PRACTICAL: 6

AIM:

Implement a secure, transparent, and efficient e-voting system using the Ethereum blockchain that allows candidates to vote from any location, ensures vote integrity, and provides real-time verifiability.

THEORY:

In traditional elections, voting is often conducted manually or electronically through centralized systems, which can suffer from issues like tampering, lack of transparency, or restricted accessibility. By using blockchain technology, particularly Ethereum, we can build a secure, transparent, and decentralized e-voting system that ensures:

- Only eligible voters can vote
- Each person can vote only once
- Votes cannot be changed or deleted
- Results are visible and verifiable in real time

Smart contracts are self-executing code stored on the blockchain. In this voting system, a smart contract handles:

- Candidate registration
- Vote casting
- Vote counting
- Declaring results

This eliminates the need for a central authority to manage the election.

Ethereum is a blockchain platform that supports smart contracts. We used it to deploy our voting contract using Solidity, Ethereum's programming language.

Remix is an online tool used for writing, deploying, and testing Ethereum smart contracts. It simulates the blockchain locally, allowing developers and students to experiment without using real cryptocurrency.

- Only the admin (the person who deployed the contract) can add candidates.
- Each voter (identified by their Ethereum address) can vote only once.
- All votes are stored on the blockchain permanently.
- Anyone can call getResults() to see who is winning.

CODE:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract Voting {
    // Structure to store information about candidates
    struct Candidate {
        uint id;
        string name;
        uint voteCount;
    }

    // Mapping from candidate ID to Candidate struct
    mapping(uint => Candidate) public candidates;
    mapping(address => bool) public voters;
    uint public candidatesCount;
    uint public totalVotes;
    address public admin;

    // Event to notify when a vote is cast
    event Voted(address indexed voter, uint candidateId);

    constructor() {
        admin = msg.sender; // The person deploying the contract is the admin
    }

    // Modifier to restrict access to the admin
    modifier onlyAdmin() {
        require(msg.sender == admin, "Only the admin can perform this action");
        _;
    }

    // Function to add a new candidate
    function addCandidate(string memory _name) public onlyAdmin {
        candidatesCount++;
        candidates[candidatesCount] = Candidate(candidatesCount, _name, 0);
    }

    // Function to allow voters to cast their vote
    function vote(uint _candidateId) public {
        // Ensure the voter has not already voted
        require(!voters[msg.sender], "You have already voted.");
        require(_candidateId > 0 && _candidateId <= candidatesCount, "Invalid
candidate.");

        voters[msg.sender] = true;
        candidates[_candidateId].voteCount++;
        totalVotes++;
```

