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 :

* Coding Phase: Pseudo Code / Flow Chart / Algorithm

Algorithm:

1.Create a governance smart contract with functions:
  -Create Proposal
  -Vote on Proposal
  -View Proposal Status
2.Deploy the contract using Remix and connect using MetaMask.
3.Cast votes from multiple Ethereum accounts to simulate DAO interaction.
4.Verify proposal acceptance or rejection based on votes.

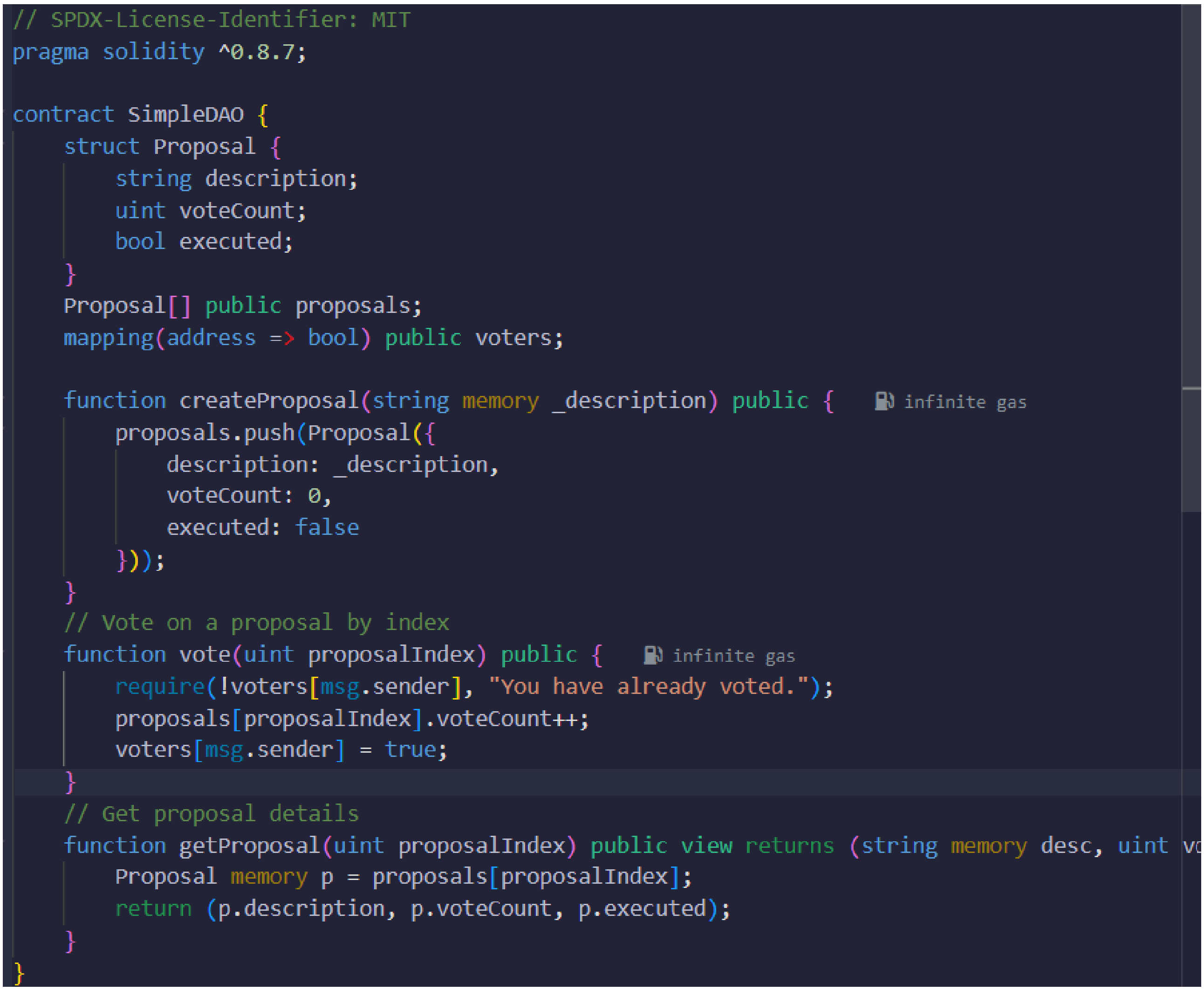
* Softwares used

1.Remix IDE
2.Solidity ^0.8.x
3.MetaMask Wallet
4.Sepolia Testnet (for multiple voting accounts).

