YİĞİT SARIOĞLU
2022400354

CMPE 483 Blockchain Programming

Homework-1, Fall 2023

1. Introduction

In this project, I have implemented a autonomous decentralized governance token contract called “MyGov”. I have completed this project alone. Erc20 interface was used for the implementation of the contract. Total amount of my government token in circulation is a 20 million. Anyone who owns at least 1 MyGov token is MyGov member and members of this token could carry out some activities like “submitting a survey”, “project proposal”. I wrote the survey submission and project proposal functions in such a way that only members can call these functions. I created an onlymember modifier for this. I used timestamp to calculate time in surveys and proposals. In other words, the time entered into these functions represents the total life of the survey and proposal.When calculating the deadline, the calculation was made by adding the number entered by the user to the current block time.

I have tracked the amount of mygov tokens in circulation by using totalnumberoftokens variable. I have already assigned the USD coin amount as a fixed value in the constructor. But I thought the mygov token amount was a variable value, but I controlled it so that it could not exceed 20 million.

I have implemented the project using the solidity programming language with Remix compiler. I used library functions to ensure that the gas usage did not exceed 24k. This would helped me write code with efficient gas usage. I have also written the gas usage values of the functions below.

2. Methods

In the coding part, I have implemented onlyowner() and onlymember() modifiers, with the help of these modifiers, I was able to control the ability to call functions.

a. **constructor(uint256 tokenSupply) ERC20("MyGov", "MG")**

This constructor initializes the MyGov governance token by setting the total token supply to the specified value. It is executed only once during the deployment of the contract and ensures that the governance token, named MyGov (MG), is created with the desired initial supply of tokens.

YİĞİT SARIOĞLU
2022400354

In my opinion(my implementation) , the tokensupply amount will be transferred to the contract owner's account with the constructor. Then users will be able to obtain tokens with the faucet function, these tokens will not exceed the total MYGOV SUPPLY amount.

b. function delegateVoteTo(address memberaddr,uint projectid) public

This function provides MyGov token holders with the ability to delegate their voting power to another member for a specific project. By calling this function, a token holder can transfer their voting weight to another member's address, allowing the delegate to vote on behalf of the token holder for the specified project.
The function cannot be called if there is no valid project or if the specified address is not a member. I have used the require function to check these.

delegateVoteTo function gas usages:

transaction cost	80207 gas
execution cost	63447 gas

c. function donateUSD(uint amount)

The donateUSD function in the MyGov contract allows users to contribute funds denominated in USD Stable Coin to the MyGov platform. Users initiate a donation by specifying the donation amount, and the function ensures that the provided amount is greater than zero and that the sender possesses sufficient USD Stable Coin balance in their account. Once these conditions are met, the function utilizes the usdcoin ERC20 token contract to transfer the specified amount of USD Stable Coin from the sender's address to the address of the MyGov contract. I ensured that the donated money was collected at the contract's own address and collected in the contract account.

donateUSD function gas usages:

transaction cost	47843 gas
execution cost	26639 gas

d. public function donateMyGovToken(uint amount)

This function within the MyGov contract facilitates the contribution of MyGov tokens by users to support the activities and initiatives within the MyGov platform. Users initiate a donation by specifying the amount of MyGov tokens they wish to contribute, and the function ensures that the provided amount is greater than zero and that the sender possesses a sufficient balance of MyGov tokens. Upon meeting these conditions, the function utilizes the intrinsic transfer method to transfer the specified amount of MyGov tokens from the sender's address to the address of the MyGov contract. This feature enables users to actively participate in

the community-driven ecosystem by donating MyGov tokens, thereby contributing to the funding pool for project proposals, surveys, and other activities.

donateMyGovToken function gas usages:

transaction cost	34596 gas
execution cost	13392 gas

e. function submitSurvey(string ipfshash,uint surveydeadline,uint numchoices, uint atmostchoice) public returns (uint surveyid)

