Development and Deployment of an Advanced Decentralized Autonomous Organization (DAO)

Abstract

This report outlines the process of designing and implementing an Advanced DAO smart contract. The DAO leverages Ethereum blockchain technology to provide a decentralized governance framework, allowing token holders to propose, vote on, and execute collective decisions. It uses an ERC20 token, "ADI," for voting power, enabling a democratic process secured by blockchain technology.

Table of Contents

Introduction

The concept of a Decentralized Autonomous Organization (DAO) represents a new paradigm in collective decision-making and resource management. By harnessing the power of smart contracts on the Ethereum blockchain, DAOs offer a transparent, immutable, and secure environment for group governance without centralized control. This project aims to create a DAO smart contract that uses a token-based voting system to enable stakeholders to guide the DAO's actions.

Objective and Project Description

The objective of this project was to create a DAO that allows members to participate in governance through a secure and automated process. The chosen project aligns with the vision of decentralized decision-making and was selected to explore the practical applications of Ethereum smart contracts. The expectation is to demonstrate the viability of blockchain technology in facilitating democratic decision-making processes.

Development Environment and Tools

- Solidity (v0.8.0)
- Remix IDE

- MetaMask (Sepolia Test Network)
- Etherscan (Sepolia Testnet Explorer)

Smart Contract Development

The AdvancedDAO contract was developed, including features such as proposal creation, voting by token weight, minimum quorum requirements, and time-bound debate periods. A secondary contract, ADI Token, was created to serve as the voting token for DAO members.

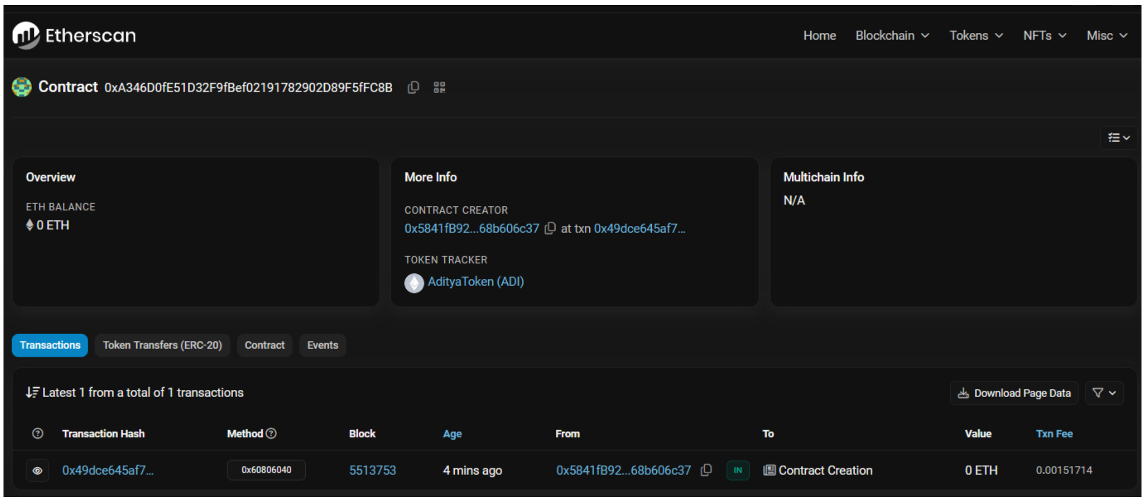
Minting ERC20 Token

An ERC20 token has been minted on the Sepolia Test Network

- **Name**: "AdityaToken"
- **Symbol**: "ADI"

The _mint function is called to create 1000 tokens and assign them to the MetaMask wallet address (*0x5841fB926aA354aF889FBf1606a9aE268b606c37*)

Link to ADI Token Minting Source Code



Contract 0xA346D0fE51D32F9fBef02191782902D89F5fFC8B

Overview

ETH BALANCE
0 ETH

More Info

CONTRACT CREATOR
0x5841fB92...68b606c37 at txn 0x49dce645af7...

