A Project Report on

Real Estate On Blockchain

Submitted in partial fulfillment of the requirements for the award of the degree of

Bachelor of Engineering

in

Computer Engineering

by

Siddhant Bhadsavale(16102038)
Himanshu Malhotra(17202006)

Under the Guidance of

Sachin Malave
Amol Kalugade

Department of Computer Engineering
A.P. Shah Institute of Technology
G.B.Road,Kasarvadavli, Thane(W), Mumbai-400615
UNIVERSITY OF MUMBAI

Academic Year 2017-2018

Approval Sheet

This Project Report entitled ***"Project Title"*** Submitted by ***"Siddhant Bhadsavale"(1610203***
Malhotra"(1720200),'is approved for the partial fulfillment of the requirenment for the award of the degree of ***Bachelor of Engineering*** in from ***University of Mumbai***.

Amol Kalugade
(Co-Guide)

Sachin Malave
(Guide)

Prof. Kiran Deshpande
Head Department of Information Technology

Place:A.P.Shah Institute of Technology, Thane
Date:

CERTIFICATE

This is to certify that the project entitled ***"Real Estate On Blockchain"*** submitted by ***"Siddhant Bhadsavale" (16102038),"Himanshu Malhotra" (17202006)*** for the partial fulfillment of the requirement for award of a degree ***Bachelor of Engineering*** in ***Computer Engineering***,to the University of Mumbai,is a bonafide work carried out during academic year 2017-2018.

Amol Kalugade
(Co-Guide)

Sachin Malave
(Guide and Head Of Dept.)

Dr. Uttam D.Kolekar
(Principal)

External Examiner(s)
1.

2.

Place:A.P.Shah Institute of Technology, Thane
Date:

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, We have adequately cited and referenced the original sources. We also declare that We have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

_____

(Signature)

_____

Siddhant Bhadsavale(16102038)
Himanshu Malhotra(17202006))

Date:

Abstract

Blockchain technologies are gaining massive momentum in the last few years. Blockchains are distributed ledgers that enable parties to trust each other to maintain a set of global states. The parties agree on the existence, values, and histories of the states. One of the most important asset in our life along with the intellectual assets is our house and we spend lot of money and time in making its proper documents. The problem in the traditional system is that it cannot be completely trusted. By creating fake documents title frauds are carried out fooling the buyers. Sometimes the same unit is sold to multiple buyers and the owners run away with the money. Using the blockchain technology we can overcome these problems as each seller and the buyer will be unique and the transactions which happen between the parties will be on the distributed ledger. By creating efficient smart contracts we can decide how the input and output state will affect the data on blockchain. In this way we can eliminate the intermediaries in the real estate industry and make the process trustworthy and transparent when it comes to the ownership.5 lines.

Contents

List of Figures

List of Abbreviations

ETH : Symbol Of Ether(Cryptocurrency)
Wei : Smallest Unit Of Ether
Gas : Minimum Transaction Fee

Chapter 1

Introduction

The blockchain is a distributed database of records of all transactions or digital event that have been executed and shared among participating parties. Each transaction verified by the majority of participants of the system. It contains every single record of each transaction. Bitcoin is the most popular cryptocurrency an example of the blockchain. Blockchain Technology first came to light when a person or Group of individuals name 'Satoshi Nakamoto' published a white paper on "Bitcoin: A peer to peer electronic cash system" in 2008. Blockchain Technology Records Transaction in Digital Ledger which is distributed over the Network thus making it incorruptible. Anything of value like Land Assets, Cars, etc. can be recorded on Blockchain as a Transaction. There is no Central Server or System which keeps the data of Blockchain. The data is distributed over Millions of Computers around the world which are connected with the Blockchain. This system allows Notarization of Data as it is present on every Node and is publicly verifiable.

Blockchain enhances trust across a business network. It's not that you can't trust those who you conduct business with it's that you don't need to when operating on a Blockchain network. The distributed ledger is shared and updated with every incoming transaction among the nodes connected to the Blockchain. All this is done in real-time as there is no central server controlling the data. There is no unauthorized access to Blockchain made possible through Permissions and Cryptography.

Because every node or participant in Blockchain has a copy of the Blockchain data, they have access to all transaction data. They themselves can verify the identities without the need for mediators. All relevant network participants must agree that a transaction is valid. This is achieved through the use of consensus algorithms. Smart Contracts which are executed based on certain conditions can be written into the platform. Blockchain Network can evolve in pace with business processes.

Chapter 2

Literature Review

Researchers claim that the blockchain technology is how people in the future, will be keeping records and histories of transactions and events and which would be very beneficial if it was applied to the property sector the benefits do not outweigh the costs for implementing such a technology.One of the easiest explaination is that we could define blockchain as a trusted network for keeping and managing the records The business logic of the blockchain is written in the smart contract.The contract is stored on the ledger on the Blockchain. So, whenever a transaction happens, a function is invoked that calls the smart contract and the processing is done. Smart Contracts are stored on the Blockchain because it is important for the contract to be available to the people making transactions. There is an IEEE paper on smart contracts by Shuai Wang,Liwei Ouyang,Yong Yuan,Xiaochun Ni,Xuan Han,Fei-Yue Wang.They have proposed the idea of the application of smart contracts in Finance,Management,Real estate even the Internet of things. The paper is entitled as Blockchain-Enabled Smart Contracts: Architecture, Applications, and Future Trends. According to this paper smart contracts can be used for smart contracts driven parallel organizational management or we can say self managing organizational man- agement. We can use this self managing property of the smart contracts can be used in real estate as it is a very senstive field when it comes to the authentication of the registrations and ownership.Nathan Shedroff have proposed the idea of self managing real esate. In this paper author has proposed that the blockchain technology can be used in real estate for keeping the ownership record with the help of smart contracts.Properly written smart contracts can keep the ownership records safe.This IEEE paper itself is entitled as Self- Managing Real Estate This project is regarding the real estate on blockchain. The land records 3are very much important asset to each and everyone as it costs a lot and it has a lot of legal procedures involved. At the end we all want these records to be safe and not to be tampered with. The traditional systems contains a database which could be manipulated or tampered with. If the data is tampered it can create a lot of problems at any extent. Using Blockchain we can ensure that the land records won't be tampered with. This project mainly focuses on storing the land records on Blockchain and thus everyone can view them on the distributed ledger. The proposed system will view land records and the transfer of ownership will be tracked easily as every transaction taking place is reflected on the ledger. .

Chapter 3

Technology Stack

3.1 Metamask

MetaMask is a bridge that allows you to visit the distributed web of tomorrow in your browser today. It allows you to run Ethereum Decentralised Applications right in your browser without running a full Ethereum node. MetaMask is a self-hosted wallet to store, send and receive ETH and ERC20. It allows you to control your funds as it is an HD wallet that provides a mnemonic phrase that you can keep as a backup. MetaMask is indeed also known as "Hot Wallet".

3.2 Ethers.js(Library)

The ethers.js library aims to be a complete and compact library for interacting with the Ethereum Blockchain and its ecosystem. It was originally designed for use with ethers.io and has since expanded into a much more general-purpose library. Ethers. js is a lightweight JavaScript library which can be used as an alternative to Web3. js to build your JavaScript frontend and interact with the Ethereum blockchain.

3.3 Node.js

As an asynchronous event driven JavaScript runtime, Node.js is designed to build scalable network applications.The node.js is required for npm which is used to run and manage the packages.

3.4 React.js

React. js is an open-source JavaScript library that is used for building user interfaces specifically for single-page applications. It's used for handling the view layer for web and mobile apps. React also allows us to create reusable UI components. React helps manage those changing states and dynamically present different views to the user based on state information. Reacts virtual browser acts like an agent between the developer and the real browser.

3.5 Solidity

Solidity is a high-level and contract-oriented language used for writing smart contracts. It is used for designing and implementing smart contracts. It was influenced by C++, Python and JavaScript and is designed to target the Ethereum Virtual Machine (EVM).

