SOLIDITY SMART CONTRACT CODE-

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

contract VotingSystem {

    struct Candidate {
        string name;
        uint256 voteCount;
    }

    mapping(uint256 => Candidate) public candidates;
    mapping(address => bool) public hasVoted;

    uint256 public totalCandidates;

    event CandidateAdded(uint256 indexed id, string name);
    event Voted(address indexed voter, uint256 indexed candidateId);

    constructor() {
        totalCandidates = 0;
    }

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

    function castVote(uint256 _candidateId) public {
        require(_candidateId > 0 && _candidateId <= totalCandidates,
"Invalid candidate ID");
        require(!hasVoted[msg.sender], "You have already voted");

        candidates[_candidateId].voteCount++;
        hasVoted[msg.sender] = true;
        emit Voted(msg.sender, _candidateId);
    }

    function queryVotes(uint256 _candidateId) public view returns
(uint256) {
        require(_candidateId > 0 && _candidateId <= totalCandidates,
"Invalid candidate ID");

        return candidates[_candidateId].voteCount;
    }
}
```

SMART CONTRACT ADDRESS

0xb5245566cff4767e933836ced8a058f345db54389888c2a80f9fb7c8ce5c8a2f

UI INTERFACE CODEpragma solidity ^0.8.0;
// SPDX-License-Identifier: MIT
<!DOCTYPE html>

```
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Voting System</title>
   <style>
      body {
         font-family: Arial, sans-serif;
         margin: 0;
         padding: 0;
      }
      .container {
         max-width: 600px;
         margin: 20px auto;
         padding: 0 20px;
      }
      .candidate {
         margin-bottom: 10px;
      }
   </style>
</head>
<body>
   <div class="container">
      <h1>Voting System</h1>
      <div id="candidates"></div>
   </div>

   <script src="https://cdn.jsdelivr.net/npm/ethereumjs-util@6.1.0/dist/ethereumjs-
util.min.js"></script>
   <script src="https://cdn.jsdelivr.net/npm/web3@1.5.3/dist/web3.min.js"></script>
   <script>
      const contractAddress = '<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Voting System</title>
   <style>
      body {
```

```
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
        }
        .container {
            max-width: 600px;
            margin: 20px auto;
            padding: 0 20px;
        }
        .candidate {
            margin-bottom: 10px;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Voting System</h1>
        <div id="candidates"></div>
    </div>

    <script src="https://cdn.jsdelivr.net/npm/ethereumjs-util@6.1.0/dist/ethereumjs-
util.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/web3@1.5.3/dist/web3.min.js"></script>
    <script>
        const contractAddress = '<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Voting System</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
        }
        .container {
            max-width: 600px;
            margin: 20px auto;
            padding: 0 20px;
        }
        .candidate {
            margin-bottom: 10px;
        }
    </style>
</head>
<body>
    <div class="container">
```

```
    <h1>Voting System</h1>
    <div id="candidates"></div>
  </div>

  <script src="https://cdn.jsdelivr.net/npm/ethereumjs-util@6.1.0/dist/ethereumjs-
util.min.js"></script>
  <script src="https://cdn.jsdelivr.net/npm/web3@1.5.3/dist/web3.min.js"></script>
  <script>
    const contractAddress =
'0xb5245566cff4767e933836ced8a058f345db54389888c2a80f9fb7c8ce5c8a2f';
    const contractABI = [
      {
        "constant": true,
        "inputs": [
          {
            "name": "_candidateId",
            "type": "uint256"
          }
        ],
        "name": "queryVotes",
        "outputs": [
          {
            "name": "",
            "type": "uint256"
          }
        ],
        "payable": false,
        "stateMutability": "view",
        "type": "function"
      }
    ];

    window.addEventListener('load', async () => {
      if (window.ethereum) {
        window.web3 = new Web3(window.ethereum);
        await window.ethereum.enable();
      } else if (window.web3) {
        window.web3 = new Web3(window.web3.currentProvider);
      } else {
        console.log('Non-Ethereum browser detected. You should consider trying MetaMask!');
        return;
      }

