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:

The main objective is to establish a connection between a basic web application (DApp) and the **Ethereum blockchain** via the **MetaMask** wallet and the **Ethers.js** library, enabling the DApp to read and display information such as the connected user's wallet address and their Ether balance.

Apparatus/Software Used:

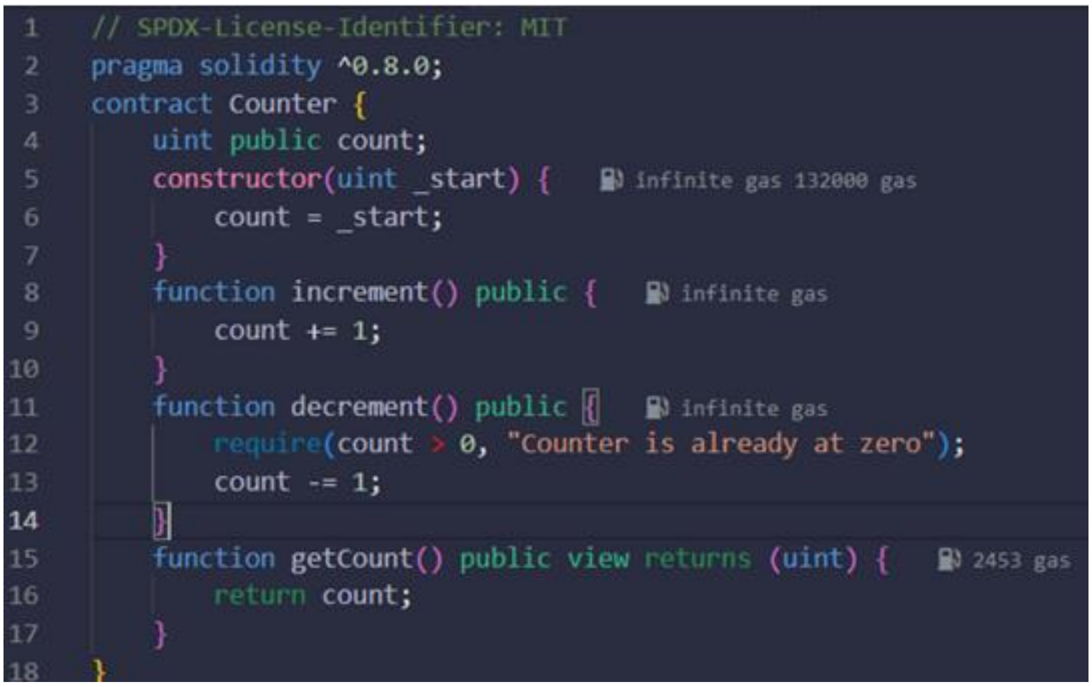
- MetaMask Wallet
- Remix IDE
- Brave browser

Theory/Concept:

- **Blockchain Abstraction: Ethers.js** is a JavaScript library that provides an easy-to-use interface to interact with the Ethereum blockchain and its compatible networks.
- **Provider:** In Ethers.js, a **Provider** is an abstraction for a connection to the Ethereum Network, offering **read-only access** to the blockchain and its status (e.g., fetching block numbers or balances).
- **Signer:** A **Signer** is an abstraction of an Ethereum Account, which can be used to **sign transactions** and messages, representing the user's wallet/private key.
- **MetaMask:** MetaMask is a browser extension that acts as a secure Ethereum wallet and injects a global window.ethereum object into the browser. This object serves as the **Web3 Provider** that Ethers.js can use to connect to the network and request the user's permission to act as the signer.
- **Connection Process:** The DApp checks for window.ethereum (MetaMask) and, if present, uses it to create a Web3Provider (in Ethers.js v5) or BrowserProvider (in Ethers.js v6). It then calls provider.send("eth_requestAccounts", []) or provider.getSigner() to prompt the user to connect and gain access to the account address and signing capabilities.

