# Blockchain Based Land Registration System


## Abstract:

Land Registration is a simple Dapp based on the Ethereum Blockchain. It can be used as an alternative to the existing approach. Here the land owner registers the land details along with the land value by providing necessary proofs. Only a registrar or government authority who is registered as the SuperAdmin and admin can do the registration process. Lands coming under a particular area can register to the system only through the admin assigned to that area. The smart contract has been written in such a way that the owner has to transfer his full asset to the buyer and no partial transaction of the asset is allowed. Even though the registration process requires a government authority, the entire process is transparent and the transaction happens between the two clients without any intermediaries.


## Introduction:

For land, being a high-valued asset, it is very important to have accurate records which identify the current owner and provide the proof that he is indeed a owner. These records can be used to:

a) protect owner's rights
b) prevent sale frauds
c) resolve disputes
d) make sure ownership is correctly transferred to a new ownership

Thus, it is crucial to maintain correctness and completeness of this information, and prevent unauthorized, fraudulent changes.

Currently people rely on third party, i.e., government agencies that are responsible for keeping track of ownership information. This third party keep all the records in the centralized database. Hence to transfer the ownership, it becomes difficult and slow to first find verify the land and then transfer the ownership.

It is possible to keep track of the property ownership if we have a distributed system which stores all the land history and share it among the interested buyers. This would remove the intermediaries. And seller can directly contact the buyer. Thereby removing the extra cost and time that is needed to be spent on the intermediaries.

At minimum a blockchain based ledger is needed that stores the transactions done in the process of land ownership transfer. This problem is solved by Satoshi Nakamoto in his paper about bitcoins when he created:

a) storing the information in a blockchain,
b) for correctness protocol rules can be used,
c) and for identifying the owner public key cryptography can be used.

Ethereum is a free open-source platform which helps developers to build and deploy decentralized applications such as smart contracts and other complicated legal and financial applications. Ethereum is kind of a programmable Bitcoin where developers can use the underlying blockchain to create markets, shared ledgers, digital organizations, and other endless

solutions application to a problem that need immutable data and agreements, all without the need for a moderator or realtor.

Hence Ethereum is best suited for creating a ledger that stores transactions during the land ownership transfer process. The aim is to create a ledger along with some smart contracts that will triggers the various events that are going to happen on the system during the process of ownership transfer.

# Existing Process:

Buying a piece of land in India is very crucial and you need to pay due diligence to the entire process. Especially, you have to make sure all the papers are clear from title defects and other legal issues.
It is best to register your land with the help of reputed legal professionals and lawyers as they scrutinize each and every document and offer the advice on whether to proceed further or not.

**Procedure for land registration:**

a) **Document verification:**
   As the first step, all the documents related to the land should be verified.

b) **Drafting of the deed:**
   Irrespective of the way you have obtained the land, it is important you have the correct deed. For instance, if it is a gift, a gift deed is required. If it is a purchased one, a sale deed mentioning the contract, payment, terms and conditions agreed by the seller and the purchaser, tenure of the payment, etc.

c) **Preparing stamp paper:**
   To execute property transaction and related documents like conveyance deed, sale deed and sale agreement, a fee has to be paid to the government. It is called stamp duty. You need to get the stamp paper from authorized vendors.

d) **Execution of the deed:**
   The deed must be executed at the Registrar's office and both the parties have to be presented to duly sign the documents. If anyone of the parties either seller or purchaser is not available, then a Power of Attorney can be given to proceed with the execution.

e) **Registration:**
   As a final step, once all the documents are reviewed and found to be perfect, the land will be registered. Personal documents like PAN, Aadhar, etc. Unlike residential and commercial buildings, for land, there is no field inspection by the authorities.

# Problems and Impact of Current System:

- **The Involvement of middlemen and brokers**
  Middlemen and brokers are an integral part of every big business as they know more about market offerings. Buyers and Sellers usually prefer to call them to build a full support team. As a result, buyers acquire a deeper understanding of the market and identify lower/higher prices for the transaction. Middlemen gather required information from traders, identify errors, interpret and facilitate the implementation of real estate transactions. Since real estate is big business, it involves a huge number of players, including brokers, lenders, intermediaries and local governments. It leads to additional costs, making the entire ecosystem expensive.

- **The increasing number of fraud cases**
  There has been several cases of imposters posing as the seller of a property. If an imposter successfully pretends as a property owner, they may receive the full amount of after completion and escape with the funds. In many of the cases, both sellers and buyers were unaware of the fraud until discovered by the Land Registration as part of a spot check exercise.

- **Time Delays**
  Land Registration takes a considerably long time to complete title registrations. There could be a gap of several months between completion and registration. Many legal problems can also arise during this long gap. For example, what if you have to serve the landlord's notice to break a lease where property has been sold. Such issues can make the entire process delayed and buyers have to wait for a long time.

- **Human error/intervention**
  Currently, updates to the Land Registration records are made manually and the accuracy of those changes depends on a particular individual. It means that the Land Registration is more vulnerable to human errors. Human intervention can increase the chances of errors in the Land Registration system.

# Role of Blockchain in current system:

- **Accelerating the Process**
  Middlemen involved in the Land Registration process hold information that you cannot access, or you might not have the license required to operate in a property transaction ecosystem. But the blockchain Land Registration platform can offer you a distributed database where anyone can record and access information without the involvement of any centralized authority. At present, the title to a property/land is just a piece of paper. You require to fill blanks in the deed, sign it, get it notarized for rubber stamping and send the documents to the government to transfer the property.
  The process looks too old and slow. However, creating a digital title with blockchain Land Registration platform can improve the process.

With the blockchain's potential to prove authenticity, homeowners can transfer the land ownership legitimately to the buyer without needing third-party verification.

- **Reducing Fraud Cases**
  In today's digital world, it is now possible for imposters to forge the documents and fake the title ownership with the editing software. Blockchain Land Registration platform will allow you to upload the title documentation to the blockchain network where signers can sign the document and other users can verify it when needed. By keeping an immutable record of transactions, blockchain can prove that you are the owner of the land title and prevent from forgery of documents. Therefore, it can be said that the blockchain Land Registration platform could serve as proof of ownership, existence, exchange and transaction.

- **Bringing Transparency with Smart Contracts**
  There are only a few people who buy property directly. The process of loan or mortgage is comparatively slower due to administrative issues. But smart contracts can make the process simpler by automating verified transactions. With the blockchain Land Registration platform, you can create a digital, decentralized ID as a seller and buyer. Doing so would make ownership transfer seamless and quicker than the traditional method. As soon as the registrar confirms the transfer of land title, smart contracts trigger to update ownership for a new buyer and transaction corresponding to it gets stored on the blockchain. In this way, it is always possible to trace back the history of ownership records.

# Methodology:

**Stakeholders involved in the Blockchain Land Registration Platform:**

1. **User:** A person who can sell or buy land and uses platform to search property request access and transfer the land title ownership.
2. **Admin:** A person who registers users and properties.
3. **Land inspector (SuperAdmin):** A person who uses the platform to manage property requests, view reports, confirm and initiate the transfer.

**Step 1: Users register to the platform**

Users who either want to sell or buy properties register to the blockchain Land Registration platform.
They are identified by there account address.
A hash for the identity information submitted by the users gets stored on the blockchain.

**Step 2: Admin upload the property specifications on the platform**

Sellers can upload properties' details like property id, value/price of property, and property owners address on the platform. The transaction corresponding to the seller's action of listing the property details is recorded on the blockchain.

Once the property's details are uploaded to the platform, SuperAdmin inspects the property and approves or rejects accordingly, then it is made available to all users who have registered.

**Step 3: Buyers request access to the listed property**

A buyer interested in any specific property can send a request to change the ownership.
SuperAdmin accept the request of transaction.
Buyers can view the previous ownership records of the property and send a request to purchase it and initiate the transfer.
Transactions corresponding to the requests made by both sellers and buyers are recorded on the blockchain to ensure authenticity and traceability.

**Step 4: Land Inspector verifies the transaction and initiates the transfer**

Land inspector verifies the details submitted by buyers and sellers and adds the authenticated records to the **blockchain land registration** platform.
The changed details of ownership gets saved in the database and transaction corresponding to it is recorded on the blockchain.
The transfer is initiated and smart contracts trigger to change title's ownership to a new buyer.

# Important Codes:

**Asset.sol**

```
pragma solidity ^0.5.0;

contract asset {

   address public creatorAdmin;
        enum Status { NotExist, Pending, Approved, Rejected }

        struct PropertyDetail {
                Status status;
                uint value;
                address currOwner;
        }

        // Dictionary of all the properties, mapped using their { propertyId:
PropertyDetail } pair.
        mapping(uint => PropertyDetail) public properties;
        mapping(uint => address) public propOwnerChange;

   mapping(address => int) public users;
   mapping(address => bool) public verifiedUsers;

        modifier onlyOwner(uint _propId) {
                require(properties[_propId].currOwner == msg.sender);
                _;
        }

        modifier verifiedUser(address _user) {
```

```
        require(verifiedUsers[_user]);
        _;
    }

    modifier verifiedAdmin() {
            require(users[msg.sender] >= 2 && verifiedUsers[msg.sender]);
            _;
    }

    modifier verifiedSuperAdmin() {
        require(users[msg.sender] == 3 && verifiedUsers[msg.sender]);
        _;
    }

    // Initializing the User Contract.
  constructor  ()  public {
            creatorAdmin = msg.sender;
            users[creatorAdmin] = 3;
            verifiedUsers[creatorAdmin] = true;
    }

    // Create a new Property.
 function createProperty(uint _propId, uint _value, address _owner)
verifiedAdmin verifiedUser(_owner) public returns (bool)  {
            properties[_propId] = PropertyDetail(Status.Pending, _value,
_owner);
            return true;
    }

    // Approve the new Property.
    function approveProperty(uint _propId) verifiedSuperAdmin public
returns (bool)  {
            require(properties[_propId].currOwner != msg.sender);
            properties[_propId].status = Status.Approved;
            return true;
    }

    // Reject the new Property.
     function rejectProperty(uint _propId) verifiedSuperAdmin public
returns (bool)  {
            require(properties[_propId].currOwner != msg.sender);
            properties[_propId].status = Status.Rejected;
            return true;
    }

    // Add new user.
     function addNewUser(address _newUser) verifiedAdmin public
returns (bool)  {
        require(users[_newUser] == 0);
        require(verifiedUsers[_newUser] == false);
        users[_newUser] = 1;
        return true;
```

```
        }

        // Add new Admin.
         function addNewAdmin(address _newAdmin) verifiedSuperAdmin
public returns (bool)  {
            require(users[_newAdmin] == 0);
            require(verifiedUsers[_newAdmin] == false);
            users[_newAdmin] = 2;
            return true;
        }

        // Add new SuperAdmin.
         function addNewSuperAdmin(address _newSuperAdmin)
verifiedSuperAdmin public returns (bool)  {
            require(users[_newSuperAdmin] == 0);
            require(verifiedUsers[_newSuperAdmin] == false);
            users[_newSuperAdmin] = 3;
            return true;
        }

        // Approve User.
         function approveUsers(address _newUser) verifiedSuperAdmin public
returns (bool)  {
            require(users[_newUser] != 0);
            verifiedUsers[_newUser] = true;
            return true;
        }
}
```

