// SPDX-License-Identifier: GPL-3.0

pragma solidity >=0.7.0<0.9.0;

contract project1

{

    constructor(){landinspectoraddress=msg.sender;}

    enum verification{notverified,verified}

    struct landregistrydetails

    {

        verification status;

        uint landid;

        string area;

        string city;

        string state;

        uint landprice;

        uint propertyid;

    }

    struct buyerdetails

    {

        address buyerid;

        string name;

        uint age;

        string city;

        uint CNIC;

        string email;

    }

    struct sellerdetails

    {

        address sellerid;

        string name;

        uint age;

        string city;

        uint CNIC;

        string email;

    }

    struct landinspectordetails

    {

        address id;

        string name;

        uint age;

        string designation;

    }

    mapping(uint=>landregistrydetails)public landregistry;

    mapping(address=>buyerdetails)public buyer;

    mapping(address=>sellerdetails)public seller;

    mapping(address=>landinspectordetails)public landinspector;

    address public landinspectoraddress;

    mapping(address=>bool)public verifyseller;

    mapping(address=>bool)public rejectseller;

    mapping(address=>bool)public verifybuyer;

    mapping(address=>bool)public rejectbuyer;

    mapping(uint=>address)public landowner;

    mapping(address=>bool)public isSellermapping;

    mapping(address=>bool)public isBuyermapping;

    modifier onlylandinspector()

    {

        require(landinspectoraddress==msg.sender,"You are not Landinspector");

        \_;

    }

    modifier onlyverifiedseller()

    {

        require(!rejectseller[msg.sender],"You are rejected");

        require(verifyseller[msg.sender]==true,"You are not verified");

        \_;

    }

    modifier onlyverifiedbuyer()

    {

        require(!rejectbuyer[msg.sender],"You are rejected");

        require(verifybuyer[msg.sender]==true,"You are not verified");

        \_;

    }

    function purchaseland(uint landid, uint amount)onlyverifiedbuyer public payable

    {

        require(seller[msg.sender].sellerid!=msg.sender,"you can't purchase the land");

        address oldowner=landowner[landid];

        require(landregistry[landid].status==verification.verified,"The land is not verified");

        require(landregistry[landid].landprice==amount,"please pay full amount");

        require(landregistry[landid].landprice==msg.value,"transfer full amount");

        payable (oldowner).transfer(amount);

        landowner[landid]=msg.sender;

    }

    function landregister(uint landid, string memory area,string memory city,string memory state,uint landprice,uint propertyid)public

    {

        landregistry[landid]=landregistrydetails(verification.notverified,landid,area,city,state,landprice,propertyid);

        landowner[landid]=msg.sender;

    }

    function verifyland(uint landid)public

    {

        landregistry[landid].status=verification.verified;

    }

    function registerbuyer(string memory name,uint age,string memory city,uint CNIC,string memory email)public

    {

        buyer[msg.sender]=buyerdetails(msg.sender,name,age,city,CNIC,email);

        isBuyermapping[msg.sender]=true;

    }

    function registerseller(string memory name,uint age,string memory city,uint CNIC,string memory email)public

    {

        require(seller[msg.sender].sellerid!=msg.sender,"you are already registered");

        seller[msg.sender]=sellerdetails(msg.sender,name,age,city,CNIC,email);

        isSellermapping[msg.sender]=true;

    }

    function updateseller(string memory name,uint age,string memory city,uint CNIC,string memory email)public

    {

        seller[msg.sender]=sellerdetails(msg.sender,name,age,city,CNIC,email);

    }

    function updatebuyer(string memory name,uint age,string memory city,uint CNIC,string memory email)public

    {

        buyer[msg.sender]=buyerdetails(msg.sender,name,age,city,CNIC,email);

    }

    function \_landinspectordetails(string memory name, uint age, string memory designation)public

    {

        landinspector[landinspectoraddress]=landinspectordetails(landinspectoraddress,name,age,designation);

    }

    function verify\_seller(address sellerid)onlylandinspector()public

    {

        verifyseller[sellerid]=true;

    }

    function reject\_seller(address sellerid)onlylandinspector()public

    {

        rejectseller[sellerid]=true;

    }

    function verify\_buyer(address buyerid)onlylandinspector()public

    {

        verifybuyer[buyerid]=true;

    }

    function reject\_buyer(address buyerid)onlylandinspector()public

    {

        rejectbuyer[buyerid]=true;

    }

    function transferowner(address newowner, uint landid)public

    {

         landowner[landid]=newowner;

    }

    function isSeller(address sellerid)public view returns(bool)

    {

        if(isSellermapping[sellerid])

            return true;

        else

            return false;

    }

    function isBuyer(address buyerid)public view returns(bool)

    {

        if(isBuyermapping[buyerid])

            return true;

        else

            return false;

    }

    function checksellerVerification(address sellerid) public view returns(bool)

    {

        if(verifyseller[sellerid]==true)

            return true;

        else

            return false;

    }

    function checkbuyerVerification(address buyerid) public view returns(bool)

    {

        if(verifybuyer[buyerid]==true)

            return true;

        else

            return false;

    }

    function checklandVerification(uint landid)public view returns(bool)

    {

        if(landregistry[landid].status==verification.verified)

            return true;

        else

            return false;

    }

    function checklandowner(uint landid) public view returns(address)

    {

        return landowner[landid];

    }

    function getlanddetails(uint landid) public view returns(uint,string memory,string memory,string memory,uint,uint)

    {

        return (

        landregistry[landid].landid,

        landregistry[landid].area,

        landregistry[landid].city,

        landregistry[landid].state,

        landregistry[landid].landprice,

        landregistry[landid].propertyid);

    }

    function checklandinspector() public view returns(address)

    {

        return landinspectoraddress;

    }

    function landPrice(uint landid) public view returns(uint)

    {

      return landregistry[landid].landprice;

    }

    function landCity(uint landid) public view returns(string memory)

    {

      return landregistry[landid].city;

    }

    function landArea(uint landid) public view returns(string memory)

    {

      return landregistry[landid].area;

    }}