3.6 Chai-Mocha Unit Testing

Mocha is a feature-rich JavaScript test framework running on Node.js and in the browser. Chai is basically the library and mocha is the framework on which we conduct our tests. Add your chapter Name

Chapter 4

Benefits

4.1 Benefits For Society

Transaction history is becoming more open with blockchain Engineering. Since blockchain is a distributed ledger form, all of the network Participants share identical documents as opposed to individual copies. Accordingly, Blockchain data is more accurate, coherent and transparent than when it is Pushed into processes heavy on paper.

Any industry where sensitive protection Data is critical — Blockchain has financial services, government , health care etc. Attempt to further improve the way vital knowledge is exchanged Preventing fraud and criminal activity. Reducing costs is a thing for most businesses

Blockchain means you don't need as many third parties or intermediaries Make guarantees, because if you can trust your trading partner, it does not matter. You just have to trust the blockchain data, instead. You won't have to, either. Review so many documents to complete a trade, because everyone will have permissioned access

Chapter 5

Project Design

5.1 Proposed System

This project focuses on making the transactions and ownership transfer trans-parent. With the help of blockchain technology there will be a unique entity for a person so there will be no duplicity. The information will be visible to all thus eliminating intermediaries. This process will reduce the time required for ownership transfer as compared to the legal paper procedures.

5.2 Flow Of Modules

5.2.1 Description

The Decentralised Application (DAPP) is created to make land registration and ownership transaction transparent and decentralised. React.js is used to implement the front end and solidity smart contract is used as backend. React is a JavaScript library for building user interfaces. Its main feature is that we can build the application in a modular fashion which helps in code repetition and makes it easier to debug. The solidity contract in the backend makes the DAPP decentralised and transparent.

5.2.2 Flow Diagram

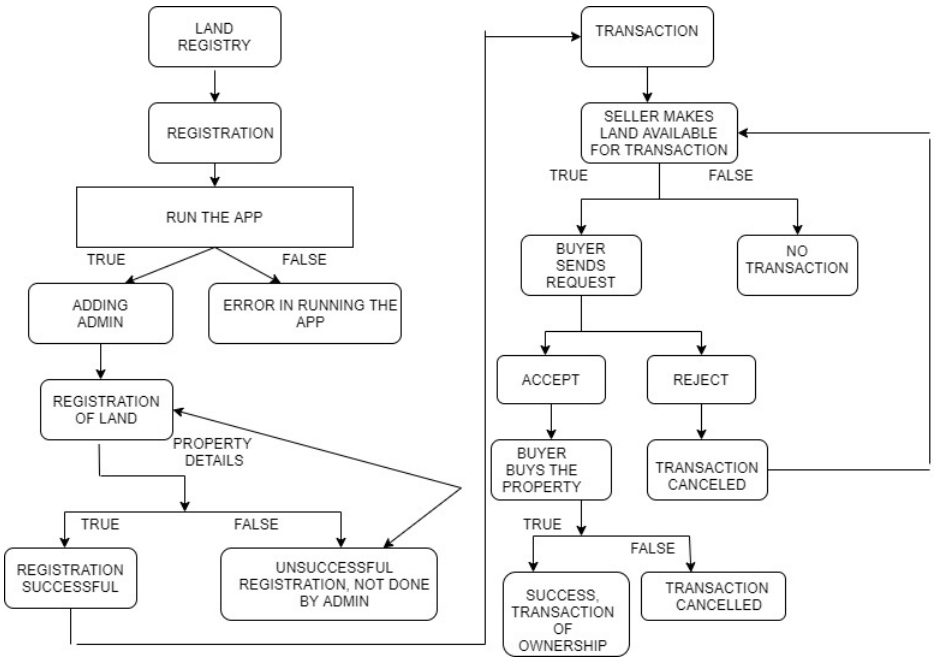


Figure 5.1: Flow Of Modules

Chapter 6

Modules Of System

6.1 Registration

The user provides the land details to the government authorities who is registered as admin in this case. The land that is being registered should be in the same area as that of the admin who is going to register the land. The admin verifies the details with the existing records and enters into the DAPP. The details that are enrolled into the DAPP are state, district, village, survey number, owner address and market value. Along with this, an ID is generated when the first four details of the land are passed. This ID is generated in the function "computeId ()" using SHA256. The values entered in the registration form are passed in to the function "Registration ()" and the details are mapped using the automatically generated ID. The mapping of the land details makes the searching process of the land simple.

6.2 Transaction / Transfer of ownership

The transaction of the property has several stages involved. The algorithm is designed in such a way that there is no need for any central authority to verify the transaction process. It is important to note that the owner of a property can sell the land as a whole, that is there is no partial transaction of the property. This is just to simplify the problem. Later on, more functionalities can be added. The following are the steps involved:

6.2.1 Make the land available:

Once the buyer and seller agree to make the ownership transaction, the seller should make the land available for transaction. The land owner passes the property ID to the function "makeAvailable ()" and the function verifies the account of owner and changes the value of "isAvailable" to true which implies that the land is open to buy.

6.2.2 Sending request to land owner

When the land is available to buy, the buyer sends a request to the land owner to buy the property. The ID of the land is stored into the function "requestToLandOwner ()". The function verifies whether the land is available to buy by checking the value of "isAvailable". If the value is true, the buyer's address is stored inside "requester" which was initially 0

address.The value of “isAvailable” is then set to false so that no more request can be sent and request status is changed from “default” to “pending”.

6.2.3 View the request

The function “viewRequest” takes the property ID as the input and returns the address of the requester. The function is for checking the address of the buyer.

6.2.4 Process the request

Once the seller views the requester address and if it is the right one, then the seller can process the request by entering property ID, request status of the function “processRequest ()”. The function, as usual, verifies whether the input is done by the owner of the land and process the request. If the requester address is not of the original buyer then the seller can reject the request.

6.2.5 Buy the property

Once the request is approved the buyer can buy the property. The buyer enters the land ID to the function “buy Property”. The function check whether the request status is approved or not and if it is approved, then it checks if the amount given is greater than the sum of market value. If the conditions are satisfied then the amount is transferred to the land owner’s account. The functions then change the ownership of the land to the buyer.

Chapter 7

Project Implementation

7.1 Code For Smart Contract

The following is the smart contract of this project. In this project the smart contract acts as the business logic which is deployed on the blockchain written in Solidity.

```
pragma solidity >=0.4.0 <0.6.0;

//Land Details
contract landRegistration{
    struct landDetails{
        string state;
        string district;
        string village;
        uint surveyNumber;
        address payable CurrentOwner;
        uint marketValue;
        bool isAvailable;
        address requester;
        reqStatus requestStatus;

    }


//request status
enum reqStatus {Default,pending,reject,approved}



//profile of a client
struct profiles{
 uint[] assetList;
}
```

```
mapping(uint => landDetails) land;
address owner;
mapping(string => address) superAdmin;
mapping(address => profiles) profile;

//contract owner
constructor() public{
 owner = msg.sender;
}
modifier onlyOwner {
 require(msg.sender == owner);
 _;
}

//adding village admins
function addSuperAdmin(address _superAdmin,
string memory _village ) onlyOwner public
{
 superAdmin[_village]=_superAdmin;
 }
//Registration of land details.
function Registration(string memory _state,string memory _district,
string memory _village,uint _surveyNumber,
address payable _OwnerAddress,uint _marketValue,uint id
) public returns(bool){
require(superAdmin[_village] == msg.sender || owner == msg.sender);
land[id].state = _state;
land[id].district = _district;
land[id].village = _village;
land[id].surveyNumber = _surveyNumber;
        land[id].CurrentOwner = _OwnerAddress;
        land[id].marketValue = _marketValue;
        profile[_OwnerAddress].assetList.push(id);
        return true;
    }
    //to view details of land for the owner
    function landInfoOwner(uint id) public view returns(string memory,string
    memory,string memory,uint256,bool,address,reqStatus){
        return(land[id].state,land[id].district,land[id].village,land[id].
   surveyNumber,land[id].isAvailable,land[id].requester,land[id].
   requestStatus);
    }
        //to view details of land for the buyer
        function landInfoUser(uint id) public view returns(address,uint,bool
   ,address,reqStatus){
        return(land[id].CurrentOwner,land[id].marketValue,land[id].
```

```
isAvailable,land[id].requester,land[id].requestStatus);
 }

 // to compute id for a land.
 function computeId(string memory _state,string memory _district,string
memory _village,uint _surveyNumber) public view returns(uint){
     return uint(keccak256(abi.encodePacked(_state,_district,_village,
_surveyNumber)))%10000000000000;
 }

 //push a request to the land owner
 function requstToLandOwner(uint id) public {
     require(land[id].isAvailable);
     land[id].requester=msg.sender;
     land[id].isAvailable=false;
     land[id].requestStatus = reqStatus.pending; //changes the status to
pending.
 }
 //will show assets of the function caller
 function viewAssets()public view returns(uint[] memory){
     return (profile[msg.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(0);
         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 buyProperty(uint property)public payable{
     require(land[property].requestStatus == reqStatus.approved);
     // require(msg.value >= (land[property].marketValue+((land[property
].marketValue)/10)));
      require(msg.value >= land[property].marketValue);
```

```
        land[property].CurrentOwner.transfer(land[property].marketValue);
        removeOwnership(land[property].CurrentOwner,property);
        land[property].CurrentOwner=msg.sender;
        land[property].isAvailable=false;
        land[property].requester = address(0);
        land[property].requestStatus = reqStatus.Default;
        profile[msg.sender].assetList.push(property); //adds the property to
    the asset list of the new owner.

    }
    //removing the ownership of seller for the land. and it is called by the
    buyProperty function
    function removeOwnership(address previousOwner,uint id)private{
        uint index = findId(id,previousOwner);
        profile[previousOwner].assetList[index]=profile[previousOwner].
   assetList[profile[previousOwner].assetList.length-1];
        delete profile[previousOwner].assetList[profile[previousOwner].
   assetList.length-1];
        profile[previousOwner].assetList.length--;
    }
    //for finding the index of a perticular id
    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;
    }
}
```

