VOTING_DAO

Overview

*The **VotingDAO** smart contract is a decentralized voting system that allows users to create proposals and vote on them. Built with Solidity, it ensures transparency and security by leveraging blockchain technology.*

Key Features

1. ***Create Proposals**: Users can create proposals with a description and voting duration.*
2. ***Vote on Proposals**: Users can cast votes either for or against active proposals.*
3. ***Track Proposals**: View active proposals and check their status (active or expired).*
4. ***Vote Tracking**: Check vote counts for/against each proposal and confirm whether a user has voted.*

Functions Overview

1. createProposal

- ***Purpose**: Allows users to create a proposal.*
- ***Parameters**:*
    - *description: A string with the proposal's description.*
    - *durationInSeconds: Duration for which the proposal remains active.*
- ***Logic**:*
    - *Validates that the duration is greater than zero.*
    - *Creates a new proposal with a unique ID.*
    - *Emits a ProposalCreated event.*

2. vote

- ***Purpose**: Enables users to vote on an active proposal.*
- ***Parameters**:*
    - *proposalId: ID of the proposal being voted on.*
    - *voteFor: Boolean (true for "yes", false for "no").*
- ***Logic**:*

- *Verifies that the proposal is active and that the user hasn't already voted.*
- *Records the vote and updates the vote counts.*
- *Emits a VoteCast event.*

3. isProposalActive

- ***Purpose**: Checks whether a proposal is still active.*
- ***Parameters**:*
  - *proposalId: ID of the proposal.*
- ***Logic**:*
  - *Returns true if the proposal is active; otherwise, false.*

4. getVoteCount

- ***Purpose**: Retrieves the number of votes for and against a proposal.*
- ***Parameters**:*
  - *proposalId: ID of the proposal.*
- ***Logic**:*
  - *Returns the counts of "for" and "against" votes.*

5. getProposal

- ***Purpose**: Returns details of a proposal.*
- ***Parameters**:*
  - *proposalId: ID of the proposal.*
- ***Logic**:*
  - *Returns the proposal's description, vote counts, and status.*

6. getAllProposalIds

- ***Purpose**: Retrieves all stored proposal IDs.*
- ***Logic**:*
  - *Returns an array of all proposal IDs.*

7. hasUserVoted

- ***Purpose**: Checks if a user has already voted on a specific proposal.*
- ***Parameters**:*
  - *proposalId: ID of the proposal.*
  - *user: Address of the user.*

- ***Logic**:*
    - *Returns true if the user has voted, otherwise false.*

Development Process

1. Requirements Gathering

Defined the key features of the contract: proposal creation, voting, vote tracking, and viewing status.

2. Contract Design

Used Solidity structs and mappings for proposal management and vote tracking.

3. Implementation

Implemented core functionality with checks for proposal expiry, vote status, and prevention of double voting.

4. Testing

Tested the contract using Hardhat with Mocha and Chai, ensuring that all edge cases (e.g., double voting, expired proposals) are handled.

Deployment Using Hardhat

Steps to Deploy:

1. ***Environment Setup**:*
    - *Install Node.js and Hardhat:*

npm install --save-dev hardhat

2. ***Create Hardhat Project**:*

npx hardhat init

3. ***Install Dependencies**:*
    - *Install OpenZeppelin Contracts:*

npm install @openzeppelin/contracts

javascript:

import { buildModule } from "@nomicfoundation/hardhat-ignition/modules";

module.exports = buildModule("VotingDAO", (m) => {

const votingDao = m.contract("VotingDAO", []);

return { votingDao };

});

4. ***Deploy the Contract****:*
    - *Run the Ignition deployment:*

*# **npx hardhat ignition deploy ./ignition/modules/Lock.js --network sepolia**(Any network).*

Testing

To Test the Contract:

1. ***Proposal Creation Tests****:*
    - *Check if proposals are created successfully.*
    - *Ensure invalid durations are rejected.*
2. ***Voting Tests****:*
    - *Test voting for active proposals.*
    - *Ensure users can't vote twice on the same proposal.*
    - *Verify that voting after the proposal expiration fails.*
3. ***Data Retrieval Tests****:*
    - *Confirm correct retrieval of proposals and vote counts.*
    - *Verify that the getAllProposalIds function returns all proposal IDs.*

Run Tests:

 # npx hardhat test

 # Deployed Addresses

*#VotingDao - **0x80897Aa9aa4e2914AeFb12591eaD2c620c9bF54a***

***Conclusion:** The **VotingDAO** contract provides a decentralized, transparent voting mechanism on Ethereum Blockchain. The application will allow users to create proposals and vote on them, simulating a basic Decentralized Autonomous Organization (DAO) mechanism.*