

ECO5073S- Fintech & Cryptocurrencies

Mmbi Khuliso MMBKHU001

MMMMMMM

MM

MMBKHU001

Git link: <https://github.com/Mukandangalo/Crypto-Tutotials>

## Practical 1: Blockchain enabled Voting system

The aim of this practical was to develop a blockchain-based voting system for the upcoming UCT student body council elections.

Requirements:

* Use Algorand smart contracts
* Voters should be able to submit their votes and view results in real time
* Algorand address should only be used once

Assumptions:

* There are 5 candidates participating in the election
* Maximum votes are 20 votes

Blockchain system

**Step 1**

Setting up the voting system. Import pyteal which is a library that enables developers to Algorand smart contracts using a python syntax. The other part defines how the votes will be managed and allows developers to keep track of the votes.

|  |
| --- |
| from pyteal import \*  def approval\_program():      # Keys for the global state      candidate1\_key = Bytes("Candidate1")      candidate2\_key = Bytes("Candidate2")      candidate3\_key = Bytes("Candidate3")      candidate4\_key = Bytes("Candidate4")      candidate5\_key = Bytes("Candidate5") |

**Step 2**

This part defines the maximum number of votes allowed for the elections and also initialize each candidate’s vote count to 0.

|  |
| --- |
| #number of votes      max\_votes = Int(20)      # Initialize the vote count for each candidate to 0      initialization = Seq([          App.globalPut(candidate1\_key, Int(0)),          App.globalPut(candidate2\_key, Int(0)),          App.globalPut(candidate3\_key, Int(0)),          App.globalPut(candidate4\_key, Int(0)),          App.globalPut(candidate5\_key, Int(0)),          Return(Int(1))      ]) |

**Step 3**

This part ensures that each vote casted by the voter matches the candidate keys in part 1 and increase their vote count by +1 for each vote.

|  |
| --- |
| handle\_vote = Seq([          Assert(Or(              Txn.application\_args[0] == candidate1\_key,              Txn.application\_args[0] == candidate2\_key,              Txn.application\_args[0] == candidate3\_key,              Txn.application\_args[0] == candidate4\_key,              Txn.application\_args[0] == candidate5\_key          )),          App.globalPut(              Txn.application\_args[0],              Add(App.globalGet(Txn.application\_args[0]), Int(1))          ),          Assert(Lt(              App.globalGet(Txn.application\_args[0]),              max\_votes          )),          Return(Int(1))      ]) |

**Step 4**

Step 4 sets up the program's structure. it starts by initializing the vote counts for candidates when the contract begins. Then, whenever someone votes, it updates the vote counts. Lastly, the code turns our easy-to-read PyTeal code into the language the Algorand blockchain understands.

|  |
| --- |
| program = Cond(          [Txn.application\_id() == Int(0), initialization],          [Txn.application\_id() != Int(0), handle\_vote]      )      return program  teal\_code = compileTeal(approval\_program(), mode=Mode.Application, version=3)  print(teal\_code) |

## Challenge - Web based blockchain voting application

**Web application**

|  |
| --- |
|  |

**Code used**

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">    <head>      <meta charset="UTF-8" />      <meta name="viewport" content="width=device-width, initial-scale=1.0" />      <title>University Election Voting System</title>      <style>        .container {          max-width: 800px;          margin: 0 auto;          padding: 20px;        }        h1 {          text-align: center;        }        #voting-form {          margin-bottom: 20px;        }        #voting-form label {          font-weight: bold;        }        #voting-form select {          margin-right: 10px;        }        #voting-results {          border-top: 2px solid #ccc;          padding-top: 20px;        }        #results-list {          list-style-type: none;          padding: 0;        }        #results-list li {          margin-bottom: 10px;        }      </style>    </head>    <body>      <div class="container">        <h1>University Election Voting System</h1>        <div id="voting-form">          <label for="candidate-select">Choose Candidate:</label>          <select id="candidate-select">            <option value="0">Candidate A</option>            <option value="1">Candidate B</option>            <option value="2">Candidate C</option>            <option value="3">Candidate D</option>            <option value="4">Candidate E</option>          </select>          <button onclick="submitVote()">Submit Vote</button>        </div>        <div id="voting-results">          <h2>Current Voting Results</h2>          <ul id="results-list">          </ul>        </div>      </div>      <script>        // Function to submit a vote        function submitVote() {          var candidateSelect = document.getElementById("candidate-select");          var selectedCandidate = candidateSelect.value;          console.log("Vote submitted for Candidate " + selectedCandidate);        }        // Function to fetch and display voting results        function displayResults() {          var dummyResults = [            { candidate: "Candidate A", votes: 2 },            { candidate: "Candidate B", votes: 2 },            { candidate: "Candidate C", votes: 5 },            { candidate: "Candidate D", votes: 6 },            { candidate: "Candidate E", votes: 3 },          ];          var resultsList = document.getElementById("results-list");          resultsList.innerHTML = "";          dummyResults.forEach((result) => {            var listItem = document.createElement("li");            listItem.textContent = result.candidate + ": " + result.votes;            resultsList.appendChild(listItem);          });        }        // Display voting results when the page loads        displayResults();      </script>    </body>  </html> |