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 : Web3 Connect – Contract Calls via Frontend

Objective/Aim:

To understand how to connect a frontend application to an Ethereum blockchain using **Web3.js**, and read data from a deployed smart contract (e.g., a public variable or a view function).

Apparatus/Software Used:

- Node.js and npm
- MetaMask Wallet (Testnet)
- Ganache (local blockchain) or public testnet (e.g., Sepolia)
- VS Code (or any IDE)
- Web3.js library
- A deployed smart contract with an ABI and address

Theory/Concept:

Web3.js is a JavaScript library that allows interaction with Ethereum nodes using HTTP or WebSocket. It acts as a bridge between frontend applications (DApps) and the Ethereum blockchain.

Key Components:

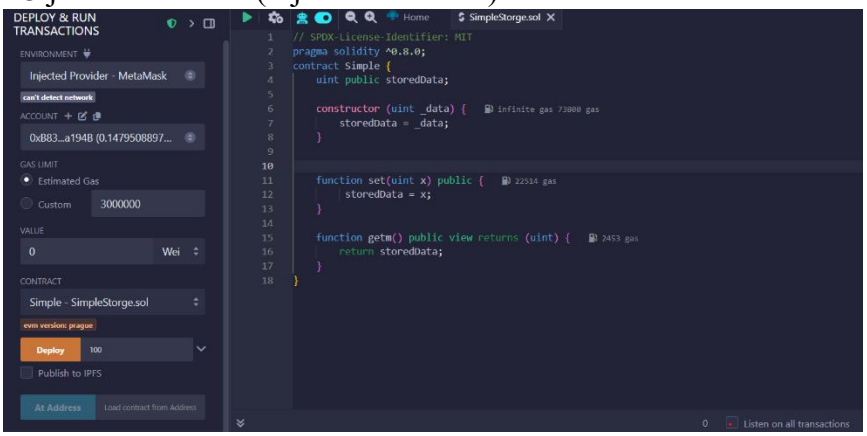
- **Provider**: Connection to Ethereum node (MetaMask, Ganache, Infura)
- **Contract**: A JavaScript object that represents an on-chain smart contract
- **ABI**: Application Binary Interface – defines how to interact with the contract
- **Contract Address**: The deployed location of the contract on the blockchain

Procedure:

1. **Install Web3.js** in your project directory:

```
PS C:\Users\pikun\OneDrive\Desktop\SimpleStorage\frontend> npm install web3

added 43 packages, and audited 1430 packages in 14s

307 packages are looking for funding
  run `npm fund` for details

9 vulnerabilities (3 moderate, 6 high)

To address all issues (including breaking changes), run:
  npm audit fix --force
```

2. Connect Web3.js to MetaMask (Injected Provider):



3. Define the contract ABI and address:

```
.env
1  REACT_APP_CONTRACT_ADDRESS=0xf13e5aE62305Dc397a5514c563d9F4bF42536214
2  REACT_APP_NETWORK=sepolia
```

4. Call a view function to read data:

```
App.js 2 | App.test.js | .env | abi.js | App.css
frontend > src > App.js > App
import React, { useState, useEffect } from 'react';
import Web3 from 'web3';
import { contractABI } from './abi'; // Make sure ABI is correct and updated

const contractAddress = '0x8C23b3CF68C79973dED1f16BD084cd6B48444030';

function App() {
  const [web3, setWeb3] = useState(null);
  const [contract, setContract] = useState(null);
  const [account, setAccount] = useState('');
  const [currentValue, setCurrentValue] = useState('');
  const [inputValue, setInputValue] = useState('');

  const connectWallet = async () => {
    if (window.ethereum) {
      try {
        const web3Instance = new Web3(window.ethereum);
        await window.ethereum.request({ method: 'eth_requestAccounts' });
        const accounts = await web3Instance.eth.getAccounts();
        const userAccount = accounts[0];

        const tempContract = new web3Instance.eth.Contract(contractABI, contractAddress);

        setWeb3(web3Instance);
        setContract(tempContract);
        setAccount(userAccount);

        alert(`Connected: ${userAccount}`);
      } catch (error) {
        console.error("Error connecting wallet", error);
        alert("Failed to connect wallet.");
      }
```

5. **Run the project in a browser** with MetaMask connected to the correct network.

Observation Table:

- MetaMask successfully connected to the DApp.
- The smart contract method (e.g., getValue()) returned the expected data.
- Console displayed:
  - Connected account: 0xB835d4728E2811F5ced713dB1D0622A101ba194B
  - Value from smart contract: 2345
- Changing the network in MetaMask (e.g., from Sepolia to Mainnet) causes the contract call to fail unless the contract is deployed on that network.

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 :
Regn. No. :

Signature of the Faculty: