BIT 04103 BLOCKCHAIN TECHNOLOGY AND APPLICATION

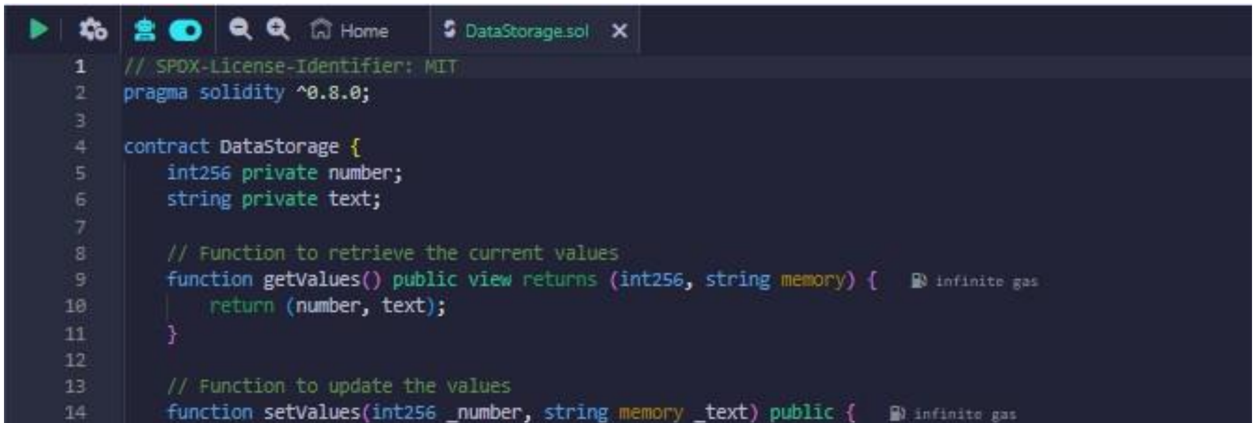
Practical Session Assignment

Group Members

1.Onyango Hillary Valentine BIT 21/03083
2.Zipporah N. Ombasa BIT 23/04181

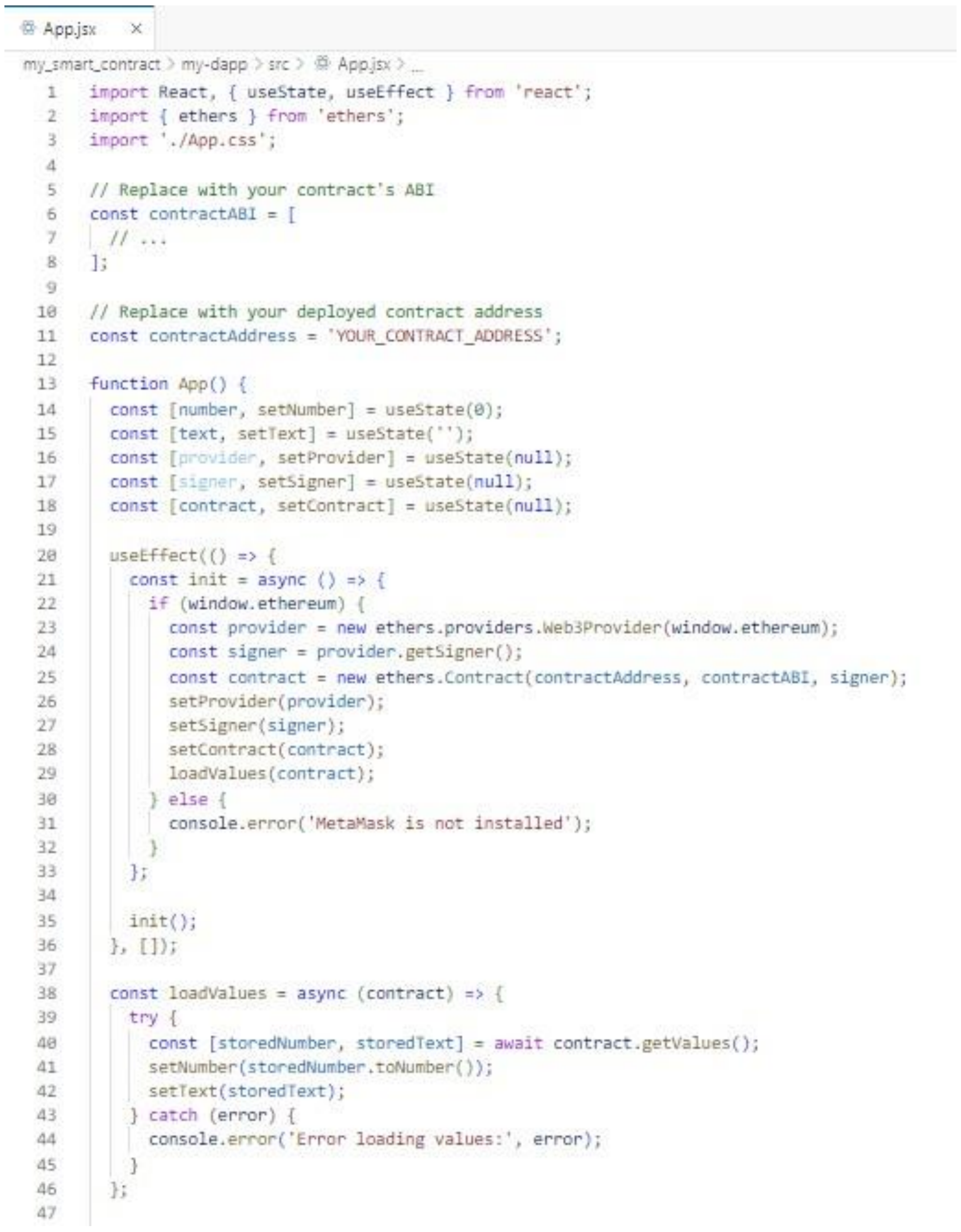
Write a smart contract with solidity where it's two inputs for int256 and string they can be blank during deployment. And a display area for the current value of the inputs in the frontend which updates when it's changed. And one can edit it from the frontend. Deploy it to Sepolia ETH testnet Create a frontend for it (preferably with vite because it'll be easier) Connect your frontend with your deployed smart contract