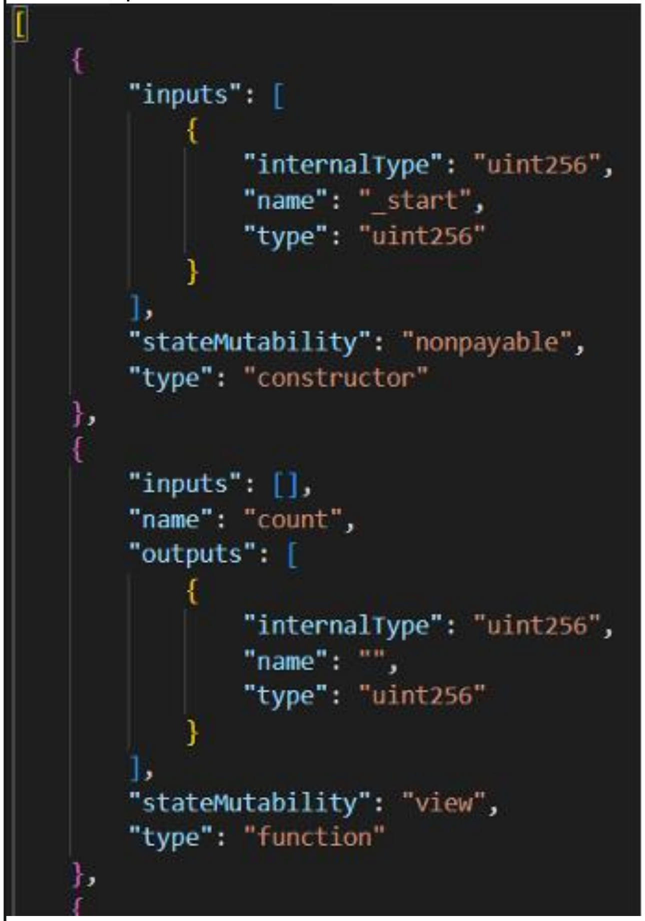
Procedure:

- **Project Setup:**
- Create a new project directory and initialize a Node.js project (npm init -y).
- Install Ethers.js: npm install ethers.
- Create an index.html file for the UI and a JavaScript file (e.g., main.js) for the logic.
- **MetaMask Setup:**
- Install the **MetaMask** browser extension and set up an account.
- Switch the network in MetaMask to a test network (e.g., Sepolia) and ensure the account has test Ether.
- **UI Development (HTML):**
- Create a simple UI with a "Connect Wallet" button and elements to display the connected **Wallet Address** and **Balance**.
- **Ethers.js Connection Logic (JavaScript):**
- In the JavaScript file, define an asynchronous function (e.g., connectWalletHandler) to handle the connection.
- **Check for MetaMask:** Use an if (window.ethereum) check to see if MetaMask is installed.
- **Create Provider:** If MetaMask exists, create an Ethers.js Web3Provider (or BrowserProvider): const provider = new ethers.providers.Web3Provider(window.ethereum);.
- **Request Accounts:** Prompt the user to connect their wallet: await provider.send("eth_requestAccounts", []);.
- **Get Signer/Address:** Obtain the Signer object and the user's address: const signer = provider.getSigner(); and const address = await signer.getAddress();.
- **Get Balance:** Fetch the balance of the connected address using the Provider/Signer and format it for display: const balance = await provider.getBalance(address);.
- **Display Results:** Update the UI elements with the retrieved Address and Balance.
- **Execution and Observation:**
- Open the index.html file in the browser.
- Click the "Connect Wallet" button.
- Approve the connection request in the MetaMask prompt.

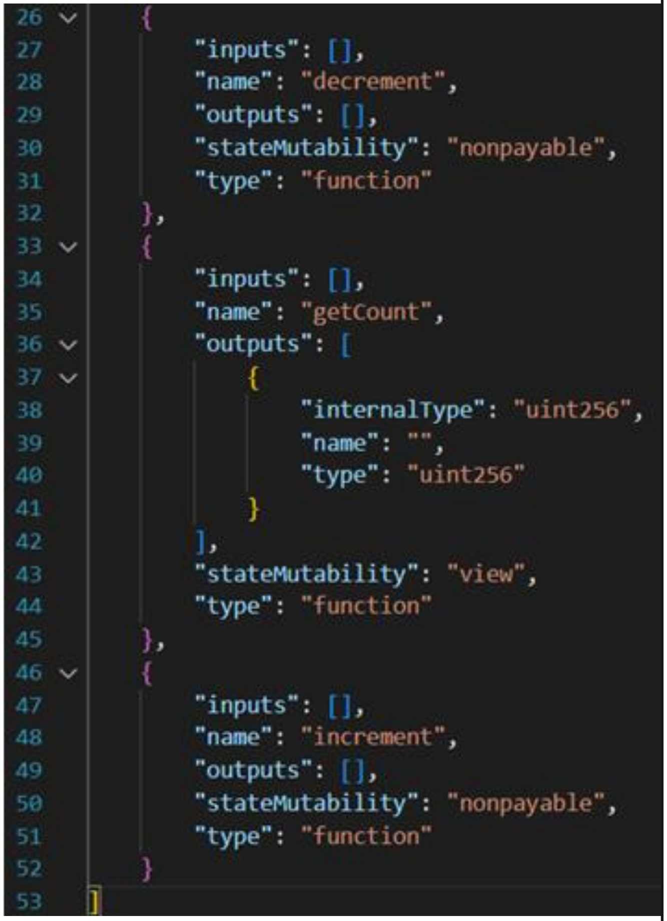


```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract Counter {
    uint public count;
    constructor(uint _start) {    infinite gas 132000 gas
        count = _start;
    }
    function increment() public {    infinite gas
        count += 1;
    }
    function decrement() public {    infinite gas
        require(count > 0, "Counter is already at zero");
        count -= 1;
    }
    function getCount() public view returns (uint) {    2453 gas
        return count;
    }
}
```