**App.js :**

```
App = {
  web3Provider: null,
  contracts: {},

  init: function() {
    if (typeof web3 !== 'undefined') {
       web3 = new Web3(web3.currentProvider);
    } else {
       web3 = new Web3(new Web3.providers.HttpProvider("http://localhost:7
545"))
    }

    web3.eth.defaultAccount = web3.eth.accounts[0];

    return App.initContract();
  },

  initContract: function() {

    var abi =[
    {
        "constant": false,
        "inputs": [
```

```
            {
                "name": "_newAdmin",
                "type": "address"
            }
        ],
        "name": "addNewAdmin",
        "outputs": [
            {
                "name": "",
                "type": "bool"
            }
        ],
        "payable": false,
        "stateMutability": "nonpayable",
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [
            {
                "name": "_propId",
                "type": "uint256"
            }
        ],
        "name": "getPropertyDetails",
        "outputs": [
            {
                "name": "",
                "type": "uint8"
            },
            {
                "name": "",
                "type": "uint256"
            },
            {
                "name": "",
                "type": "address"
            }
        ],
        "payable": false,
        "stateMutability": "view",
        "type": "function"
    },
    {
        "constant": false,
        "inputs": [
            {
                "name": "_propId",
                "type": "uint256"
            }
        ],
        "name": "rejectProperty",
        "outputs": [
            {
                "name": "",
```

```
            "type": "bool"
        }
    ],
    "payable": false,
    "stateMutability": "nonpayable",
    "type": "function"
},
{
    "constant": false,
    "inputs": [
        {
            "name": "_newUser",
            "type": "address"
        }
    ],
    "name": "approveUsers",
    "outputs": [
        {
            "name": "",
            "type": "bool"
        }
    ],
    "payable": false,
    "stateMutability": "nonpayable",
    "type": "function"
},
{
    "constant": false,
    "inputs": [
        {
            "name": "_newSuperAdmin",
            "type": "address"
        }
    ],
    "name": "addNewSuperAdmin",
    "outputs": [
        {
            "name": "",
            "type": "bool"
        }
    ],
    "payable": false,
    "stateMutability": "nonpayable",
    "type": "function"
},
{
    "constant": true,
    "inputs": [],
    "name": "creatorAdmin",
    "outputs": [
        {
            "name": "",
            "type": "address"
        }
    ],
```

```
    "payable": false,
    "stateMutability": "view",
    "type": "function"
},
{
    "constant": false,
    "inputs": [
        {
            "name": "_propId",
            "type": "uint256"
        },
        {
            "name": "_value",
            "type": "uint256"
        },
        {
            "name": "_owner",
            "type": "address"
        }
    ],
    "name": "createProperty",
    "outputs": [
        {
            "name": "",
            "type": "bool"
        }
    ],
    "payable": false,
    "stateMutability": "nonpayable",
    "type": "function"
},
{
    "constant": true,
    "inputs": [
        {
            "name": "",
            "type": "address"
        }
    ],
    "name": "users",
    "outputs": [
        {
            "name": "",
            "type": "int256"
        }
    ],
    "payable": false,
    "stateMutability": "view",
    "type": "function"
},
{
    "constant": false,
    "inputs": [
        {
            "name": "_newUser",
```

```
            "type": "address"
        }
    ],
    "name": "addNewUser",
    "outputs": [
        {
            "name": "",
            "type": "bool"
        }
    ],
    "payable": false,
    "stateMutability": "nonpayable",
    "type": "function"
},
{
    "constant": true,
    "inputs": [
        {
            "name": "",
            "type": "address"
        }
    ],
    "name": "verifiedUsers",
    "outputs": [
        {
            "name": "",
            "type": "bool"
        }
    ],
    "payable": false,
    "stateMutability": "view",
    "type": "function"
},
{
    "constant": false,
    "inputs": [
        {
            "name": "_propId",
            "type": "uint256"
        }
    ],
    "name": "approveProperty",
    "outputs": [
        {
            "name": "",
            "type": "bool"
        }
    ],
    "payable": false,
    "stateMutability": "nonpayable",
    "type": "function"
},
{
    "constant": true,
    "inputs": [
```

```
            {
                "name": "",
                "type": "uint256"
            }
        ],
        "name": "properties",
        "outputs": [
            {
                "name": "status",
                "type": "uint8"
            },
            {
                "name": "value",
                "type": "uint256"
            },
            {
                "name": "currOwner",
                "type": "address"
            }
        ],
        "payable": false,
        "stateMutability": "view",
        "type": "function"
    },
    {
        "inputs": [],
        "payable": false,
        "stateMutability": "nonpayable",
        "type": "constructor"
    }
];

App.contracts.asset =  web3.eth.contract(abi).at('0x98C1933D362F34d9F40
eA47d190314681E7ef468');

    return App.bindEvents();

  },

bindEvents: function() {
    $(document).on('click', '#btn-PropAdd', App.createProperty);
    $(document).on('click', '#btn-PropApprove', App.approveProperty);
    $(document).on('click', '#btn-PropReject', App.rejectProperty);
    $(document).on('click', '#btn-PropReqChange', App.reqchangeOwnership);
    $(document).on('click', '#btn-
PropAcptChange', App.approveChangeOwnership);
    $(document).on('click', '#btn-PropValChange', App.changeValue);
    $(document).on('click', '#btn-PropSearch', App.getPropertyDetails);
    $(document).on('click', '#btn-PropAddUser', App.adduser);
    $(document).on('click', '#btn-PropApproveUser', App.approveUsers);
    },

  createProperty: function(event) {
    event.preventDefault();
```

```
    var PropId = $('#PropAdd #PropId').val();
    var PropVal = $('#PropAdd #PropVal').val();
    var PropOwner = $('#PropAdd #PropOwner').val();


    web3.eth.getAccounts(function(error, accounts) {
                                        if (error) {
                                       console.log(error);
                                        }

      App.contracts.asset.createProperty(PropId, PropVal, PropOwner, {gas: 10
00000, gasPrice: web3.toWei(20, 'gwei')}, function(error, result){
        if(!error)
            console.log(JSON.stringify(result));
        else
            console.error(error);
      });



    });

  },

  approveProperty: function(event) {
    event.preventDefault();
    var PropId = $('#PropSearchform #PropSearch').val();
    web3.eth.getAccounts(function(error, accounts) {
                        if (error) {
                    console.log(error);
                        }
                        App.contracts.asset.approveProperty(PropId, {gas: 1000000,
gasPrice: web3.toWei(20, 'gwei')}, function(error, result){
                          if(!error)
                              console.log(JSON.stringify(result));
                          else
                              console.error(error);
                        });});

  },
  rejectProperty: function(event) {
    event.preventDefault();
    var PropId = $('#PropSearchform #PropSearch').val();
    web3.eth.getAccounts(function(error, accounts) {
                        if (error) {
                    console.log(error);
                        }
                        App.contracts.asset.rejectProperty(PropId, {gas: 1000000, ga
sPrice: web3.toWei(20, 'gwei')}, function(error, result){
                          if(!error)
                              console.log(JSON.stringify(result));
                          else
                              console.error(error);
                        });});
```

```
  },
  reqchangeOwnership: function(event) {
    event.preventDefault();
    var PropId = $('#PropSearchform #PropSearch').val();
    var NewOwner = $('#PropReqChangeform #PropReqChange').val();
    web3.eth.getAccounts(function(error, accounts) {
                    if (error) {
                  console.log(error);
                    }
                    App.contracts.asset.changeOwnership(PropId, NewOwner, {g
as: 1000000, gasPrice: web3.toWei(20, 'gwei')}, function(error, result){
                      if(!error)
                          console.log(JSON.stringify(result));
                      else
                          console.error(error);
                    });});

  },
  approveChangeOwnership: function(event) {
    event.preventDefault();
    var PropId = $('#PropSearchform #PropSearch').val();
    web3.eth.getAccounts(function(error, accounts) {
                    if (error) {
                  console.log(error);
                    }
                    App.contracts.asset.approveChangeOwnership(PropId, {gas:
1000000, gasPrice: web3.toWei(20, 'gwei')}, function(error, result){
                      if(!error)
                          console.log(JSON.stringify(result));
                      else
                          console.error(error);
                    });});

  },
  changeValue: function(event) {
    event.preventDefault();
    var PropId = $('#PropSearchform #PropSearch').val();
    var NewVal = $('#PropValChangeform #PropValChange').val();
    web3.eth.getAccounts(function(error, accounts) {
                    if (error) {
                  console.log(error);
                    }
                    App.contracts.asset.changeValue(PropId, NewVal, {gas: 1000
000, gasPrice: web3.toWei(20, 'gwei')}, function(error, result){
                      if(!error)
                          console.log(JSON.stringify(result));
                      else
                          console.error(error);
                    });});

  },
  getPropertyDetails: function() {
  //  event.preventDefault();
    var PropId = $('#PropSearchform #PropSearch').val();
    web3.eth.getAccounts(function(error, accounts) {
```

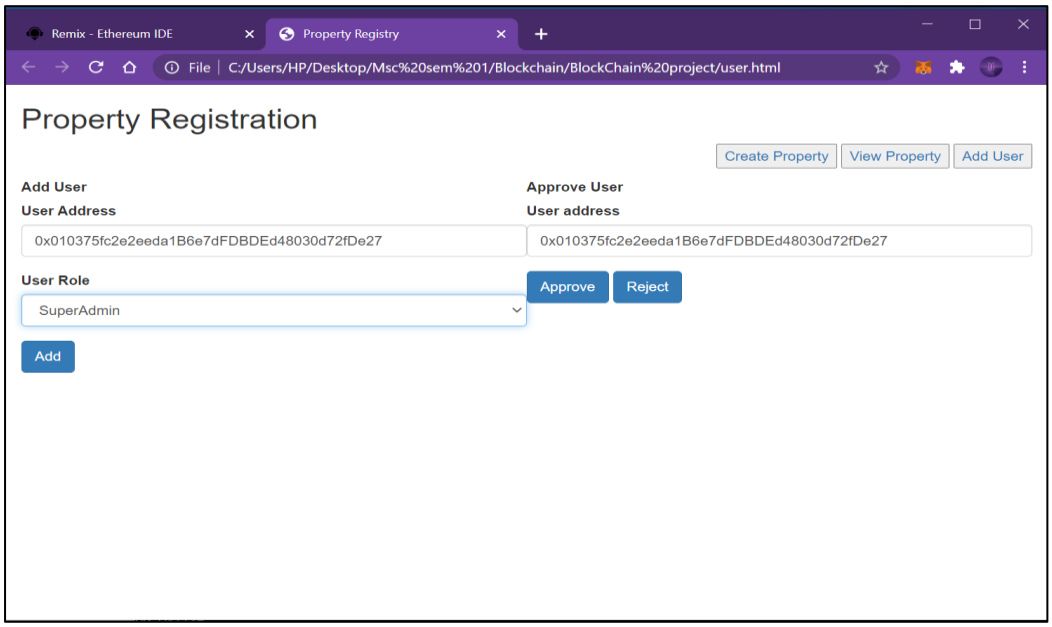
```
                if (error) {
                    console.log(error);
                }


                App.contracts.asset.getPropertyDetails(PropId, {gas: 100000
0, gasPrice: web3.toWei(20, 'gwei')}, function(error, result){
                if(!error)
                {       console.log(JSON.stringify(result));

                        var state = result[0];

                        $('#resultPropID').text(PropId);

                        if(state == 0)
                        {$('#resultPropStatus').text('NotExist');
                        $('#resultPropFunc1 #btn-
PropApprove').attr('disabled', true);
                        $('#resultPropFunc2 #btn-
PropReject').attr('disabled', true);
                        $('.sec1').html('');
                        $('.sec2').html('');
                        $('.sec3').html('');}
                        else if (state == 1)
                        {
                        $('#resultPropStatus').text('Pending');
                        $('#resultPropFunc1').html('<a href="#" id="btn-
PropApprove" class="btn btn-primary">Approve</a>');
                        $('#resultPropFunc2').html('<a href="#" id="btn-
PropReject" class="btn btn-primary">Reject</a>');
                        $('.sec1').html('');
                        $('.sec2').html('');
                        $('.sec3').html('');}
                        else if(state == 2)
                        {$('#resultPropStatus').text('Approved');

                        $('#resultPropFunc1 #btn-
PropApprove').attr('disabled', true);
                        $('#resultPropFunc2 #btn-
PropReject').attr('disabled', true);
                        $('.sec1').html('');
                        $('.sec2').html('');
                        $('.sec3').html('');
                        }
                        else if(state == 3)
                        {$('#resultPropStatus').text('Rejected');
                        $('#resultPropFunc1 #btn-
PropApprove').attr('disabled', true);
                        $('#resultPropFunc2 #btn-
PropReject').attr('disabled', true);
                        $('.sec1').html('');
                        $('.sec2').html('');
                        $('.sec3').html('');}

                        $('#resultPropValue').text(result[1]);
```

```
                    $('#resultPropOwner').text(result[2]);

                }
                else
                console.error(error);
  });  });
 },
 adduser: function(event) {
  event.preventDefault();
  var useraddress = $('#adduser #PropAddUser').val();
  web3.eth.getAccounts(function(error, accounts) {
                if (error) {
              console.log(error);
                }
                if($('#adduser #PropAddUserrole').val() == "User")
                {App.contracts.asset.addNewUser(useraddress, {gas: 100000
0, gasPrice: web3.toWei(20, 'gwei')}, function(error, result){
                  if(!error)
                     console.log(result);
                  else
                     console.error(error);
                });
                }
                else if ($('#adduser #PropAddUserrole').val() == "Admin")
                {App.contracts.asset.addNewAdmin(useraddress, {gas: 10000
00, gasPrice: web3.toWei(20, 'gwei')}, function(error, result){
                  if(!error)
                     console.log(JSON.stringify(result));
                  else
                     console.error(error);
                });
                }
                else if ($('#adduser #PropAddUserrole').val() == "SuperAdmin"
)
                {App.contracts.asset.addNewSuperAdmin(useraddress, {gas:
1000000, gasPrice: web3.toWei(20, 'gwei')}, function(error, result){
                  if(!error)
                     console.log(JSON.stringify(result));
                  else
                     console.error(error);
                });
                }


              });

 },
 approveUsers: function(event) {
  event.preventDefault();
  var useraddress = $('#adduser #PropAddUser').val();
  web3.eth.getAccounts(function(error, accounts) {
                if (error) {
              console.log(error);
                }
```

