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 understand how a frontend application connects to the Ethereum blockchain using **Web3.js**, interact with a deployed smart contract, and perform basic read/write blockchain operations from the browser.

1. Apparatus/Software Used:

a. Laptop/PC
b. Web browser (Chrome/Brave)
c. MetaMask wallet extension
d. Local blockchain (Ganache) or Testnet (Sepolia/Polygon Amoy)
e. Web3.js library
f. HTML/CSS/JavaScript frontend

Theory/Concept:

Web3.js is a JavaScript library that allows frontend applications to communicate with the Ethereum blockchain.
It interacts with nodes using JSON-RPC and enables:

Key Features:

- Connecting to MetaMask or any Web3 provider
- Reading blockchain data (balance, contract state, events)
- Writing transactions (sending crypto, calling smart contract functions)
- Listening for events emitted from smart contracts

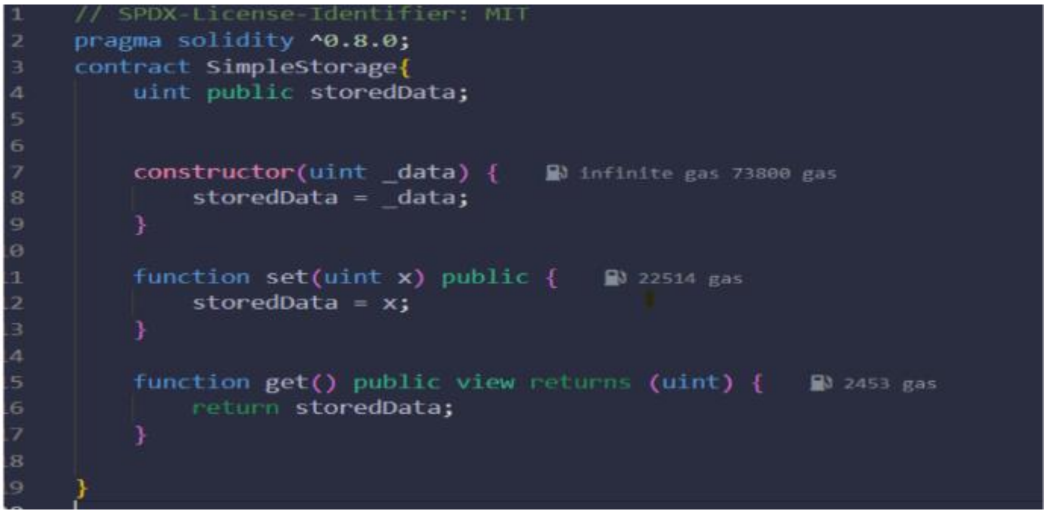
How Web3.js Works:

1. **User connects wallet**
   MetaMask injects window.ethereum into the browser.
2. **Frontend creates Web3 instance:**
3. const web3 = new Web3(window.ethereum);
4. **Contract ABI + Address are loaded:**
5. const contract = new web3.eth.Contract(ABI, contractAddress);
6. **Read-only calls** use .call()
7. **Write transactions** use .send() and require gas & confirmation

Web3.js is essential in **Decentralized Applications (DApps)** because it bridges the **smart contract** and the **user interface**.

Procedure:

Step:1 Create a smart contract in remix **IDE**



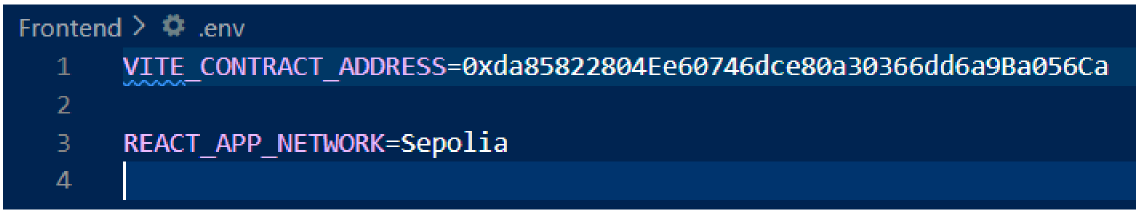
```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract SimpleStorage{
    uint public storedData;


    constructor(uint _data) {     infinite gas 73800 gas
        storedData = _data;
    }

    function set(uint x) public {     22514 gas
        storedData = x;
    }

    function get() public view returns (uint) {     2453 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**

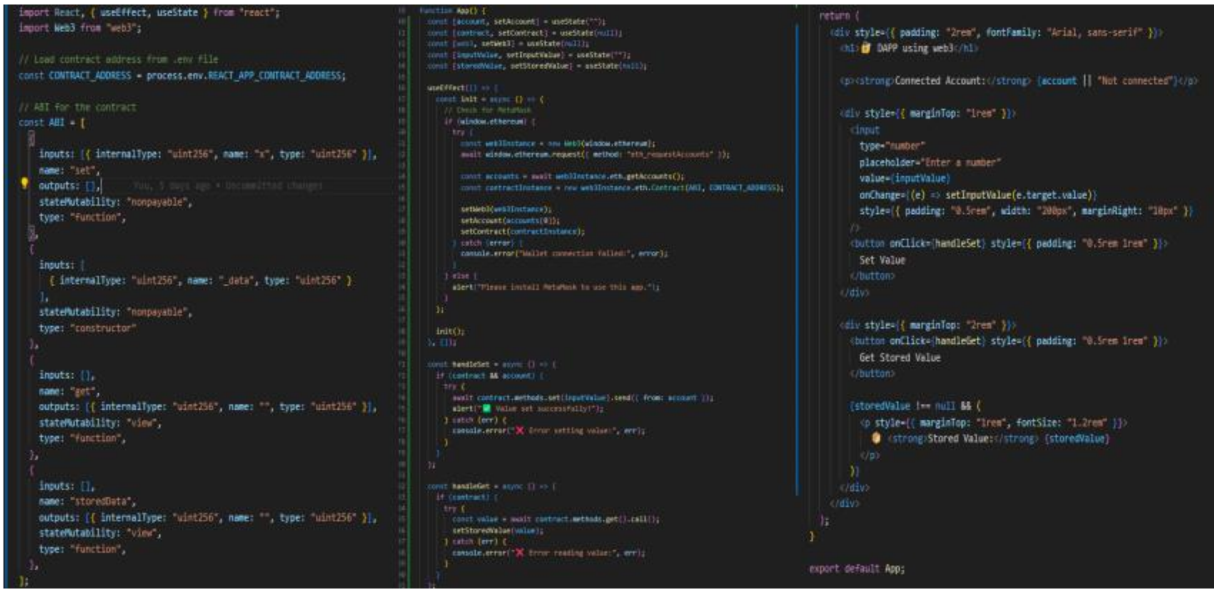


Frontend > .env

