ASSIGNMENT – 5

(Blockchain Technology)

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

Submitted to: Mr. Shashikant

Name: Ishit Singh

Class: 3NC1

Roll No.: 102115023

Semester: Jan'24-May'24

Problem Statement :

Design and Develop a Decentralized application for the Lottery auction and declare the winner.

IMAGES :

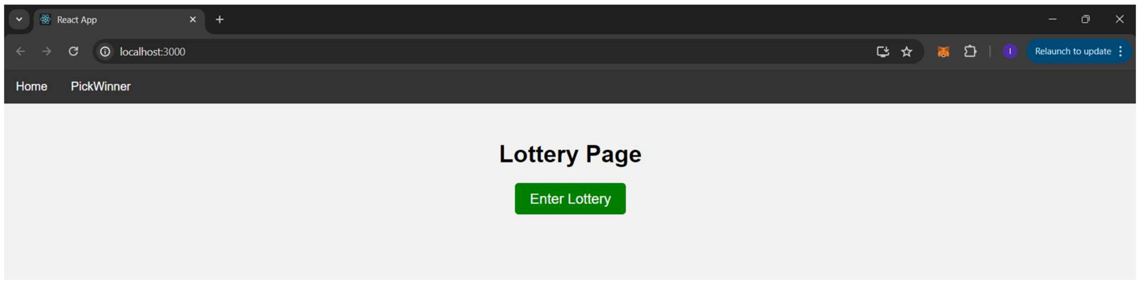


Fig a) Starting page of the Decentralised Lottery Application