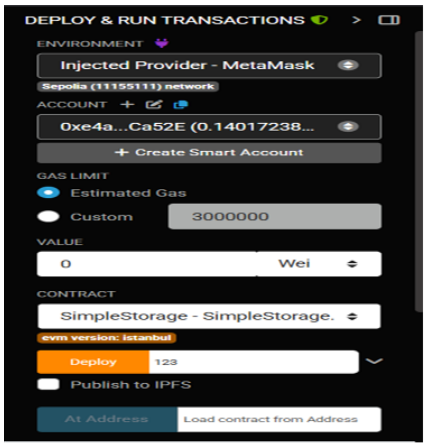
After compile the smart contract there is a ABI of the smart contract



```
[
    {
        "inputs": [
            {
                "internalType": "uint256",
                "name": "_start",
                "type": "uint256"
            }
        ],
        "stateMutability": "nonpayable",
        "type": "constructor"
    },
    {
        "inputs": [],
        "name": "count",
        "outputs": [
            {
                "internalType": "uint256",
                "name": "",
                "type": "uint256"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    },
    {
```



```
    {
        "inputs": [],
        "name": "decrement",
        "outputs": [],
        "stateMutability": "nonpayable",
        "type": "function"
    },
    {
        "inputs": [],
        "name": "getCount",
        "outputs": [
            {
                "internalType": "uint256",
                "name": "",
                "type": "uint256"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    },
    {
        "inputs": [],
        "name": "increment",
        "outputs": [],
        "stateMutability": "nonpayable",
        "type": "function"
    }
]
```



DEPLOY & RUN TRANSACTIONS

ENVIRONMENT: Injected Provider - MetaMask

Sepolia (11155111) network

ACCOUNT: 0xe4a...Ca52E (0.14017238...

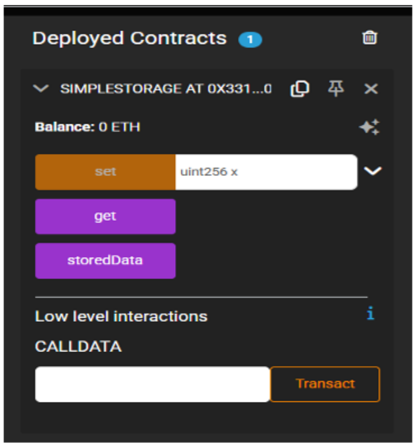
GAS LIMIT: Estimated Gas [✓] / Custom 3000000

VALUE: 0 Wei

CONTRACT: SimpleStorage - SimpleStorage.

