script>
    <script src="app.js"></script>
</body>

</html>
```

Name : Pratik Tiruwa
Seat No : 31031523035

2. app.js

```
const contractAddress = ""; // Replace with your deployed contract address
const contractABI = []; // Use ABI from compiled contract

let web3;
let contract;
```

Name : Pratik Tiruwa
Seat No : 31031523035

```
window.addEventListener("load", async () => {
    if (window.ethereum) {
        web3 = new Web3(window.ethereum);
        await window.ethereum.enable();
    } else {
        console.log("MetaMask not detected. Please install MetaMask.");
    }

    contract = new web3.eth.Contract(contractABI, contractAddress);
});

async function vote(can) {
    var canM = can;
    const accounts = await web3.eth.getAccounts();
    const voter = accounts[0];

    contract.methods.vote(canM).send({
        from: voter
    })
};

async function checkResult() {
    const accounts = await web3.eth.getAccounts();

    contract.methods.getWinner()
        .call({ from: accounts[0] })
        .then((winner) => {
            document.getElementById("result").innerText = `${winner}`;
        });
}s
```

Name : Pratik Tiruwa
Seat No : 31031523035

3. voting.sol

```
// SPDX-License-Identifier: MIT
pragma solidity 0.8.19;

contract voting {
    mapping(string => uint256) public c;
    mapping(address => bool) public voters;
    string[] public cn;

    constructor() {
        cn = ["Can1", "Can2", "Can3"];
    }

    function vote(string memory caNm) public {
        require(!voters[msg.sender], "Already Voting Done.");
        bool ce = false;
        for (uint256 i = 0; i < cn.length; i++) {
            if (keccak256(bytes(caNm)) == keccak256(bytes(cn[i]))) {
                ce = true;
                break;
            }
        }
        require(ce, "Candidate does not exist.");
        c[caNm]++;
        voters[msg.sender] = true;
    }

    function getVoterC(string memory canM) public view returns (uint256) {
        return c[canM];
    }

    function getWinner() public view returns (string memory) {
        string memory winner;

        uint256 temp = 0;
```

```
        for (uint256 j = 0; j < cn.length; j++) {
            if (getVoterC(cn[j]) > temp) {
                temp = getVoterC(cn[j]);
                winner = cn[j];
            }
        }
        return winner;
    }

    function showPercentage(string memory canM) public view returns (uint256) {
        uint256 total;
        for (uint256 i = 0; i < cn.length; i++) {
            total = total + getVoterC(cn[i]);
        }

        uint256 per = getVoterC(canM) * (100 / total);
        return per;
    }
}
```

4. 1_deploy.js

```
const vote = artifacts.require("voting");

module.exports = async function (deployer) {
  await deployer.deploy(vote);
  const instance = await vote.deployed();
  console.log("Contract deployed at:", instance.address);
};
```

Name : Pratik Tiruwa
Seat No : 31031523035

5. Output: