Deadline: 12/07/2022

Task Voting

**Description:**

The smart-contract designed to create a voting for a number of candidate(candidates are added by admin at the moment of creation of voting). Contract should accept votes and finalize voting, after admin created it. To participate in voting user must send 0.01 ETH along with a vote() function call. The winner of the vote takes 90% of accumulated ETH, and other 10% goes to Owner of the contract as a fee.

**Tasks:**

Create a smart-contract, please think thoroughly of the naming in your code.

Cover your contract with set of unit-test. Use hardhat plugin "solidity-coverage".

Create a deployment script and deploy contract to any evm-compatible test network.

Create tasks to call the deployed contract.

**Functionality:**

only the owner can create a voting

a voting lasts 3 days

any user can vote for any candidate by sending 0.01 ETH. Users can participate in voting only once.

after a voting has ended anyone should be able to finish it. Winner selection and reward distribution should be done within a single function call.

All ETH will be sent to the winner minus a 10% fee. These 10% remain on the platform

10% from every voting stay on the contract as a fee, until Owner withdraws them. Only owner should be able to withdraw fees.

information about any voting should be quarriable.

Develope unit tests (js or ts)

use solidity-coverage plugin to get full coverage

develope a deployment script to the rinkeby test network

create "task" scripts to call functions from the testnet

**Check-list:**

create project with the hardhat framework.

hardhat.config.js(ts) is configured correctly, hardhat has detailed documentation for each setting.

all private data (private key, mnemonic, Infura access keys, Alchemy, etc.) is saved in a file with extension \*.env, which shouldn't be pushed to git.

make sure all \*.sol files are stored at "~/contracts" directory.

the contract has a function to create a voting.

the contract has a commission withdrawal function.

the contract has a function to withdraw the owner's fees.

the contract has a function to finish a voting.

the contract has additional view functions for fetching information about voting and participants.

the solidity-coverage plugin is installed and configured in the project.

the "~/test" directory contains unit-tests that provide 100% contract coverage for all indicators (statements, branch, functions, lines).

the project is published for all users on github/gitlab/bitbucket.

the "~/scripts" directory contains a script for publishing the contract to a testnet.

the "~/tasks" directory contains hardhat task for every possible interaction with the deployed contract (add a vote, participate in a vote, end a vote, etc.).

**Note:**

contract supports several votings at the same time.

try to use optimal algorithms with O(1) complexity

**Materials:**

This article can answer some questions. (link)

Building Decentralized Blockchain Applications. Narayan Prusty (link)

Ethereum development environment for professionals (link)

A library for secure smart contract development (link)

Solidity docs (link)

**FYI**

The project is important to upload to github/gitlab/bitbucket

Create an account for the proposed services if you don't have one yet.

Create a public repository

Link your public repo in the answer.👇

