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:

T o understand the **Decentralized Autonomous Organization (DAO)** concept by simulating a governance process involving proposal creation, voting, and decision execution using smart contracts.

Apparatus/Software Used:

Laptop/PC
Node.js and npm
Visual Studio Code (VS Code)
MetaMask Wallet (connected to Sepolia / Goerli Testnet)
Hardhat / Truffle Framework
Solidity Compiler

Theory/Concept:

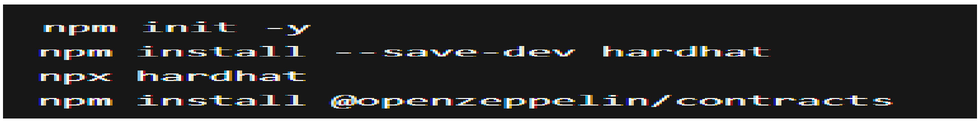
A DAO (Decentralized Autonomous Organization) is a blockchain-based organization governed by smar t contracts instead of centralized leadership.
Members of a DAO hold governance tokens that grant voting rights and decision-making power on proposals related to project development, fund allocation, or policy changes.
Key Concepts:
Governance Token:
Represents voting power within the DAO.
Proposal Creation:
Members can suggest changes or decisions, which are recorded as proposals on-chain.
Voting Mechanism:
Token holders vote “Yes” or “No” within a fixed time frame.
Quorum and Execution:
A proposal passes if it meets the required voting threshold, after which the smart contract executes the decision automatically.

Procedure

Setup Environment:

- Install Node.js, Hardhat, and MetaMask.
- Connect MetaMask to the Sepolia test network.

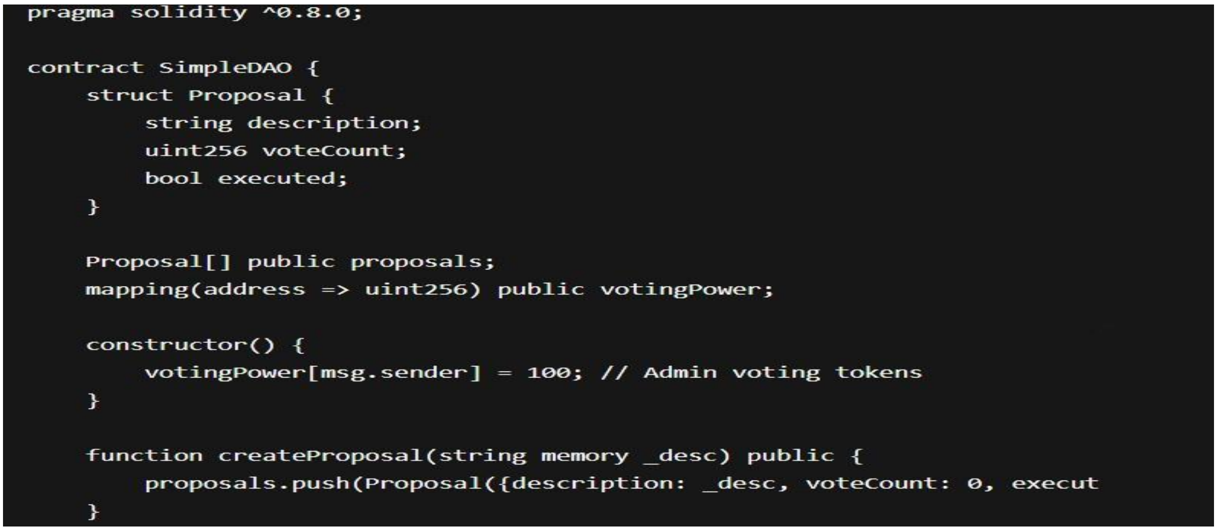
Initialize Hardhat Project:



```
npm init -y
npm install --save-dev hardhat
npx hardhat
npm install @openzeppelin/contracts
```

Write the DAO Smart Contract:

- Create SimpleDAO.sol in the contracts folder.
- Example:



