Centurion University
Shaping Lives... Empowering Communities...

School: ..... Campus: .....

Academic Year: ..... Subject Name: ..... Subject Code: .....

Semester: ..... Program: ..... Branch: ..... Specialization: .....

Date: .....

Applied and Action Learning

(Learning by Doing and Discovery)

Name of the Experiement : DAO in Action – Governance Simulation

Objective/Aim:

To simulate decentralized governance by interacting with a DAO smart contract on a testnet. The goal is to connect a wallet, create a proposal, vote on it, and observe how on-chain governance decisions are executed.

Apparatus/Software Used:

- MetaMask Wallet
- Brave Web Browser
- Testnet Governance dApp (example DAO interface)
- Ethereum Sepolia Testnet

Theory/Concept:

A DAO (Decentralized Autonomous Organization) is a community-driven governance structure controlled by smart contracts rather than centralized authorities. Token holders participate in the decision-making process by creating proposals and voting on them.

Key Concepts

- Governance Token: Allows holders to vote on proposals.
- Proposal: A suggested action, rule update, or change.
- Voting Mechanism: Token-based system where votes are counted on-chain.
- Execution: If the proposal passes the required quorum, it is executed automatically by the smart contract.

Procedure:

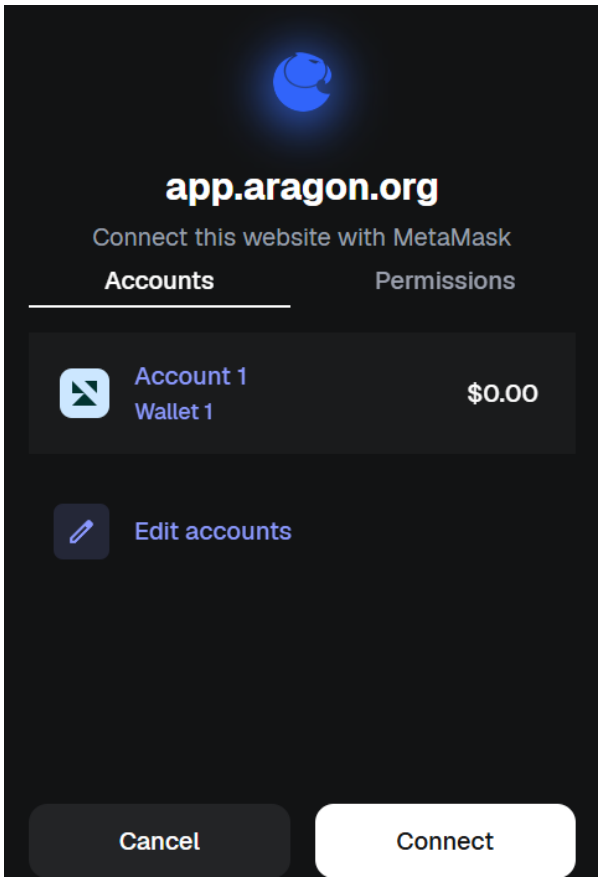
1. Open MetaMask and switch to the Sepolia testnet.
2. Visit the DAO governance dApp interface in your browser.
3. Click “Connect Wallet” and link MetaMask to the governance application.
4. Navigate to the Proposals section.
5. Click “Create New Proposal”.
6. Enter a proposal title (e.g., “Increase Community Treasury Allocation”).
7. Write a brief description and submit the proposal.
8. Approve the MetaMask transaction to publish the proposal on-chain.
9. Once published, go to the Voting section.
10. Select the proposal and choose “Vote For” or “Vote Against”.
11. Confirm the voting transaction in MetaMask.
12. Wait for the proposal’s voting period to complete.
13. After voting ends, view the result and click “Execute Proposal” if it has passed the required quorum.
14. Confirm the final transaction in MetaMask to execute the proposal.

