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 – Ethers.js and MetaMask UI

Coding Phase : Pseudo Code/Flow Chart/Algorithm

1. Start React project using `npx create-react-app`.
2. Install `ether.js` library.
3. Create `.env` file with:
4. In `app.js`:

- Import Web3 and connect to MetaMask.
- Load contract using ABI & address from `.env`.
- Fetch `storedData` using `contract.methods.get().call()`.
- Send transaction using `contract.methods.set(value).send()`.

5. Test the frontend by setting and getting values.

Apparatus/Software Used:

- Node.js & npm
- React.js
- MetaMask
- **Network**: Sepolia Testnet

Testing Phase:

- Deployed `SimpleStorage` contract to Sepolia using Remix.
- Noted the contract address & ABI.
- Created `.env` file to store sensitive data.
- Connected frontend to MetaMask.
- Verified:
- Reading stored value works.
- Writing new value updates blockchain data.

Implementation Phase: Final Output (no error)

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).
- Then run cd simple-storage

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:

```jsx
import React, { useState } from 'react';
import { ethers } from 'ethers';
import { ToastContainer, toast } from 'react-toastify';
import 'react-toastify/dist/ReactToastify.css';
import { FaSpinner } from 'react-icons/fa';

const contractAddress = process.env.REACT_APP_CONTRACT_ADDRESS;

const simpleStorageABI = [
  {
    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: [{ internalType: "uint256", name: "_data", type: "uint256" }],
    name: "set",
    outputs: [],
    stateMutability: "nonpayable",
    type: "function"
  },
  {
    inputs: [],
    name: "storedData",
    outputs: [{ internalType: "uint256", name: "", type: "uint256" }],
    stateMutability: "view",
    type: "function"
  }
];

function App() {
  const [wallet, setWallet] = useState(null);
  const [contract, setContract] = useState(null);
  const [value, setValue] = useState('');
  const [storedValue, setStoredValue] = useState(null);
  const [loading, setLoading] = useState(false);

  const connectWallet = async () => {
    if (!window.ethereum) {
      toast.error('Please install MetaMask.');
      return;
    }

    try {
      const provider = new ethers.BrowserProvider(window.ethereum);
      const signer = await provider.getSigner();
      const address = await signer.getAddress();

      const contract = new ethers.Contract(contractAddress, simpleStorageABI, signer);

      setWallet(address);
      setContract(contract);
      toast.success("Wallet connected!");

      const data = await contract.get();
      setStoredValue(data.toString());
    } catch (err) {
```

```jsx
      toast.error("Failed to connect.");
      console.error(err);
    }
  };

  const updateValue = async () => {
    if (!value || !contract) return toast.error("Missing input or contract");

    try {
      setLoading(true);
      const tx = await contract.set(value);
      await tx.wait();
      toast.success("Value updated!");
      const newVal = await contract.get();
      setStoredValue(newVal.toString());
      setValue('');
    } catch (err) {
      toast.error("Transaction failed.");
      console.error(err);
    } finally {
      setLoading(false);
    }
  };

  return (
    <div style={{ padding: 20, fontFamily: 'Arial', maxWidth: 600, margin: 'auto' }}>
      <ToastContainer />
      <h2>📦 Simple Storage DApp</h2>

      {!wallet ? (
        <button onClick={connectWallet}>🦊 Connect MetaMask</button>
      ) : (
        <>
          <p><strong>Connected:</strong> {wallet}</p>
          <p><strong>Stored Value:</strong> {storedValue}</p>

          <input
            type="number"
            placeholder="Enter value"
            value={value}
            onChange={e => setValue(e.target.value)}
          />
          <button onClick={updateValue} disabled={loading} style={{ marginLeft: 10 }}>
            {loading ? <FaSpinner className="spin" /> : 'Set Value'}
          </button>
        </>
      )}
      <style>{`
        .spin {
          animation: spin 1s linear infinite;
          display: inline-block;
        }
        @keyframes spin {
          from { transform: rotate(0deg); }
          to { transform: rotate(360deg); }
        }
      `}</style>
    </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

- Then connect the meta mask.
- Then Enter some value and set value .

Observations :

- Ether.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 Student:

Name :

Regn. No. :

Signature of the Faculty:

**As applicable according to the experiment. Two sheets per experiment (10-20) to be used.*