Voting System

Votingsystem.sol

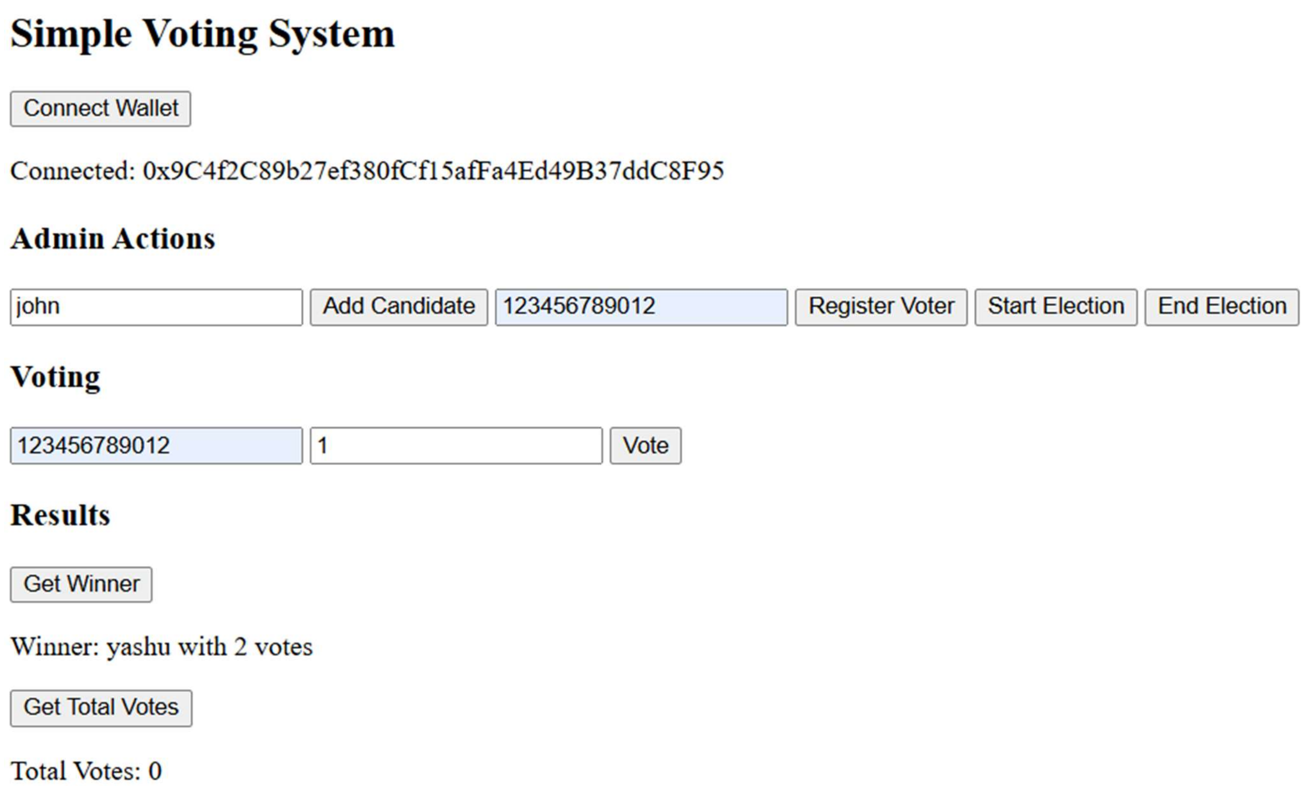
```solidity
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract SimpleVoting {
    struct Candidate {
        uint id;
        string name;
        uint voteCount;
    }

    address public admin;
    bool public electionOngoing;
    uint public candidateCount;
    uint public totalVotes;  // Count total votes cast

    mapping(uint => Candidate) public candidates;
    mapping(string => bool) public voters;  // Aadhaar number as unique ID
    mapping(string => bool) public hasVoted; // Track if Aadhaar holder has voted

    event ElectionStarted();
    event ElectionEnded();
    event Voted(string aadhaar, uint candidateId);
    event CandidateAdded(uint id, string name);
    event VoterRegistered(string aadhaar);

    modifier onlyAdmin() {
        require(msg.sender == admin, "Only admin can perform this action");
        _;
    }

    modifier electionActive() {
        require(electionOngoing, "Election is not active");
        _;
    }

    constructor() {
        admin = msg.sender;
        totalVotes = 0; // Initialize vote count to zero
    }

    function addCandidate(string memory _name) public onlyAdmin {
        candidateCount++;
        candidates[candidateCount] = Candidate(candidateCount, _name, 0);
        emit CandidateAdded(candidateCount, _name);
    }

    function registerVoter(string memory _aadhaar) public onlyAdmin {
        require(!voters[_aadhaar], "Voter already registered");
        voters[_aadhaar] = true;
        emit VoterRegistered(_aadhaar);
    }

    function startElection() public onlyAdmin {
        electionOngoing = true;
        emit ElectionStarted();
    }

    function endElection() public onlyAdmin {
        electionOngoing = false;
        emit ElectionEnded();
    }

    function vote(string memory _aadhaar, uint _candidateId) public electionActive {
        require(voters[_aadhaar], "You must be a registered voter");
        require(!hasVoted[_aadhaar], "You have already voted");
        require(candidates[_candidateId].id != 0, "Candidate does not exist");

        hasVoted[_aadhaar] = true;
        candidates[_candidateId].voteCount++;
        totalVotes++;  // Increase the total vote count

        emit Voted(_aadhaar, _candidateId);
    }

    function getWinner() public view returns (string memory winnerName, uint
winnerVoteCount) {
        require(!electionOngoing, "Election must be ended first");

        uint maxVotes = 0;
        uint winnerId = 0;

        for (uint i = 1; i <= candidateCount; i++) {
            if (candidates[i].voteCount > maxVotes) {
                maxVotes = candidates[i].voteCount;
                winnerId = i;
            }
        }

        return (candidates[winnerId].name, maxVotes);
    }

    function getTotalVotes() public view returns (uint) {
        return totalVotes; // Returns the total number of votes cast
    }
}
```