TOKEN TRACKER
AdityaToken (ADI)

Multichain Info

N/A

Transactions | Token Transfers (ERC-20) | Contract | Events

Latest 1 from a total of 1 transactions

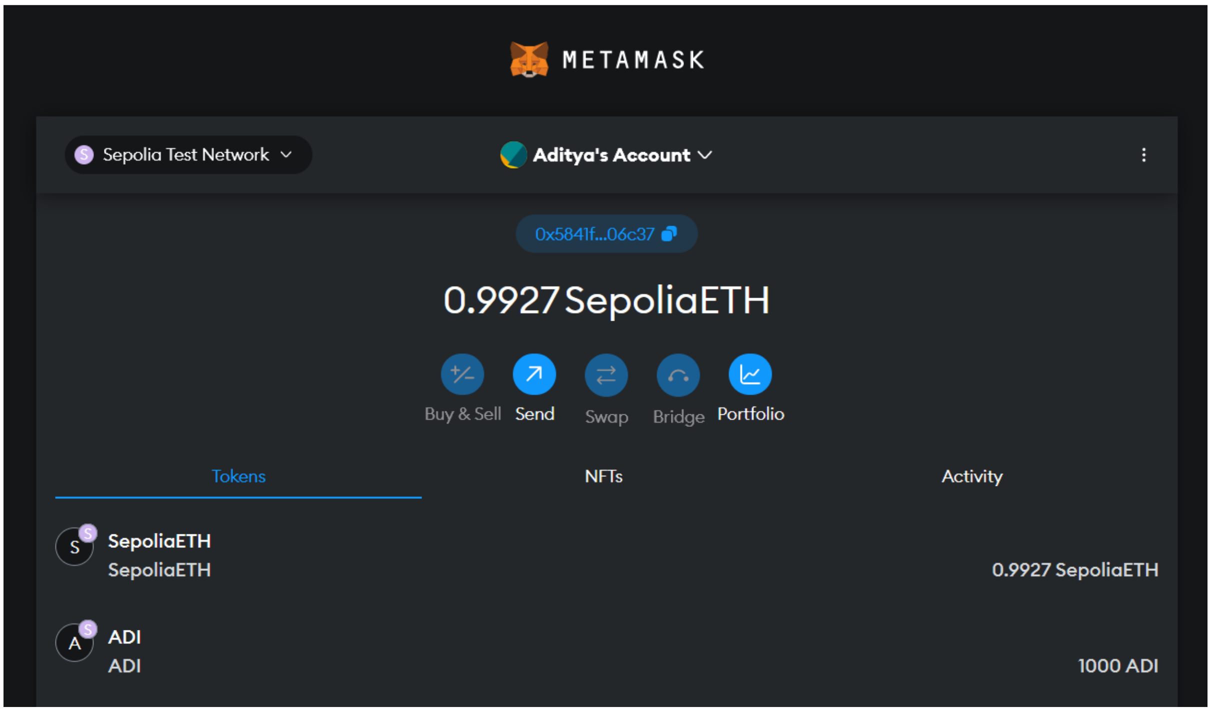
Download Page Data

	Transaction Hash	Method	Block	Age	From		To	Value	Txn Fee
	0x49dce645af7...	0x60806040	5513753	4 mins ago	0x5841fB92...68b606c37	IN	Contract Creation	0 ETH	0.00151714

(Fig: Etherscan transaction for minting the token, Address: 0xA346D0fE51D32F9fBef02191782902D89F5fFC8B)

Etherscan link for minted ADI token:
https://sepolia.etherscan.io/address/0xa346d0fe51d32f9fbef02191782902d89f5ffc8b

The newly minted token is then added to the MetaMask account.



SepoliaETH SepoliaETH	0.9927 SepoliaETH
ADI ADI	1000 ADI

(Fig: MetaMask Account balance showing 1000 ADI newly minted token)

We have now Successfully minted and added the ADI ERC20 token to our account.

