CMPE 483 Blockchain Programming

Homework-1, Fall 2022

(This homework can be done in groups of at most 3 students)

(due Nov. 28th)

In this project, you will implement an autonomous decentralized governance token contract called MyGov. Governance token allows crowds to collectively manage a decentralized autonomous organization (DAO). The following links provide information about governance tokens:

- https://academy.binance.com/en/articles/what-are-governance-tokens
- https://www.coindesk.com/learn/what-is-a-governance-token/

A list of major Governance Tokens is available here:

- https://www.coingecko.com/en/categories/governance

You can use OpenZeppelin ERC20 token contract to implement the governance token. OpenZeppelin contracts are available at:

https://github.com/OpenZeppelin/openzeppelin-contracts/tree/master/contracts/token/ERC20

Your governance token contract will provide the following functionality:

1. All standard ERC20 functions should be provided by MyGov. Additionally, interface functions given in Table 1 should be implemented and provided.
2. MyGov will allow provide three transactional activities (i) Donation to MyGov (ii) Survey service and (iii) Project Proposal, Voting and Funding.
3. Anyone who owns at least 1 MyGov token is MyGov member.
4. Members can carry out activities (i) (ii), (iii).
5. Anyone can carry out activity (i), i.e. you do not need to be a member to carry out (i).
6. Anyone can call get information functions. View functions are open to everyone, i.e. you do not need to be a member to get information.
7. Submitting Project Proposal costs 5 MyGov tokens and 0.1 Ether.
8. Submittting Survey costs 2 MyGov tokens and 0.04 Ether.
9. MyGov token supply is 10 million (fixed).
10. Donations can be accepted in ethers and MyGov tokens only. Ethers can be granted to winning Project proposals. MyGov tokens are given away via faucet function.
11. MyGov tokens are distributed via a faucet function. Faucet gives 1 token to an address. If the address obtained a token before, it cannot get token from faucet any longer.
12. Each member can give 1 vote for each proposal. Members can delegate vote. Members who voted or delegated vote cannot reduce their MyGov balance to zero until the voting deadlines.
13. In order to get a Project proposal funded, at least 1/10 of the members must vote yes AND there should be sufficient ether amount in the MyGov contract. Project proposer must call reserveProjectGrant function in order to reserve the funding by the proposal deadline. If the project proposer does not reserve by the deadline, funding is lost. Also, if there is not sufficient ether in MyGov contract when trying to reserve, funding is lost. In order to receive payments according to the schedule, at least 1/100 of the members should vote yes. If at least 1/100 do not vote yes, then the current as well as the remaining payments are terminated, i.e. project is no longer funded.

The Governance Token implementation should provide the following interfaces (functions) to the external world (in addition to the standard ERC20 functions). You can also implement additional functions as you like.

Table 1
constructor(uint tokensupply) function delegateVoteTo(address memberaddr,uint projectid) public function donateEther() public payable function donateMyGovToken(uint amount) public function voteForProjectProposal(uint projectid,bool choice) public function voteForProjectPayment(uint projectid,bool choice) public function submitProjectProposal(string ipfshash, uint votedeadline,uint [] paymentamounts, payable uint [] payschedule) public returns (uint projectid) function submitSurvey(string ipfshash,uint surveydeadline,uint numchoices, uint atmostchoice) payable public returns (uint surveyid) function takeSurvey(uint surveyid,uint [] choices) public function reserveProjectGrant(uint projectid) public function withdrawProjectPayment(uint projectid) public function getSurveyResults(uint surveyid) public view returns(uint numtaken, uint [] results) function getSurveyInfo(uint surveyid) public view returns(string ipfshash, uint surveydeadline,uint numchoices, uint atmostchoice) function getSurveyOwner(uint surveyid) public view returns(address surveyowner) function getIsProjectFunded(uint projectid) public view returns(bool funded) function getProjectNextPayment(uint projectid) public view returns(int next) function getProjectOwner(uint projectid) public view returns(address projectowner) function getProjectInfo(uint activityid) public view returns(string ipfshash, uint votedeadline,uint [] paymentamounts, uint [] payschedule) function getNoOfProjectProposals() public view returns(uint numproposals) function getNoOfFundedProjects () public view returns(uint numfunded) function getEtherReceivedByProject (uint projectid) public view returns(uint amount) function getNoOfSurveys() public view returns(uint numsurveys)

function faucet()

.

Grading

Your project will be graded according to the following criteria:

Documentation (written document describing how you implemented your project and also showing the correctness of your implementation). You should also provide average gas usages for the interface functions.	30%
Comments in your code	10%
Correctly functioning Solidity code, test scripts, and tests.	60%

Homework Submission:

Please submit your project to Moodle. Before you submit your project, please timestamp (notarize) your project zip file at https://certify.bloxberg.org/ Do NOT lose the certification and the submitted project zip file. The certification is a proof that your project zip file existed during the time of submission.

Please also answer the following questions and submit it with your project.

Task Achievement Table	Yes	Partially	No
I have prepared documentation with at least 6 pages.			
I have provided average gas usages for the interface functions.			
I have provided comments in my code.			
I have developed test scripts, performed tests and submitted test scripts as well documented test results.			
I have developed smart contract Solidity code and submitted it.			
Function delegateVoteTo is implemented and works.			
Function donateEther is implemented and works.			
Function donateMyGovToken is implemented and works.			
Function voteForProjectProposal is implemented and works.			
Function voteForProjectPayment is implemented and works.			
Function submitProjectProposal is implemented and works.			
Function submitSurvey is implemented and works.			
Function submitSurvey is implemented and works.			
Function takeSurvey is implemented and works.			
Function reserveProjectGrant is implemented and works.			
Function withdrawProjectPayment is implemented and works.			
Function getSurveyResults is implemented and works.			
Function getSurveyInfo is implemented and works.			
Function getSurveyOwner is implemented and works.			
Function getIsProjectFunded is implemented and works.			
Function getProjectNextPayment is implemented and works.			
Function getProjectOwner is implemented and works.			
Function getProjectInfo is implemented and works.			
Function getNoOfProjectProposals is implemented and works.			
Function getNoOfFundedProjects is implemented and works.			
Function getEtherReceivedByProject is implemented and works.			
Function getNoOfSurveys is implemented and works.			
I have tested my smart contract with 100 addresses and documented the results of these tests.			
I have tested my smart contract with 200 addresses and documented the results of these tests.			
I have tested my smart contract with 300 addresses and documented the results of these tests.			
I have tested my smart contract with more than 300 addresses and documented the results of these tests.			