7.2 Tests for smart contract

There are various tools for testing the smart contract. The smart contracts could also be tested on the remix IDE after deploying on the testnetwork.

Chai and Mocha are quite popular for javascript testing. Chai is the assertion library where as Mocha is the testing framework of Javascript for testing the smart contract functions

```
/// @dev importing packages required
const assert = require('assert');
const ethers = require('ethers');
const ganache = require('ganache-cli');
const { parseTx } = require('../helpers');
```

```
/// @dev when you make this true, the parseTx helper will output transaction
    gas consumption and logs
const DEBUG_MODE = false;

/// @dev initialising development blockchain
const provider = new ethers.providers.Web3Provider(ganache.provider({
   gasLimit: 80000000 }));

/// @dev importing build file
const simpleStorageJSON = require('../build/SimpleStorage_landRegistration.
   json');

/// @dev initialize global variables
let accounts, simpleStorageInstance;
let mValue;
/// @dev this is a test case collection
describe('Ganache␣Setup', async() => {

  /// @dev this is a test case. You first fetch the present state, and
   compare it with an expectation. If it satisfies the expectation, then
   test case passes else an error is thrown.
  it('initiates␣ganache␣and␣generates␣a␣bunch␣of␣demo␣accounts..Like␣a␣local
   ␣blockchain', async() => {

    /// @dev for example in this test case we are fetching accounts array.
    accounts = await provider.listAccounts();

    /// @dev then we have our expection that accounts array should be at
   least having 1 accounts
    assert.ok(accounts.length >= 1, 'atleast␣2␣accounts␣should␣be␣present␣in
   ␣the␣array');
  });
});

/// @dev this is another test case collection
describe('Land␣Registration␣Contract', () => {

  /// @dev describe under another describe is a sub test case collection
  describe('Land␣Registration␣Setup', async() => {

    /// @dev this is first test case of this collection
    it('deploys␣Simple␣Storage␣contract␣from␣first␣account␣with␣initial␣
   storage:␣landRegistration', async() => {

      /// @dev you create a contract factory for deploying contract. Refer
   to ethers.js documentation at https://docs.ethers.io/ethers.js/html/
      const SimpleStorageContractFactory = new ethers.ContractFactory(
```

```
      simpleStorageJSON.abi,
      simpleStorageJSON.evm.bytecode.object,
      provider.getSigner(accounts[0])
    );
    simpleStorageInstance =  await SimpleStorageContractFactory.deploy();

    assert.ok(simpleStorageInstance.address, 'conract␣address␣should␣be␣
  present');
  });

  /// @dev this is second test case of this collection
  // it('value should be set properly while deploying', async() => {

  //   /// @dev you access the value at storage with ethers.js library of
  our custom contract method called getValue defined in contracts/
  SimpleStorage.sol
  //   const currentValue = await simpleStorageInstance.functions.getValue
  ();

  //   /// @dev then you compare it with your expectation value
  //   assert.equal(