Smart Contract Design

The AdvancedDAO smart contract is a great example of how blockchain can change the way we make decisions together. It's made to be clear and work well, making sure that everyone's voice can be heard fairly.

- **Proposal System:** The most important part of the DAO is that it lets people with tokens suggest new ideas. Then, all the other people with tokens can vote on these ideas. This means that everyone gets to help make decisions.
- **Voting Based on Tokens:** The voting system is designed so that the more tokens you have, the more your vote counts. This means your influence in the DAO's choices is linked to how many tokens you have. You can vote 'yes' or 'no' on new ideas, and the majority wins.
- **Debating Period - Time to Talk and Think:** Every new idea gets a certain amount of time for discussion before voting ends. This gives everyone a chance to think about it and share their thoughts. The DAO can change how long this period is, making sure people don't rush into decisions.
- **Quorum – The Minimum Number of Votes Needed:** For any decision to be valid, a certain number of votes must be reached. This ensures that a good number of people are involved in the decision. It's not just a few people deciding, but a larger group, which makes the outcome more valid.
- **Making Decisions Automatically:** After the discussion time is over, if most votes are 'yes' and enough people have voted, the decision is made automatically. This means the system itself carries out the will of the people, without anyone having to do it manually.
- **Security Measures:** The DAO has special security rules to make sure everything runs smoothly. For example, only the person who made the DAO can do certain things, which prevents anyone from making changes they shouldn’t be able to.

In simple terms, the AdvancedDAO is a tool for people to make decisions together in a fair, open, and secure way on the blockchain. It makes sure that the more you're invested in the DAO, the more you get to influence its direction. It also has rules to make sure that decisions are made only when enough people agree and that the system itself is safe from anyone trying to mess with it.

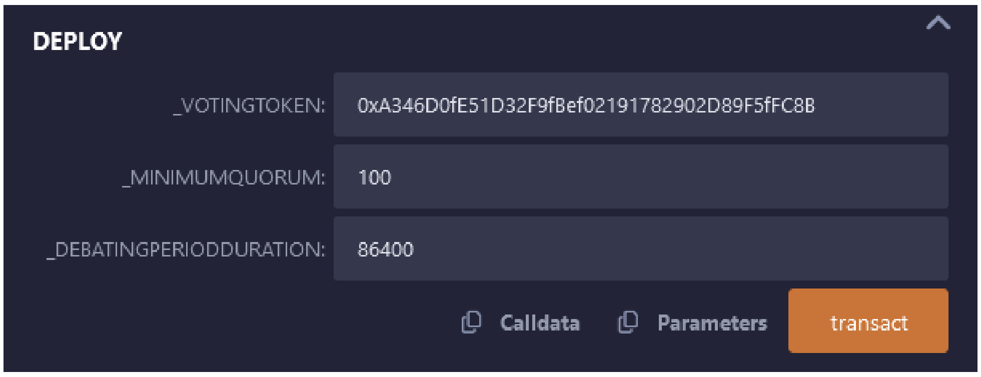
Development Process

The DAO was developed iteratively, focusing on functionality, security, and user experience. OpenZeppelin contracts were used to secure the token interactions. After writing the contract, it was compiled and deployed using Remix IDE to the Sepolia Test Network. Extensive testing was conducted to ensure that the contract performed as expected.

Link to DAO source code

Deployment and Testing

Tests were performed on the Sepolia Test Network to simulate the creation, voting, and execution of proposals. The DAO contract successfully allowed the creation of proposals by the owner, voting by token holders, and automatic execution of proposals after the deadline. Voting power was accurately weighted according to the number of ADI tokens held by each voter.


