import React, { useState, useEffect } from "react";

import { ethers } from "ethers";

import VerifiedEventTicketsABI from "./VerifiedEventTickets.json"; // ABI JSON generated on compile

const CONTRACT\_ADDRESS = "0xYourDeployedContractAddress"; // Replace with your contract address

const TICKET\_PRICE = "0.05"; // in ETH, adjust to match contract

function App() {

const [provider, setProvider] = useState(null);

const [signer, setSigner] = useState(null);

const [contract, setContract] = useState(null);

const [account, setAccount] = useState(null);

const [ticketIdToVerify, setTicketIdToVerify] = useState("");

const [status, setStatus] = useState("");

useEffect(() => {

if (window.ethereum) {

const ethProvider = new ethers.BrowserProvider(window.ethereum);

setProvider(ethProvider);

} else {

setStatus("Please install MetaMask");

}

}, []);

const connectWallet = async () => {

try {

const accounts = await window.ethereum.request({ method: "eth\_requestAccounts" });

setAccount(accounts[0]);

const signer = await provider.getSigner();

setSigner(signer);

const contractInstance = new ethers.Contract(CONTRACT\_ADDRESS, VerifiedEventTicketsABI, signer);

setContract(contractInstance);

setStatus("Wallet connected");

} catch (err) {

setStatus("Connection failed");

console.error(err);

}

};

const buyTicket = async () => {

if (!contract) return setStatus("Contract not loaded");

try {

setStatus("Buying ticket...");

const tx = await contract.buyTicket({ value: ethers.parseEther(TICKET\_PRICE) });

await tx.wait();

setStatus("Ticket purchased!");

} catch (err) {

setStatus("Purchase failed");

console.error(err);

}

};

const verifyTicket = async () => {

if (!contract) return setStatus("Contract not loaded");

if (!ticketIdToVerify) return setStatus("Enter ticket ID to verify");

try {

setStatus("Verifying ticket...");

const tx = await contract.verifyTicket(ticketIdToVerify);

await tx.wait();

setStatus(`Ticket ${ticketIdToVerify} verified!`);

} catch (err) {

setStatus("Verification failed");

console.error(err);

}

};

return (

<div style={{ padding: 20, fontFamily: "Arial" }}>

<h2>Event Ticketing DApp</h2>

{!account ? (

<button onClick={connectWallet}>Connect Wallet</button>

) : (

<>

<p>Connected account: {account}</p>

<button onClick={buyTicket}>Buy Ticket ({TICKET\_PRICE} ETH)</button>

<hr />

<h3>Verify Ticket (Owner only)</h3>

<input

type="number"

placeholder="Ticket ID"

value={ticketIdToVerify}

onChange={(e) => setTicketIdToVerify(e.target.value)}

/>

<button onClick={verifyTicket}>Verify Ticket</button>

</>

)}

<p>Status: {status}</p>

</div>

);

}

export default App;