// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract PropertyDetail{

address public owner;

struct Property {

string propertyId;

string name;

string location;

string discription;

address currentOwner;

}

mapping(string => Property) public properties;

mapping(address => mapping(string => bool)) public hasAccess;

event PropertyAdded(

string indexed propertyId,

string name,

string location,

address indexed owner

;

event PropertyTransferred(

string indexed propertyId,

address indexed from,

address indexed to

);

constructor() {

owner = msg.sender;

}

modifier onlyOwner() {

require(msg.sender == owner, "Only contract owner can call this");

\_;

}

modifier hasPropertyAccess(string memory propertyId) {

require(

hasAccess[msg.sender][propertyId],

"You don't have access to this property"

);

\_;

}

function addProperty(

string memory propertyId,

string memory name,

string memory location,

string memory \_description

) external onlyOwner {

require(

bytes(properties[propertyId].propertyId).length == 0,

"Property already exists"

);

properties[propertyId] = Property({

propertyId: propertyId,

name: name,

location: location,

discription : \_description,

currentOwner: owner

});

hasAccess[owner][propertyId] = true;

emit PropertyAdded(propertyId, name, location, owner);

}

function transferProperty(

string memory propertyId,

address newOwner

) external hasPropertyAccess(propertyId) {

require(newOwner != address(0), "Invalid new owner");

address currentOwner = properties[propertyId].currentOwner;

properties[propertyId].currentOwner = newOwner;

hasAccess[currentOwner][propertyId] = false;

hasAccess[newOwner][propertyId] = true;

emit PropertyTransferred(propertyId, currentOwner, newOwner);

}

function getPropertyDetails(

string memory propertyId

) external view returns (string memory, string memory, address) {

Property memory prop = properties[propertyId];

return (prop.name, prop.location, prop.currentOwner);

}

}

Connector.js

const { ethers } = require("ethers");

const abi = [

{

"inputs": [],

"stateMutability": "nonpayable",

type": "constructor"

},

{

"anonymous": false,

"inputs": [

{

"indexed": true,

"internalType": "string",

"name": "propertyId",

"type": "string"

},

{

"indexed": false,

"internalType": "string",

"name": "name",

"type": "string"

},

{

"indexed": false,

"internalType": "string",

"name": "location",

"type": "string"

},

{

"indexed": true,

"internalType": "address"

name": "owner",

"type": "address"

}

],

"name": "PropertyAdded",

"type": "event"

},

{

"anonymous": false,

"inputs": [

{

"indexed": true,

"internalType": "string",

"name": "propertyId",

"type": "string"

},

{

"indexed": true,

"internalType": "address",

"name": "from",

"type": "address"

},

{

"indexed": true,

"internalType": "address",

"name": "to",

type": "address"

}

],

"name": "PropertyTransferred",

"type": "event"

},

{

"inputs": [

{

"internalType": "string",

"name": "propertyId",

"type": "string"

},

{

"internalType": "string",

"name": "name",

"type": "string"

},

{

"internalType": "string",

"name": "location",

"type": "string"

},

{

"internalType": "string",

"name": "\_description",

type": "string"

}

],

"name": "addProperty",

"outputs": [],

"stateMutability": "nonpayable",

"type": "function"

},

{

"inputs": [

{

"internalType": "string",

"name": "propertyId",

"type": "string"

}

],

"name": "getPropertyDetails",

"outputs": [

{

"internalType": "string",

"name": "",

"type": "string"

},

{

"internalType": "string",

name": ""

type": "string"

},

{

"internalType": "address",

"name": "",

"type": "address"

}

],

"stateMutability": "view",

"type": "function"

},

{

"inputs": [

{

"internalType": "address",

"name": "",

"type": "address"

},

{

"internalType": "string",

"name": "",

"type": "string"

}

],

"name": "hasAccess",

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": "string",{

"internalType": "address",

"name": "currentOwner",

"type": "address"

}

],

"stateMutability": "view",

"type": "function"

},

{

"inputs": [

{

"internalType": "string",

"name": "propertyId",

"type": "string"

},

{

"internalType": "address",

"name": "newOwner",

"type": "address"

}

],

"name": "transferProperty",

"outputs": [],

"stateMutability": "nonpayable"

type": "function"

}

]

if (!window.ethereum) {

alert('Meta Mask Not Found')

window.open("https://metamask.io/download/")

}

export const provider = new ethers.providers.Web3Provider(window.ethereum);

export const signer = provider.getSigner();

export const address = "0xA074a1f2E1285E292cC4a0aD8c14a1037071993F"

export const contract = new ethers.Contract(address, abi, signer)