DEPLOY

_VOTINGTOKEN: 0xA346D0fE51D32F9fBef02191782902D89F5fFC8B

_MINIMUMQUORUM: 100

_DEBATINGPERIODDURATION: 86400

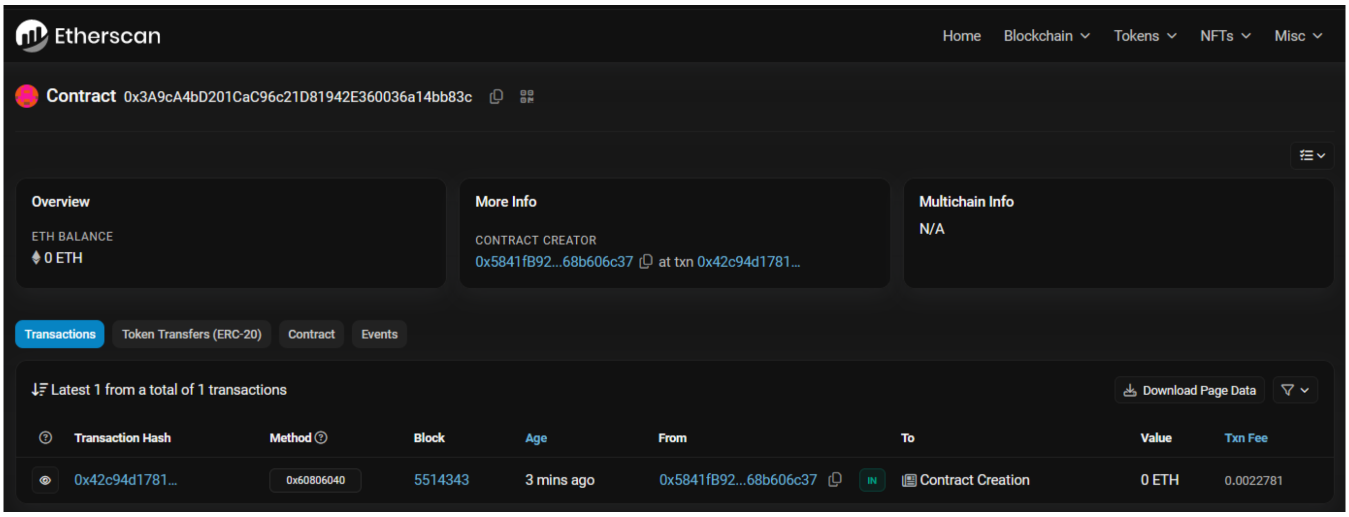
(Fig: Input Parameters for voting DAO contract created)

The Input parameters have the following meaning:

- **_VOTINGTOKEN**: This parameter signifies the contract address for the token to be used for voting which in our case is the newly minted ERC20 token 'ADI'.
  The contract address cannot be directly fed here and instead, we will have to convert it to checksum. Used an online tool to convert Ethereum Contract Address to its Checksum value (Link in References)

- **_MINIMUMQUORUM**: This is the minimum number of votes that must be in agreement for a decision to be valid. We have set it to 10% of the total supply and since we have 1000 ADI, we set it to 100 ADI.

- **_DEBATINGPERIODDURATION**: This is the duration that a proposal remains open for voting. It's specified in seconds. We have set the debating period to be 24 hours (24 hours * 60 minutes * 60 seconds = 86400 seconds).



Contract 0x3A9cA4bD201CaC96c21D81942E360036a14bb83c

Overview

ETH BALANCE
0 ETH

More Info

CONTRACT CREATOR
0x5841fB92...68b606c37 at txn 0x42c94d1781...

Multichain Info
N/A

Latest 1 from a total of 1 transactions

Transaction Hash	Method	Block	Age	From		To	Value	Txn Fee
0x42c94d1781...	0x60806040	5514343	3 mins ago	0x5841fB92...68b606c37	IN	Contract Creation	0 ETH	0.0022781

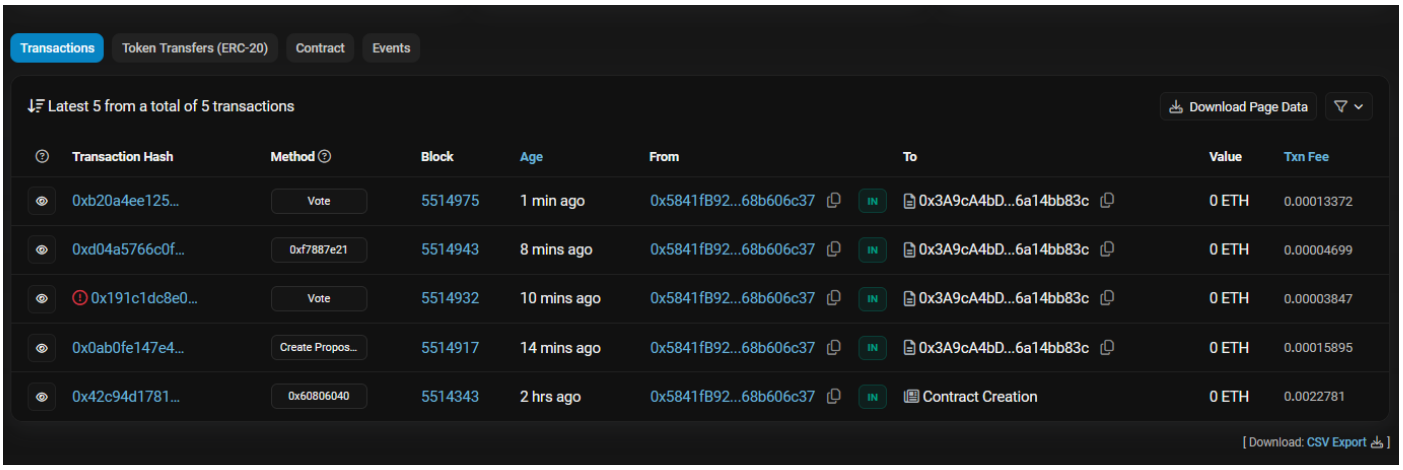
(Fig: Etherscan transaction for DAO deployment to Sepolia Blockchain)

Etherscan link for DAO on Sepolia Test Network:
https://sepolia.etherscan.io/address/0x3a9ca4bd201cac96c21d81942e360036a14bb83c

The Created DAO has several Methods which can be called:

1. **createProposal(string memory _description)**
   - Allows the owner to create a new proposal with the provided description. It records the proposal's deadline based on the current block timestamp and debating period duration.
2. **vote(uint256 _proposalId, bool _support)**
   - Enables a token holder to cast a vote on a proposal. The vote can be in support of (`_support = true`) or against (`_support = false`) the proposal. Each token holder's vote is weighted by their token balance.
3. **executeProposal(uint256 _proposalId)**
   - After the debating period has ended, this function can be called to execute the proposal. It checks if the proposal has reached the minimum quorum and if the majority of votes are in favor. It then marks the proposal as executed.
4. **emergencyStopProposal(uint256 _proposalId)**
   - This administrative function allows the owner to stop a proposal in case of an emergency. It immediately marks the proposal as executed to prevent further actions.
5. **changeMinimumQuorum(uint256 _newMinimumQuorum)**
   - An owner-only function that allows the adjustment of the minimum quorum required for proposals to be considered valid.
6. **changeDebatingPeriodDuration(uint256 _newDebatingPeriodDuration)**
   - An owner-only function that allows the changing of the duration for which a proposal can be debated and voted on.
7. **renounceOwnership()**
   - Part of the `Ownable` contract from OpenZeppelin; allows the current owner to transfer control of the contract to no one, effectively leaving the contract without an owner.
8. **transferOwnership(address newOwner)**
   - Another `Ownable` function, this method allows the current owner to transfer ownership of the contract to a new address.

While testing the DAO, I have First created a Proposal “Hello World” and Changed the Minimum Quorum to 2 and then Voted in Support of the proposal.