(a) Smart contract. (DataStorage.sol)

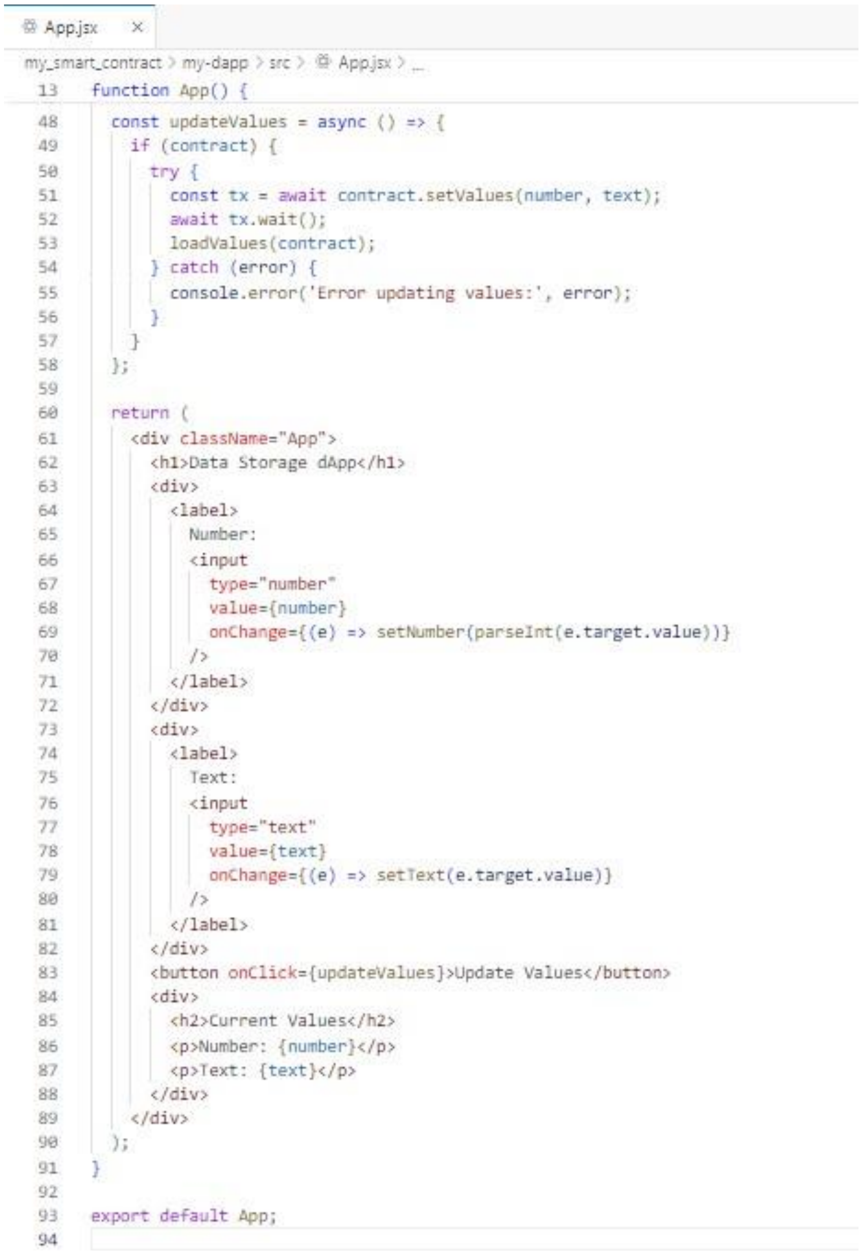


```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract DataStorage {
    int256 private number;
    string private text;

    // Function to retrieve the current values
    function getValues() public view returns (int256, string memory) {    infinite gas
        return (number, text);
    }

    // Function to update the values
    function setValues(int256 _number, string memory _text) public {    infinite gas
```

