St. Francis Institute of Technology, Mumbai-400 103
Department Of Information Technology

A.Y. 2025-2026
Class: BE-ITA/B, Semester: VIII
Subject: BlockChain Lab

Experiment – 3

1. **Aim:** To implement smart contract using Remix IDE and deploy it using MetaMask.
2. **Objective:** To …
   - explore working of Local ethereum network.
   - explore working of MetaMask wallet
   - test a smart contract by using Remix IDE
3. **Lab outcome:** After performing the experiment, the students will be able to **implement** smart contracts in Ethereum using different development frameworks (PO3, PSO2, BL3)
4. **Prerequisite:**
   - Fundamental knowledge of blockchain
   - Knowledge of the Ethereum platform and Remix IDE
   - Familiarity with the Solidity programming language

5. **Requirements:** The following are the requirements –
Remix IDE, MetaMask wallet etc.

6. **Pre-Experiment Theory:**

What is MetaMask?

MetaMask is a browser plugin that serves as a cryptocurrency wallet and enables users to store Ether and other ERC-20 tokens. It also enables users to access the Web 3 ecosystem of dapps. It relies on browser integration and good design to serve as one of the main gateways to the world of Web3, decentralized finance (DeFi) and NFTs.

7. **Laboratory Exercise Steps to be implemented.**

To Follow the procedure given below to build smart contract in Remix IDE

1. Download and install the official MetaMask extension/ plugin.
2. Open Remix IDE in Google Chrome.
3. Write smart contract by creating new file under contract folder, with .sol extension.
4. Write your contract code using solidity language.
5. Click on compiler icon, choose compiler version or keep default and click on compile button.
6. For deployment…

In Remix IDE choose environment Injected Provider- MetaMask and then deploy the said contract.

7. Under deployed contract, get the output of your contract.

Program Code

Write a smart contract StudentMarks.sol to create student database with attributes, id, name, dept, sub1marks, sub2Marks etc. and deploy it using Remix and MetaMask.

Post Experimental Exercise-

Questions:

- List down the details of different MetaMask networks

Network	Currency	Type	Key Features
Ethereum Mainnet	ETH	Mainnet	Original blockchain, most secure, highest gas fees, largest DeFi ecosystem
Sepolia	SepoliaETH	Testnet	Ethereum testnet for developers, free test ETH from faucets
Polygon (Matic)	MATIC	Sidechain	Low fees (~$0.01), fast transactions, EVM-compatible, PoS consensus
Binance Smart Chain	BNB	Mainnet	Low fees, high speed, centralized validators, DeFi & NFT friendly
Avalanche C-Chain	AVAX	Mainnet	Fast finality, subnet architecture, eco-friendly consensus

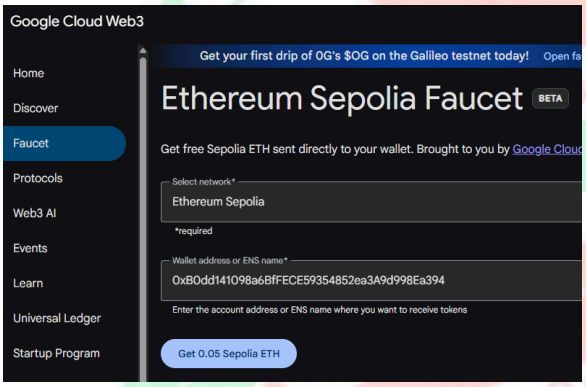
- Create any one game using smart contracts and deploy it.

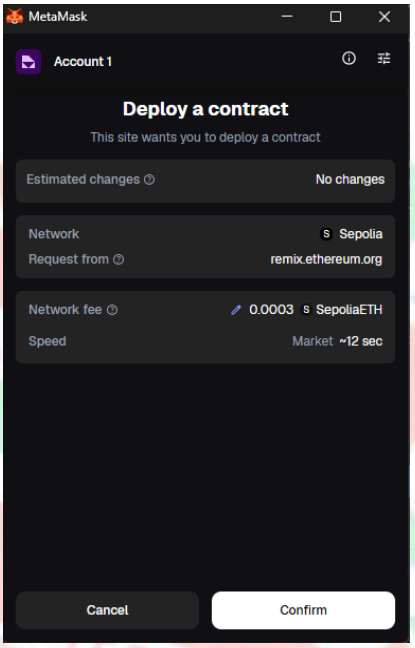
A. Results/Observations/Program output:

Present the program input/output results if any and comment on the same.