Transactions | Token Transfers (ERC-20) | Contract | Events

Latest 5 from a total of 5 transactions

Download Page Data

Transaction Hash	Method	Block	Age	From		To	Value	Txn Fee
0xb20a4ee125...	Vote	5514975	1 min ago	0x5841fB92...68b606c37	IN	0x3A9cA4bD...6a14bb83c	0 ETH	0.00013372
0xd04a5766c0f...	0xf7887e21	5514943	8 mins ago	0x5841fB92...68b606c37	IN	0x3A9cA4bD...6a14bb83c	0 ETH	0.00004699
0x191c1dc8e0...	Vote	5514932	10 mins ago	0x5841fB92...68b606c37	IN	0x3A9cA4bD...6a14bb83c	0 ETH	0.00003847
0x0ab0fe147e4...	Create Propos...	5514917	14 mins ago	0x5841fB92...68b606c37	IN	0x3A9cA4bD...6a14bb83c	0 ETH	0.00015895
0x42c94d1781...	0x60806040	5514343	2 hrs ago	0x5841fB92...68b606c37	IN	Contract Creation	0 ETH	0.0022781

[Download: CSV Export]

(Fig: Etherscan transaction for Creating Proposal, Changing Minimum Quorum and Voting in favour)

I again tried to vote for the same proposal but this time the DAO didn’t allow and this is a feature of the contract which does not allow double voting from the same address.

Challenges and Solutions

During development, two primary challenges were faced: preventing the reuse of voting power across different proposals and ensuring that token holders could not vote more than once on the same proposal. These issues were critical to maintaining a fair and democratic voting process.

The solutions implemented were both effective and elegant. To tackle the reuse of voting power, the contract was designed to map addresses to Boolean values within each proposal, indicating whether that address had already cast a vote. This mapping served as a checkpoint, preventing repeat voting, and ensuring that each token holder's voice was heard exactly once.

Further, the voting power was tied to the token balance at the time of the vote, mitigating the risk of vote manipulation through token transfer. This implementation ensured that voting power was not only accurately represented but also locked in at the crucial moment of decision-making.

These solutions underscored the project's commitment to fairness and integrity, solidifying the AdvancedDAO as a paradigm of trust and security in the realm of decentralized governance.

Future Improvements

While the developed DAO smart contract serves as a robust framework for decentralized governance, there are several avenues for enhancement and additional features that could be implemented to expand its capabilities and improve security:

1. **Voting Help**: Let people who don't have time or don't know much about the proposals give their votes to someone they trust. This helps ensure that all voices are heard, even if they're not casting the vote themselves.
2. **Handle More Users**: Improve the contract so it can work well even with lots of proposals and voters, without costing too much in fees.
3. **Easy Updates**: Change the way the DAO's code is organized so that it's easier to add new things or make changes in the future without starting from scratch.

4. **Working with Other Contracts**: Make the DAO able to talk to other contracts. This could let it manage money, work with other blockchain programs, or do several steps in one go.
5. **Wait Before Acting**: Put in a waiting period after a decision is made so there's time to get ready for what will happen next.
6. **Check for Safety**: Have experts check the code for any problems before it's used for real. Think about rewarding people who find and report these issues.
7. **Rewards for Participation**: Consider ways to encourage more people to take part in voting, like giving them small rewards or allowing them to earn more votes over time.
8. **Flexible Voting Requirements**: Adjust the number of votes needed to make a decision based on how many people have been taking part recently. This helps to keep things fair as the DAO grows or changes.
9. **Private Voting**: Use special tech to let people vote without others knowing what they choose. This can stop people from being pushed into voting a certain way.
10. **Joining Forces with Fast Blockchain Layers**: Connect with new blockchain technology that can process many transactions quickly and cheaply to make the DAO run smoother and faster.

Conclusion

The DAO smart contract developed provides a foundational governance framework for collective decision-making. The DAO concept has far-reaching implications for decentralized governance and represents a significant step forward in blockchain-based organizational structures.

