```
pracma solidity a.4.17;
11Land Details
contract
    LandRegistration{  st
    ruct landDetails{
        string state;
        strin1 district;
        strin1 location;
        strin1 landMark;
        uint256 plotNo;
        address payable CurrentOWner;
        uint priceSelling;
        bool isAvailable;
        address requester;
        reqStatus requestStatus;
    }
    enum reqStatus {Default,pending,reject,approved}
    //profile of a client
    struct profiles{
        uint[]assetlist;
        }
    mapping (uint => landDetails) Land;
    address owner;
    m1pping(string => address)manager;
    mapping (address => profiles) profile;

  //contract o..ner
  constructor () public{
      owner • 11sg.sender;
  }
  110dif ier onlyOwner {
      require( 111si.serder == owner);
      _; }
  / ladcirg locat!O" .,.ager
  function add!olanager ( address -1"anag2r,string MllOry _location } onl>'°"""tr public {
  unagtr(_location]•_l"a nager;  }
    Deahtrato  o• land deta: s
  function register (string 111110ry _state,strinc 11NOry _district ,
      strina 11tciory  _:ocahon,strina llellOl'y  _la!ld"ar1e,uint2S6 J>lotNo ,
      add,..ss payable _  .nerAddr ess,uint _pr ceSelllng,uint       er
      ) public returns (bool ) {
      requir1(111n1ger(_location] == l"Sg.sede•     °"ner ==
      ug.sender); land [Nu•ber) .state = _state;
      land [Nu111ber) .district = _district
      ;       land[Nu111btr].location       =
      _location ;   land[Nu111ber) .landMark
      = _landMark;  land[Nu111ber] .plotNo =
      _plotNo;
      land[Number J .CurrentOwner = _OwnerAddress;
      land[Number) .priceSelling = _priceSelling;
      profile (_OWnerAddress  ].assetlist. push(Number);
      return  true;  }

  //to view details of land for tne o•ner
  function Owner(uint Nu'lber) public view retums(strinc me11ory,strin1 11t110ry,
  strinc   ne1ory,strin1   1H110ry,uint256,bool,addrtss,reqStatus)
  {
      return(land[tlumber]   .state,land[taimber].district,land[Nullber].   location,
      land [Nullber ].hndJolirk,hnd[NllWer). plotNo,
      land[Nullbtr]. isAvailable, land[Nur.iber). requester ,land(Nullbtr]. requestStatus);

        'to view d"ails o• land for te o..> er
      function Buyer(uint i.d) public view retums(address,uint,bool,address,reqStatus)
      {
        return(land[Nu&berJ .Currento.ner, land[IMlber).priceSelling,  land[Ndler J.isAvailable,
        land[Ntmber] .requester,land [tll.Dber].requestStatus );


   I to CCII .te unique llJ cer fo  a 1;inc.
  f unction coetlU11ber(string 1e.:>ry _state,string MtnOry _district ,
  strinc 1e1ory _village ,uint _plotr.o ) public view retums (uint)
  {
      return   uint (keccak256( block .dif&iculty,now,_state,_district,_location,  _plotNo)))1eeeeeeeeeeee ;


  //push a request to the land owner
  f unction requstToland(Mner (uint Humber ) public {
```

require(land[Number]. isAvailable);

```
    land [Number].requester=msg.sender  ;
    land [Number].isAvailable=false;
    land [Number].requestStatus = reqstatus.pending; //changes the status to
    pending.
}
//will show assets of the function caller
function viewAssets()public  view returns(uint
    []memory){ return
    (profile(qsg.sender].assetlist);
}
//viewing request for the lands
function viewRequest(uint property)public  view returns(address){
    return(land [property].requester);
}
//processing request for the land by accepting or rejecting
function processRequest(uint property ,reqStatus status)public
{
    require(land [property].Currentowner ==
    msg.sender); land
    [property].requestStatus=status;
    if(status == reqStatus.reject){
        land(property].requester = address(e);
        land(property].requeststatus = reqStatus.Default;
    }
}
//availing land for sale.
function makeAvailable(uint property)public {
    require(land [property).CurrentOwner ==
    msg.sender); land [property].isAvailable=true;
//buying the approved property
function purchaseland (uint property)public payable{
   require(land(property].requestStatus == reqStatus.approved);
   require(msg.value   >=  (land[property].marketValue+((land
   [property].priceSelling)/10))); land
   [property).Current(}wner.transfer(land[property].marketValue);
   removeOWnership(land[property].CurrentOWner,property);
   land (property).Currente>wner=msg.sender;
   land  (property).isAvailablt=false;
   land [property).requester = addN!ss(0);
   land [property).requestStatus = reqStatus
   .Default;
   profile[msg.sender].assetlist.push(property)
   ;  }
//reoving the o•nership of seller for the land.and it is called by the purchaseland
function function  remove0."'1ership(address previousowner,uint  id)private{
   uint index = findid(id,previousOwner);
   profile[previousOwner).assetlist[index)=profile[previouse>wner).assetlist[profile[previous().o
   iner)
   .assetlist.length-l];
   delete
   profile[previousOWner).assetlist(profile(previousOwner].assetlist.length -
   1); profile[previousowner).assetlist.length--;  }
```

```
//for finding the index of a perticular Number
function findId(uint id,address user)public view
    returns(uint){  uint i;
    for(i=0;i<profile[user].assetlist.length;i++){
     if(profile[user).assetlist[i)== id)
           return i; }
    return i;
}
```