```
        emit Voted(msg.sender, _candidateId);
    }

    // Function to get the results of the election
    function getResults() public view returns (string memory winnerName, uint
winnerVoteCount) {
        uint winningVoteCount = 0;
        for (uint i = 1; i <= candidatesCount; i++) {
            if (candidates[i].voteCount > winningVoteCount) {
                winningVoteCount = candidates[i].voteCount;
                winnerName = candidates[i].name;
                winnerVoteCount = winningVoteCount;
            }
        }
    }
}
```

OUTPUT:

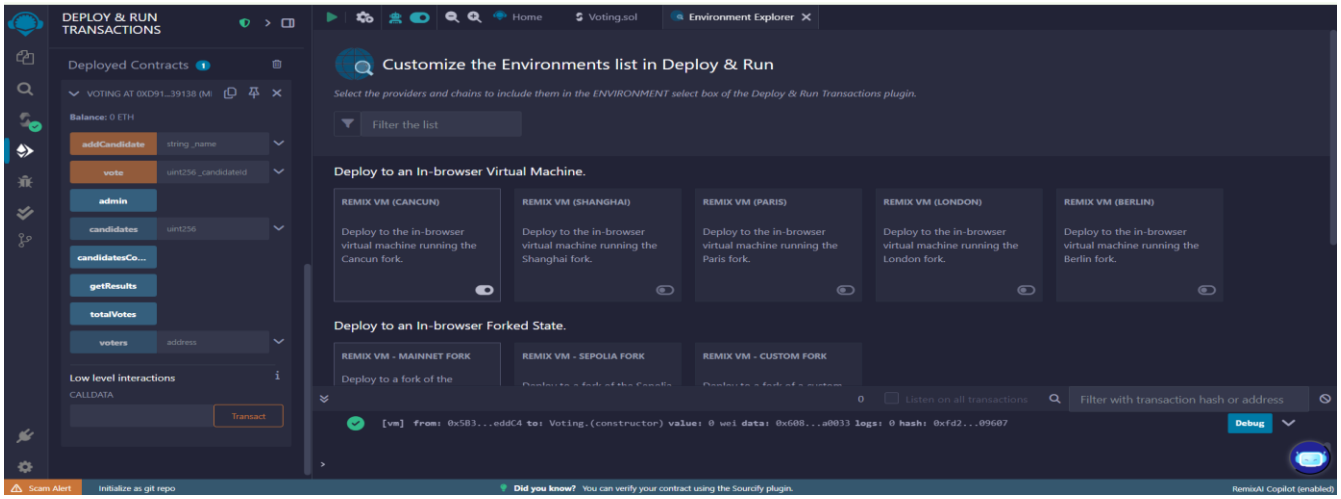

Figure 1:Deploy the contract

Figure 2:Add candidate

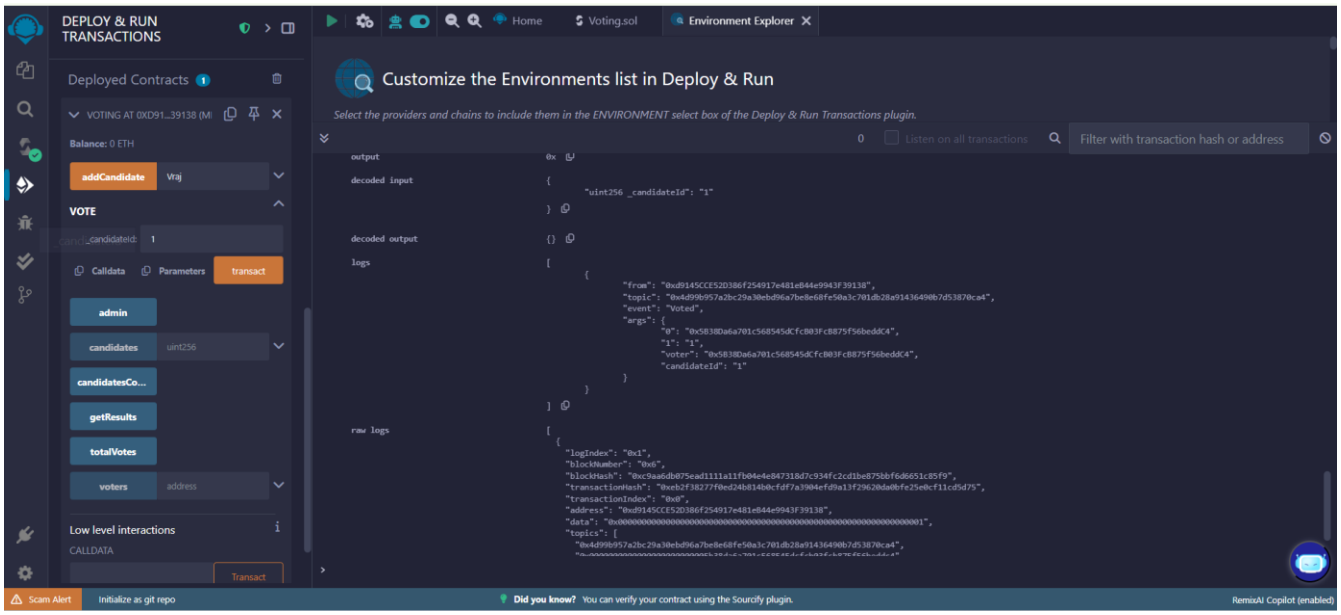

Figure 3:Vote for candidate

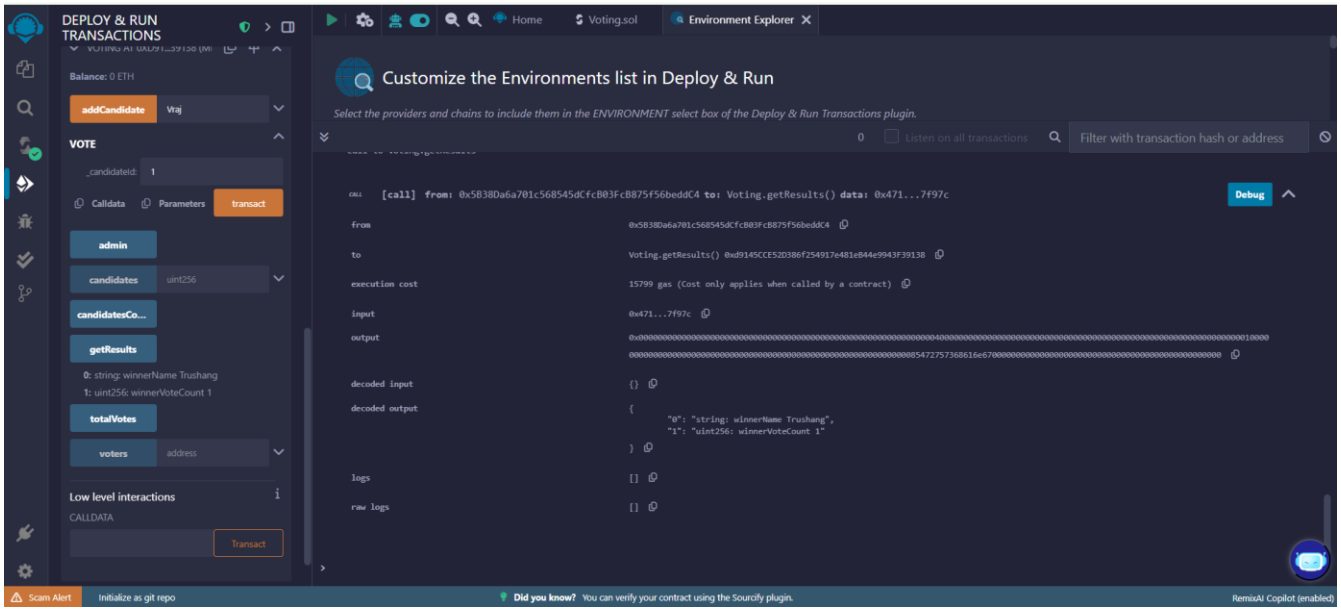

Figure 4:Get the result

LATEST APPLICATIONS:

- National-Level Pilots
- University & Student Elections
- Online Polling & Contests

LEARNING OUTCOME:

In this practical, I developed and deployed a smart contract for an e-voting system using Ethereum and Remix IDE. I added candidates, cast votes, and retrieved results. This enhanced my understanding of blockchain-based voting, Solidity programming, and decentralized systems, while gaining hands-on experience with smart contract deployment and testing.

REFERENCES:

1. https://remix.ethereum.org/