



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 Experiment :** Frontend Connect – Web3.js Integration

### \* Coding Phase: Pseudo Code / Flow Chart / Algorithm

#### ALGORITHM:

- Start
- Open remix IDE write the smart contract in SimpleStorage.sol
- Compile the smart contract in remix
- Copy the generated abi and save it somewhere
- Deploy the contract in sepolia testnet using metamask
- Copy the deployed contract address
- Create a react frontend project using create react app
- Add the contract address and network information in .env file
- Install web3.js
- Use the ABI and contract address to connect the frontend with smart contract 11.End

### \* Software used

1. Metamask wallet
2. Remix IDE
3. Brave Browser

## \* Testing Phase: Compilation of Code (error detection)

### Smart contract solidity code

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

    constructor(uint _data) {
        storedData = _data;
    }
    function set(uint x) public {
        storedData = x;
    }
    function get() public view returns (uint) {
        return storedData;
    }
}
```

### ABI key

```
export 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": "x",
        "type": "uint256"
      }
    ],
    "name": "set",
    "outputs": [],
    "stateMutability": "nonpayable",
```

```
    "type": "function"
  },
  {
    "inputs": [],
    "name": "storedData",
    "outputs": [
      {
        "internalType": "uint256",
        "name": "",
        "type": "uint256"
      }
    ],
    "stateMutability": "view",
    "type": "function"
  }
]
```

## \* Testing Phase: Compilation of Code (error detection)

After compilation deploy the smart contract in sepolia test network using metamask

### DEPLOY & RUN TRANSACTIONS

0xDAF...CCb81 (0 ETH)

GAS LIMIT

☒ Estimated Gas

☐ Custom 3000000

VALUE

0 Wei

CONTRACT

Counter - contracts/simpstr.sol

evm version: prague

☒ Verify Contract on Explorers

DEPLOY & VERIFY

\_start: 123

Calldata Parameters transaction

Address of contract

At Address Load contract from Address

### MetaMask

Account 1  
Sepolia

## Deploy a contract

This site wants you to deploy a contract

Estimated changes ? No changes

Request from ? remix.ethereum.org

Network fee ? 0.0003 s SepoliaETH

Speed 🦊 Market ~12 sec

Cancel Confirm

**\* Implementation Phase: Final Output (no error)**

Now we have to work on our frontend first create a folder for your frontend then open terminal and install react modules needed for the project. Now create `Abi.js` file in the `src` folder where we have to store the abi for our smart contract and now create a `.env` file and store the contract address and network information.

```
C: > Users > amitk > OneDrive > Desktop > SIMPLESTORAGE_FRONTEND > addfile.js > lab4 ipfs > .env
1  1.REACT_APP_CONTRACT_ADDRESS=0xdafe4dfdda3e39e0580d206bcfb59b93caeccb81
2  2.REACT_APP_NETWORK=sepolia
```

```

1 import React, { useEffect, useState } from 'react';
2 import Web3 from 'web3';
3 import { simpleStorageABI } from './abi';
4 import { ToastContainer, toast } from 'react-toastify';
5 import 'react-toastify/dist/react-toastify.css';
6 import { Fuelminer } from 'react-lottie/fa';
7
8 const contractAddress = process.env.REACT_APP_CONTRACT_ADDRESS;
9
10 function App() {
11   const [walletAddress, setWalletAddress] = useState(null);
12   const [web3, setWeb3] = useState(null);
13   const [contract, setContract] = useState(null);
14   const [storedValue, setStoredValue] = useState(null);
15   const [inputValue, setInputValue] = useState('');
16   const [loading, setLoading] = useState(false);
17
18   const connectWallet = async () => {
19     if (window.ethereum) {
20       try {
21         const web3Instance = new Web3(window.ethereum);
22         await window.ethereum.request({ method: 'eth_requestAccounts' });
23         const accounts = await web3Instance.eth.getAccounts();
24
25         const contractInstance = new web3Instance.eth.Contract(simpleStorageABI, contractAddress);
26
27         setWalletAddress(accounts[0]);
28         setWeb3(web3Instance);
29         setContract(contractInstance);
30
31         toast.success('Wallet connected!');
32         fetchStoredValue(contractInstance);
33       } catch (err) {
34         toast.error('Connection failed.').
35         console.error(err);
36       }
37     } else {
38

```

```

48 toast.error("Please install MetaMask.");
49 }
50 };
51
52 const disconnectWallet = () => {
53   setWalletAddress(null);
54   setWeb3(null);
55   setContract(null);
56   setStoredValue(null);
57   toast.info("Wallet disconnected.");
58 };
59
60 const fetchStoredValue = async (contractRef = contract) => {
61   try {
62     if (contractRef) {
63       const value = await contractRef.methods.get().call();
64       setStoredValue(value.toString());
65     }
66   } catch (err) {
67     toast.error("Failed to fetch data.");
68     console.error(err);
69   }
70 };
71
72 const handleSet = async () => {
73   if (contract && inputValue && web3 && walletAddress) {
74     try {
75       setLoading(true);
76       toast.info("Transaction submitted...");
77       await contract.methods.set(inputValue).send({ from: walletAddress });
78       setInputValue("");
79       toast.success("Value updated successfully!");
80       fetchStoredValue();
81     } catch (err) {
82       toast.error("Transaction failed.");
83     }
84   }
85 };

```

