

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

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

(Learning by Doing and Discovery)

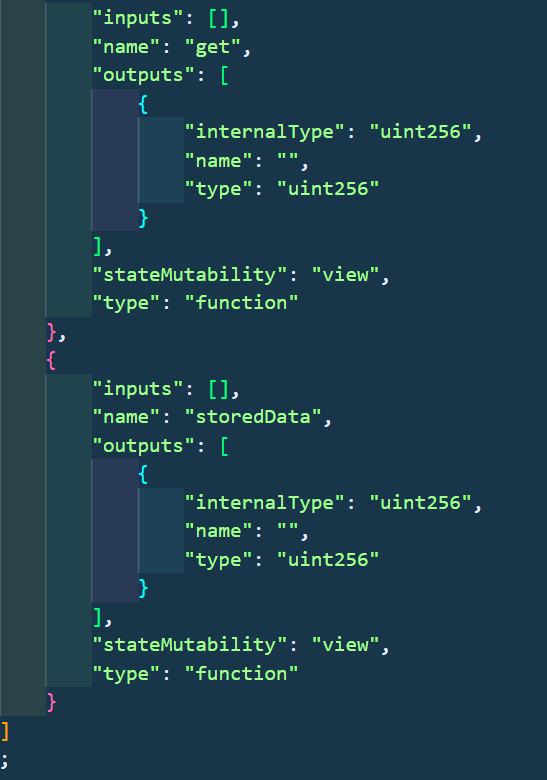
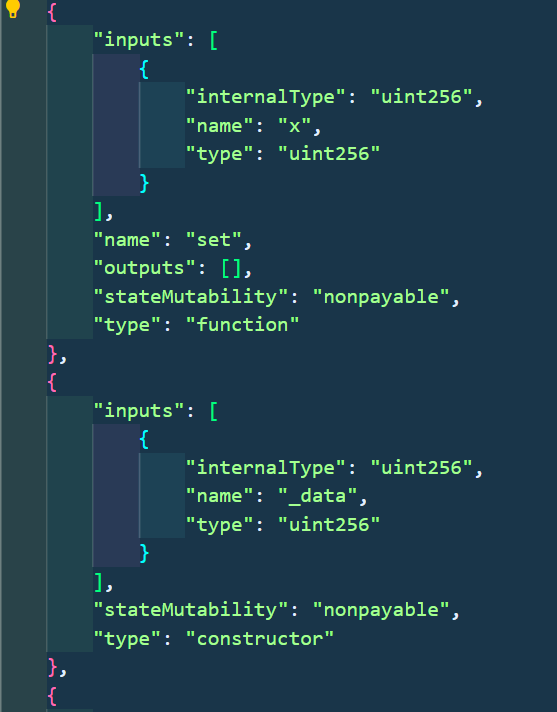
**\* Coding Phase: Pseudo Code / Flow Chart / Algorithm**

**Algorithm:**

1. Open Remix IDE and write the SimpleStorage.sol smart contract.
2. Compile the smart contract using the Solidity compiler in Remix.
3. Copy the generated ABI after successful compilation.
4. Deploy the contract to the Sepolia Testnet using MetaMask.
5. Copy the deployed contract address.
6. Create a React frontend project using create-react-app.
7. Add the contract address and network information to the .env file.
8. Install web3.js to interact with the blockchain.
9. Use the ABI and contract address to connect the frontend with the smart contract.
10. Design the UI in App.js using ethers.js to store and retrieve data.

