



School: ............................................................................................................. Campus: ....................................................... Academic Year: ...................... Subject Name: ........................................................... Subject Code: ..........................

Semester: ............... Program: ........................................ Branch: ......................... Specialization: .......................... Date: .....................................

(Learning by Doing and Discovery)

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

ALGORITHM:

Start.

1. Initialize a React project using **npx** create-react-app.
2. Install **Web3.js** with **npm** install web3.
3. Create a **.env** file and store your contract address (and any needed keys) safely.
4. In **src/App.js:**

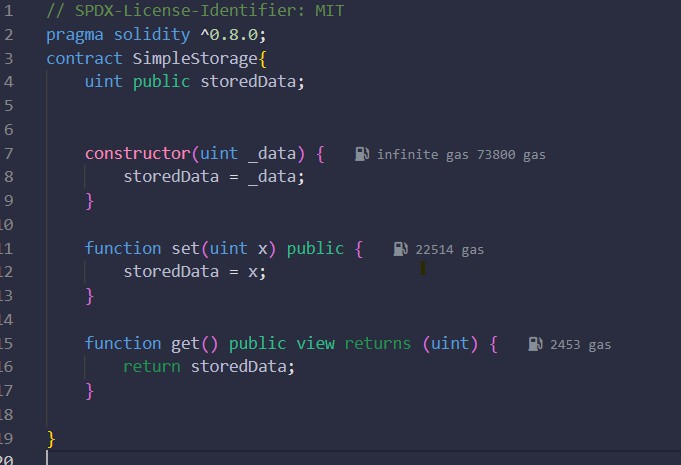
* Import Web3 and connect to MetaMask.
* Load the contract using ABI + address from **.env**.
* Read data via contract.methods.get().call().
* Write/update via contract.methods.set(value).send().

1. Run the app and verify read/write from the UI.

# \* Software used

* Node.js & npm
* React.js
* Web3.js
* MetaMask
* Network: **Sepolia Testnet**

**Step-1:**  
 Smart Contract Deployment (Remix + Sepolia)



1. Open Remix IDE and paste the SimpleStorage contract.
2. Compile the contract.
3. Deploy to Sepolia (via MetaMask).
4. Copy the contract address and ABI for the frontend.

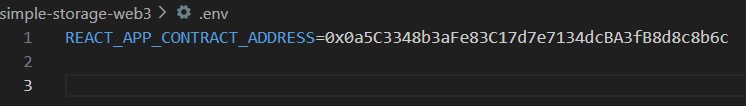
**Step-2:**In VS Code terminal:

* npx create-react-app simple-storage-web3
* cd simple-storage-web3
* npm install web3

**Step 3:**

Create a .env File.

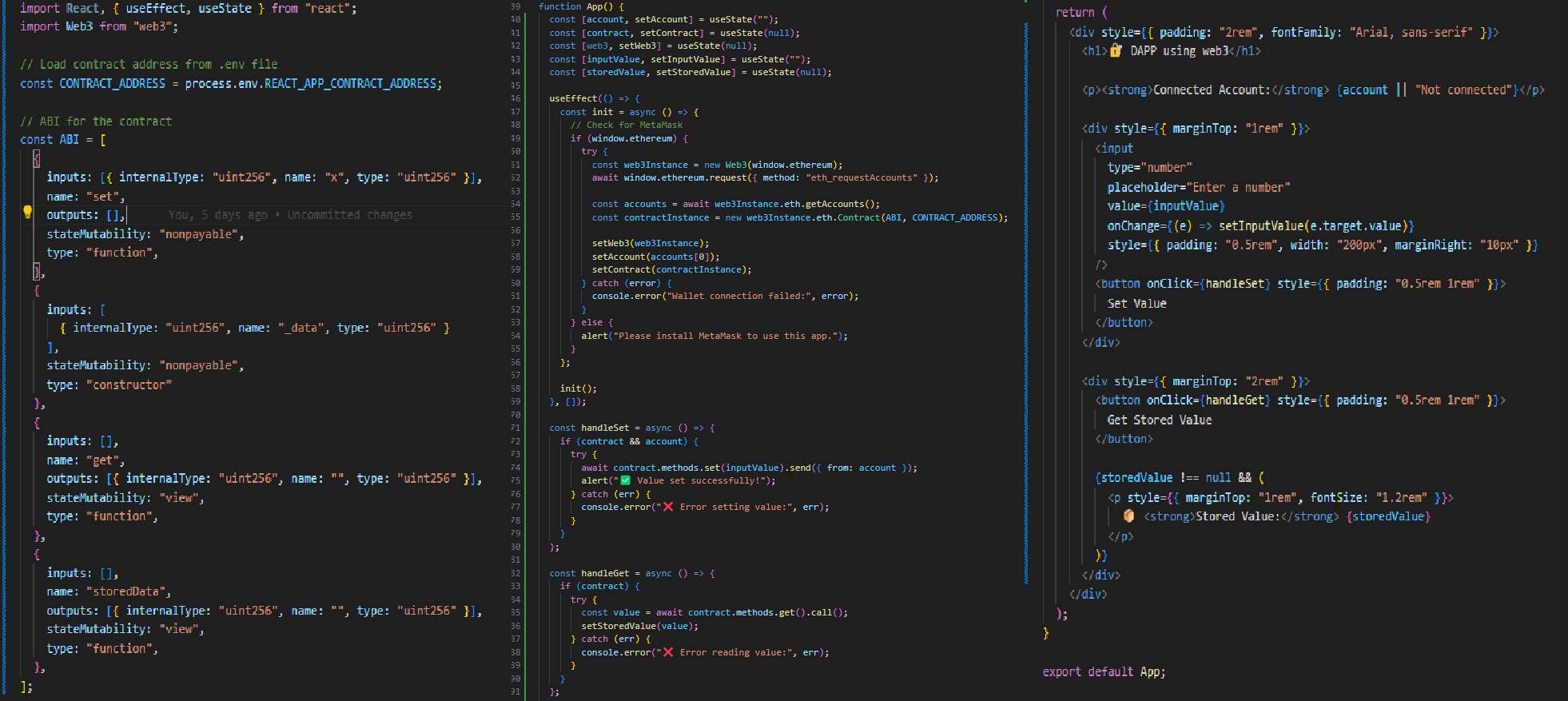
* Create .env in project root and add the deployed **contract address** (and any keys you keep locally).
* Write The deployed contract address from Remix or blockchain explorer.



