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

* Coding Phase: Pseudo Code / Flow Chart / Algorithm

ALGORITHM:

1.Initialize the environment using Remix IDE and connect MetaMask to the Sepolia Test Network.
2.Define a DAO (Decentralized Autonomous Organization) smart contract that allows proposal creation and voting by token holders.
3.Create mappings and data structures to store proposal details such as description, vote counts, and execution status.
4.Implement a function createProposal() allowing members to submit new governance ideas.
5.Add a function voteOnProposal() enabling participants to cast votes for or against a proposal.
6.Include logic to prevent double voting and restrict voting to valid DAO members.
7.Implement executeProposal() to finalize results once the voting period ends.
8.Compile and deploy the contract on Remix using the Sepolia test network.
9.Test proposal creation, voting, and execution functions to simulate DAO governance.

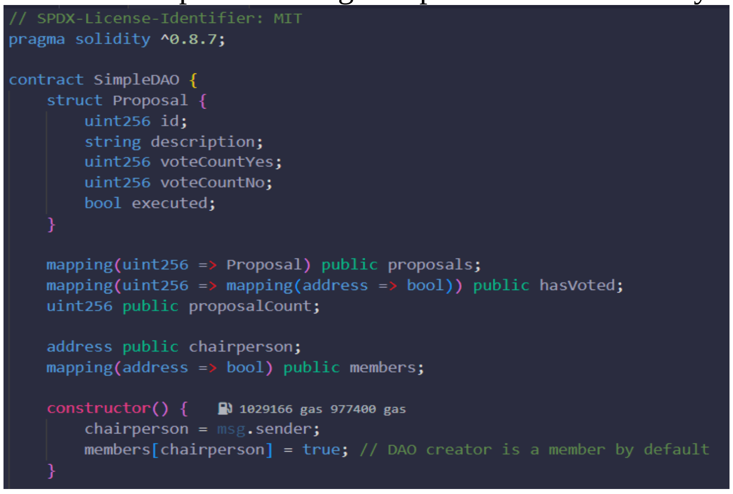
* Software used

1.Remix IDE
2.MetaMask Wallet
3.Solidity
4.Sepolia Testnet

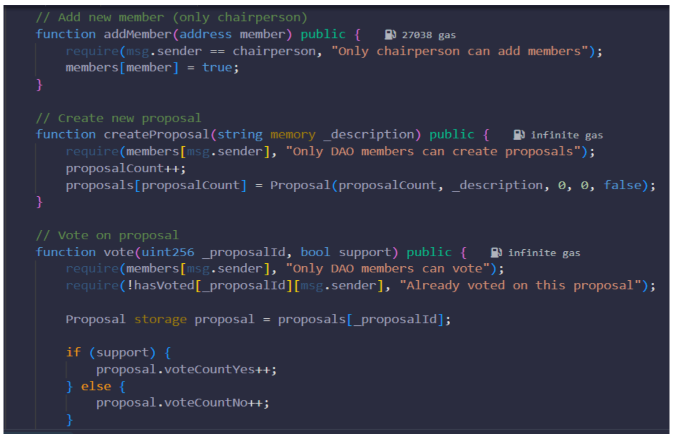
*As applicable according to the experiment.
Two sheets per experiment (10-20) to be used.

* Testing Phase: Compilation of Code (error detection)

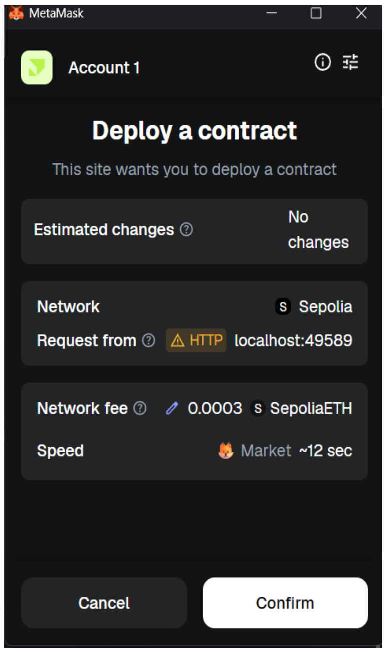
1.Deployed the SimpleDAO contract on Remix IDE using the Sepolia Test Network.
2.Created multiple governance proposals with the createProposal() function.
3.Used different member accounts to cast votes using the vote() function.
4.Verified correct vote tallies and proposal outcomes through the getProposalStatus() function.
5.Ensured duplicate voting was prevented successfully using the hasVoted mapping check.



```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.7;

contract SimpleDAO {
    struct Proposal {
        uint256 id;
        string description;
        uint256 voteCountYes;
        uint256 voteCountNo;
        bool executed;
    }

    mapping(uint256 => Proposal) public proposals;
    mapping(uint256 => mapping(address => bool)) public hasVoted;
    uint256 public proposalCount;

    address public chairperson;
    mapping(address => bool) public members;

    constructor() {    1029166 gas 977400 gas
        chairperson = msg.sender;
        members[chairperson] = true; // DAO creator is a member by default
    }
```



```
    // Add new member (only chairperson)
    function addMember(address member) public {    27038 gas
        require(msg.sender == chairperson, "Only chairperson can add members");
        members[member] = true;
    }

    // Create new proposal
    function createProposal(string memory _description) public {    infinite gas
        require(members[msg.sender], "Only DAO members can create proposals");
        proposalCount++;
        proposals[proposalCount] = Proposal(proposalCount, _description, 0, 0, false);
    }

    // Vote on proposal
    function vote(uint256 _proposalId, bool support) public {    infinite gas
        require(members[msg.sender], "Only DAO members can vote");
        require(!hasVoted[_proposalId][msg.sender], "Already voted on this proposal");

        Proposal storage proposal = proposals[_proposalId];

        if (support) {
            proposal.voteCountYes++;
        } else {
            proposal.voteCountNo++;
        }
```



MetaMask

Account 1

Deploy a contract

This site wants you to deploy a contract

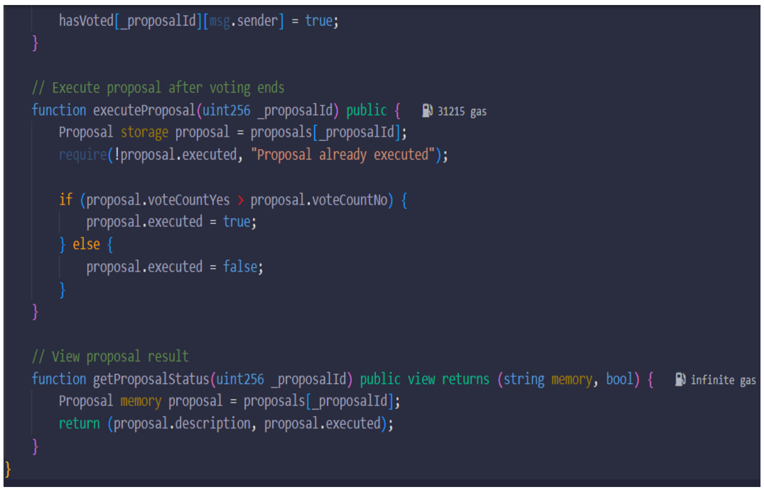
Estimated changes — No changes

Network — Sepolia

Request from — HTTP localhost:49589

Network fee — 0.0003 SepoliaETH

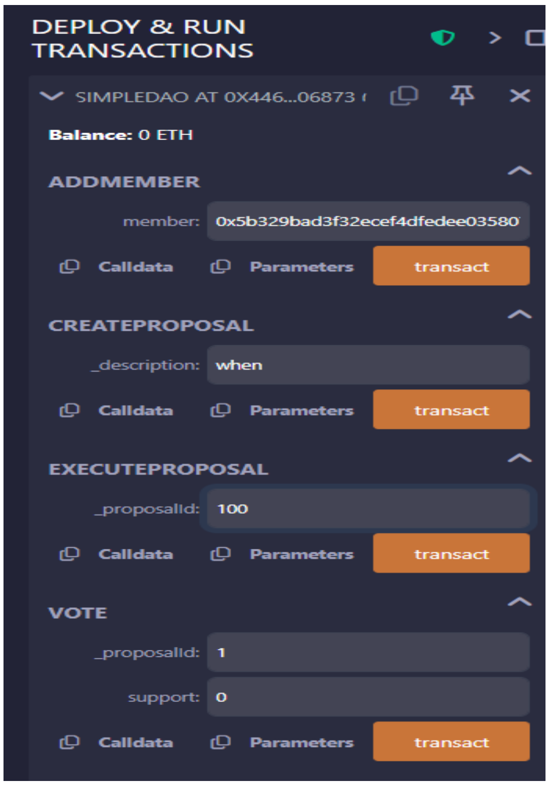
Speed — Market ~12 sec



```
        hasVoted[_proposalId][msg.sender] = true;
    }

    // Execute proposal after voting ends
    function executeProposal(uint256 _proposalId) public {    31215 gas
        Proposal storage proposal = proposals[_proposalId];
        require(!proposal.executed, "Proposal already executed");

        if (proposal.voteCountYes > proposal.voteCountNo) {
            proposal.executed = true;
        } else {
            proposal.executed = false;
        }
    }

    // View proposal result
    function getProposalStatus(uint256 _proposalId) public view returns (string memory, bool) {    infinite gas
        Proposal memory proposal = proposals[_proposalId];
        return (proposal.description, proposal.executed);
    }
}
```

*As applicable according to the experiment.
Two sheets per experiment (10-20) to be used.

* Implementation Phase: Final Output (no error)



DEPLOY & RUN TRANSACTIONS

SIMPLEDAO AT 0X446...06873

Balance: 0 ETH

ADDMEMBER

member: 0x5b329bad3f32ecef4dfedee03580

Calldata Parameters transact

CREATEPROPOSAL

_description: when

Calldata Parameters transact

EXECUTEPROPOSAL

_proposalId: 100

Calldata Parameters transact

VOTE

_proposalId: 1

support: 0

Calldata Parameters transact

* Observations

1.The DAO contract successfully demonstrated decentralized decision-making through voting mechanisms.

2.The simulation confirmed that proposals, voting, and execution were handled securely and transparently using smart contracts.

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 :

Signature of the Faculty:

Regn. No. :

******As applicable according to the experiment.*
Two sheets per experiment (10-20) to be used.