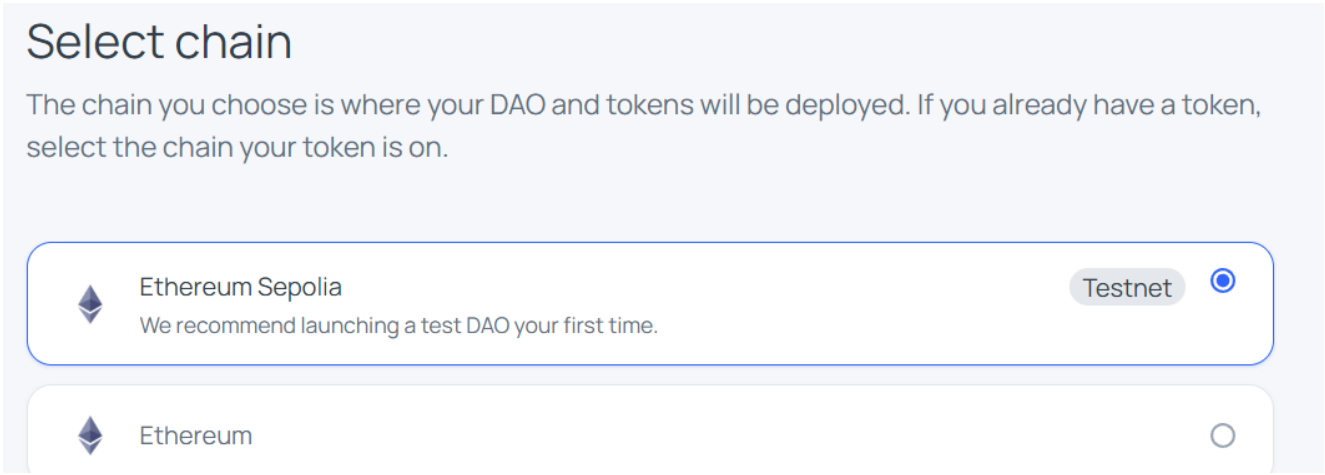
Solidity contract

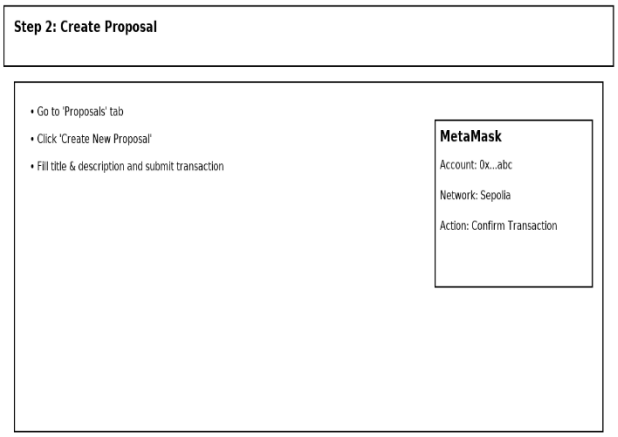
C: > Users > HP > Downloads > SimpleDAO.sol

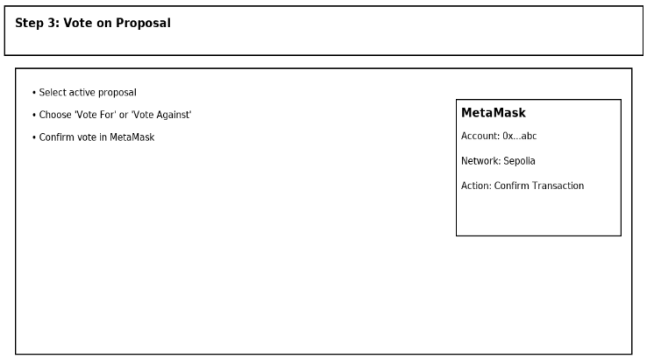
```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.19;

/*
 SimpleDAO — minimal governance contract for educational labs.
 - One address = one vote (no token weighting) for simplicity.
 - Owner (deployer) can add/remove members eligible to create proposals and vote.
 - Proposal lifecycle: Created -> Voting -> Tally -> Executable(if passed)
 - Voting period and quorum are configurable at deployment.
*/

import "@openzeppelin/contracts/access/Ownable.sol";

contract SimpleDAO is Ownable {
    struct Proposal {
        uint256 id;
        address proposer;
        string description;      // human-readable description
        uint256 yesVotes;
        uint256 noVotes;
        uint256 startBlock;
        uint256 endBlock;
        bool executed;
        mapping(address => bool) voted; // track who voted
    }

    uint256 public proposalCount;
    mapping(uint256 => Proposal) private proposals;
    mapping(uint256 => bool) private _exists; // helper for existence check

    // members mapping: addresses eligible to create proposals and vote
    mapping(address => bool) public isMember;
```