**Step-4:**

Replace the contents of src/App.js to:

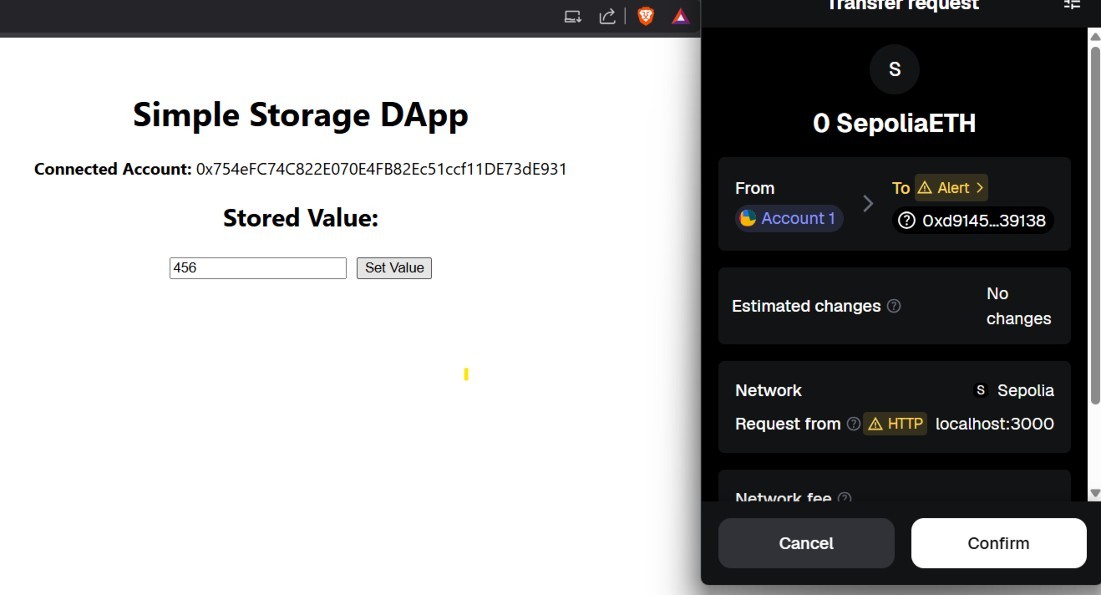
* initialize Web3 with window.ethereum,
* request MetaMask accounts,
* instantiate the contract with ABI + address,
* implement get (read) and set (write) handlers.



**Step 5:**

Run & Verify

* Start the app with npm start and open <http://localhost:3000>.
* Click **Connect** in MetaMask when prompted.
* Test **Read**: call get() and confirm the current stored value displays in the UI.
* Test **Write**: enter a new value, call set(value), approve the transaction in MetaMask, wait for confirmation, then call get() again to see the updated value.



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

Applied and Action Learning

**Note:**

* Ensure the selected MetaMask network is Sepolia during both deploy and testing.
* If reads fail: recheck contract **address/ABI** and that web3.eth.Contract(ABI, ADDRESS) is set after accounts load.
* If writes fail: confirm the connected account has Sepolia ETH and you’ve approved the MetaMask prompt.

# \* Observations

Web3.js successfully connects the React frontend to the blockchain and interacts with the contract.

MetaMask cleanly handles account access and transaction approvals.

|  |  |  |  |
| --- | --- | --- | --- |
| **Rubrics** |  |  |  |
| 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:***

***Signature of the Faculty:***