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

      displayCandidates(contract);
    });
```

```
    async function displayCandidates(contract) {
      const candidatesElement = document.getElementById('candidates');
      const totalCandidates = await contract.methods.totalCandidates().call();

      for (let i = 1; i <= totalCandidates; i++) {
        const candidateName = await contract.methods.candidates(i).call();
        const voteCount = await contract.methods.queryVotes(i).call();
        const candidateElement = document.createElement('div');
        candidateElement.classList.add('candidate');
        candidateElement.innerHTML = `
          <strong>${candidateName}</strong> - Votes: ${voteCount}
          <button onclick="castVote(${i})">Vote</button>
        `;
        candidatesElement.appendChild(candidateElement);
      }
    }

    async function castVote(candidateId) {
      const accounts = await web3.eth.getAccounts();
      const hasVoted = await contract.methods.hasVoted(accounts[0]).call();
      if (!hasVoted) {
        contract.methods.castVote(candidateId).send({ from: accounts[0] })
          .on('transactionHash', () => {
            alert('Vote successful!');
            window.location.reload();
          })
          .on('error', (error) => {
            alert('Vote failed: ' + error.message);
          });
      } else {
        alert('You have already voted!');
      }
    }
  </script>
</body>
</html>
';
    const contractABI = [
      {
        "constant": true,
        "inputs": [
          {
            "name": "_candidateId",
            "type": "uint256"
          }
        ],
        "name": "queryVotes",
        "outputs": [
```

```
        {
          "name": "",
          "type": "uint256"
        }
      ],
      "payable": false,
      "stateMutability": "view",
      "type": "function"
    }
];

window.addEventListener('load', async () => {
    if (window.ethereum) {
        window.web3 = new Web3(window.ethereum);
        await window.ethereum.enable();
    } else if (window.web3) {
        window.web3 = new Web3(window.web3.currentProvider);
    } else {
        console.log('Non-Ethereum browser detected. You should consider trying MetaMask!');
        return;
    }

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

    displayCandidates(contract);
});

async function displayCandidates(contract) {
    const candidatesElement = document.getElementById('candidates');
    const totalCandidates = await contract.methods.totalCandidates().call();

    for (let i = 1; i <= totalCandidates; i++) {
        const candidateName = await contract.methods.candidates(i).call();
        const voteCount = await contract.methods.queryVotes(i).call();
        const candidateElement = document.createElement('div');
        candidateElement.classList.add('candidate');
        candidateElement.innerHTML = `
            <strong>${candidateName}</strong> - Votes: ${voteCount}
            <button onclick="castVote(${i})">Vote</button>
        `;
        candidatesElement.appendChild(candidateElement);
    }
}

async function castVote(candidateId) {
    const accounts = await web3.eth.getAccounts();
    const hasVoted = await contract.methods.hasVoted(accounts[0]).call();
    if (!hasVoted) {
```

```
            contract.methods.castVote(candidateId).send({ from: accounts[0] })
              .on('transactionHash', () => {
                alert('Vote successful!');
                window.location.reload();
              })
              .on('error', (error) => {
                alert('Vote failed: ' + error.message);
              });
          } else {
            alert('You have already voted!');
          }
        }
      </script>
    </body>
    </html>
    ';
        const contractABI = [
          {
            "constant": true,
            "inputs": [
              {
                "name": "_candidateId",
                "type": "uint256"
              }
            ],
            "name": "queryVotes",
            "outputs": [
              {
                "name": "",
                "type": "uint256"
              }
            ],
            "payable": false,
            "stateMutability": "view",
            "type": "function"
          }
        ];

        window.addEventListener('load', async () => {
          if (window.ethereum) {
            window.web3 = new Web3(window.ethereum);
            await window.ethereum.enable();
          } else if (window.web3) {
            window.web3 = new Web3(window.web3.currentProvider);
          } else {
            console.log('Non-Ethereum browser detected. You should consider trying MetaMask!');
            return;
          }
```

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

        displayCandidates(contract);
    });

    async function displayCandidates(contract) {
        const candidatesElement = document.getElementById('candidates');
        const totalCandidates = await contract.methods.totalCandidates().call();

        for (let i = 1; i <= totalCandidates; i++) {
            const candidateName = await contract.methods.candidates(i).call();
            const voteCount = await contract.methods.queryVotes(i).call();
            const candidateElement = document.createElement('div');
            candidateElement.classList.add('candidate');
            candidateElement.innerHTML = `
                <strong>${candidateName}</strong> - Votes: ${voteCount}
                <button onclick="castVote(${i})">Vote</button>
            `;
            candidatesElement.appendChild(candidateElement);
        }
    }

    async function castVote(candidateId) {
        const accounts = await web3.eth.getAccounts();
        const hasVoted = await contract.methods.hasVoted(accounts[0]).call();
        if (!hasVoted) {
            contract.methods.castVote(candidateId).send({ from: accounts[0] })
                .on('transactionHash', () => {
                    alert('Vote successful!');
                    window.location.reload();
                })
                .on('error', (error) => {
                    alert('Vote failed: ' + error.message);
                });
        } else {
            alert('You have already voted!');
        }
    }
  </script>
</body>
</html>
```

SMART CONTRACT ADDRESS FOR THE UI INTERFACE
0xfe8bc019acd9fab9f3c3ff7f7bc1b54e5b227400c2ea523276c49996529f258d