evm version: istanbul

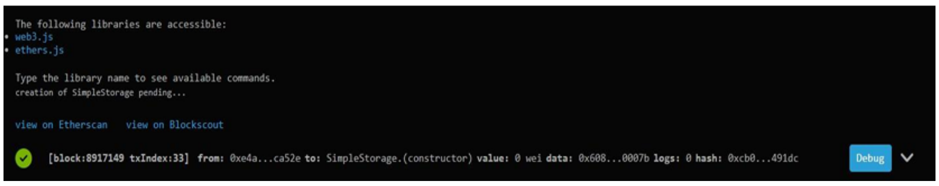
Deploy 123



Deployed Contracts 1

SIMPLESTORAGE AT 0X331...0

Balance: 0 ETH



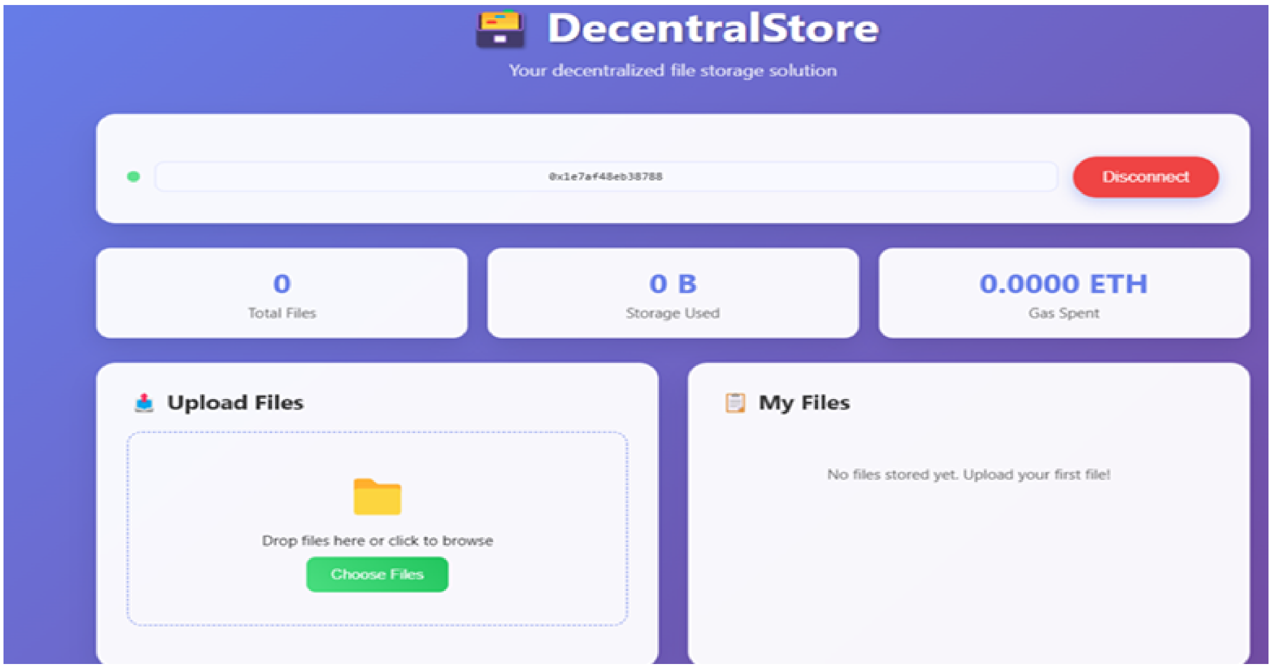
```
The following libraries are accessible:
• web3.js
• ethers.js

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
```

In this Smart contract we have two accessible libraries one is ether.js and another is web3.js we have to work on ether.js


DecentralStore

Your decentralized file storage solution

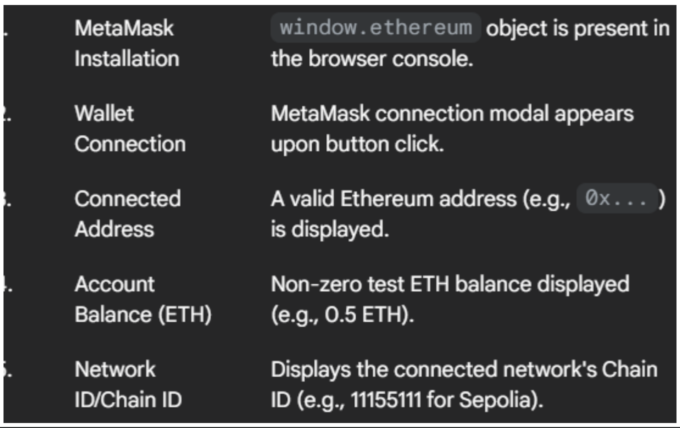
0x1e7af48eb38788

0	0 B	0.0000 ETH
Total Files	Storage Used	Gas Spent

No files stored yet. Upload your first file!

This is the frontend

Observation



MetaMask Installation	`window.ethereum` object is present in the browser console.
Wallet Connection	MetaMask connection modal appears upon button click.
Connected Address	A valid Ethereum address (e.g., `0x...`) is displayed.
Account Balance (ETH)	Non-zero test ETH balance displayed (e.g., 0.5 ETH).
Network ID/Chain ID	Displays the connected network's Chain ID (e.g., 11155111 for Sepolia).

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:

Name :

Signature of the Faculty:

Regn. No. :