The **submitSurvey** function in the MyGov contract allows registered members to submit new surveys to the MyGov platform. To initiate a survey submission, a member provides essential details such as the **ipfshash** (for example it could be link to the web address) representing the survey data, the deadline for survey participation (surveydeadline), the number of choices available in the survey (numchoices), and the maximum number of choices a participant can make (atmostchoice). I have implemented this function as consisting of single question and multiple options. Participants could make more than one selection, that is determined by atmostchoice . The function also enforces payment requirements, ensuring that the submitting member has a balance of at least 2 MyGov tokens and 5 USD Stable Coins to cover the associated costs. Upon successful payment verification, the function transfers the specified amounts of USD Stable Coin and MyGov tokens to the contract address, representing the survey submission fees. Subsequently, a new survey instance is created using the createSurvey function from the MyGovSurveyLibrary, capturing the survey details and storing it in the surveys array. The unique identifier (surveyid) of the newly created survey is returned to the member at the end.

example input : "survey.com",300,5,1
in this example , survey has a name “survey.com” and its deadline is 300 sec later than its creation. So I can say that the total lifespan of the survey is 300 seconds. total number of choices is 5, it means that there are 5 different choice, and users could select at most 1 of them.

submitSurvey function gas usages:

transaction cost	223376 gas
execution cost	201412 gas

f. function faucet() public notReceivedTokenFromFaucet

The **faucet function** serves as a token distribution mechanism within the MyGov contract, allowing users to receive a small amount of MyGov tokens. To prevent abuse, certain conditions are enforced. Firstly, the function checks whether the caller is the owner of the contract, ensuring that contract owners cannot avail tokens from the faucet. Additionally, it

verifies if the address has already received tokens from the faucet, prohibiting multiple claims from the same address. Upon successful validation, the function mints one MyGov token to the caller's address using the _mint function, increments the address's membership status in the members mapping, and adds the address to the member_adress array if it is not already included. Furthermore, the function initializes the voting balances for the caller across all existing proposals, assigning a default voting balance of 1 to each proposal.

faucet function gas usages:

transaction cost	132749 gas
execution cost	111685 gas

g. function takeSurvey(uint surveyid,uint [] choices) public

This function in the MyGov contract is designed to allow MyGov members to actively participate in surveys created on the platform. This function is restricted to registered members, as indicated by the onlyMember modifier, ensuring that only valid members can submit their responses to a survey. The function takes two main parameters: surveyid, which identifies the specific survey the member wants to participate in, and choices, an array representing the member's chosen responses to the survey questions.

The function enforces several conditions to ensure the validity of the survey response. It checks that the number of choices made by the member is within the acceptable range (greater than zero and less than or equal to the maximum allowed choices). Additionally, it verifies that the current timestamp is before the survey deadline, preventing members from participating in surveys that have already concluded.

Upon successful validation of the input parameters, the function proceeds to handle the survey-taking logic. It utilizes the takeSurvey function from the MyGovSurveyLibrary to update the survey results based on the member's selected choices

takeSurvey function gas usages:

transaction cost	81344 gas
execution cost	59720 gas

h. function getSurveyResults(uint surveyid) public view returns(uint numtaken, uint [] results)

YİĞİT SARIOĞLU
2022400354

This function in the MyGov contract serves as a query mechanism for retrieving the results of a specific survey identified by its unique identifier (surveyid). This function is designated as a view function, indicating that it does not modify the state of the contract and can be called without incurring any gas costs. (not any transaction cost)

The function begins by validating that the provided *surveyid* is within the valid range of existing surveys, preventing attempts to retrieve results for non-existent surveys. Upon successful validation, the function retrieves two pieces of information related to the specified survey: the total number of participants (numtaken) and an array of results (results). The numtaken variable represents the count of members who have participated in the survey. The results array contains the aggregated responses to the survey questions. Each element in the array corresponds to a specific choice in the survey, and the value at each index represents the cumulative count of members who selected that particular choice. This array is crucial for analyzing the distribution of responses and deriving meaningful insights from the survey data.