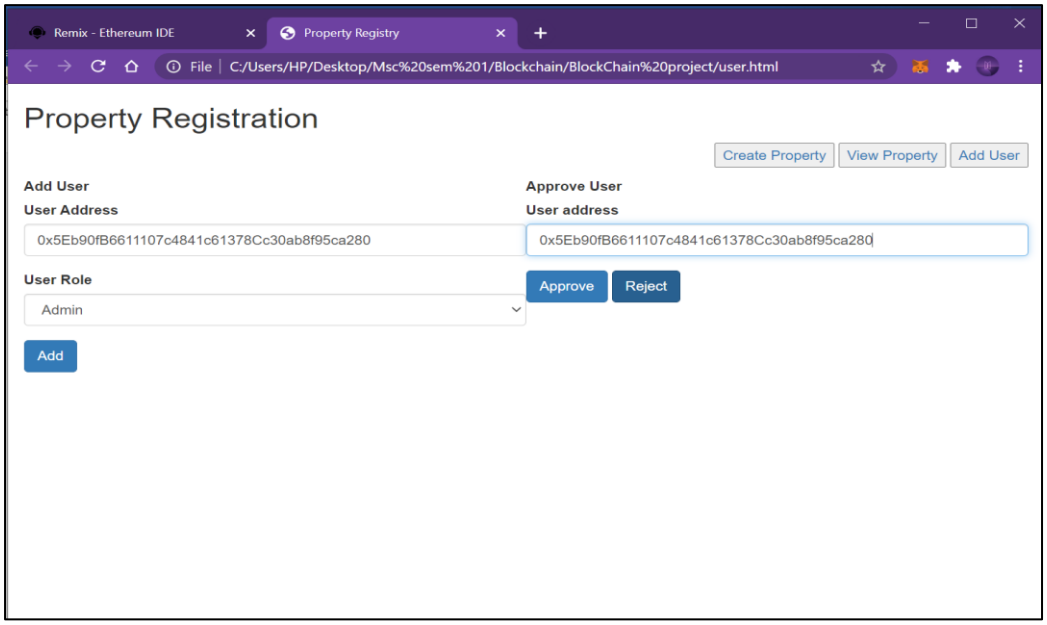
```
                    App.contracts.asset.approveUsers(useraddress, {gas: 100000
0, gasPrice: web3.toWei(20, 'gwei')}, function(error, result){
                     if(!error)
                        console.log(JSON.stringify(result));
                     else
                        console.error(error);
                    });});

  }
};
$(function() {
  $(window).load(function() {

    App.init();
  });
});
```

# Results:

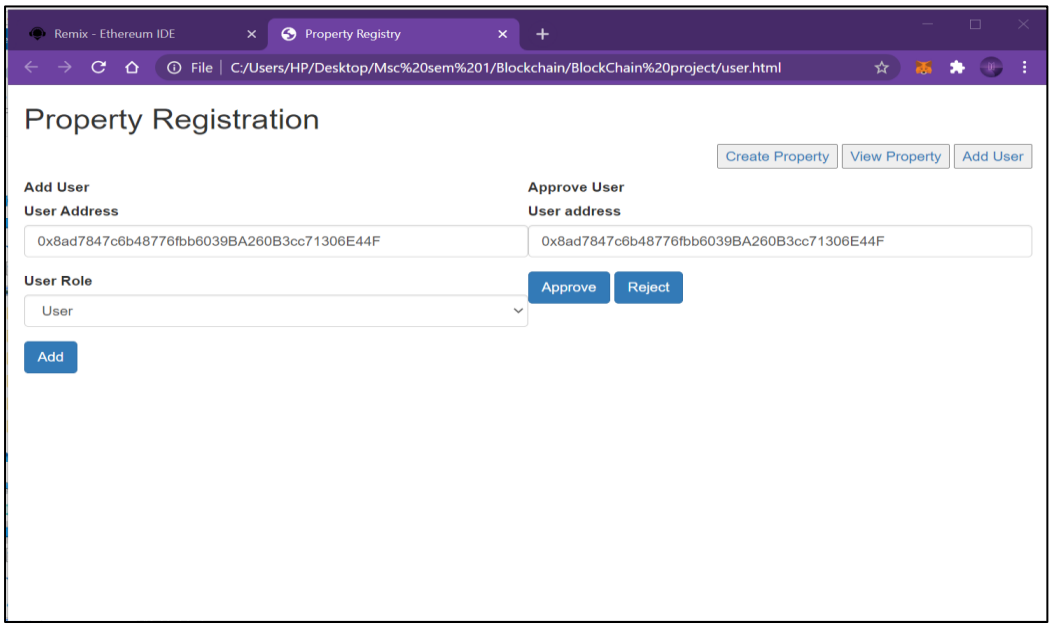
**Adding SuperAdmin :**



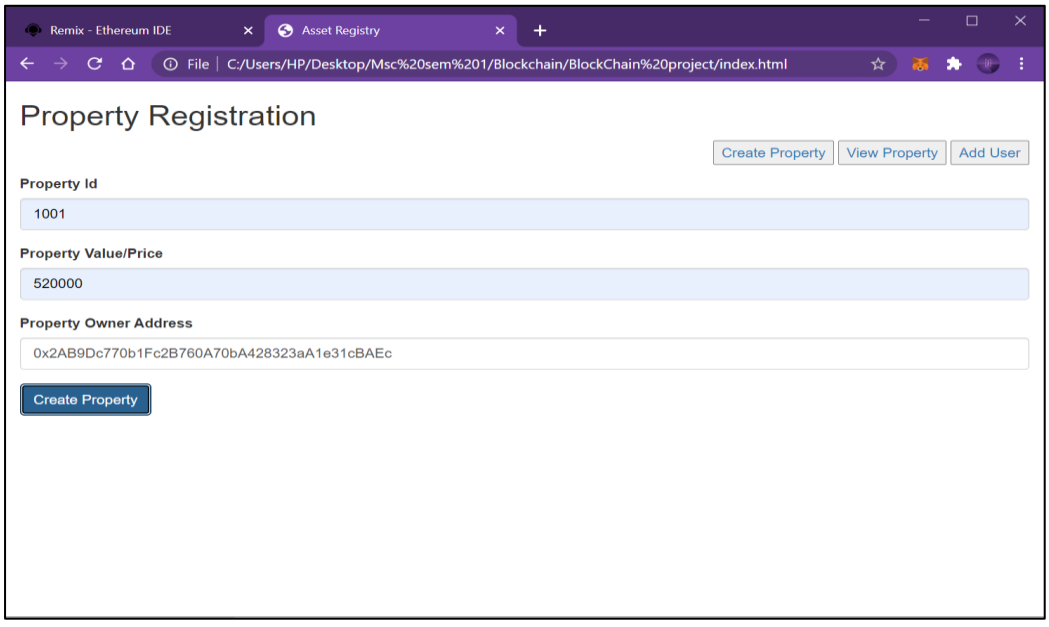
## Adding Admin :