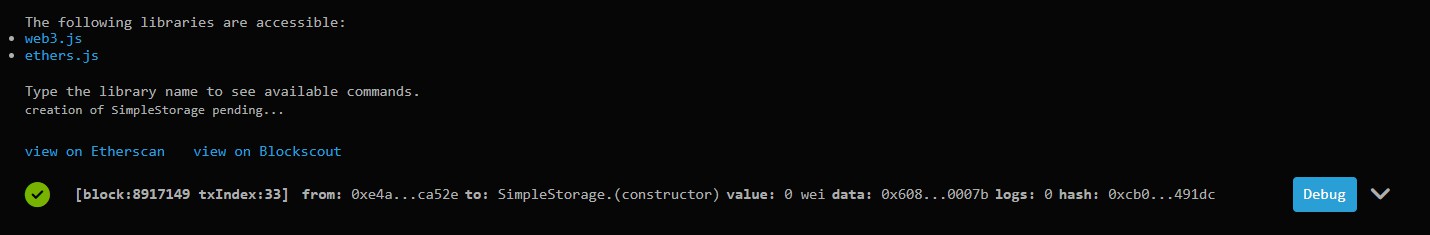
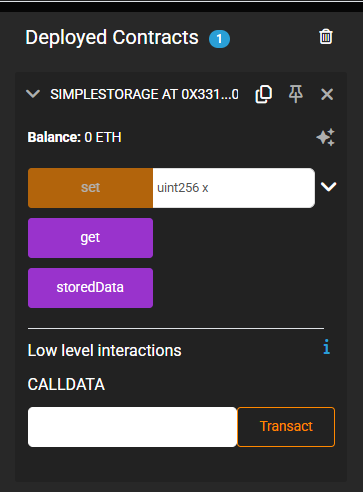
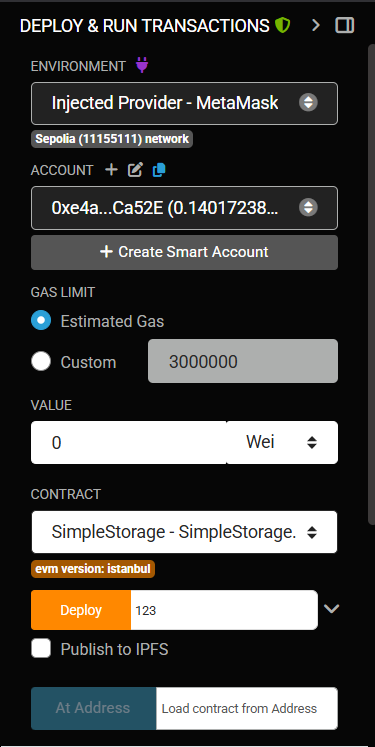
# \* Softwares used

1. MetaMask Wallet
2. Remix IDE
3. Brave browser



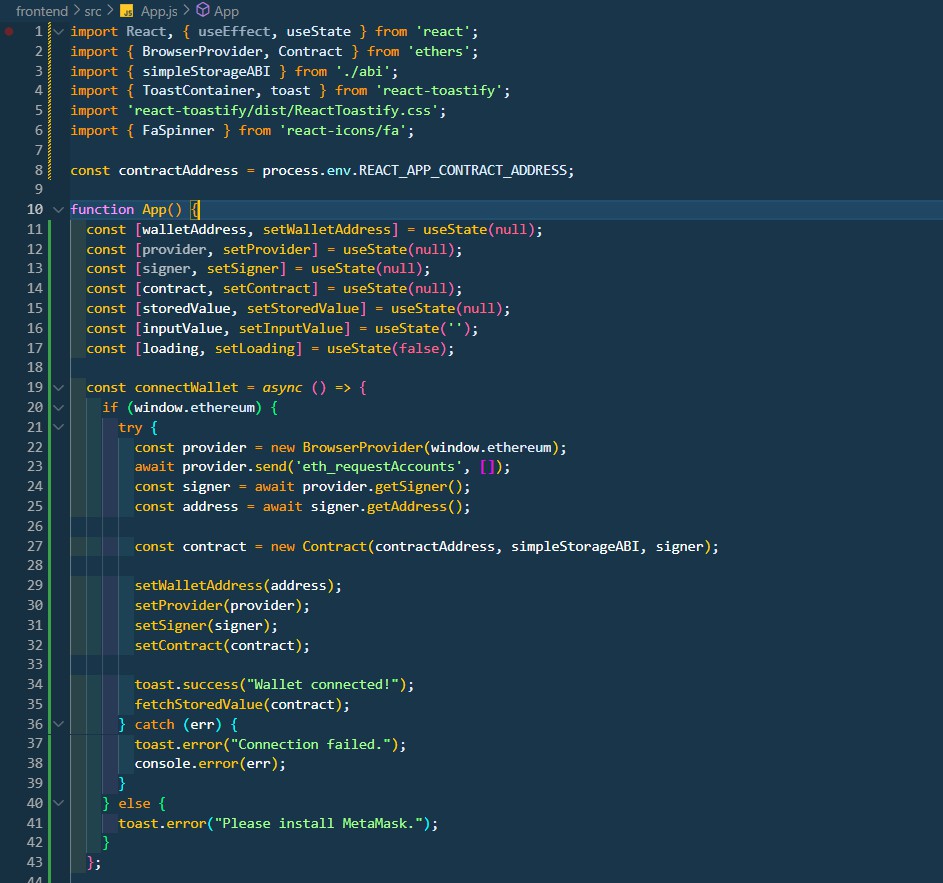
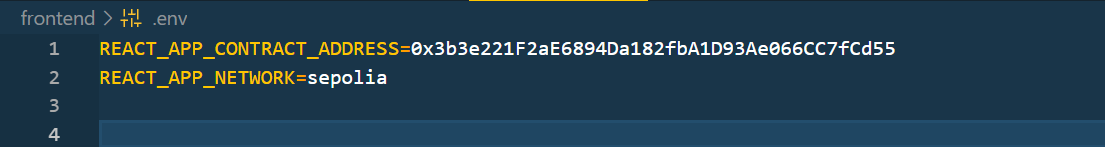
Go to remix ide and write a smart contract on simplestorage.sol and compile it

