

BLOCKCHAIN PROOF-OF-CONCEPT

PROPOSAL

Blockchain Development individual assignment.

Voting

Prepared by: Luka Lazarashvili

Due date: 25 December, 18:00 Tbilisi time.

Dear Mr Alexander,

We would like to present a smart contracts-based blockchain solution to the discussed use-case. Please get acquainted with the proposed prototype to understand the structure and process of our solution below.

SUMMARY

A voting dapp application that runs on a blockchain and allows users to participate in voting or decision-making processes. Some possible features of a voting dapp might include:

- **Secure and transparent voting:** Because the dapp runs on a blockchain, all votes are recorded in an immutable ledger, making it difficult for anyone to tamper with the results. This can help to ensure the integrity of the voting process.
- **Anonymity:** We can encrypt users and they can vote anonymously, which can help to protect their privacy and prevent vote-buying or other forms of coercion.
- **Accessibility:** A voting dapp can be accessed from any device with an internet connection, making it easier for people to participate in the voting process regardless of their location.
- **Efficient decision-making:** Because the dapp is decentralized, it can be used to facilitate decision-making processes in a variety of contexts, including corporate governance, community voting, and even political elections.
- **Customization:** A voting dapp can be customized to fit the specific needs and requirements of a particular organization or community. For example, it could be configured to allow only certain groups of users to vote, or to require a certain percentage of votes to pass a particular measure.

	<code>contract Voting { // voting contract for register as voters and vote</code>
--	---

```

address public owner; // owner address

mapping(address => Voter) voters; // all registered voters

Voter[] votersArr; // all registered voters (array for get length)

bool votingStatus; // to check if voter vote

Candidates candidatesContract; //

event ChangeVotingStatus(); //event on votinnng status change

event SCFeedback(); //event on scfeedback


modifier isOwner() // to check owner

modifier is18() // to check if voter is 18+

modifier isZeroBalance() // to check user balance to prevent fake accaunts spam

modifier isVoterNew() // to check if user is registered

modifier hasVote() // to check if voter has left vote

modifier isVotingTrue // to check if voting is on

modifier isVotingFalse // ti check if voting is of


constructor() {

    owner = msg.sender; //find owner address

}


function addCandidatesContract(Candidates _candidatesContract) public
isOwner // add candidate contract by owner

function viewAllVoters() public view returns (uint, Voter[] memory) //
returns all voter

function checkVotingProcess() public view returns (bool) // returns
voting status

function startVotingProcess() external isOwner isVotingFalse // owner
starts voting process and emit voting status event

function stopVotingProcess() external isOwner isVotingTrue // owner
stops voting process and emit voting status event

function vote(uint8 _participantNumber) external
isVoterRegistered( msg.sender) isVotingTrue hasVote(msg.sender) // check