execution cost	17663 gas (Cost only applies when called by a contract)

i. function getSurveyInfo(uint surveyid) public view returns(string ipfshash, uint surveydeadline,uint numchoices, uint atmostchoice)

The function within the MyGov contract serves as a read-only function designed to provide comprehensive information about a specific survey identified by its unique identifier (surveyid). By utilizing this function, users or external applications can obtain essential details regarding the surveyed topic, including the ipfshash representing the survey data, the deadline for survey participation (surveydeadline), the total number of available choices (numchoices), and the maximum number of choices allowed per participant (atmostchoice). This function enhances transparency and accessibility, offering stakeholders the ability to retrieve crucial metadata about a survey without altering the contract's state.. The returned information is instrumental for users seeking to understand the parameters and constraints associated with a particular survey.

execution cost	13368 gas (Cost only applies when called by a contract)

j. function getNoOfSurveys() public view returns(uint numsurveys)

This function in the MyGov contract is a view function designed to provide a quick and efficient way for users or external applications to obtain the total number of surveys available on the MyGov platform. (not any transaction cost)

execution cost	13368 gas (Cost only applies when called by a contract)

k. function getSurveyOwner(uint surveyid) public view returns(address surveyowner)

This function in the provided contract is designed to retrieve the owner's address for a specific survey identified by the given survey ID. It ensures that the provided survey ID is valid by checking if it is within the bounds of the existing surveys. Once validated, the function returns the address of the owner who have created this survey. It is a view function that, does not change any states, so there would be not any transaction cost.

execution cost	5212 gas (Cost only applies when called by a contract)

l. function submitProjectProposal(string ipfshash, uint votedeadline,uint [] paymentamounts, uint [] payschedule) public returns (uint projectid)

This function in the provided contract facilitates the submission of project proposals to the decentralized governance system. To initiate a proposal, a member must provide essential details such as the ipfshash representing the proposal content, the voting deadline in seconds (for example if you write 1000, it survives 1000 seconds) , payment amounts, and the corresponding payment schedule. The function enforces that the ipfshash is not empty and then ensures the submission cost is met, requiring 5 MyGov tokens and 50 USD stable coins. Upon successful validation, a new project proposal is created, indexed by the proposal ID, and stored in the 'proposals' array. The payment schedule is updated to reflect the block timestamp plus the specified durations. Additionally, the voting balances for each member are reset for the new proposal, allowing members to cast their votes in the upcoming governance process.

ex: "www.project.com",50, [10,20], [100,200]
This input creates a project proposal that its voting deadline is 50 secs after the creation. And after it is funded, the first payment will be done 100 secs after the creation and second payment will be done after the 200 secs. In order to make these payments, the withdrawprojectpayment function must be called.

submitProjectProposal function gas usages:

transaction cost	399857 gas
execution cost	377053 gas

m. **public function voteForProjectProposal(uint projectid,bool choice)**

YİĞİT SARIOĞLU
2022400354

The `voteForProjectProposal` function is designed to facilitate the voting process for MyGov members on a specific project proposal. The function requires the member to be a MyGov token holder, ensuring that only eligible participants can contribute to the decision-making process. The `projectid` parameter identifies the targeted project proposal within the array of proposals. Before allowing the member to vote, the function checks if the proposal's voting deadline has passed and if the project has not been funded already, ensuring that votes are cast within the designated timeframe and under the correct circumstances. The function then delegates the actual voting mechanism to a separate helper function called `votingMechanism`, passing along the project ID, the member's voting choice (either `true` or `false`), and the member's address (`msg.sender`). This separation of concerns enhances readability and modular design in the contract. The `votingMechanism` function is expected to handle the core logic of updating the vote counts and ensuring that members can vote only once for a given project proposal.

