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 : Frontend Connect – Web3.js Integration

Objective/Aim:

To establish a connection between a web-based frontend and a blockchain smart contract using the **Web3.js** library. The goal is to allow users to interact with blockchain data — such as reading balances or executing transactions — directly from their web browser.

Apparatus/Software Used:

- Visual Studio Code
- Node.js + npm
- Web3.js Library
- MetaMask Wallet
- Ethereum Testnet

Theory/Concept:

Web3.js is a JavaScript library that enables communication between a web application and the Ethereum blockchain (or other EVM-compatible networks).
It works on the **JSON-RPC protocol**, which allows frontend applications to send requests to blockchain nodes and receive responses.

Using Web3.js, developers can:

- Connect browser wallets such as **MetaMask**
- **Read** blockchain data (e.g., account balances, contract variables)
- **Write** data (e.g., sending tokens or updating contract state)
- **Deploy** and **interact** with smart contracts directly from web pages

This integration is a key part of decentralized applications (DApps), as it removes the need for centralized servers to process blockchain-related actions.

.

Procedure:

Step 1: Create a smart contract in remix IDE

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

contract SimpleStorage {
    uint256 private storedData;

    // Set the value of storedData
    function set(uint256 x) public {    22492 gas
        storedData = x;
    }

    // Get the value of storedData
    function get() public view returns (uint256) {    2431 gas
        return storedData;
    }
}
```

Step 2: Create a React app in VS Code.

- Open VS Code.
- Open a terminal inside of VS Code.
- Run this code (npx create-react-app simple-storage-web3).
- Then run cd simple-storage-web3.
- Then install web3 like run this code(npm install web3).

Step 3: Create a .env File.

- Write The deployed contract address from Remix or blockchain explorer.

```
.env
REACT_APP_CONTRACT_ADDRESS =0x30DFDC7d8D9437A22Fe7DEa570C1477ac8aC9592
REACT_APP_NETWORK= sepolia
```

Step 4: Connect in src/App.js

- Replace App.js with something like:

```
import React, { useEffect, useState } from "react";
import Web3 from "web3";

// Load contract address from .env file
const CONTRACT_ADDRESS = process.env.REACT_APP_CONTRACT_ADDRESS;

// ABI for the contract
const ABI = [
  {
    inputs: [{ internalType: "uint256", name: "x", type: "uint256" }],
    name: "set",
    outputs: [],
    stateMutability: "nonpayable",
    type: "function",
  },
  {
    inputs: [
      { internalType: "uint256", name: "_data", type: "uint256" }
    ],
    stateMutability: "nonpayable",
    type: "constructor"
  },
  {
    inputs: [],
    name: "get",
    outputs: [{ internalType: "uint256", name: "", type: "uint256" }],
    stateMutability: "view",
    type: "function",
  },
  {
    inputs: [],
    name: "storedData",
    outputs: [{ internalType: "uint256", name: "", type: "uint256" }],
    stateMutability: "view",
    type: "function",
  },
];

function App() {
  const [account, setAccount] = useState("");
  const [contract, setContract] = useState(null);
  const [web3, setWeb3] = useState(null);
  const [inputValue, setInputValue] = useState("");
  const [storedValue, setStoredValue] = useState(null);

  useEffect(() => {
    const init = async () => {
      // Check for MetaMask
      if (window.ethereum) {
        try {
          const web3Instance = new Web3(window.ethereum);
          await window.ethereum.request({ method: "eth_requestAccounts" });

          const accounts = await web3Instance.eth.getAccounts();
          const contractInstance = new web3Instance.eth.Contract(ABI, CONTRACT_ADDRESS);

          setWeb3(web3Instance);
          setAccount(accounts[0]);
          setContract(contractInstance);
        } catch (error) {
          console.error("Wallet connection failed:", error);
        }
      } else {
        alert("Please install MetaMask to use this app.");
      }
    };

    init();
  }, []);

  const handleSet = async () => {
    if (contract && account) {
      try {
        await contract.methods.set(inputValue).send({ from: account });
        alert("✅ Value set successfully!");
      } catch (err) {
        console.error("❌ Error setting value:", err);
      }
    }
  };

  const handleGet = async () => {
    if (contract) {
      try {
        const value = await contract.methods.get().call();
        setStoredValue(value);
      } catch (err) {
        console.error("❌ Error reading value:", err);
      }
    }
  };

  return (
    <div style={{ padding: "2rem", fontFamily: "Arial, sans-serif" }}>
      <h1>🦊 DAPP using web3</h1>

      <p><strong>Connected Account:</strong> {account || "Not connected"}</p>

      <div style={{ marginTop: "1rem" }}>
        <input
          type="number"
          placeholder="Enter a number"
          value={inputValue}
          onChange={(e) => setInputValue(e.target.value)}
          style={{ padding: "0.5rem", width: "200px", marginRight: "10px" }}
        />
        <button onClick={handleSet} style={{ padding: "0.5rem 1rem" }}>
          Set Value
        </button>
      </div>

      <div style={{ marginTop: "2rem" }}>
        <button onClick={handleGet} style={{ padding: "0.5rem 1rem" }}>
          Get Stored Value
        </button>

        {storedValue !== null && (
          <p style={{ marginTop: "1rem", fontSize: "1.2rem" }}>
            📦 <strong>Stored Value:</strong> {storedValue}
          </p>
        )}
      </div>
    </div>
  );
}

export default App;
```

Step 5: Run the App

- In terminal: npm start

Step 6: After run this open React app at http://localhost:3000

Observation Table:

Function	Action	Expected Result	Actual Result	Status
Connect Wallet	Click connect button	MetaMask connects successfully	✅ Success	Pass
Read Data	Call getValue()	Returns stored contract value	✅ Correct value	Pass
Write Data	Call setValue()	Transaction processed and confirmed	✅ Mined successfully	Pass
Update Display	Refresh contract state	UI updates with new data	✅ Updated	Pass

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 Faculty:

Signature of the Student:

Name :
Regn.No.