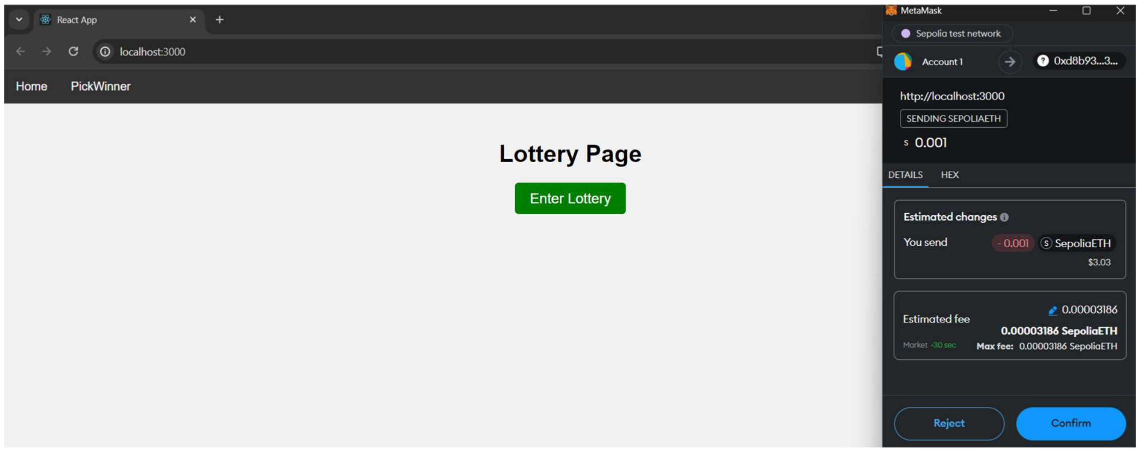


Fig b) Account-1 Entering the lottery by giving 0.001ether as fees

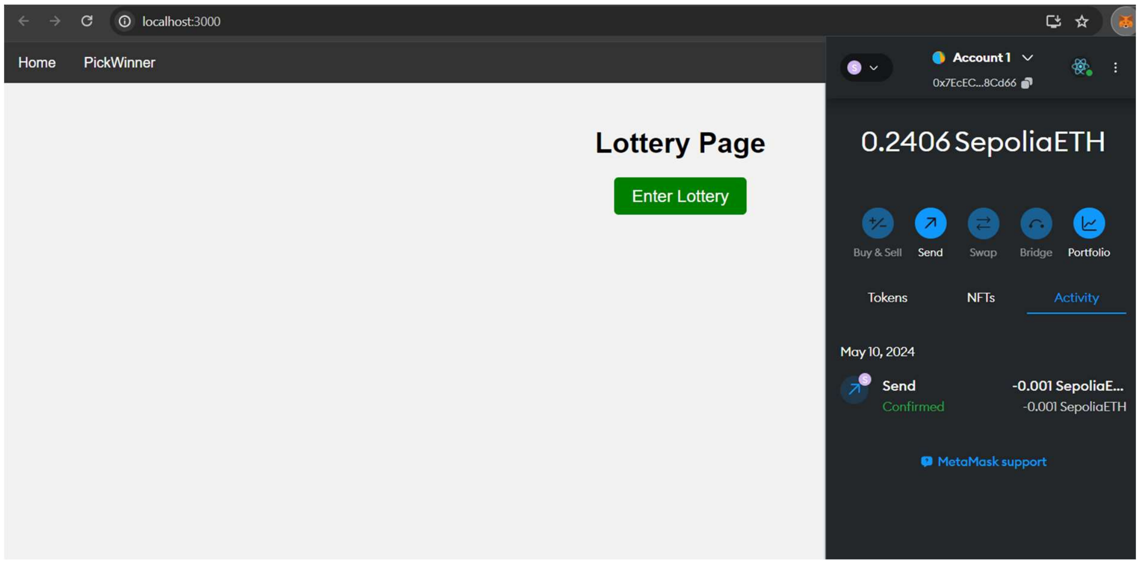
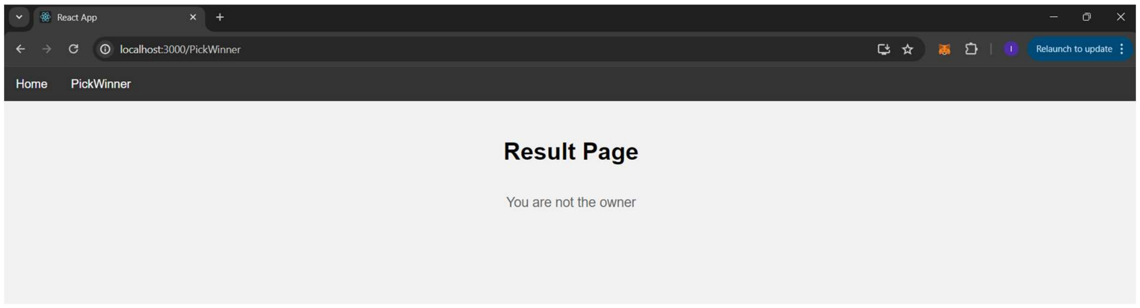


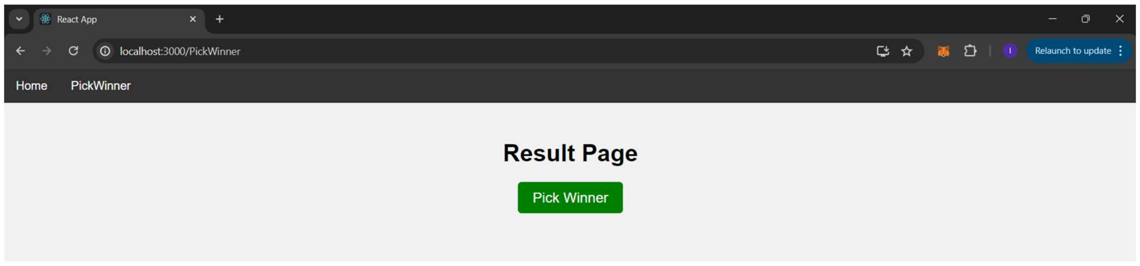
Fig c) Account-1 enters the lottery (similarly do for account-2 & account-3)



Result Page

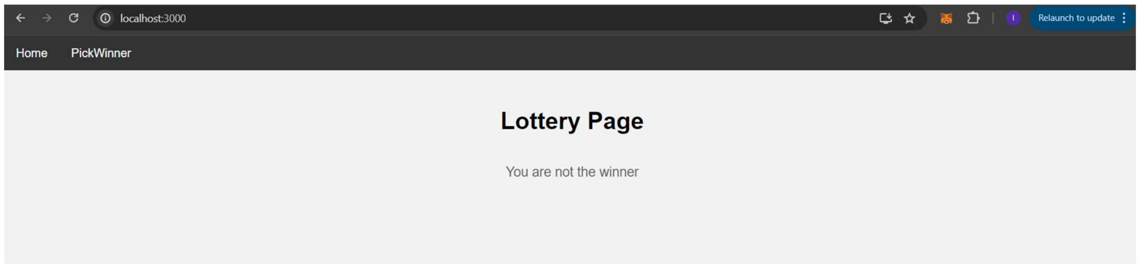
You are not the owner

Fig d) Account-2 & Account-3 NOT the owner of lottery, so they can't choose winner