```
VITE_CONTRACT_ADDRESS=0xda85822804Ee60746dce80a30366dd6a9Ba056Ca

REACT_APP_NETWORK=Sepolia

```

Step 4: Connect in src/App.js

- Replace App.js with something like:

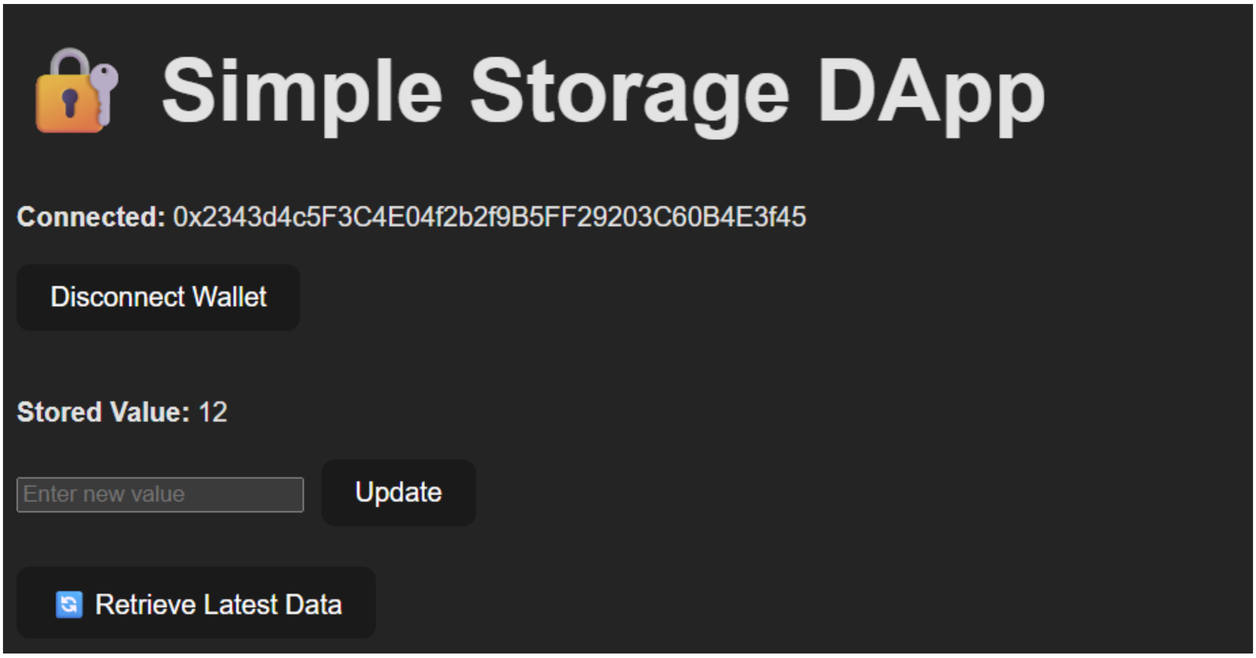


```
import React, { useEffect, useState } from "react";
import Web3 from "web3";

// 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: [],      You, 5 days ago • Uncommitted changes
    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.



🔐 Simple Storage DApp

Connected: 0x2343d4c5F3C4E04f2b2f9B5FF29203C60B4E3f45

Stored Value: 12

Observation

- Web3.js successfully connected frontend to blockchain.
- MetaMask allowed account access and transaction confirmation.
- Updating values from frontend reflected immediately on blockchain

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.