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



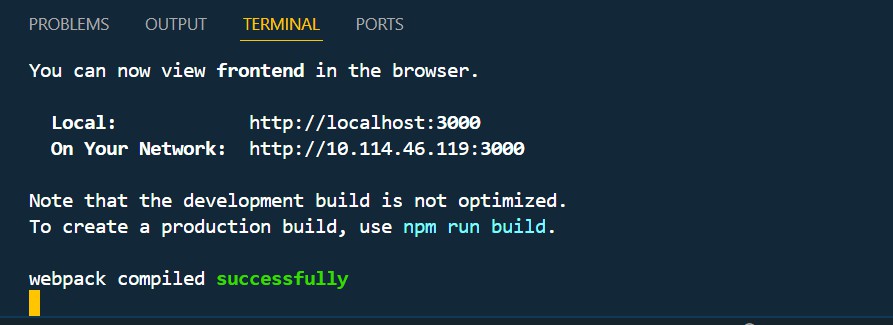
After compilation ,deploy the smart contract and choose the enviornment as injector provider-metamask then give some value and start deploy

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



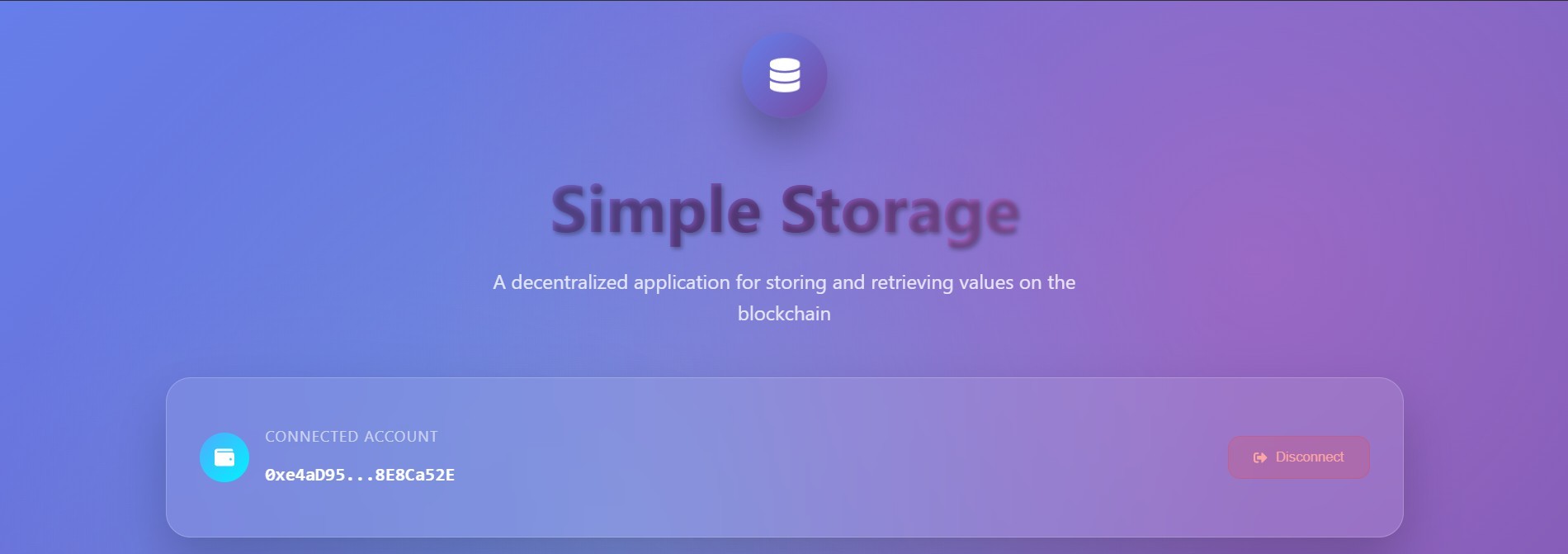
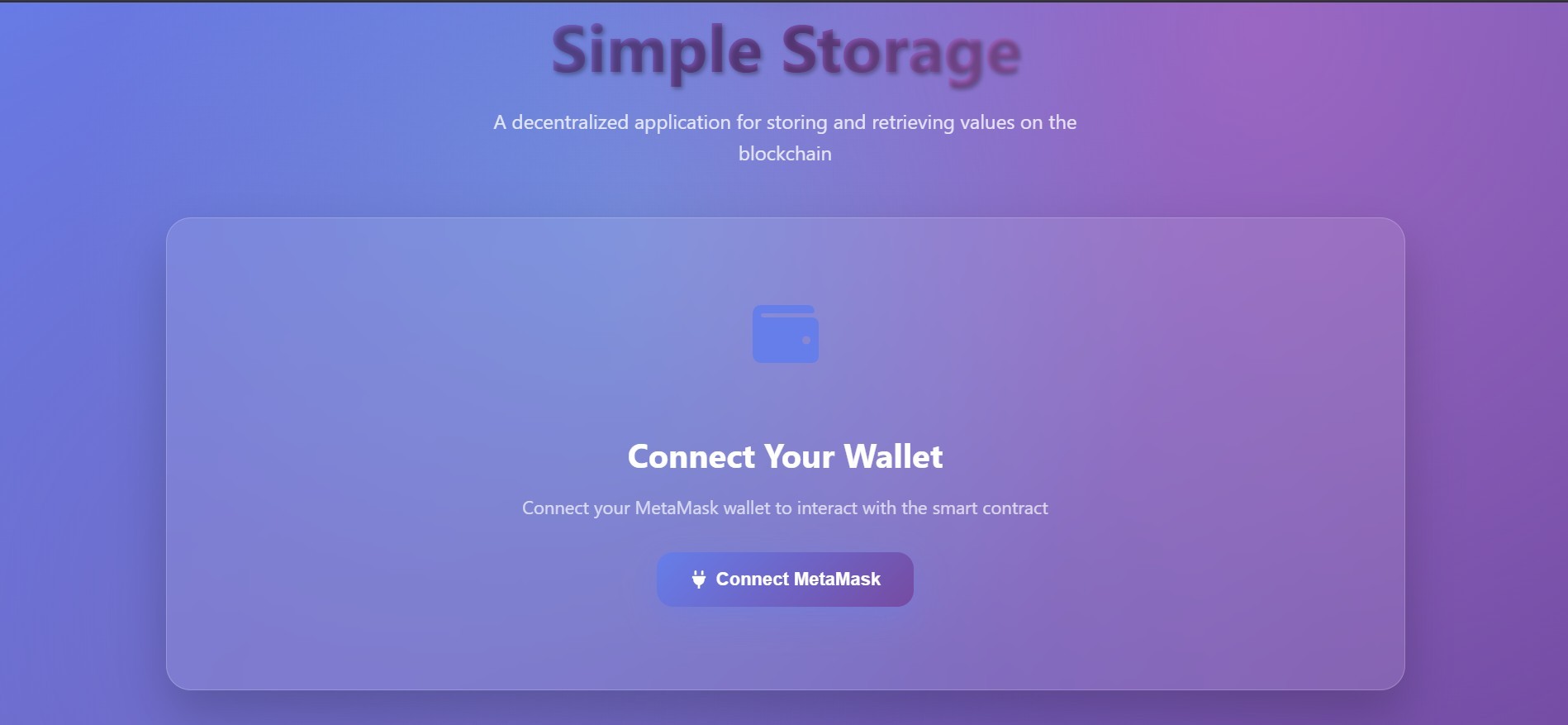
Now we have to work on frontend first create a folder for your frontend then open terminal to install the react modules . Then create a ABI.js file inside your src folder where we have to store the abi of our smart contract and then create a .env file in the root of the project folder to dtore contract address and tectnet network

Now in App.js write your frontend code and wallet connection code importing web3.