Result Page

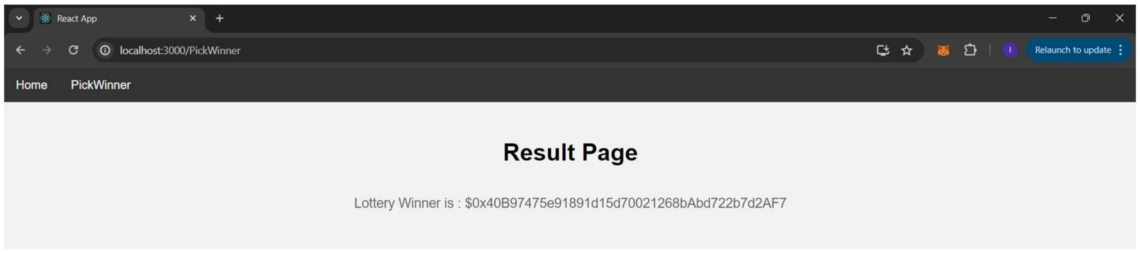
Fig e) Account-1 the OWNER of lottery, so it CAN choose the winner



Lottery Page

You are not the winner

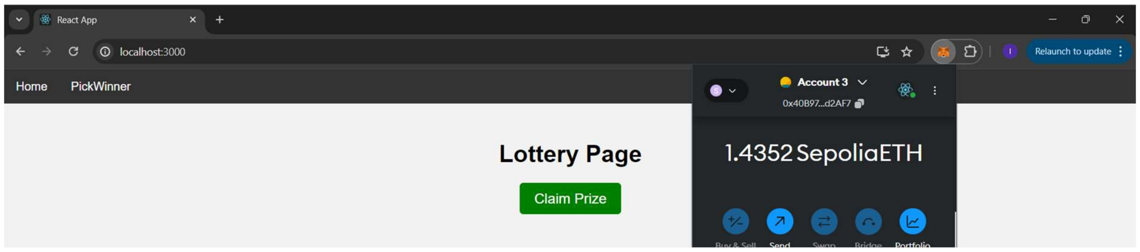
Fig f) Account-1 & Account-2 NOT the lottery winners



Result Page

Lottery Winner is : $0x40B97475e91891d15d70021268bAbd722b7d2AF7

Fig g) Address of winner account displayed on it's PickWinner Page (here Account-3 WINS)



Lottery Page

Fig h) Account-3 can claim prize of 0.003 ether on its Home Page

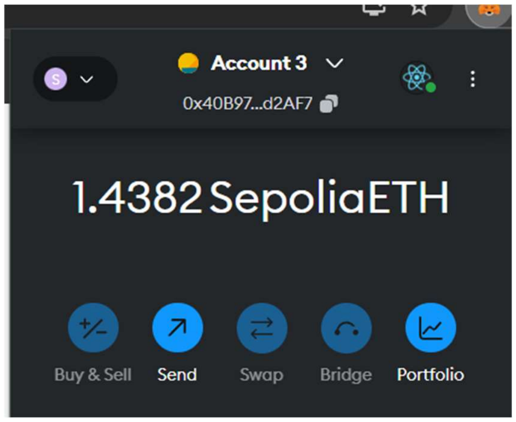


Fig i) Amount of 0.003 ether received by Account-3

CODES :

1. LOTTERY Contract

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.13;

contract Lottery{
    address public owner;
    address payable[] public players;
    address payable winner;
    bool public isComplete; // To check if all of the lottery is completed
    bool public isClaimed; // Once an account has claimed the price he/she
can't claim again

    constructor(){
        owner = msg.sender;
        isComplete = false;
        isClaimed = false;
    }

    modifier onlyOwner(){
        require(msg.sender == owner);
        _;
    }

    function status() public view returns(bool){
        return isComplete;
    }

    function getOwner() public view returns(address){
        return owner;
    }

    function getWinner() public view returns(address){
        return winner;
    }

    function getPlayers() public view returns(address payable[] memory){
        return players;
    }

    function enterLottery() public payable{
        require(msg.value >= 0.001 ether,"Minimum Entry fees in lottery");
        require(isComplete == false); // Lottery is NOT Completed
        players.push(payable(msg.sender));
    }

    function pickWinner() public payable onlyOwner(){
        require(players.length > 0, "No Players in the Lottery");
        require(isComplete == false); // Lottery is NOT Completed

        winner = players[random() % players.length];

        delete players; // Reset the lottery by initializing an empty array of
players
        isComplete = true;
    }

    function random() private view returns (uint) {
        return uint(keccak256(abi.encodePacked(block.prevrandao,
block.timestamp, players.length)));
    }

    function claimPrize() public{
        require(msg.sender == winner);
        require(isComplete);
        uint contractBalance = address(this).balance;
        (bool sent,) = winner.call{value : contractBalance}("");
        require(sent == true,"Failed to send Ethers to Winner");
        isClaimed = true;
    }

