



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 Experiment :** React Start – DApp Frontend Scaffolding

### **Objective/Aim:**

o set up a basic React-based decentralized application (DApp) frontend using Vite/React, connect it with MetaMask, and prepare a functional skeleton for interacting with blockchain smart contracts.

### **Apparatus/Software Used:**

- Node.js & npm
- VS Code
- MetaMask Wallet
- Brave (or Chrome) Web Browser
- Ethereum Sepolia Testnet
- 

### **Theory/Concept:**

#### **React (Frontend Framework)**

React is a JavaScript library used to build interactive UIs. For DApps, React is used to create a responsive interface that communicates with blockchain networks via Web3 libraries.

#### **Web3 / Ethers.js**

To allow the browser to communicate with smart contracts, the DApp uses Ethers.js or Web3.js. These libraries let developers connect to MetaMask, read blockchain data, and execute transactions.

#### **MetaMask**

MetaMask acts as a bridge between the DApp and the Ethereum testnet. It provides:

- User authentication
- Signing and sending transactions
- Switching between networks

## Procedure:

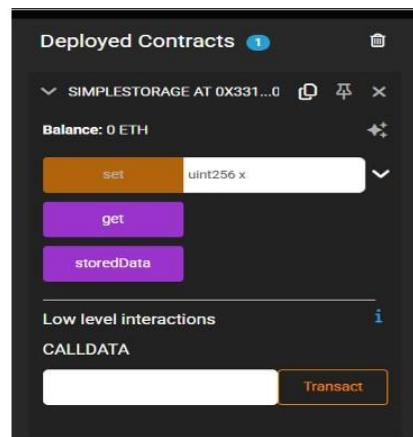
1. Open your command prompt or terminal.
2. Create a new React project using Vite:  
3. `npm create vite@latest my-dapp --template react`
4. Move into the project folder:  
5. `cd my-dapp`
6. Install project dependencies:  
7. `npm install`
8. Install Ethers.js for blockchain interactions:  
9. `npm install ethers`
10. Open the project in VS Code.
11. In src, create a file named `connectWallet.js` to handle wallet connection logic.
12. Add basic MetaMask connection code:  
13. `import { ethers } from "ethers";`
- 14.
15. `export async function connectWallet() {`
16. `if (window.ethereum) {`
17. `const accounts = await window.ethereum.request({ method: "eth_requestAccounts" });`
18. `return accounts[0];`
19. `} else {`
20. `alert("MetaMask not installed");`
21. `}`
22. `}`
23. Open App.jsx and import the wallet function to display a “Connect Wallet” button.
24. Run the development server:  
  `npm run dev`
11. Open the local address shown in the terminal (e.g., <http://localhost:5173>).
12. Click Connect Wallet and confirm the connection in MetaMask.

```

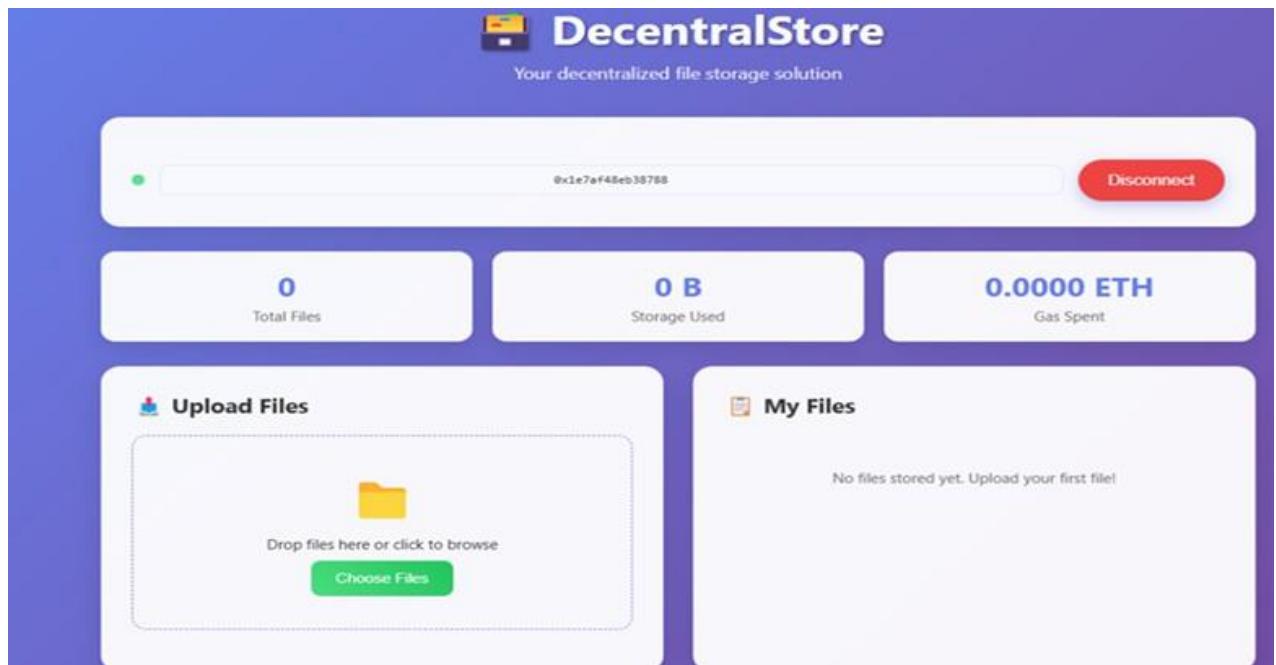
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.0;
3 contract Counter {
4     uint public count;
5     constructor(uint _start) {    ┌ infinite gas 132000 gas
6         count = _start;
7     }
8     function increment() public {    ┌ infinite gas
9         count += 1;
10    }
11    function decrement() public [    ┌ infinite gas
12        require(count > 0, "Counter is already at zero");
13        count -= 1;
14    ]
15    function getCount() public view returns (uint) {    ┌ 2453 gas
16        return count;
17    }
18 }
```

After compile the smart contract there is a ABI of the smart contract

```
[{"inputs": [{"internalType": "uint256", "name": "_start", "type": "uint256"}], "stateMutability": "nonpayable", "type": "constructor"}, {"inputs": [], "name": "count", "outputs": [{"internalType": "uint256", "name": "", "type": "uint256"}], "stateMutability": "view", "type": "function"}, {"inputs": [], "name": "decrement", "outputs": [], "stateMutability": "nonpayable", "type": "function"}, {"inputs": [], "name": "getCount", "outputs": [{"internalType": "uint256", "name": "", "type": "uint256"}], "stateMutability": "view", "type": "function"}, {"inputs": [], "name": "increment", "outputs": [], "stateMutability": "nonpayable", "type": "function"}]
```



In this Smart contract we have two accessible libraries one is ether.js and another is web3.js we have to work on ether.js



**This is the frontend**

## Observation

Observation	Result
React project scaffold creation	Successful
Ethers.js installation	Successful
MetaMask connection	Successfully connects to Sepolia testnet
Wallet address visibility	Address displayed on UI after connection

## 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		
<b>Total</b>	<b>50</b>		

***Signature of the Student:***

Name :

***Signature of the Faculty:***

Regn. No. :