voteForProjectProposal function gas usage:

transaction cost	81109 gas
execution cost	59777 gas

n. function voteForProjectPayment(uint projectid,bool choice) public

The `voteForProjectPayment` function enables MyGov members to express their preference on a specific project payment proposal. Each member, whether directly or through delegation, can cast a vote indicating approval or rejection. The `projectid` parameter identifies the target project payment proposal. Members must ensure that the voting deadline for the proposal has passed before participating in the vote. The boolean parameter `choice` allows members to vote positively (`true`) or negatively (`false`). This function ensures that members or their delegated representatives can only vote once for a given payment proposal. The outcome of the vote influences whether the proposed payment will be executed, based on the overall consensus of the MyGov community.

voteForProjectPayment function gas usage:

transaction cost	64018 gas
execution cost	42674 gas

o. function reserveProjectGrant(uint projectid) public

The `reserveProjectGrant` function is responsible for allowing project proposers to reserve the grant for their project, subject to specific conditions. The function first verifies the validity of the project ID, ensuring it is within the range of the available proposals. It then checks whether the sender of the transaction is the designated project proposer, ensuring that only the proposer can trigger the grant reservation.

YİĞİT SARIOĞLU
2022400354

To prevent double funding, the function verifies that the project proposal has not been funded already. Additionally, it checks that the proposal's deadline has passed (within the timing limit), ensuring that the grant reservation occurs within the specified timeframe. The requirement for at least 1/10 of the members to have voted "yes" is a key condition to secure community support before proceeding with the funding.

The function also checks whether there is a sufficient balance of USD stable coins in the MyGov contract to cover the grant amount calculated by the `calculateGrantAmount` function. If these conditions are met, the totalGrantedMoney variable is updated to reflect the increased granted amount, and the project is marked as funded. The `resetVotingBalances` function is then called to reset the voting balances for the project, ensuring that members can vote on subsequent proposals independently.

reserveProjectGrant function gas usage:

transaction cost	137687 gas
execution cost	116495 gas

p. function withdrawProjectPayment(uint projectid) public

This function is designed to allow project owners to withdraw payments from their funded projects, subject to specific conditions. It begins by validating the project ID to ensure it falls within the range of available proposals. The function then checks whether the project is indeed funded, preventing withdrawals from non-funded projects.

The time condition is crucial, ensuring that the withdrawal occurs only when the payment deadline has passed. I set it to be withdrawn the money within 60 seconds after the deadline. The function also verifies that the sender is the designated project owner, limiting payment withdrawals to the rightful owner.

Additional checks ensure that there is sufficient community support for the payment, requiring at least 1/100 of the members to have voted "yes" for project payments. The function also ensures there are remaining payments to be withdrawn by checking the nextPaymentIndex against the total number of payments in the payschedule.

The amount to be withdrawn is calculated based on the payment schedule, and the function checks if there is enough USD stable coin balance in the MyGov contract to cover the payment. If all conditions are met, the payment amount is transferred to the project owner, and the payedAmount is updated accordingly. The nextPaymentIndex is then incremented to track the progress of payments.

If there are more requests than the amount of payment to be made, the function checks and prevents this.

vithdrawprojectPayment function gas usage:

transaction cost	86531 gas
execution cost	65339 gas

q. function donationMyGOVBalance() public view

This function is a view function that returns the balance of donated MyGov tokens to the contract. Since this function does not change the state, there will be no gas usage.

transaction cost : 0

execution cost	2642 gas (Cost only applies when called by a contract)

r. function donationUSDBalance() public view

This function is a view function that returns the balance of donated USD tokens to the contract. Since this function does not change the state, there will be no gas usage. Both donationmygovbalance() and donationUSDBalance() are helper functions in the contract.

transaction cost :0

execution cost	8628 gas (Cost only applies when called by a contract)

3. Conclusion

I worked alone on this project and learned how to develop smart contracts and solidity language. While developing the project, I made sure that it was gas efficient. Also, I explained almost the entire project by writing comments to my code. I also wrote unit tests and test scripts that test the project.

YİĞİT SARIOĞLU
2022400354

4. Task Achievement Table

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 donateUSD 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 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.	+		

YİĞİT SARIOĞLU
2022400354

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 getUSDReceivedByProject 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.		+	