References and Links

- Ethereum Solidity Documentation
- OpenZeppelin Smart Contracts Library
- Remix IDE Documentation
- Ethereum Contract Address to Checksum
- Minting ERC20 Token
- Etherscan Transaction for Minting ADI Token
- Etherscan Transaction for DAO

Source Code

- **Code For Minting ADI token – ADI_Token.sol**
- **Code for DAO implementation - AdvancedDAO.sol**

Minting ERC20 ADI Token Code

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

import "@openzeppelin/contracts/token/ERC20/ERC20.sol";

contract MyToken is ERC20 {
    constructor() ERC20("AdityaToken", "ADI") {
        _mint(msg.sender, 1000*10**18);
    }
}
```

DAO Voting Contract code

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

import "@openzeppelin/contracts/token/ERC20/IERC20.sol";
import "@openzeppelin/contracts/access/Ownable.sol";

contract AdvancedDAO is Ownable {
    IERC20 public votingToken;
    uint256 public minimumQuorum;
    uint256 public debatingPeriodDuration;

    struct Proposal {
        string description;
        bool executed;
        uint256 deadline;
        uint256 forVotes;
        uint256 againstVotes;
        mapping(address => bool) voted;
    }

    Proposal[] public proposals;

    event ProposalCreated(uint256 id, string description, uint256 deadline);
    event Voted(uint256 proposalId, bool position, address voter, uint256 weight);
    event ProposalExecuted(uint256 id, bool result);

    // The Ownable constructor is called implicitly if there are no parameters needed.
    constructor(address _votingToken, uint256 _minimumQuorum, uint256
_debatingPeriodDuration) Ownable(msg.sender) {
        votingToken = IERC20(_votingToken);
        minimumQuorum = _minimumQuorum;
        debatingPeriodDuration = _debatingPeriodDuration;
    }

    function createProposal(string memory _description) public onlyOwner {
        uint256 deadline = block.timestamp + debatingPeriodDuration;
        proposals.push();
        uint256 proposalId = proposals.length - 1;
        Proposal storage newProposal = proposals[proposalId];
        newProposal.description = _description;
        newProposal.executed = false;
        newProposal.deadline = deadline;
        emit ProposalCreated(proposalId, _description, deadline);
    }

    function vote(uint256 _proposalId, bool _support) public {
        Proposal storage proposal = proposals[_proposalId];
        require(block.timestamp < proposal.deadline, "Voting is closed.");
        require(!proposal.voted[msg.sender], "Already voted.");
        require(!proposal.executed, "Proposal already executed.");

        uint256 voterBalance = votingToken.balanceOf(msg.sender);
```

```solidity
        require(voterBalance > 0, "No voting tokens held.");

        if (_support) {
            proposal.forVotes += voterBalance;
        } else {
            proposal.againstVotes += voterBalance;
        }

        proposal.voted[msg.sender] = true;
        emit Voted(_proposalId, _support, msg.sender, voterBalance);
    }

    function executeProposal(uint256 _proposalId) public {
        Proposal storage proposal = proposals[_proposalId];
        require(block.timestamp >= proposal.deadline, "Debating period not over.");
        require(!proposal.executed, "Proposal already executed.");
        require(proposal.forVotes + proposal.againstVotes >= minimumQuorum, "Minimum
quorum not met.");

        proposal.executed = true;
        bool result = proposal.forVotes > proposal.againstVotes;

        emit ProposalExecuted(_proposalId, result);
    }

    function emergencyStopProposal(uint256 _proposalId) public onlyOwner {
        Proposal storage proposal = proposals[_proposalId];
        require(!proposal.executed, "Proposal already executed.");
        proposal.executed = true;
        emit ProposalExecuted(_proposalId, false);
    }

    function changeMinimumQuorum(uint256 _newMinimumQuorum) public onlyOwner {
        minimumQuorum = _newMinimumQuorum;
    }

    function changeDebatingPeriodDuration(uint256 _newDebatingPeriodDuration) public
onlyOwner {
        debatingPeriodDuration = _newDebatingPeriodDuration;
    }

    // Additional functions and logic can be implemented as needed
}
```