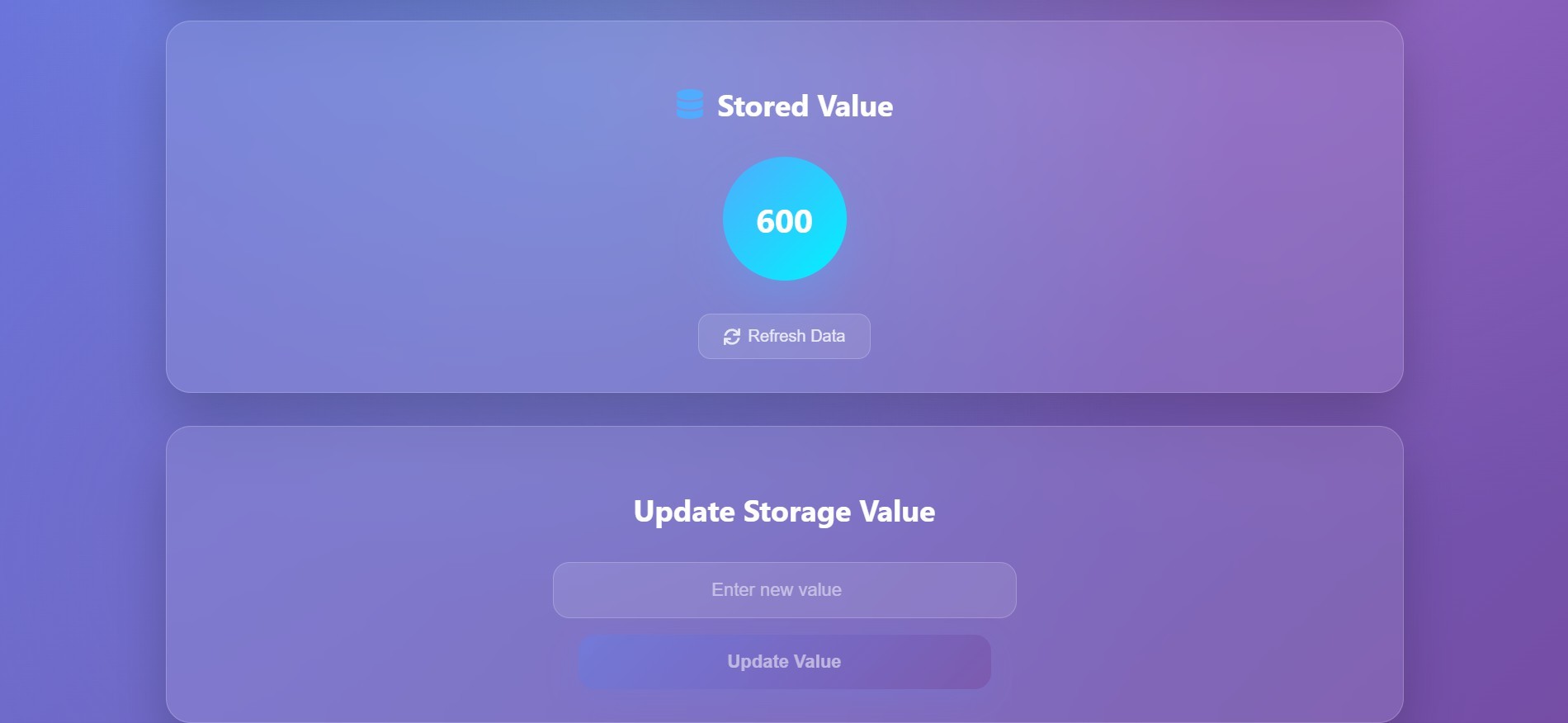
After write all the coder now for frontend design write the css .after writing all coder now install the web3 packages inside your frontend folder to install all the packages of web3 the command is -npm install web3

after installing all the packages now to run the frontend write the command npm start



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

Applied and Action Learning



Now in this frontend you can update value and check stored value

# \* Observations

1. Ethers.js provides a lightweight and modular approach for interacting with Ethereum smart contracts.
2. It simplifies wallet connection and contract function calls using a clean and modern syntax.
3. The library ensures better security and improved developer experience compared to older Web3.js practices.



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