}
```

-- Building Frontend of Decentralised Application ---------------------------------

2. APP.JS [Frontend folder connecting components of app]

```
import React from 'react';
import HomePage from './HomePage';
import PickWinnerPage from './PickWinnerPage';
import {BrowserRouter, Routes, Route, Link} from 'react-router-dom';
import './App.css';
```

```
function App() {
  return(
    <BrowserRouter>
    <div>
      <nav>
        <ul>
          <li>
            <Link to="/">Home</Link>
          </li>
          <li>
            <Link to="/PickWinner">PickWinner</Link>
          </li>
        </ul>
      </nav>

      <Routes>
        <Route path="/PickWinner" element={<PickWinnerPage />}></Route>
        <Route path="/" element={<HomePage />}></Route>
      </Routes>
    </div>
    </BrowserRouter>
  )
}

export default App;
```

3. HOMEPAGE.JS [Code for building the home page of frontend]

```
import {useState,useEffect}from 'react';
import { ethers } from 'ethers';
import constants from './constants';

function HomePage() {
    const [currentAccount, setCurrentAccount] = useState("");
    const [contractInstance, setContractInstance] = useState('');
    const [statusCompletion, setStatusCompletion] = useState(false);
    const [isWinner, setisWinner] = useState('');

    // whenever our website runs --> functions under useeffect run again
    useEffect(() => {
        // 1. Function connecting to the blockchain network
        const querryBlockchain = async () => {
            // If Metamask is Installed & connected
            if (typeof window.ethereum !== 'undefined') {
                const provider = new
ethers.providers.Web3Provider(window.ethereum);
```

```
                try {
                    const signer = provider.getSigner();
                    const userAddress = await signer.getAddress();
                    console.log(userAddress);
                    setCurrentAccount(userAddress);
                    window.ethereum.on('accountChanged', (accounts) => {
                        if(accounts.length() > 0){
                            setCurrentAccount(accounts[0]);
                            console.log(currentAccount);
                        }
                        else{
                            console.log('No accounts available after account
changed');
                        }
                    })
                } catch (err) {
                    console.error(err);
                }
            }
            else {
                alert('Please install Metamask to use this application');
            }
        };


        // 2. Function connecting to the smart contract
         const contractConnection = async () => {
            const provider = new
ethers.providers.Web3Provider(window.ethereum);
            const signer = provider.getSigner();
            const contractCopy = new
ethers.Contract(constants.contractAddress,constants.contractAbi,signer);
            setContractInstance(contractCopy);
            const currStatus = await contractInstance.status();
            setStatusCompletion(currStatus);
            const winner = await contractInstance.getWinner();
            if(winner === currentAccount){
                setisWinner(true);
            }
            else{
                setisWinner(false);
            }
         }

        querryBlockchain();
        contractConnection();

    }, [currentAccount]);
    const claimPrize = async () => {
        const tx = await contractInstance.claimPrize();
        await tx.wait();
    }

    const enterLottery = async () => {
        const amountToSend = ethers.utils.parseEther('0.001');
        const tx = await contractInstance.enterLottery({value:
amountToSend,});
        await tx.wait();
    }

    return (
        <div className="container">
            <h1>Lottery Page</h1>
            <div className="button-container">
                {
                    statusCompletion == true ? (
                        isWinner == true ? (
                            <button className="enter-button"
onClick={claimPrize}>
                                Claim Prize
                            </button>
                        ) : (
                            <p>You are not the winner</p>
                            )
                    ) : (
                            <button className="enter-button"
onClick={enterLottery}>
                                Enter Lottery
                            </button>
                        )
                }
            </div>
        </div>
    );

}

export default HomePage;
```

4. PICKWINNERPAGE.JS [Code for building a pickwinner page for frontend]

```
import React, {useEffect, useState} from 'react';
import {ethers} from 'ethers';
import constants from './constants';
```