```

73 console.error(err);
74 } finally {
75   setloading(false);
76 }
77 }
78 };
79
80 return (
81   <div style={{
82     padding: '38px',
83     fontFamily: 'Segoe UI, sans-serif',
84     background: 'linear-gradient(to right, #0f2027, #202943, #2c3e50)',
85     color: 'white',
86     minWidth: '100vh'
87   }}>
88     <toastcontainer />
89     <div style={{
90       maxWidth: '500px',
91       margin: 'auto',
92       background: 'linear-gradient(to right, #0f2027, #202943)',
93       padding: '38px',
94       borderRadius: '15px',
95       boxShadow: '0 10px 20px rgba(0,0,0,0.3)'
96     }}>
97       <div style={{
98         textAlign: 'center',
99         marginbottom: '20px',
100        color: 'white'
101      }}> Simple Storage v0.0.1
102    </div>
103    <div>
104      <button
105        onClick={connectWallet}
106        style={buttonStyle}>
107        Connect MetaMask wallet

```

```

100 </button>
101 } </
102 O
103
104 <strong>Connect</strong> <span style={{ color: '#0000FF' }}>{{urlAddress}}</span></p>
105 <button onClick={disconnect} style={{ ...buttonStyle, backgroundColor: '#FF0000', margin: '10px' }}>
106   Disconnect
107 </button>
108
109 <strong>Store Value</strong> <span style={{ color: '#0000FF' }}>{{storeValue}}</span></p>
110
111 <div style={{ margin: '10px' }}>
112   <input
113     type="text"
114     placeholder="Enter new value"
115     value={inputValue}
116     onChange={e => setInputValue(e.target.value)}
117     style={inputStyle}
118   />
119   <button
120     onClick={update}
121     disabled={loading}
122     style={{ ...buttonStyle, margin: '10px' }}
123   >
124     {loading ? 'Loading' : 'Update'}
125   </button>
126 </div>
127
128 <div>
129   <button
130     onClick={() => fetchStoreValue()}
131     style={{ ...buttonStyle, backgroundColor: '#0000FF', margin: '10px' }}
132   >
133     Fetch latest data
134   </button>
135 </div>
136
137 </
138 </

```

## \* Implementation Phase: Final Output (no error)

Now open terminal and run the project by writing the code `npm start`

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Compiled successfully!

You can now view frontend in the browser.

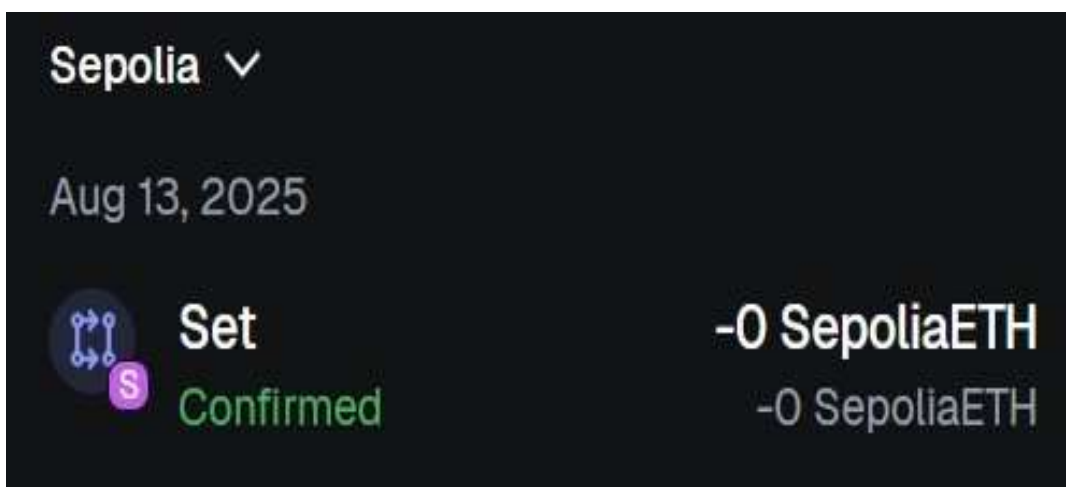
  Local:            http://localhost:3000
  On Your Network:  http://10.99.38.54:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```



Connecting the wallet with the DApp



## \* Implementation Phase: Final Output (no error)

Applied and Action Learning

Now your wallet is successfully connected with your DApp



## \* Observations

1. The lab demonstrates how to integrate a blockchain smart contract with a frontend application using Web3.js, enabling real-time interaction between users and the blockchain.
2. It highlights connecting wallets, reading/writing contract data, and handling blockchain events from the UI.

## ASSESSMENT

| Rubrics  | Full Mark | Marks Obtained | Remarks |
|--|-----------|----------------|---------|
| Concept  | 10        |                |         |
| Planning and Execution/<br>Practical Simulation/ Programming | 10        |                |         |
| Result and Interpretation                                    | 10        |                |         |
| Record of Applied and Action Learning                        | 10        |                |         |
| Viva   | 10        |                |         |
| <b>Total</b>   | <b>50</b> |                |         |

**Signature of the Student:**

Name :

Regn. No. :

**Signature of the Faculty:**

Page No.....

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