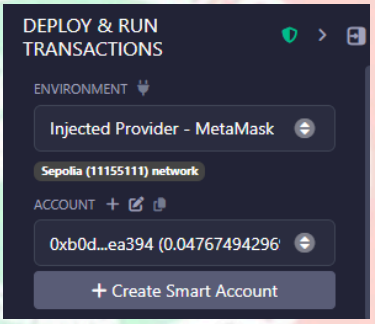
B. Conclusion:

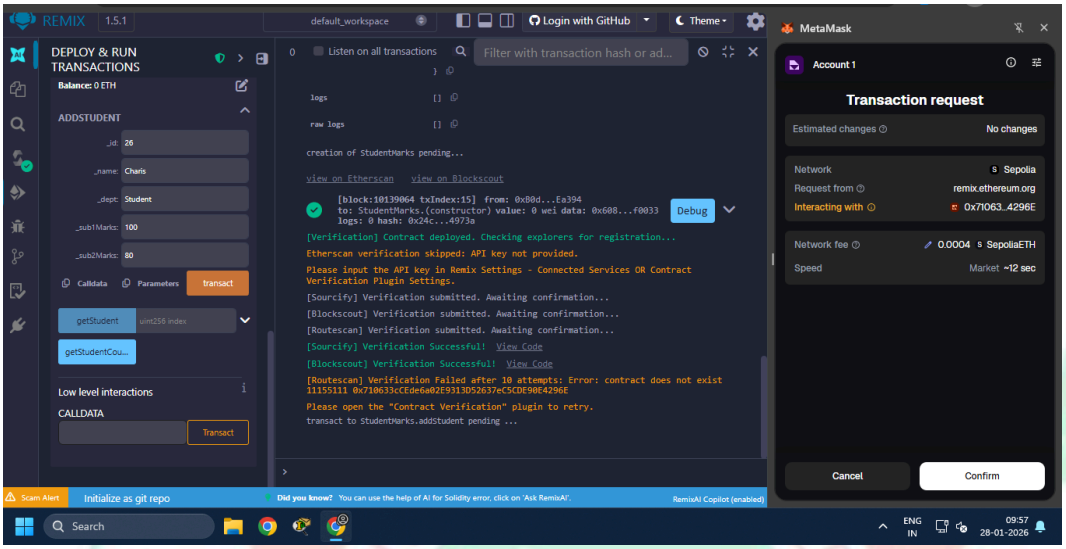
- Write what was performed in the experiment.
- Write which tools you used to perform the experiment
- Write what you inferred from the output obtained.

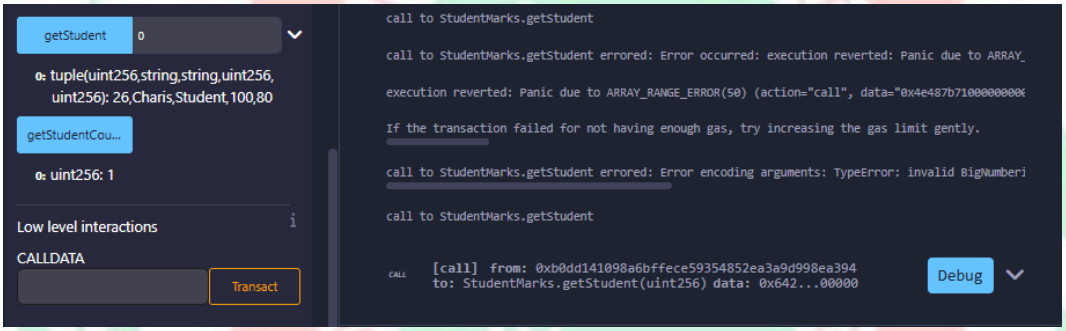





```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract StudentMarks {
    struct Student {
        uint256 id;
        string name;
        string dept;
        uint256 sub1Marks;
        uint256 sub2Marks;
    }

    Student[] private students;

    function addStudent(
        uint256 _id,
        string memory _name,
        string memory _dept,
        uint256 _sub1Marks,
        uint256 _sub2Marks
    ) public {
        students.push(Student({
            id: _id,
            name: _name,
            dept: _dept,
            sub1Marks: _sub1Marks,
            sub2Marks: _sub2Marks
        }));
    }

    function getStudent(uint256 index) public view returns (Student memory) {
        return students[index];
    }

    function getStudentCount() public view returns (uint256) {
        return students.length;
    }
}
```

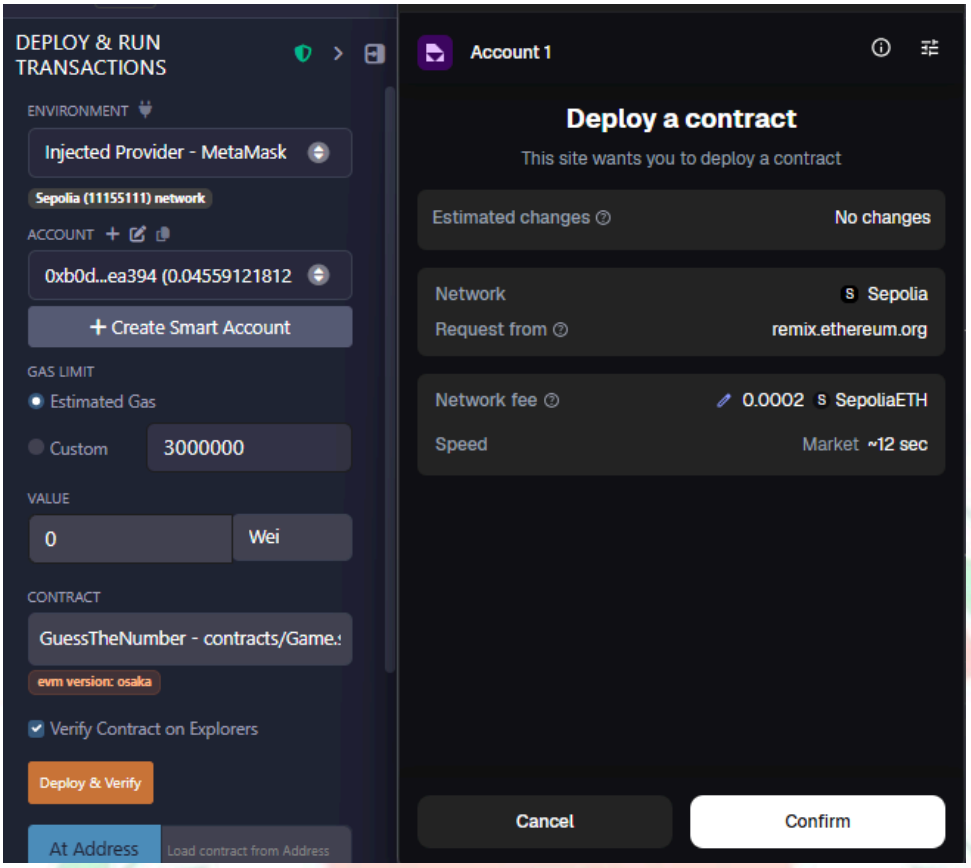


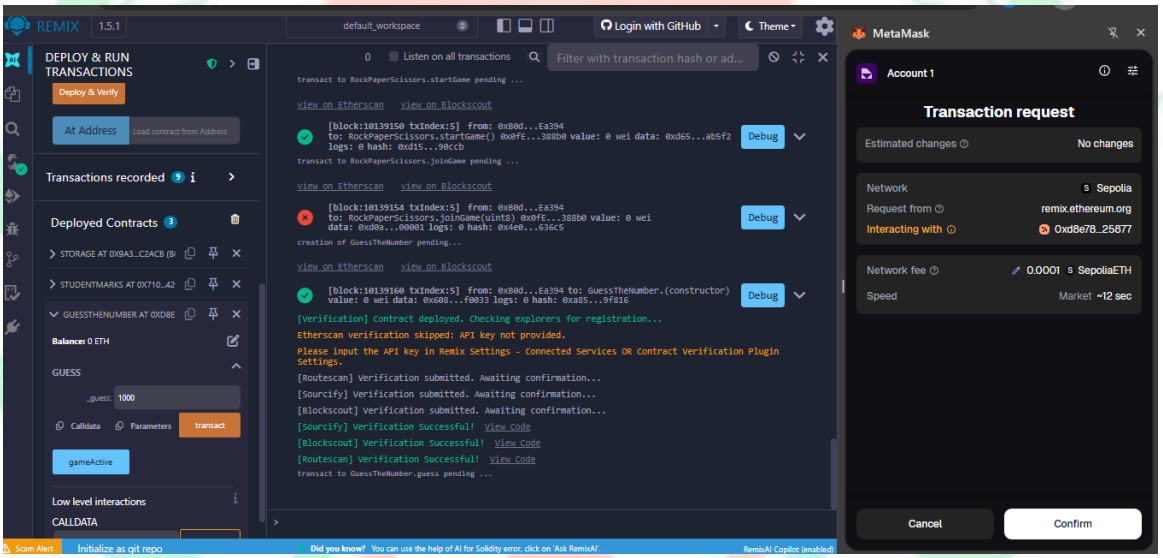




POST EXP- Game

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

contract GuessTheNumber {
    uint private secretNumber;
    bool public gameActive;

    event Result(string message);

    constructor() {
        secretNumber = block.timestamp % 10 + 1; // Random 1-10
        gameActive = true;
    }

    function guess(uint _guess) external {
        require(gameActive, "Game over");
        if (_guess == secretNumber) {
            emit Result("You win!");
        } else {
            emit Result(_guess > secretNumber ? "Too high!" : "Too low!");
        }
    }
}
```





```
logs                    [
                            {
                                "from": "0xd8e788b3F5CE2D6719EF82AAC325198355D25877",
                                "topic":
"0xf5250111a819ab87d27490e1837d465f4bc000cfd9316fc90fde27c791d8026d",
                                "event": "Result",
                                "args": {
                                    "0": "Too high!"
                                }
                            }
```