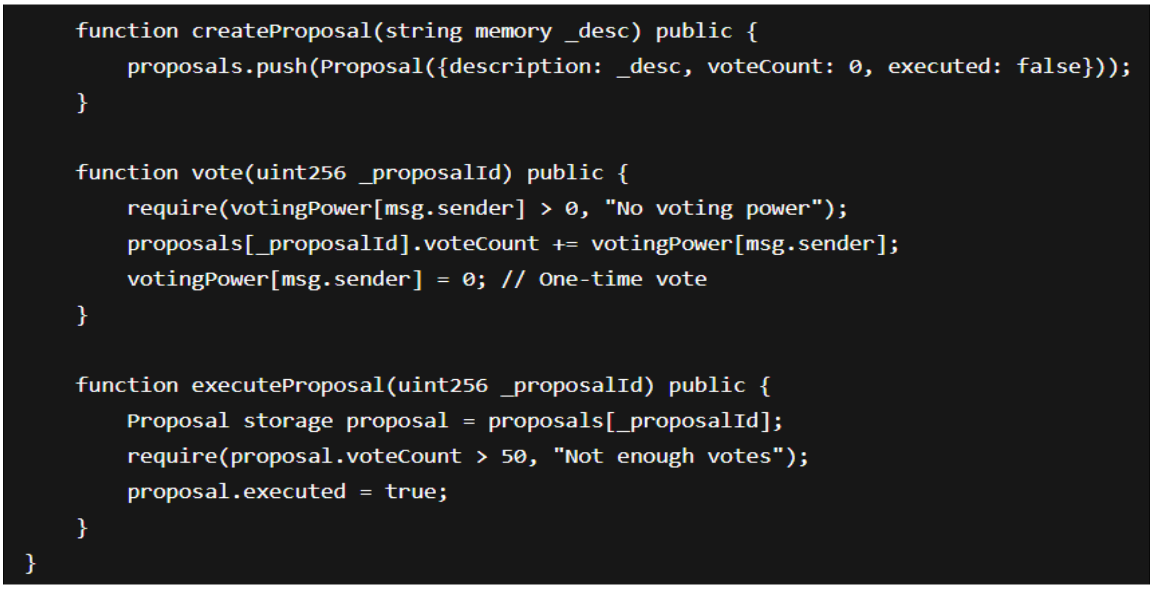
```
pragma solidity ^0.8.0;

contract SimpleDAO {
    struct Proposal {
        string description;
        uint256 voteCount;
        bool executed;
    }

    Proposal[] public proposals;
    mapping(address => uint256) public votingPower;

    constructor() {
        votingPower[msg.sender] = 100; // Admin voting tokens
    }

    function createProposal(string memory _desc) public {
        proposals.push(Proposal({description: _desc, voteCount: 0, execut
    }
```



```
    function createProposal(string memory _desc) public {
        proposals.push(Proposal({description: _desc, voteCount: 0, executed: false}));
    }

    function vote(uint256 _proposalId) public {
        require(votingPower[msg.sender] > 0, "No voting power");
        proposals[_proposalId].voteCount += votingPower[msg.sender];
        votingPower[msg.sender] = 0; // One-time vote
    }

    function executeProposal(uint256 _proposalId) public {
        Proposal storage proposal = proposals[_proposalId];
        require(proposal.voteCount > 50, "Not enough votes");
        proposal.executed = true;
    }
}
```

Compile and Deploy the Contract:

```
npx hardhat compile
npx hardhat run scripts/deploy.js --network sepolia
```

Simulate DAO Actions:

- Use Hardhat console or frontend UI to:
  - Create a proposal (e.g., “Fund new project”).
  - Vote using different wallet addresses.
  - Execute proposal after reaching quorum.

Record Governance Flow:

- Note how proposals and votes are tracked on-chain.
- Observe smart contract’s automatic execution after majority approval.

Observation Table:

- Brainstorming led to multiple creative blockchain gaming concepts.
- NFT-based ownership and token economies introduced unique monetization models.
- Polygon and Solana emerged as efficient choices due to lower transaction fees.
- Integration of smart contracts for in-game logic proved practical and secure.

ASSESSMENT

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

Signature of the Student:

Name :

Signature of the Faculty:

Regn. No.