**Practical 6**

**Aim:** Setup and deploy Voting Smart contract in solidity programming language using Remix IDE.

**Background Information:**

**1. Remix IDE:**

Remix is an online Integrated Development Environment for Ethereum smart contract development.

Remix IDE is a popular tool for developing and deploying smart contracts on Ethereum and other compatible blockchains.

It provides a user-friendly interface for writing, compiling, deploying, and testing smart contracts.

**2. Solidity:**

Solidity is a high-level programming language used for writing smart contracts on the Ethereum blockchain.

It is statically typed and supports inheritance, libraries, and complex user-defined types.

**3. Voting Smart Contract:**

A Voting Smart Contract allows users to vote for candidates in an election or for various options in a decision-making process.

It typically involves maintaining a list of candidates, recording votes, and ensuring security and transparency in the voting process.

**Installation steps:**

**Prerequisites:**

Remix IDE: Access the Remix Integrated Development Environment (IDE) here **(**<https://remix.ethereum.org/>).

Solidity Compiler: Ensure the Solidity compiler is available within Remix IDE.

To set up and deploy a Voting Smart Contract using Solidity programming language in Remix IDE, follow these steps:

**Steps:**

**1. Open Remix IDE:**

Navigate to the [Remix IDE website] (https://remix.ethereum.org/) in your web browser.

2. **Create a New File:**

- Click on the "+" button on the left-hand side panel to create a new file.

- Name the file `Voting.sol` or any desired name for your contract.

**3. Write the Smart Contract Code:**

-Write the Solidity code for the Voting Smart Contract in the created file.

-Ensure to define necessary variables, mappings, functions, and constructor (if needed).

-In the newly created file, write your Solidity code for the voting smart contract.

-Here's a basic example:

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

/\*\*

\* @title Storage

\* @dev Store & retrieve value in a variable

\* @custom:dev-run-script scripts/deploy\_with\_ethers.ts

\*/

contract Voting {

struct Candidate {

uint id;

string name;

uint voteCount;

}

mapping(uint => Candidate) public candidates;

mapping(address => bool) public hasVoted;

uint public candidatesCount;

constructor() {

addCandidate("Candidate 1");

addCandidate("Candidate 2");

}

function addCandidate(string memory \_name) private {

candidatesCount++;

candidates[candidatesCount] = Candidate(candidatesCount, \_name, 0);

}

function vote(uint \_candidateId) public {

require(\_candidateId > 0 && \_candidateId <= candidatesCount, "Invalid candidate ID");

require(!hasVoted[msg.sender], "You have already voted");

hasVoted[msg.sender] = true;

candidates[\_candidateId].voteCount++;

emit Voted(\_candidateId, candidates[\_candidateId].name);

}

event Voted(uint indexed candidateId, string candidateName);

}

**4. Compile the Contract:**

- Go to the "Solidity Compiler" tab in Remix IDE.

- Select the version of Solidity used in your contract (e.g., `0.8.0`).

- Click on the "Compile Voting.sol" button. Ensure there are no compilation errors.

**5. Deploy & Run Transactions:**

- Move to the "Deploy & Run Transactions" tab in Remix IDE.

- Under the "Deploy" section:

  - Select the contract you want to deploy from the dropdown list (e.g., `Voting`).

  - Adjust constructor arguments if your contract requires any.

- Click on the "Deploy" button.

**6.Generating UML Diagram:**

For creating a UML diagram:

-Right click on ‘Voting.sol’ file.

-then click on Generate UML

**Conclusion:**

- Testing: Ensure to test the contract extensively in Remix IDE using various scenarios.

- Security: Validate inputs, handle edge cases, and consider best security practices.

- Gas Fees: Be aware of gas costs when deploying and interacting with the contract on the Ethereum network.

Always thoroughly review, test, and validate your smart contract code before deploying it to a live Ethereum network to avoid any potential vulnerabilities or issues.

Thus, the setup and deploy Voting Smart contract in solidity programming language using Remix IDEis performed successfully