(b) Integration of frontend with smart contract. (App.jsx)



```jsx
import React, { useState, useEffect } from 'react';
import { ethers } from 'ethers';
import './App.css';

// Replace with your contract's ABI
const contractABI = [
  // ...
];

// Replace with your deployed contract address
const contractAddress = 'YOUR_CONTRACT_ADDRESS';

function App() {
  const [number, setNumber] = useState(0);
  const [text, setText] = useState('');
  const [provider, setProvider] = useState(null);
  const [signer, setSigner] = useState(null);
  const [contract, setContract] = useState(null);

  useEffect(() => {
    const init = async () => {
      if (window.ethereum) {
        const provider = new ethers.providers.Web3Provider(window.ethereum);
        const signer = provider.getSigner();
        const contract = new ethers.Contract(contractAddress, contractABI, signer);
        setProvider(provider);
        setSigner(signer);
        setContract(contract);
        loadValues(contract);
      } else {
        console.error('MetaMask is not installed');
      }
    };

    init();
  }, []);

  const loadValues = async (contract) => {
    try {
      const [storedNumber, storedText] = await contract.getValues();
      setNumber(storedNumber.toNumber());
      setText(storedText);
    } catch (error) {
      console.error('Error loading values:', error);
    }
  };

```

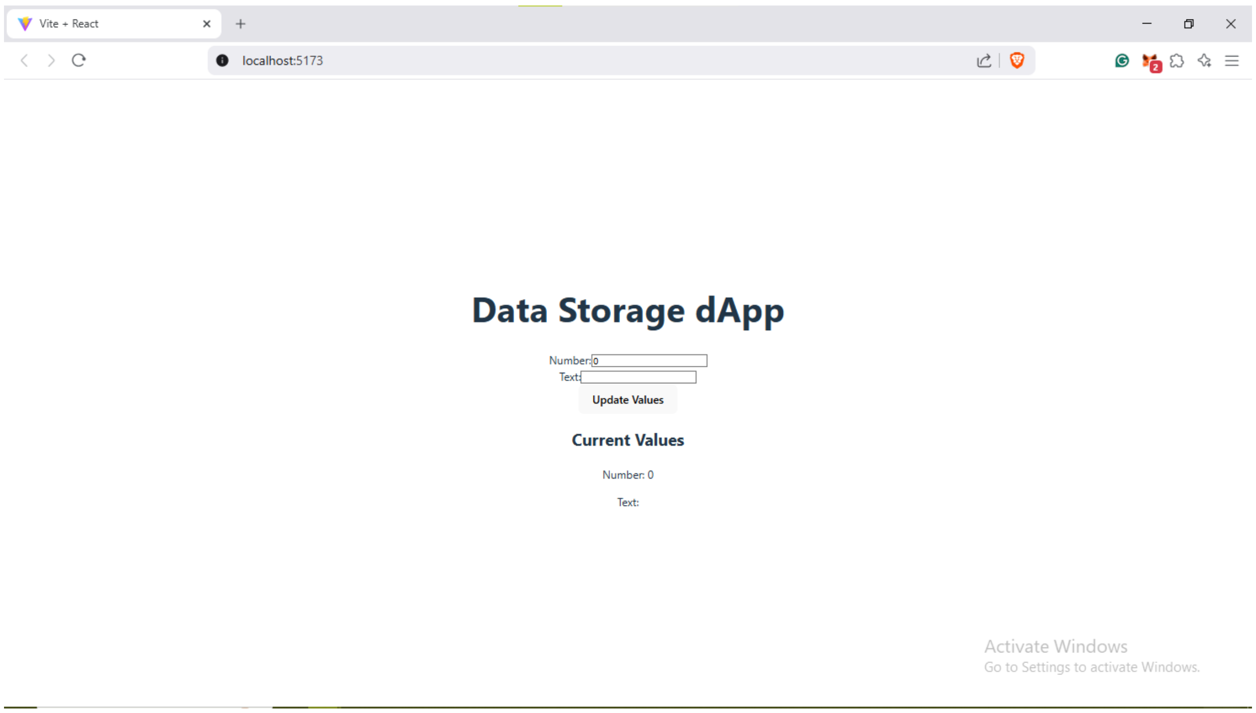


App.jsx

my_smart_contract > my-dapp > src > App.jsx > ...

```
function App() {
  const updateValues = async () => {
    if (contract) {
      try {
        const tx = await contract.setValues(number, text);
        await tx.wait();
        loadValues(contract);
      } catch (error) {
        console.error('Error updating values:', error);
      }
    }
  };

  return (
    <div className="App">
      <h1>Data Storage dApp</h1>
      <div>
        <label>
          Number:
          <input
            type="number"
            value={number}
            onChange={(e) => setNumber(parseInt(e.target.value))}
          />
        </label>
      </div>
      <div>
        <label>
          Text:
          <input
            type="text"
            value={text}
            onChange={(e) => setText(e.target.value)}
          />
        </label>
      </div>
      <button onClick={updateValues}>Update Values</button>
      <div>
        <h2>Current Values</h2>
        <p>Number: {number}</p>
        <p>Text: {text}</p>
      </div>
    </div>
  );
}

export default App;
```

(c) Display area



Data Storage dApp

Number: 0

Text:

Update Values

Current Values

Number: 0

Text: