

CMPE 483 Blockchain Programming

Homework-1, Spring 2025

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

(due April 11th)

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/>

As real life examples, a list of major governance tokens for the game sector is available here:

- <https://www.coingecko.com/en/categories/gaming-governance-token>

You can use OpenZeppelin ERC20 token contract to implement the governance token. OpenZeppelin contracts are also 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 4000 TL tokens.
8. Submitting Survey costs 2 MyGov tokens and 1000 TL tokens.
9. MyGov token supply is 10 million (fixed).
10. Donations can be accepted in TL and MyGov tokens only. TL tokens 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 TL token 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 TL token 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

```

function delegateVoteTo(address memberaddr,uint projectid) public
function donateTLToken() public
function donateMyGovToken(uint amount) public
function voteForProjectProposal(uint projectid,bool choice) public
function voteForProjectPayment(uint projectid,bool choice) public
function submitProjectProposal(string weburl, uint votedeadline,uint [] paymentamounts,
                                uint [] payschedule) public returns (uint projectid)
function submitSurvey(string weburl,uint surveydeadline,uint numchoices, uint atmostchoice)
                                public returns (uint surveyid)
function takeSurvey(uint surveyid,uint [] choices) public
function reserveProjectGrant(uint projectid) public
function withdrawProjectTLPayment(uint projectid) public
function getSurveyResults(uint surveyid) public view returns(uint numtaken, uint [] results)
function getSurveyInfo(uint surveyid) public view returns(string weburl,
                                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 getProjectNextTLPayment(uint projectid) public view returns(int next)
function getProjectOwner(uint projectid) public view returns(address projectowner)
function getProjectInfo(uint activityid) public view returns(string weburl,
                                uint votedeadline,uint [] paymentamounts, uint [] payschedule)
function getNoOfProjectProposals() public view returns(uint numproposals)
function getNoOfFundedProjects () public view returns(uint numfunded)
function getTLReceivedByProject (uint projectid) public view returns(uint amount)
function getNoOfSurveys() public view returns(uint numsurveys)

```

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, hardhat deployment and test scripts, and tests.	60%

Homework Submission:

Please submit your project to Moodle. Please also answer the following questions and submit it with your project. Note that you should perform a demonstration of your project to me. An online survey will be sent in order to schedule demo dates and times for each group during the finals.

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.			
I have used Hardhat and ethers.js for smart contract development and testing.			
I have implemented Chai assertions for testing.			
Function delegateVoteTo is implemented and works.			
Function donateTLToken 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 withdrawProjectTLPayment 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 getProjectNextTLPayment 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 getTLReceivedByProject 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.			