**Blockchain Technology Lab**

**Lab – 8**

**Aim : Create a application based smart contract in Remix-IDE**

**Theory:**

This smart contract, MessageBoard, allows users to post and retrieve messages on the Ethereum blockchain. Each message includes the content, a custom senderID provided by the user, and a timestamp. Users can interact with the contract to post messages, retrieve the latest message, or filter messages by a specific senderID. The contract stores all messages in a public array, ensuring transparency and immutability. Designed for decentralized communication, it showcases how blockchain can be used for persistent and verifiable message storage.

**Steps:**

**1.** Open Remix-IDE, create a new workspace and use the below code:

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract MessageBoard {

    struct Message {

        string text;

        string senderID; // Manually entered sender ID

        uint256 timestamp;

    }

    Message[] public messages; // Array to store all messages

    event MessagePosted(string text, string senderID, uint256 timestamp);

    // Function to post a new message with a senderID

    function postMessage(string calldata newText, string calldata newSenderID) public {

        Message memory newMessage = Message({

            text: newText,

            senderID: newSenderID,

            timestamp: block.timestamp

        });

        messages.push(newMessage);

        emit MessagePosted(newText, newSenderID, block.timestamp);

    }

    // Function to get the latest message

    function getMessage() public view returns (string memory, string memory, uint256) {

        require(messages.length > 0, "No messages found");

        Message memory latest = messages[messages.length - 1];

        return (latest.text, latest.senderID, latest.timestamp);

    }

    // Function to get all messages by a specific senderID

    function getMessagesBySenderID(string calldata senderID) public view returns (Message[] memory) {

        uint256 count = 0;

        // Count how many messages the senderID has

        for (uint256 i = 0; i < messages.length; i++) {

            if (keccak256(bytes(messages[i].senderID)) == keccak256(bytes(senderID))) {

                count++;

            }

        }

        // Create an array to store messages from the specified senderID

        Message[] memory senderMessages = new Message[](count);

        uint256 index = 0;

        for (uint256 i = 0; i < messages.length; i++) {

            if (keccak256(bytes(messages[i].senderID)) == keccak256(bytes(senderID))) {

                senderMessages[index] = messages[i];

                index++;

            }

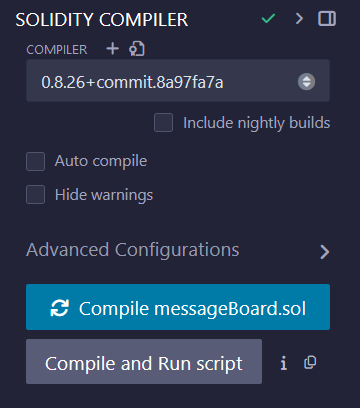
        }

        return senderMessages;

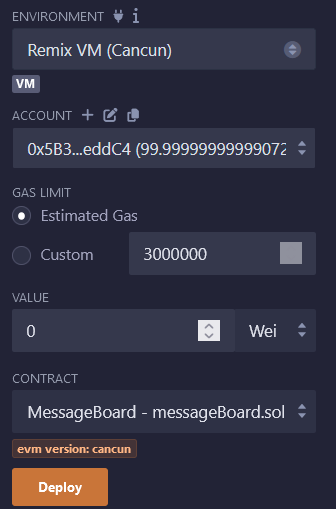
    }

}

**2.** Compile the Contract



**3.** Switch to the Deploy & Run Transactions tab and click deploy.



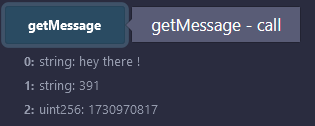
**4.** Interact with the contract

1. Posting a message

A screenshot of a chat

Description automatically generated

2. Receiving latest message



3. get message by senderID

A screenshot of a computer

Description automatically generated