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

* Testing Phase: Compilation of Code (error detection)

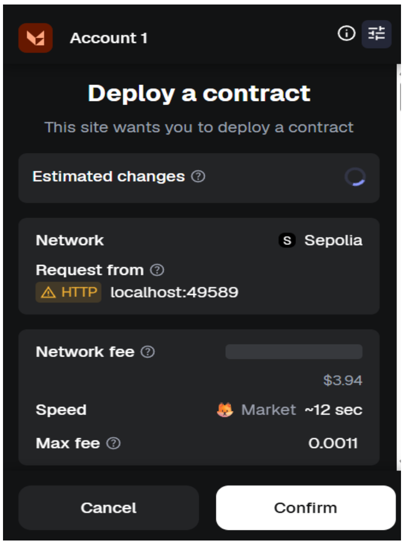
1.Contract deployed using Remix in JavaScript VM.
2.Created 2 proposals using the createProposal() function.
3.Simulated multiple votes from different accounts using vote() function.
4.Verified vote counts increased correctly using getProposal().
5.No double-voting was possible due to voters mapping checks.



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

contract SimpleDAO {
    struct Proposal {
        string description;
        uint voteCount;
        bool executed;
    }
    Proposal[] public proposals;
    mapping(address => bool) public voters;

    function createProposal(string memory _description) public {    infinite gas
        proposals.push(Proposal({
            description: _description,
            voteCount: 0,
            executed: false
        }));
    }
    // Vote on a proposal by index
    function vote(uint proposalIndex) public {    infinite gas
        require(!voters[msg.sender], "You have already voted.");
        proposals[proposalIndex].voteCount++;
        voters[msg.sender] = true;
    }
    // Get proposal details
    function getProposal(uint proposalIndex) public view returns (string memory desc, uint vo
        Proposal memory p = proposals[proposalIndex];
        return (p.description, p.voteCount, p.executed);
    }
}
```



Account 1

Deploy a contract

This site wants you to deploy a contract

Estimated changes

Network: S Sepolia

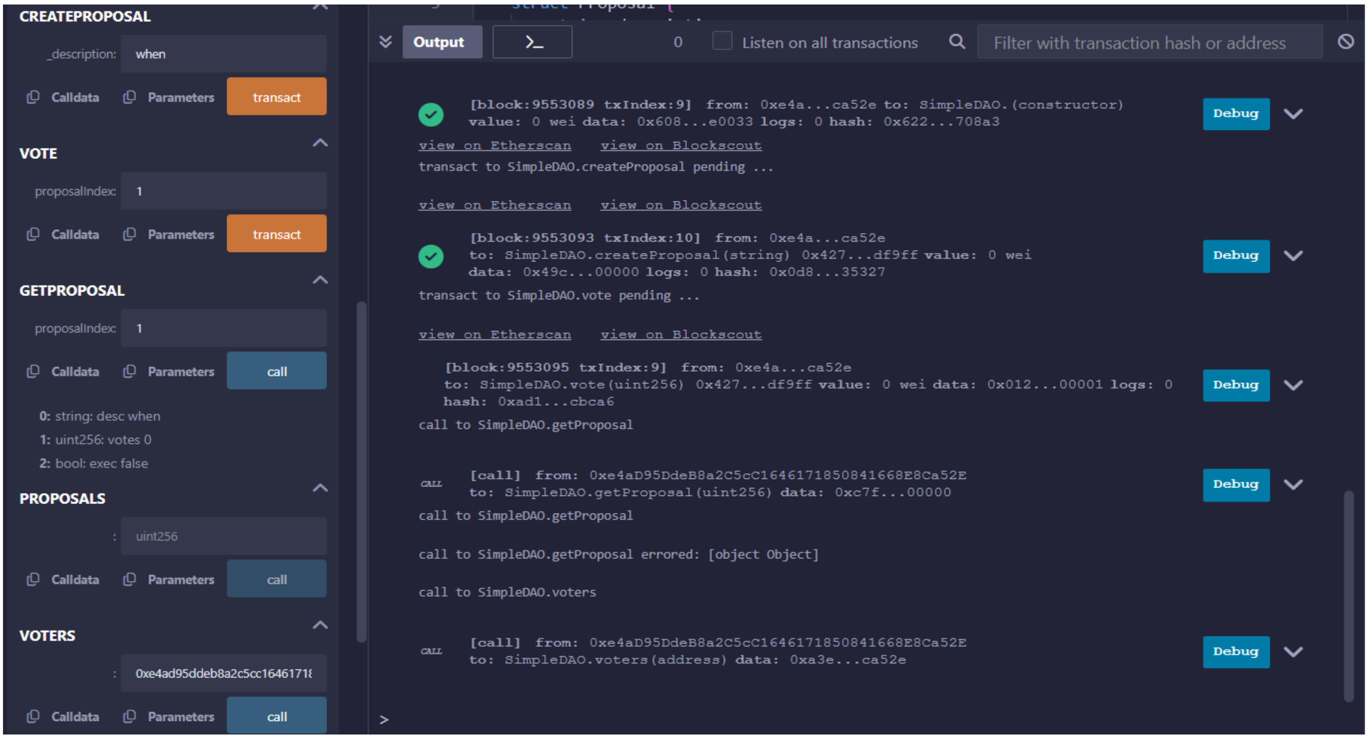
Request from: ⚠ HTTP localhost:49589

Network fee: $3.94

Speed: Market ~12 sec

Max fee: 0.0011

* Implementation Phase: Final Output (no error)



CREATEPROPOSAL
_description: when
Calldata | Parameters | transact

VOTE
proposalIndex: 1
Calldata | Parameters | transact

GETPROPOSAL
proposalIndex: 1
Calldata | Parameters | call

0: string: desc when
1: uint256: votes 0
2: bool: exec false

PROPOSALS
: uint256
Calldata | Parameters | call

VOTERS
: 0xe4ad95ddeb8a2c5cc164617118
Calldata | Parameters | call

Output
0
Listen on all transactions
Filter with transaction hash or address
Debug