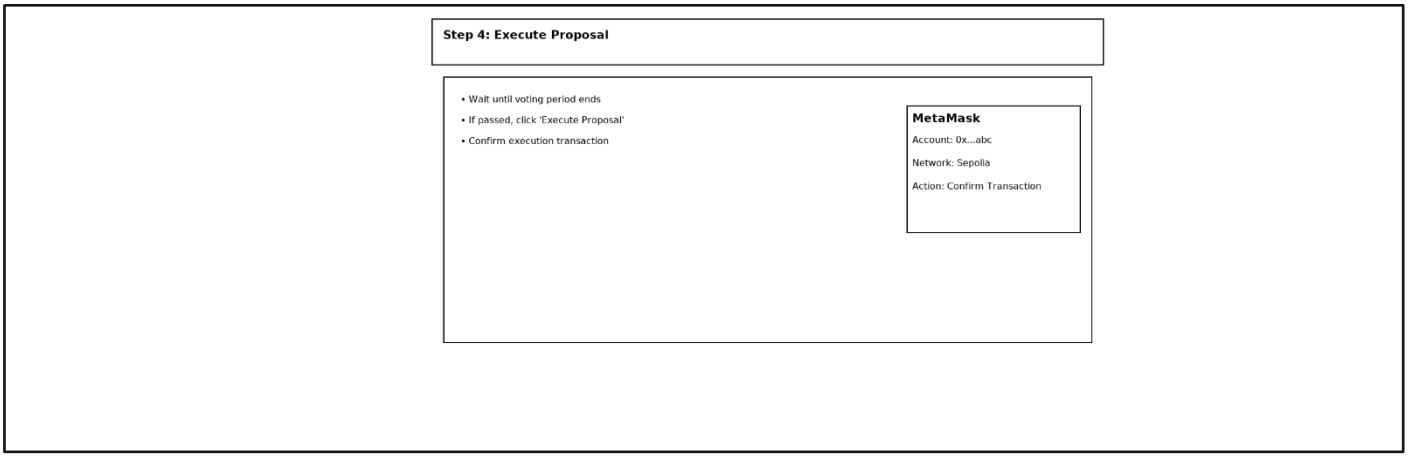
```
// Tally and execute (callable by anyone once voting ended)
function executeProposal(uint256 _id) external {
    require(_exists[_id], "No such proposal");
    Proposal storage p = proposals[_id];
    require(block.number > p.endBlock, "Voting not ended");
    require(!p.executed, "Already executed");

    p.executed = true;
    bool passed = false;
    if (p.yesVotes >= quorum && p.yesVotes > p.noVotes) {
        passed = true;
        // Note: In this minimal DAO we do not perform arbitrary on-chain actions.
        // In real DAOs proposals can include encoded calls to execute.
    }

    emit ProposalExecuted(_id, passed);
}

// Read helpers
function getProposalMeta(uint256 _id) external view returns (
    uint256 id, address proposer, string memory description, uint256 yesVotes, uint256 noVotes, uint256 startBl
) {
    require(_exists[_id], "No such proposal");
    Proposal storage p = proposals[_id];
    return (p.id, p.proposer, p.description, p.yesVotes, p.noVotes, p.startBlock, p.endBlock, p.executed);
}

// get simple existence
function proposalExists(uint256 _id) external view returns (bool) {
    return _exists[_id];
}
```


app.aragon.org
Connect this website with MetaMask
Accounts
Permissions
Account 1
Wallet 1
$0.00
Edit accounts
Cancel
Connect


Select chain
The chain you choose is where your DAO and tokens will be deployed. If you already have a token, select the chain your token is on.
Ethereum Sepolia
We recommend launching a test DAO your first time.
Testnet
Ethereum


Step 2: Create Proposal
• Go to 'Proposals' tab
• Click 'Create New Proposal'
• Fill title & description and submit transaction
MetaMask
Account: 0x...abc
Network: Sepolia
Action: Confirm Transaction


Step 3: Vote on Proposal
• Select active proposal
• Choose 'Vote For' or 'Vote Against'
• Confirm vote in MetaMask
MetaMask
Account: 0x...abc
Network: Sepolia
Action: Confirm Transaction



Observation

Activity	Observation
Wallet Connection	MetaMask successfully connects to the governance dApp.
Proposal Creation	New proposal appears in the proposal list after on-chain confirmation.
Voting	Votes are recorded and visible on the DAO dashboard.
Execution	Passed proposals execute correctly on-chain through the DAO contract.

ASSESSMENT

Rubrics	Full Mark	Marks Obtained	Remarks
Concept	10		
Planning and Execution/ Practical Simulation/ Programming	10		
Result and Interpretation	10		
Record of Applied and Action Learning	10		
Viva	10		
Total	**50**		

Signature of the Student:
Name :
Regn. No.

Signature of the Faculty: