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 : Connect the dots – Ether.js find MetaMask UI

Objective/Aim:

To understand the the basics of how to connect the frontend and the smart contract using web3.js

Apparatus/Software Used:

- Laptop / PC
- Remix IDE
- Metamask
- Etherscan

Theory/Concept:

Introduction

- Blockchain Data Access – Web3.js allows developers to read data stored on the blockchain (Ethereum, BSC, Polygon, etc.) such as account balances, transaction details, and block info.
- No Gas Fees for Reading – Reading (calling eth_call) is free because it doesn't change the blockchain state, unlike transactions (eth_sendTransaction).
- Functions for Reading – Common Web3.js methods include web3.eth.getBalance(), web3.eth.getBlock(), web3.eth.getTransaction(), and contract read functions using myContract.methods.methodName().call().
- Smart Contract Interaction – You can connect to deployed smart contracts using their ABI and address, then read variables and return values from functions without modifying the state.
- Use Cases – Reading is used for showing token balances, NFT metadata, transaction history, block details, and DApp dashboards.

Procedure:

Step 1.Open Remix IDE and write the SimpleStorage.sol smart contract.
Step 2.Compile the smart contract using the Solidity compiler in Remix.
Step 3.Copy the generated ABI after successful compilation.
Step 4.Deploy the contract to the Sepolia Testnet using MetaMask.
Step 5.Copy the deployed contract address.
Step 6.Create a React frontend project using create-react-app.
Step 7.Add the contract address and network information to the .env file.
Step 8.Install ether.js to interact with the blockchain.
Step 9.Use the ABI and contract address to connect the frontend with the smart contract.
Step 10.Design the UI in App.js using Web3.js to store and retrieve data.

```
1  // SPDX-License-Identifier: MIT
2  pragma solidity ^0.8.0;
3  contract Counter {
4      uint public count;
5      constructor(uint _start) {    infinite gas 132000 gas
6          count = _start;
7      }
8      function increment() public {    infinite gas
9          count += 1;
10     }
11     function decrement() public {    infinite gas
12         require(count > 0, "Counter is already at zero");
13         count -= 1;
14     }
15     function getCount() public view returns (uint) {    2453 gas
16         return count;
17     }
18 }
```

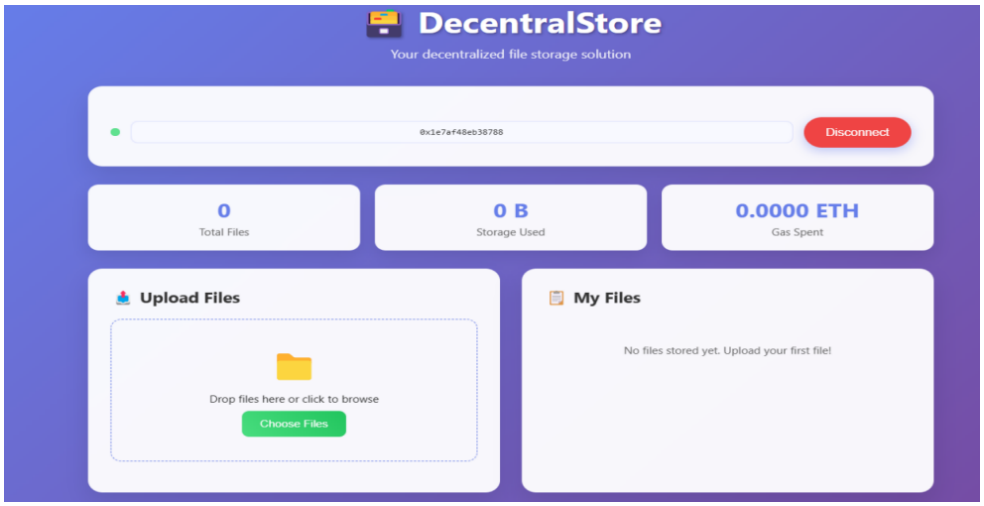
Smart contract

```
The following libraries are accessible:
• web3.js
• ethers.js

Type the library name to see available commands.
creation of SimpleStorage pending...

view on Etherscan   view on Blockscout

[block:8917149 txIndex:33] from: 0xe4a...ca52e to: SimpleStorage.(constructor) value: 0 wei data: 0x608...0007b logs: 0 hash: 0xcb0...491dc   Debug
```



Observation

1.Ethers.js provides a lightweight and modular approach for interacting with Ethereum smart contract.
2. It simplifies wallet connection and contract function calls using a clean and modern syntax.
3.The library ensures better security and improved developer experience compared to older Web3.js practices.

ASSESSMENT

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

Signature of the Student:

Signature of the Faculty:

Name :

Regn. No. :