



Centurion  
UNIVERSITY  
Mysore 570 010  
Karnataka, India

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 : Connect the Dots – Ethers.js and MetaMask UI

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

- | Setup MetaMask wallet in the browser.
- | Open Remix IDE and write the smart contract (SimpleStorage.sol).
- | Compile the smart contract to generate the **ABI**.
- | Deploy the contract on **Sepolia Testnet** using MetaMask (Injected Provider).
- | Copy the **deployed contract address**.
- | Create a **React frontend project** using create-react-app.
- | Inside src, create ABI.js to store the smart contract ABI.
- | Create a .env file in the project root to store **contract address** and **network details**.
- | Install **ethers.js** (primary library for blockchain interaction).
- | In App.js, implement wallet connection and contract interaction logic using **ethers.js**.
- | Build a **UI** to store and retrieve values from the contract.
- | Run the project with npm start and verify blockchain interactions through MetaMask prompts.

### Software used

1. MetaMask Wallet
2. Remix IDE.
3. MS Word.
4. Brave for researching.

Page No.....

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

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

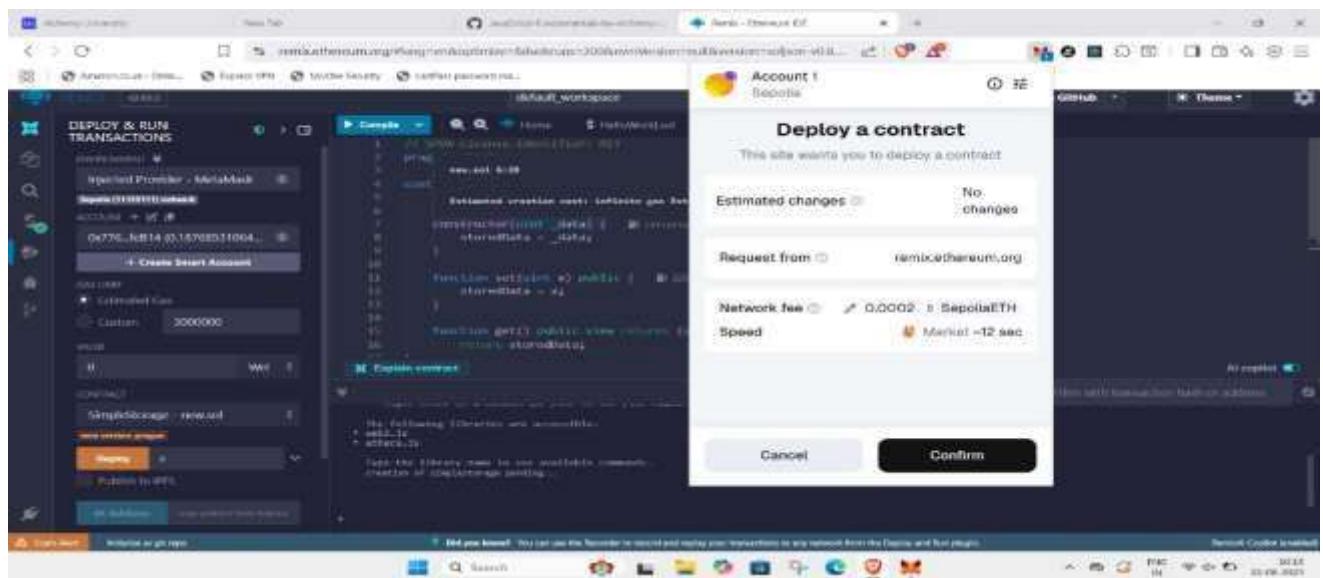
- Smart contract successfully compiled in Remix IDE without errors.
- Contract deployed on **Sepolia Testnet** using MetaMask.
- ABI and contract address correctly imported into React frontend.
- .env file securely holds contract address and network details.
- Wallet connection established via MetaMask in the frontend.
- User can **store** values in the contract and **retrieve** them through the UI.
- Frontend interacts smoothly with blockchain using **ethers.js** (preferred over web3.js for simplicity & security).
- Final project runs successfully with `npm start` showing no errors.



```

1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.0;
3
4 contract SimpleStorage {
5     uint public storedData;
6
7     constructor(uint _data) {
8         storedData = _data;
9     }
10
11    function set(uint x) public {
12        storedData = x;
13    }
14
15    function get() public view returns (uint) {
16        return storedData;
17    }
18 }

```



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

```

// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract SimpleStorage {
    uint256 value;

    function get() public view returns (uint256) {
        return value;
    }

    function set(uint256 newValue) public {
        value = newValue;
    }
}

```

The screenshot shows a code editor with a Solidity smart contract named `SimpleStorage`. The contract has a state variable `value` of type `uint256`. It includes two functions: `get` (view function returning `value`) and `set` (function taking a `newValue` parameter and setting `value` to it). The code is annotated with `// SPDX-License-Identifier: MIT` at the top.

**Simple Storage DApp**

Connect Metamask Wallet

Account 1: Sepolia

Transaction request

Estimated changes: No changes

Request from: HTTP localhost:3000

Interacting with: Alert: 0x90944...BB4B5

Network fee: 0 SepoliaETH

Speed: Market ~12 sec

Stored Value: 2324242

Enter new value:  Update

Retrieve Latest Data

Cancel Confirm

Wallet connected!

**Simple Storage DApp**

Connected: 0x7760106495a804b2DE289dc8010Bc5b2a61feB14

Disconnect Wallet

Stored Value: 2324242

Enter new value:  Update

Retrieve Latest Data

### \* Observations:

- | Contract successfully deployed on **Sepolia Testnet**.
- | **MetaMask wallet connection** worked smoothly with the frontend.
- | Frontend UI allowed **real-time interaction** with the smart contract (store/retrieve values).
- | **ethers.js** ensured secure and simplified interaction compared to web3.js.
- | .env file securely handled contract details and network configuration.
- | Data stored in the contract remained **persistent on the blockchain**.
- | The project acted as a **scaffold for future DApp development**.

### 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 :**

**Regn. No. :**

***Signature of the Faculty:***

Page No.....

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