```
function PickWinner() {
    const [owner,setOwner] = useState('');
    const [currentAccount,setCurrentAccount] = useState('');
    const [contractInstance,setContractInstance] = useState('');
    const [isOwnerConnected,setIsOwnerConnected] = useState(false);
    const [winner,setWinner] = useState('');
    const [statusCompletion,setStatusCompletion] = useState(false);

    useEffect(() => {
        // 1. Function connecting to the blockchain network
        const querryBlockchain = async () => {
            // If Metamask is Installed & connected
            if (typeof window.ethereum !== 'undefined') {
                const provider = new
ethers.providers.Web3Provider(window.ethereum);
                try {
                    const signer = provider.getSigner();
                    const userAddress = await signer.getAddress();
                    console.log(userAddress);
                    setCurrentAccount(userAddress);
                    window.ethereum.on('accountChanged', (accounts) => {
                        if(accounts.length() > 0){
                            setCurrentAccount(accounts[0]);
                            console.log(currentAccount);
                        }
                        else{
                            console.log('No accounts available after account
changed');
                        }
                    })
                } catch (err) {
                    console.error(err);
                }
            }
            else {
                alert('Please install Metamask to use this application');
            }
        };


        // 2. Function connecting to the smart contract
         const contractConnection = async () => {
            const provider = new
ethers.providers.Web3Provider(window.ethereum);
            const signer = provider.getSigner();
            const contractCopy = new
ethers.Contract(constants.contractAddress,constants.contractAbi,signer);
            setContractInstance(contractCopy);
            const status = await contractCopy.status();
            setStatusCompletion(status);
            const winner = await contractCopy.getWinner();
            setWinner(winner);
            const owner = await contractCopy.getOwner();
            setOwner(owner);
            if(owner === currentAccount){
                setIsOwnerConnected(true);
            }
            else{
                setIsOwnerConnected(false);
            }
         }

        querryBlockchain();
        contractConnection();
    }, [currentAccount]);

    const pickWinnerForLottery = async () => {
        const tx = await contractInstance.pickWinner();
        // await tx.wait();
    }

    return (
        <div className="container">
            <h1>Result Page</h1>
            <div className="button-container">
                {
                    statusCompletion === true ? (
                        <p>Lottery Winner is : ${winner}</p>
                    ) : (
                        isOwnerConnected === true ? (
                            <button className="enter-button"
onClick={pickWinnerForLottery}>
                                Pick Winner
                            </button>
                        ) : (<p>You are not the owner</p>)
                    )
                }
            </div>
        </div>
    );

}

export default PickWinner;
```

5. INDEX.CSS [Designing of the frontend pages]

```
body {
  margin: 0;
  font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto',
'Oxygen',
    'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',
    sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}

code {
  font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New',
    monospace;
}
```

6. CONSTANTS.JS [Contains the contract address and contract ABI]

```
const contractAddress = "0x19C0585B672FC14B044B96Bcb9520980DD400f6C";
const contractAbi = [
    {
        "inputs": [],
        "name": "claimPrize",
        "outputs": [],
        "stateMutability": "nonpayable",
        "type": "function"
    },
    {
        "inputs": [],
        "name": "enterLottery",
        "outputs": [],
        "stateMutability": "payable",
        "type": "function"
    },
    {
        "inputs": [],
        "name": "pickWinner",
        "outputs": [],
        "stateMutability": "payable",
        "type": "function"
    },
    {
        "inputs": [],
        "stateMutability": "nonpayable",
        "type": "constructor"
    },
    {
        "inputs": [],
        "name": "getOwner",
        "outputs": [
            {
                "internalType": "address",
                "name": "",
                "type": "address"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    },
    {
        "inputs": [],
        "name": "getPlayers",
        "outputs": [
            {
                "internalType": "address payable[]",
                "name": "",
                "type": "address[]"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    },
    {
        "inputs": [],
        "name": "getWinner",
        "outputs": [
            {
                "internalType": "address",
                "name": "",
                "type": "address"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    },
    {
        "inputs": [],
        "name": "isClaimed",
        "outputs": [
            {
                "internalType": "bool",
                "name": "",
                "type": "bool"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    },
    {
        "inputs": [],
        "name": "isComplete",
        "outputs": [
            {
                "internalType": "bool",
                "name": "",
                "type": "bool"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    },
    {
        "inputs": [],
        "name": "owner",
        "outputs": [
            {
                "internalType": "address",
                "name": "",
                "type": "address"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    },
    {
        "inputs": [
            {
                "internalType": "uint256",
                "name": "",
                "type": "uint256"
            }
        ],
        "name": "players",
        "outputs": [
            {
                "internalType": "address payable",
                "name": "",
                "type": "address"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    },
    {
        "inputs": [],
        "name": "status",
        "outputs": [
            {
                "internalType": "bool",
                "name": "",
                "type": "bool"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    }
];

export default {contractAddress, contractAbi};
```