```
[block:9553089 txIndex:9] from: 0xe4a...ca52e to: SimpleDAO.(constructor)
value: 0 wei data: 0x608...e0033 logs: 0 hash: 0x622...708a3
view on Etherscan   view on Blockscout
transact to SimpleDAO.createProposal pending ...

view on Etherscan   view on Blockscout
[block:9553093 txIndex:10] from: 0xe4a...ca52e
to: SimpleDAO.createProposal(string) 0x427...df9ff value: 0 wei
data: 0x49c...00000 logs: 0 hash: 0x0d8...35327
transact to SimpleDAO.vote pending ...

view on Etherscan   view on Blockscout
  [block:9553095 txIndex:9] from: 0xe4a...ca52e
  to: SimpleDAO.vote(uint256) 0x427...df9ff value: 0 wei data: 0x012...00001 logs: 0
  hash: 0xad1...cbca6
call to SimpleDAO.getProposal

CALL [call] from: 0xe4aD95DdeB8a2C5cC1646171850841668E8Ca52E
     to: SimpleDAO.getProposal(uint256) data: 0xc7f...00000
call to SimpleDAO.getProposal

call to SimpleDAO.getProposal errored: [object Object]

call to SimpleDAO.voters

CALL [call] from: 0xe4aD95DdeB8a2C5cC1646171850841668E8Ca52E
     to: SimpleDAO.voters(address) data: 0xa3e...ca52e

>
```

* Observations

1.DAO smart contracts enable decentralized and transparent decision-making.
2.Voting process is handled via blockchain, making it immutable and verifiable.
3.Basic DAO contracts can be extended with token-weighted voting and execution logic.
4.Testing in Remix with multiple accounts is useful for simulating real DAOs.

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:

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