Votingsystem.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Simple Voting System</title>
    <script src="https://cdn.jsdelivr.net/npm/web3/dist/web3.min.js"></script>
</head>
<body>
    <h2>Simple Voting System</h2>

    <!-- Connect Wallet -->
    <button onclick="connectWallet()">Connect Wallet</button>
    <p id="wallet-address">Not connected</p>

    <h3>Admin Actions</h3>
    <input type="text" id="candidate-name" placeholder="Candidate Name">
    <button onclick="addCandidate()">Add Candidate</button>

    <input type="text" id="aadhaar-register" placeholder="Aadhaar Number">
    <button onclick="registerVoter()">Register Voter</button>

    <button onclick="startElection()">Start Election</button>
    <button onclick="endElection()">End Election</button>

    <h3>Voting</h3>
    <input type="text" id="aadhaar-vote" placeholder="Enter Aadhaar Number">
    <input type="number" id="candidate-id" placeholder="Candidate ID">
    <button onclick="vote()">Vote</button>

    <h3>Results</h3>
    <button onclick="getWinner()">Get Winner</button>
    <p id="winner-result">Winner: Not Yet Declared</p>

    <button onclick="getTotalVotes()">Get Total Votes</button>
    <p id="total-votes">Total Votes: 0</p>

    <script>
        let web3;
        let contract;
        const contractAddress = "0x777be2d41433395b29e14C6B6645Df4781d77952"; //
Replace with deployed contract address
        const abi = [
    {
        "inputs": [
            {
                "internalType": "string",
                "name": "_name",
                "type": "string"
            }
        ],
        "name": "addCandidate",
        "outputs": [],
        "stateMutability": "nonpayable",
        "type": "function"
    },
    {
        "inputs": [],
        "stateMutability": "nonpayable",
        "type": "constructor"
    },
    {
        "anonymous": false,
        "inputs": [
            {
                "indexed": false,
                "internalType": "uint256",
                "name": "id",
                "type": "uint256"
            },
            {
                "indexed": false,
                "internalType": "string",
                "name": "name",
                "type": "string"
            }
        ],
        "name": "CandidateAdded",
        "type": "event"
    },
    {
        "anonymous": false,
        "inputs": [],
        "name": "ElectionEnded",
        "type": "event"
    },
    {
        "anonymous": false,
        "inputs": [],
        "name": "ElectionStarted",
        "type": "event"
    },
    {
        "inputs": [],
        "name": "endElection",
        "outputs": [],
        "stateMutability": "nonpayable",
        "type": "function"
    },
    {
        "inputs": [
            {
                "internalType": "string",
                "name": "_aadhaar",
                "type": "string"
            }
        ],
        "name": "registerVoter",
        "outputs": [],
        "stateMutability": "nonpayable",
        "type": "function"
    },
    {
        "inputs": [],
        "name": "startElection",
        "outputs": [],
        "stateMutability": "nonpayable",
        "type": "function"
    },
    {
        "inputs": [
            {
                "internalType": "string",
                "name": "_aadhaar",
                "type": "string"
            },
            {
                "internalType": "uint256",
                "name": "_candidateId",
                "type": "uint256"
            }
        ],
        "name": "vote",
        "outputs": [],
        "stateMutability": "nonpayable",
        "type": "function"
    },
    {
        "anonymous": false,
        "inputs": [
            {
                "indexed": false,
                "internalType": "string",
                "name": "aadhaar",
                "type": "string"
            },
            {
                "indexed": false,
                "internalType": "uint256",
                "name": "candidateId",
                "type": "uint256"
            }
        ],
        "name": "Voted",
        "type": "event"
```