```

voter vote and voting status if success voter can vot

```
function registerAsVoter(string calldata _fullname,string calldata
_ideticalNumber,uint8 _age) external isVoterNew(msg.sender) is18(_age)
isZeroBalance // if user is 18+, don't have vote account and have balance
can register as voter
```

```
function donateToContractOwner() public payable //function to donate owner
some eths
```

```
receive() external payable() // to catch unespected ethereums
}
```

```
contract Candidates{ //Candidates contract to manage candidates by owner
```

```
    address public owner; //owner address
```

```
    mapping(uint8=> Candidate) candidates; // all registered candidates
```

```
    Candidate[] candidatesArr; //all registered candidates (array for get length)
```

```
    uint8 maxCandidates = 10; // max candidate number to prevent lots of gas usage from array
iterations
```

```
    address votingAddress; // address from voting contract
```

```
    event CalculateVotes(Candidate[]); // emit when calculate votes
```

```
    event WinnerCandidate(Candidate); // emit when find winnner
```

```
    modifier checkMaxCandidates() // to check maximum candidates from array length
```

```
    modifier isOwner() // to check owner
```

```
    modifier isSCSafe() // to add securite level, check votins address
```

```
function addVotingContract(address _votingAddress) public isOwner // owner must add voting
contract address when deploy contract to check if votes is validate
```

```
function showAllCanidates() public view returns (uint, Candidate[] memory) // everyone can see
all candidates
```

```
function updateCandidateVote(uint8 _participantNumber) external isSCSafe // +1 vote on
candidate if vote is from known contract
```

```

    function addCandidate(uint8 _participantNumber, uint256 _identicalNumber, string calldata
    _fullname, string calldata _slogan) external isOwner checkMaxCandidates // only owner can add
    candidate

    function deleteCandidate(uint8 index) external isOwner // only owner can delete candidate

    function calculateVotes() external isOwner // owner can invoke function that calculating all vote
    and finding winner

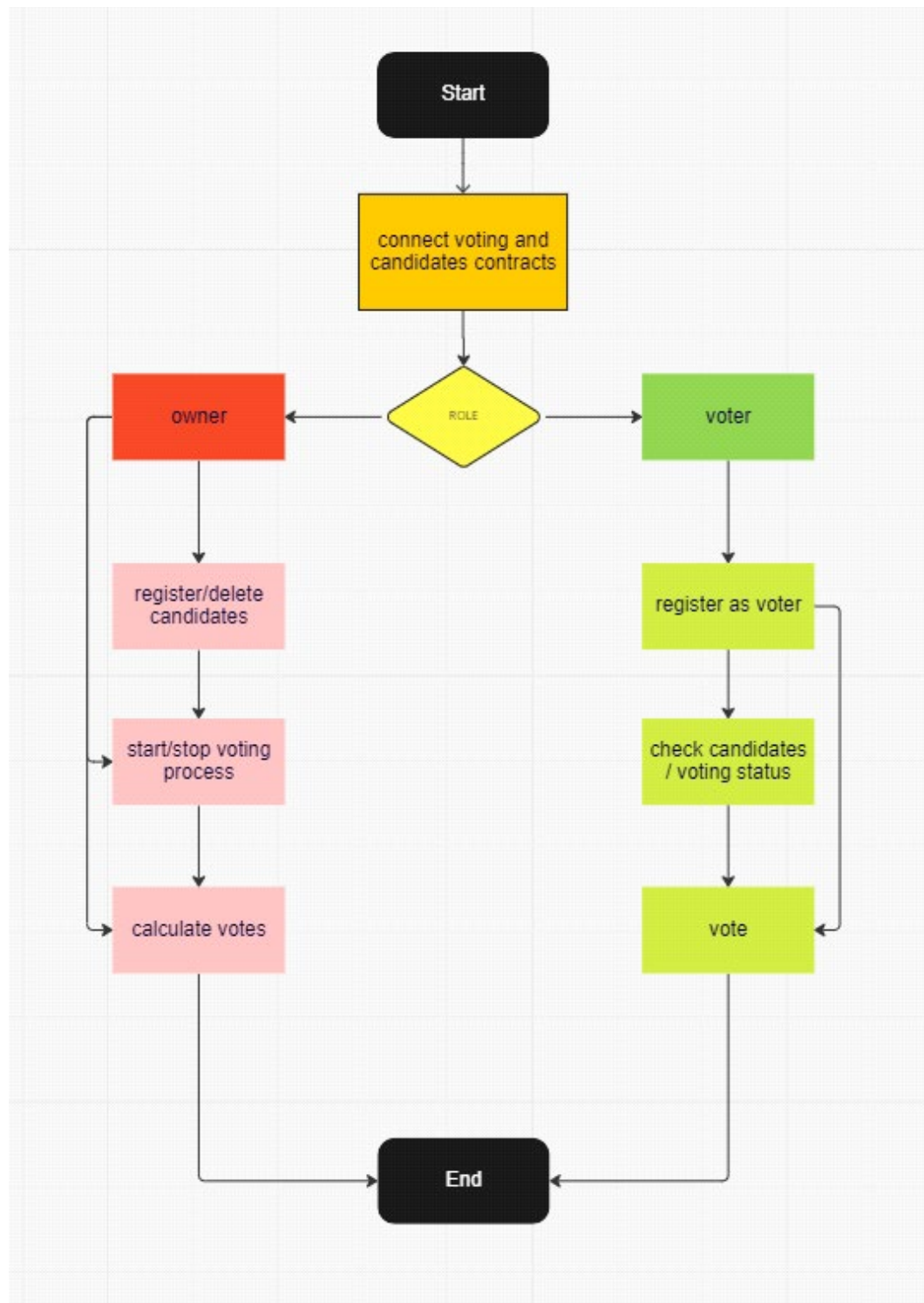
    receive() external payable() // to catch unsuspected ethereums
}

```

Smart contracts explanation

- Voting contract is used to register users as voters and vote candidate when owner starts voting process
- Candidates contract is used to add or delete candidates from owner, to calculate votes and find winner, all can check candidates list.

Smart contracts process flow



owner deploy voting and candidates contracts and connect to each other

adding or deleting candidates in candidates contract (max 10 candidate), and start voting process in voting contract

users can register as voter in voting contract

after owner start voting process, voters(users) can vote to only one candidate

owner can stop and calculate result any time

EXPECTED BUSINESS RESULTS

Overall, a voting dapp has the potential to revolutionize the way decisions are made in a variety of contexts, by providing a secure, transparent, and efficient platform for voting and decision-making.

Luka Lazarashvili