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

* Coding Phase: Pseudo Code / Flow Chart / Algorithm

ALGORITHM:

- Start
- Open remix IDE write the smart contract in SimpleStorage.sol
- Compile the smart contarct 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

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

* 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;
    }
}
```

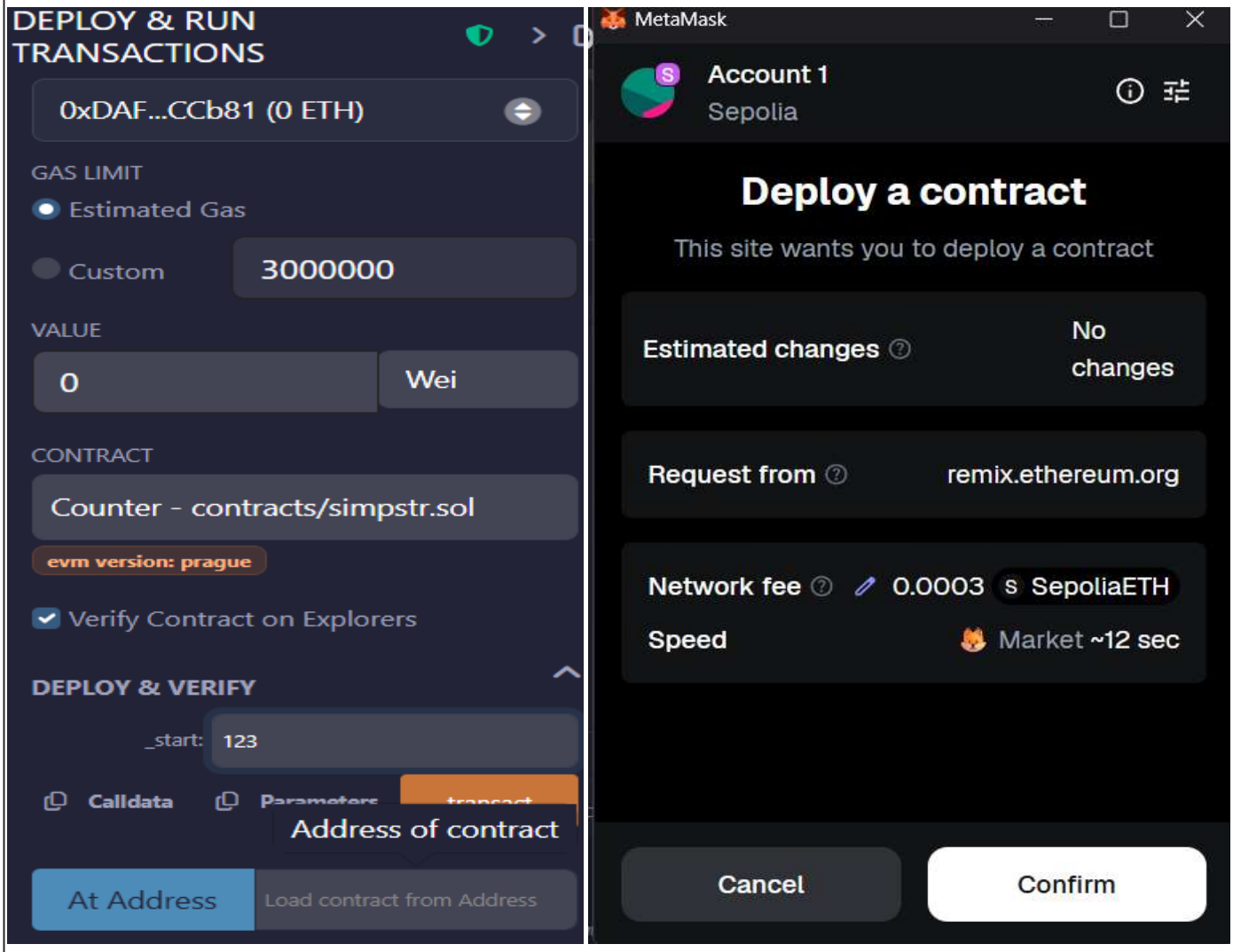
ABI key

```
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"
    }
]
```


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

* Testing Phase: Compilation of Code (error detection)

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



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

* 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.REACT_APP_CONTRACT_ADDRESS=0xdafe4dfdda3e39e0580d206bcfb59b93caeccb81
2.REACT_APP_NETWORK=sepolia
```

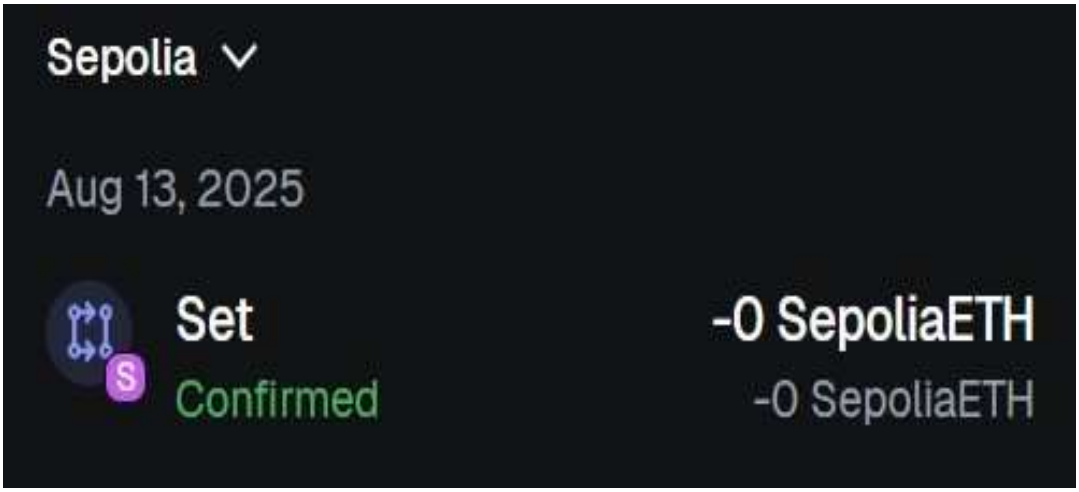
*As applicable according to the experiment.
Two sheets per experiment (10-20) to be used.

* Implementation Phase: Final Output (no error)

Now open terminal and run the project by writting 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



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

* Implementation Phase: Final Output (no error)

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/ 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.*