```json
    },
    {
        "anonymous": false,
        "inputs": [
            {
                "indexed": false,
                "internalType": "string",
                "name": "aadhaar",
                "type": "string"
            }
        ],
        "name": "VoterRegistered",
        "type": "event"
    },
    {
        "inputs": [],
        "name": "admin",
        "outputs": [
            {
                "internalType": "address",
                "name": "",
                "type": "address"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    },
    {
        "inputs": [],
        "name": "candidateCount",
        "outputs": [
            {
                "internalType": "uint256",
                "name": "",
                "type": "uint256"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    },
    {
        "inputs": [
            {
                "internalType": "uint256",
                "name": "",
                "type": "uint256"
            }
        ],
        "name": "candidates",
        "outputs": [
            {
                "internalType": "uint256",
                "name": "id",
                "type": "uint256"
            },
            {
                "internalType": "string",
                "name": "name",
                "type": "string"
            },
            {
                "internalType": "uint256",
                "name": "voteCount",
                "type": "uint256"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    },
    {
        "inputs": [],
        "name": "electionOngoing",
        "outputs": [
            {
                "internalType": "bool",
                "name": "",
                "type": "bool"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    },
    {
        "inputs": [],
        "name": "getTotalVotes",
        "outputs": [
            {
                "internalType": "uint256",
                "name": "",
                "type": "uint256"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    },
    {
        "inputs": [],
        "name": "getWinner",
        "outputs": [
            {
                "internalType": "string",
                "name": "winnerName",
                "type": "string"
            },
            {
                "internalType": "uint256",
                "name": "winnerVoteCount",
                "type": "uint256"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    },
    {
        "inputs": [
            {
                "internalType": "string",
                "name": "",
                "type": "string"
            }
        ],
        "name": "hasVoted",
        "outputs": [
            {
                "internalType": "bool",
                "name": "",
                "type": "bool"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    },
    {
        "inputs": [],
        "name": "totalVotes",
        "outputs": [
            {
                "internalType": "uint256",
                "name": "",
                "type": "uint256"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    },
    {
        "inputs": [
            {
                "internalType": "string",
                "name": "",
                "type": "string"
            }
        ],
        "name": "voters",
        "outputs": [
            {
                "internalType": "bool",
                "name": "",
                "type": "bool"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    }
];  // Replace with contract ABI

        async function connectWallet() {
            if (window.ethereum) {
                web3 = new Web3(window.ethereum);
                await window.ethereum.request({ method: "eth_requestAccounts" });
                const accounts = await web3.eth.getAccounts();
                document.getElementById("wallet-address").innerText = "Connected: " +
accounts[0];
                contract = new web3.eth.Contract(abi, contractAddress);
            } else {
                alert("MetaMask not detected!");
            }
        }

        async function addCandidate() {
            const name = document.getElementById("candidate-name").value;
            const accounts = await web3.eth.getAccounts();
            await contract.methods.addCandidate(name).send({ from: accounts[0] });
            alert("Candidate Added!");
        }

        async function registerVoter() {
            const aadhaar = document.getElementById("aadhaar-register").value;
            const accounts = await web3.eth.getAccounts();
            await contract.methods.registerVoter(aadhaar).send({ from: accounts[0] });
            alert("Voter Registered!");
        }

        async function startElection() {
            const accounts = await web3.eth.getAccounts();
            await contract.methods.startElection().send({ from: accounts[0] });
            alert("Election Started!");
        }

        async function endElection() {
            const accounts = await web3.eth.getAccounts();
            await contract.methods.endElection().send({ from: accounts[0] });
            alert("Election Ended!");
        }

        async function vote() {
            const aadhaar = document.getElementById("aadhaar-vote").value;
            const candidateId = document.getElementById("candidate-id").value;
            const accounts = await web3.eth.getAccounts();
            await contract.methods.vote(aadhaar, candidateId).send({ from: accounts[0]
});
            alert("Vote Casted!");
        }

        async function getWinner() {
            const result = await contract.methods.getWinner().call();
            document.getElementById("winner-result").innerText = `Winner: ${result[0]}
with ${result[1]} votes`;
        }

        async function getTotalVotes() {
            const votes = await contract.methods.getTotalVotes().call();
            document.getElementById("total-votes").innerText = "Total Votes: " + votes;
        }
    </script>
</body>
</html>
```

Output:



Simple Voting System

Connected: 0x9C4f2C89b27ef380fCf15afFa4Ed49B37ddC8F95

Admin Actions

john | 123456789012

Voting

123456789012 | 1

Results

Winner: yashu with 2 votes

Total Votes: 0