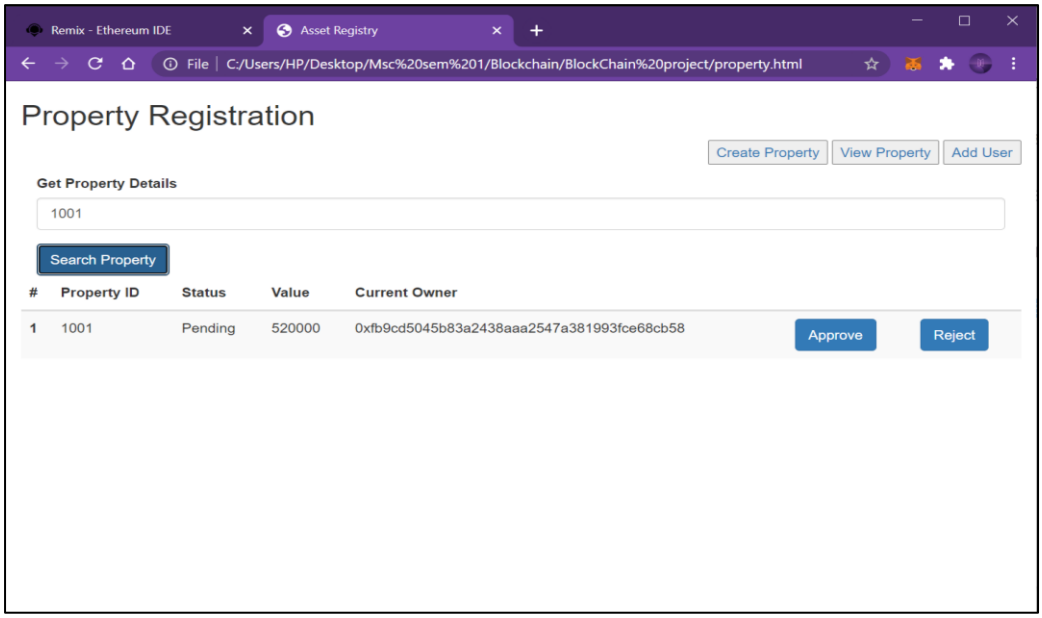

## Adding user :



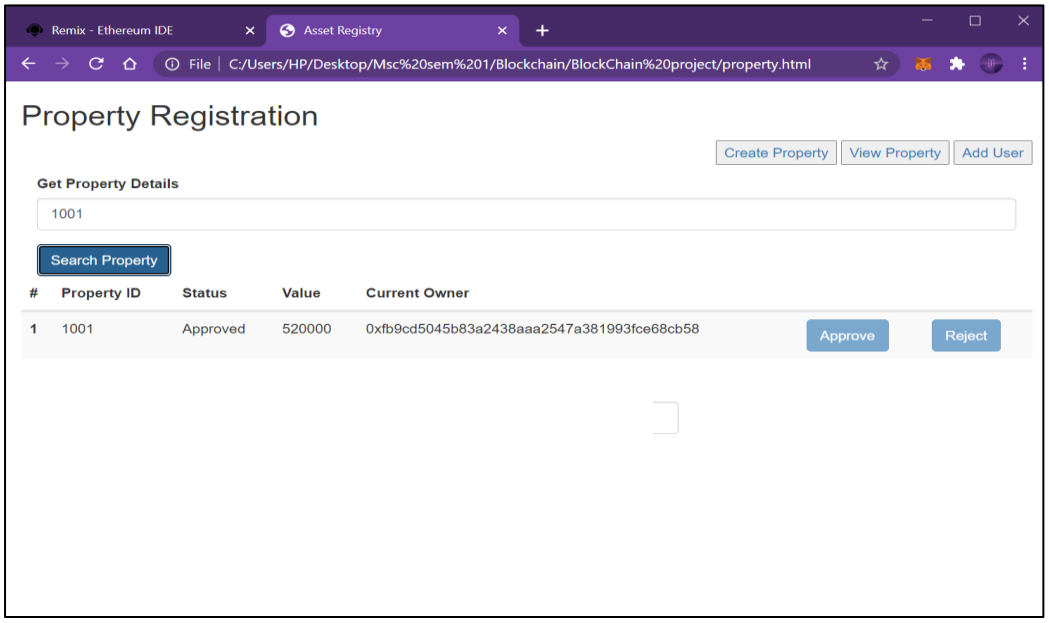

## Creating Property:

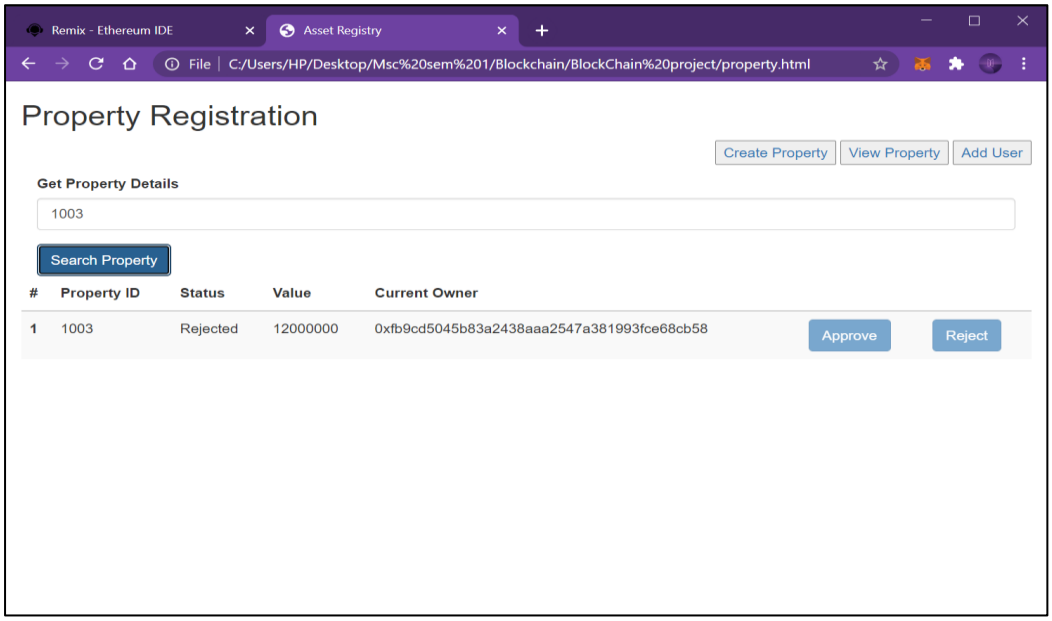


## Land Rgistration request :

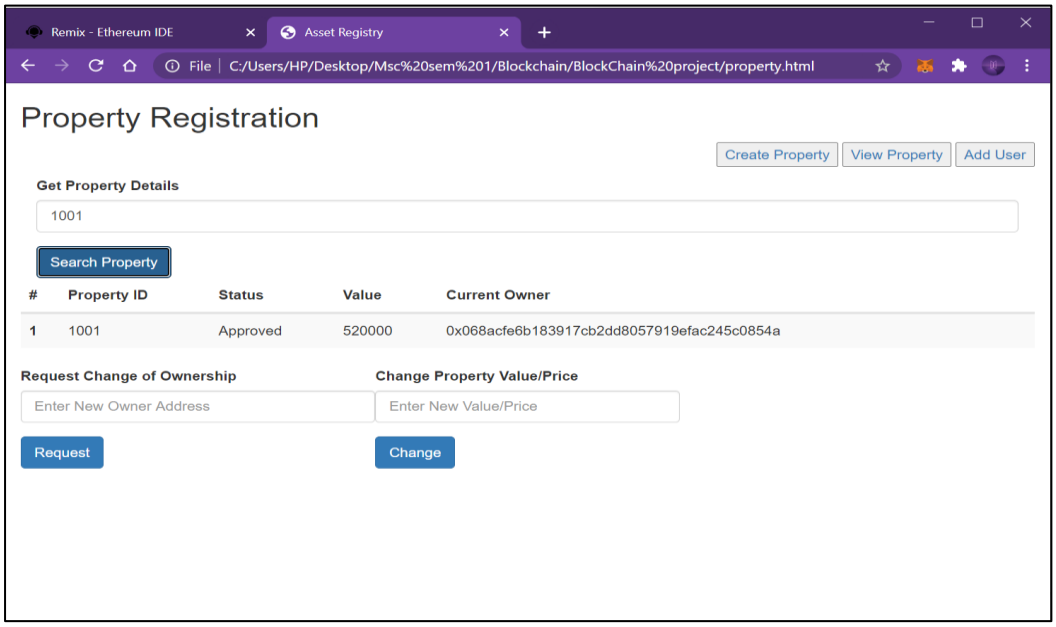


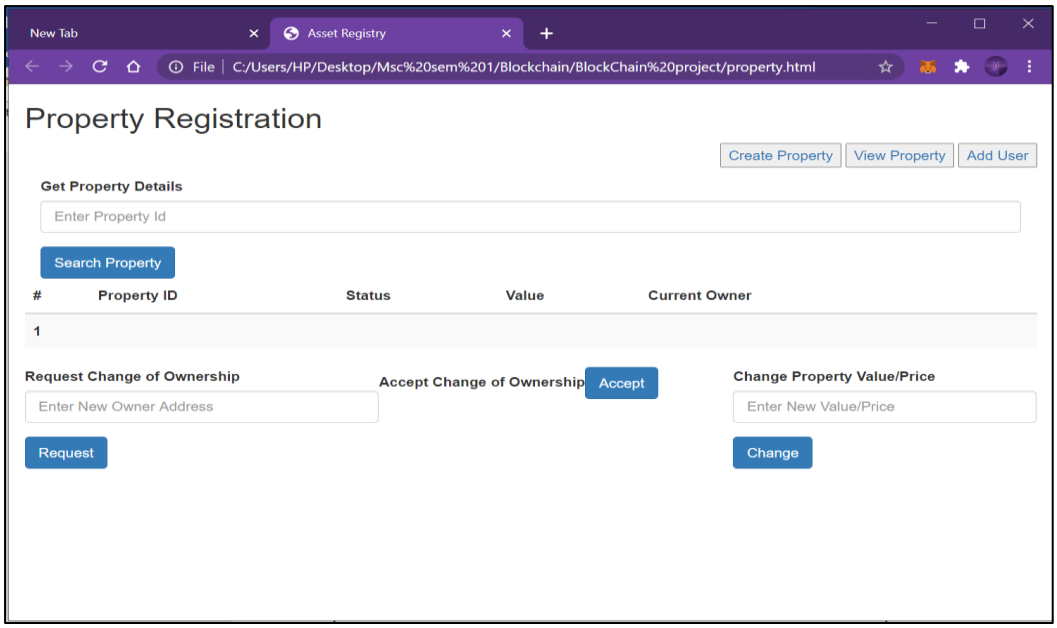

## Land Rgistration status:

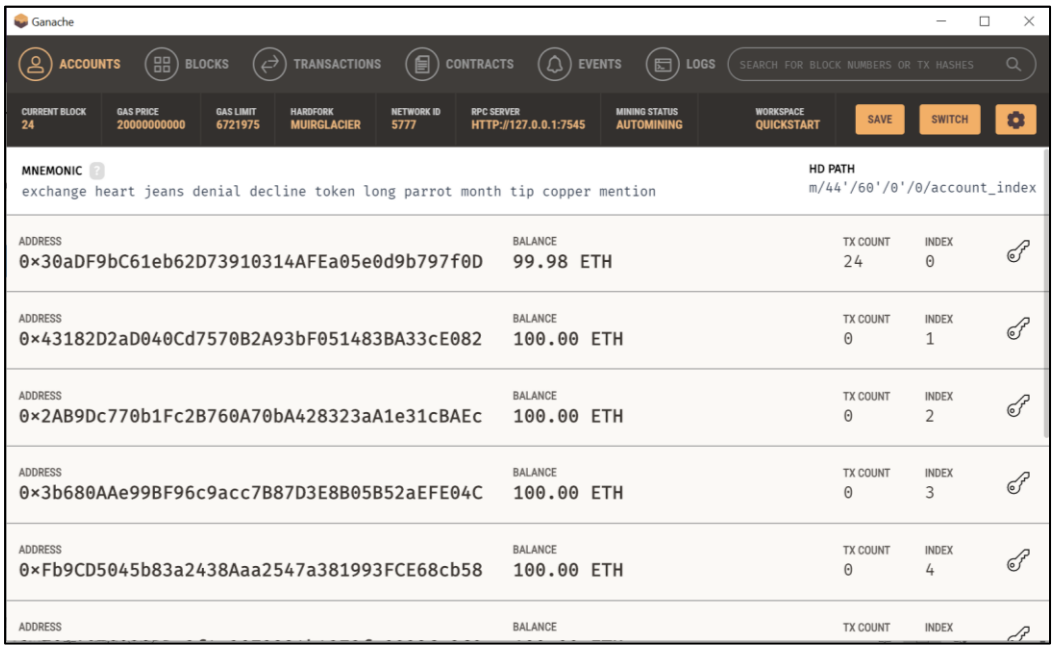


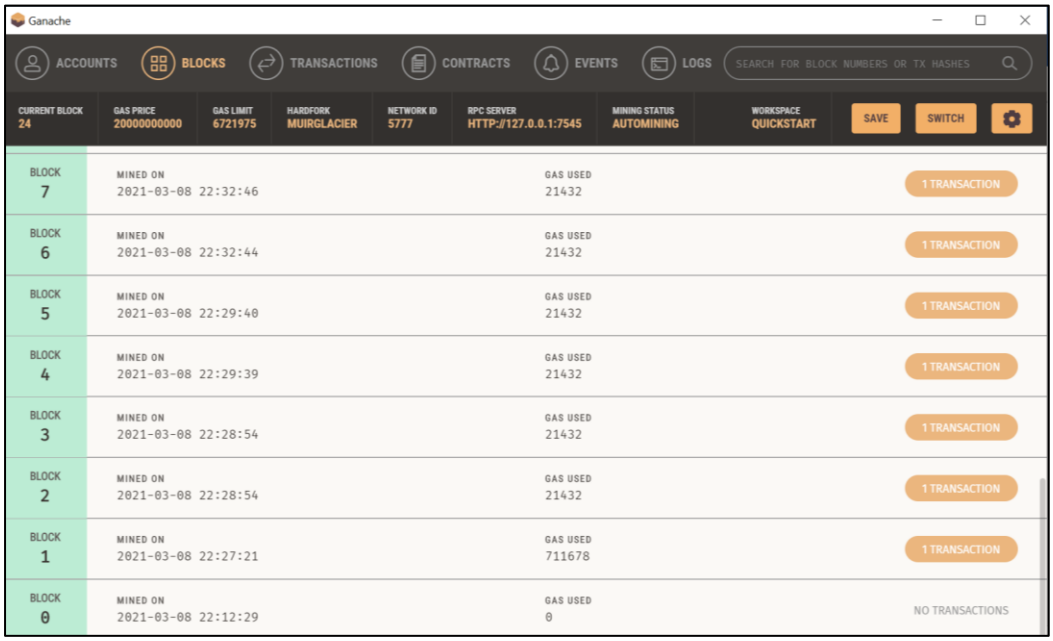
## Change property owner:



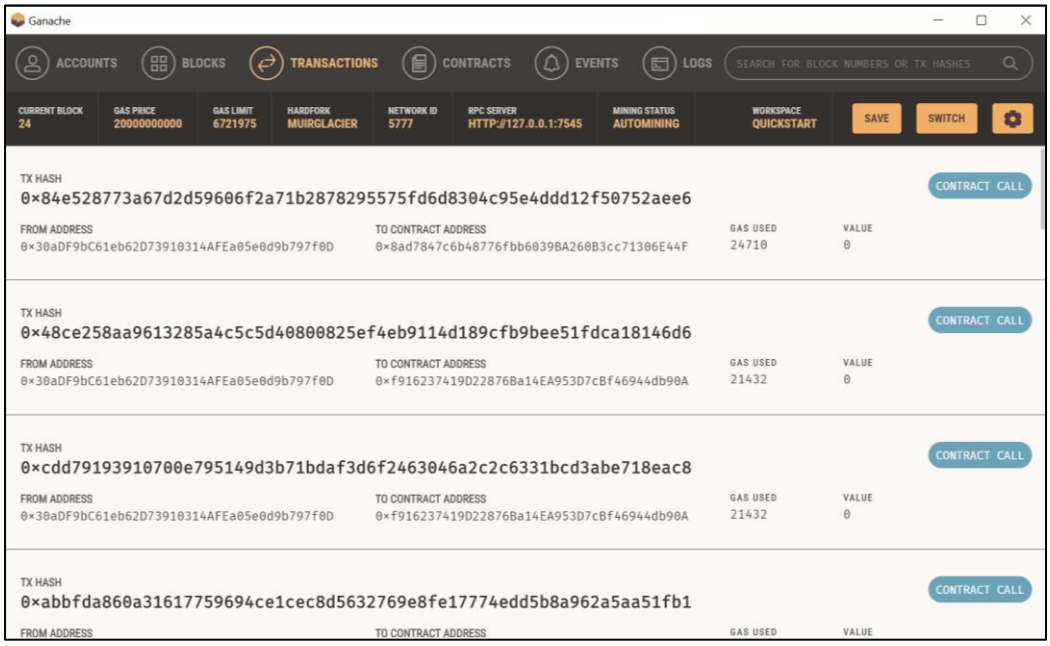



**Blocks :**




**Transcations:**




# Conclusion:

Land registration systems are crucial to protect an individual's fundamental right to property, a right enshrined in national constitutions and international conventions. On the other hand, there exist several limitations with land registration systems currently implemented by countries which affect land title certainty. Advancement in technology may aid with providing a more tamper-proof and secure registry which would ultimately lead to further protection over an individual's property rights.

Blockchain technology has many characteristics which may provide for further certainty over the contents which are to be stored in the Land Registration. Having a complete record is a must, as otherwise doubts and uncertainties will always be present. Thus, it is of utmost importance that a country's legislation

requires that anything affecting property rights be recorded within the registry (regardless of the system used). This includes *inter alia* the recordation of property transfers, lease agreements, overriding interests, usufruct rights, etc. It is only after this is ensured that a blockchain-based land registration system would be able to provide us, with certainty, a true picture of what rights are held by a person (legal or natural) over a specific property.

# Bibliography:


> bitcoin.pdf (bitcoin.org)
> NPTEL: Blockchain in Government -V (Tax Payments and Land Registry Records) (youtube.com)
> Ethereum Blockchain on webpage(medium.com)
> Smart Contract in Ethereum with Ganache and Remix IDE (c-sharpcorner.com
> Web3.js Tutorial (youtube.com)
> Introduction to Smart Contracts (soliditylang.org)