**README.md FILE**

**Project Description:**

* Smart Contract – Create a Blockchain Voting System
* The online voting system must ensure the following:
  + Correct assignment of the voting rights
  + Automatic vote counting
  + Transparent process
* One contract should be created for each vote.
* The contract creator should provide the voting rights to each address in order their owners to use their votes.
* There is an extra rule for whole process, the votes will be weighted. That means that all the votes are not equal. The contract creator will assign to each voter a priority number pNo and when the voter casts her vote, this vote will appear pNo times in the ballot box.
* There is the freedom for any assumptions to be made regarding the voting system.

**Project Designing – Programming – Deployment**

**Phase 1: Designing of the Blockchain Online Voting System project**

To create the electronic Blockchain Voting System asked in the description of the project the Solidity programming language was used to implement the whole procedure as a Smart Contract on Ethereum.

Solidity is an object-oriented programming language for writing smart contracts. Its is used for implementing smart contracts on various blockchain platforms, mostly, Ethereum.

**Basic Terminology:**

* **Blockchain**: It is a structure of recording information in a manner that is impossible for the Confidentiality, Integrity and Availability of the data to be changed, hacked, or undermined in any possible way. Typically, this structure is referred to as a “digital ledger” which is consisted of blocks stored in several databased called the “chain” that is connected in a peer-to-peer nodes network.
* **Ethereum “smart contract”**: It is a project written in a blockchain high-level programming language (in our case Solidity) which includes functions (code) and state (data) and resides for a specific address on the Ethereum blockchain.
* **Ether (ETH)**: It is the native cryptocurrency of the Ethereum blockchain platform.
* **Gas**: It the necessary amount of Ether (ETH) that will be used as the fee for a specific transaction on Ethereum to be successfully conducted.
* **Solidity**: It is an object-oriented, high-level language for implementing smart contracts, that is designed to target the Ethereum Virtual Machine (EVM).
* **Modifiers**: Can be used with Solidity programming language to change the behaviour of functions in a declarative way. For example, you can use a modifier to automatically check a condition prior to executing the function. Modifiers are inheritable properties of contracts and may be overridden by derived contracts, but only if they are marked virtual.
* **Functions**: Can be used with Solidity programming language as executable units of code. Functions are usually defined inside a contract, but they can also be defined outside of contracts. Function Calls can happen internally or externally and have different levels of visibility towards other contracts. Functions accept parameters and return variables to pass parameters and values between them.
* **Remix IDE**: It is an open-source web or desktop tool that helps developers to write Solidity contracts straight from the browser. Remix-IDE is available at [remix.ethereum.org](file:///C:\Users\chrsm\Desktop\Smart%20Voting%20System%20with%20Solidity\remix.ethereum.org).

**Requirements:**

The Smart Contract Blockchain Voting System must ensure the following:

* The assignment of the voting rights should be guaranteed.
* Each candidate’s votes should be automatically counted.
* The whole process should be fluent and transparent.
* One contract should be created for each vote.
* Voting rights are provided to each address to be used by their owners.
* The contract creator will assign to each voter a priority number pNo and when the voter casts his/her vote, this vote will appear pNo times in the ballot box.

**Assumptions:**

The following assumptions have been made in advance to settle the base for the designing of the blockchain voting system Solidity code (Specific details regarding the Solidity source code schema will be explained in the relevant Schema section):

* The electronic Blockchain Voting System is created to be used for the upcoming municipal elections and only for the municipality of Samos. This Voting System is going to be used purely for pilot testing to be fully integrated into future electoral processes.
* There is one Election Authority which is responsible for the creation of the Ethereum voting Smart Contract. The same Election Authority is also the only responsible for the following:
  + Starting the election process and defining its duration
  + Registering the candidates of the election.
  + Register the Voters that will vote for each Candidate and assign a Priority Number for each of them.
* Only registered voters can vote for a candidate.
* Each Voter can vote only once for each Candidate.
* Any vote before the start datetime and after end datetime will not be accepted by the Smart Contract.
* Election result can be audited in real time in the Blockchain.
* Any vote to an unregistered candidate will not be valid.
* Everything from the registrations to the voting and counting the results is open to audition by anyone, for the transparency of the whole process to be guaranteed.
* The whole electronic voting system project will be written, compiled, and deployed straight from the browser with the use of [Remix IDE](file:///C:\Users\chrsm\Desktop\Smart%20Voting%20System%20with%20Solidity\remix.ethereum.org).

**Actors:**

The main actors of the whole Ethereum schema have as follows:

* Election Authority (only one)
* Candidates (as many as possible)
* Voters (as many as possible)

**Phase 2: Creating the Ethereum Online Voting System Solidity Programming Schema**

* Before writing the code for the modifiers and functions, the first step was to declare all the necessary to the project variables and headers, as follows:
  + Very important for the smooth compilation of the contract is the declaration of the **license** under which the manipulation of the project will reside and the **version** **pragma** of the **compiler**.
  + The preferred **licences** and **compiler’s** **version** declaration follows the statements of the declaration for the necessary to the **functions** of the voting system **variables**. Very important in this stage was the declaration of the three **mapping** **references**.

**Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα**

Figure - Initial Variables/Headers/Mappings Declarations

* + As we can see in Figure 1 four **mapping** **references** with their explanation and usage referenced inside each line of code.
* The **constructor()** **public** of the project was created for the **electionAuthority** state variable of the contract.

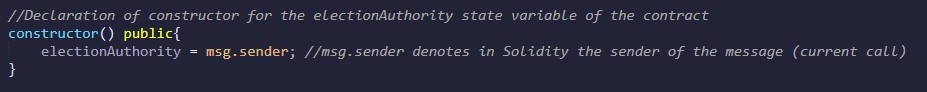
****

Figure - constructor() public Declaration

* The next step was to create the code for the **modifier** **conditional** **functions** of the **Ethereum** voting system. The details of each modifier’s usability are included as references inside the code.

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

Figure - modifier() conditional functions

* **Basic Schema explanation**:
  + The Election Authority is responsible for the publishing of the Smart Contract’s code on the Ethereum Blockchain. The same authority publicly announces the contract’s address.

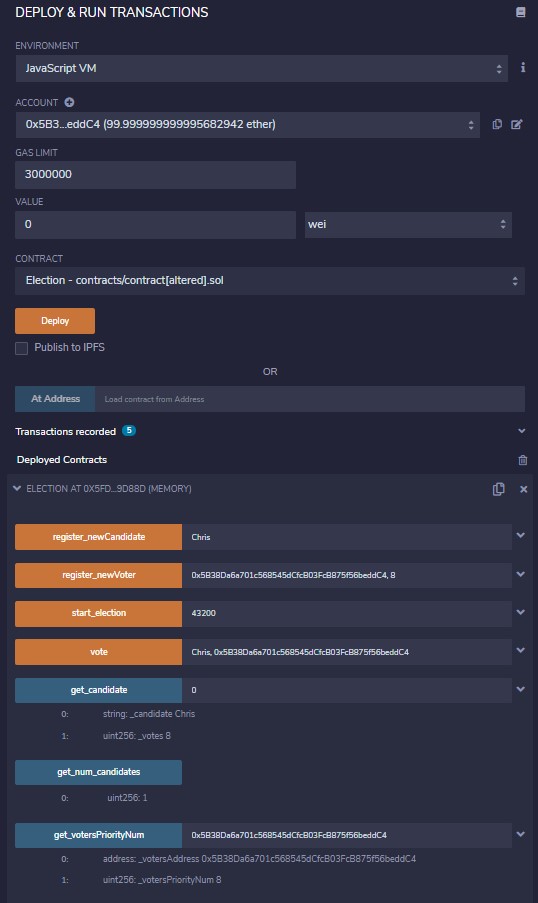


Figure - Election Authority Publishes the Smart Contract on Ethereum Blockchain

* + Only the Election Authority with the use of the **function register\_newCandidate (string memory id) public**, records the ids of the candidates. All Candidate unique names are publicly listed in the Blockchain using the same function. Election Authority also stores and publicly announces a record of association between each candidate ID and their provided name.
  + The Election Authority with the use of the **function register\_newVoter(address addr, uint pNo) public**, records the ids and Ethereum addresses of the voters. All Voters unique Ethereum addresses are publicly listed in the Blockchain using the same function. As opposed to the registration of new Candidates, the Election Authority stores the IDs to prevent double registration but does not keep a record of association between the IDs and the provided Ethereum addresses. That is enough to guarantee the anonymity of the citizen’s voting right.

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

Figure - register\_newCandidate and register\_newVoter Solidity Functions

* + Election starts when Election Authority sets the duration of the election using the **function start\_election(uint duration) public**. Very important in this step is that the duration of the election should be set to **43200** in **Linux** **Epoch** **time**. In **real** **lifetime** it is **12 hours or 720** **seconds**.

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

Figure - function start\_election(uint duration)

* + Voters vote for their preferred candidate's id with the use of their Ethereum address and the whole process is publicly listed in the Blockchain using the function **function vote (string memory id, address addr) public**. Very important for the whole procedure is to make clear that no record is kept for association between the voter’s IDs and the provided Ethereum addresses.

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

Figure - function vote (string memory id, address addr) public

* + Finally, three more functions have been created, as follows:
    - **function get\_num\_candidates() public view returns(uint):** Returns the total number of candidates that take part to the elections.
    - **function get\_candidate(uint i) public**

**view returns(string memory \_candidate, uint \_votes):** Returns the id and number of votes of each candidate.

* + - **function get\_votersPriorityNum(address addr) public**

**view returns(address \_votersAddress, uint \_votersPriorityNum):** Returns the id and number of votes of each candidate.

**Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα**

Figure - Smart Contract Voting System get() functions