  //     currentValue,
  //     'hello world',
  //     'value set while deploying must be visible when get'
  //   );
  // });
});

describe('Land␣Registration␣Functionality', async  => {

  /// @dev this is first test case of this collection
  it('should␣compute␣the␣Id', async() => {

    /// @dev now get the value at storage
    const currentValue = await simpleStorageInstance.functions.computeId('
  ','','',1);

    /// @dev then comparing with expectation value
    assert.equal(
      currentValue,
      9459944778998
    );
  });
  it("Registering␣the␣land␣by␣accounts[0]", async() => {
    let id = await simpleStorageInstance.functions.computeId('maharashtra'
  ,'thane','kalyan',21);
    let register = await parseTx(simpleStorageInstance.functions.
```

```
  Registration('maharashtra','thane','kalyan',21,accounts[1],20,id),
  DEBUG_MODE)
     assert(register);

       //     const receipt = await parseTx(instance.functions.setValue('hi
  '), DEBUG_MODE);

   });

});
describe('Checking␣successful␣LandTransaction',async()=>{
   it("checking␣successful␣registration",async()=>{
     let value = 20;

     let marketValue = ethers.utils.parseEther(value.toString());
     mValue = marketValue;
     let id = await simpleStorageInstance.functions.computeId('maharashtra'
  ,'thane','kalyan',21);
      await parseTx(simpleStorageInstance.functions.Registration('
  maharashtra','thane','kalyan',21,accounts[1],marketValue,id), DEBUG_MODE
  )
     let landInfo = await simpleStorageInstance.functions.landInfoUser(id);
     assert.equal(landInfo[0],accounts[1]);
   });
   it("checking␣the␣availability␣before␣making␣available,␣by␣default␣it␣is␣
  unavailable",async()=>{
     let id = await simpleStorageInstance.functions.computeId('maharashtra'
  ,'thane','kalyan',21);
     let landInfo = await simpleStorageInstance.landInfoUser(id);
     assert.equal(landInfo[2],false);

   })
   it("checking␣that␣the␣buyer␣can␣only␣make␣request␣if␣the␣property␣is␣
  available␣for␣sale", async() => {
     let id = await simpleStorageInstance.functions.computeId('maharashtra'
  ,'thane','kalyan',21);
     let state;
     try {
       let instance1 = simpleStorageInstance.connect(provider.getSigner(
  accounts[1]))

       await instance1.functions.requstToLandOwner(id);
       state =false;

     } catch (error) {
       state = true;
```

```
    }
    assert(state);
  })
  it("checking␣the␣availability␣for␣buying␣a␣land␣after␣make␣it␣avialable"
, async() => {
    let id = await simpleStorageInstance.functions.computeId('maharashtra'
,'thane','kalyan',21);
   let instance1 = simpleStorageInstance.connect(provider.getSigner(
accounts[1]))
   let instance2 = simpleStorageInstance.connect(provider.getSigner(
accounts[2]))
   await instance1.functions.makeAvailable(id);
    let landInfo = await instance2.functions.landInfoUser(id);
    assert.equal(landInfo[2],true);
  })

  it("checking␣that␣the␣request␣for␣land␣works!!", async() => {
    let id = await simpleStorageInstance.functions.computeId('maharashtra'
,'thane','kalyan',21);
    let instance2 = simpleStorageInstance.connect(provider.getSigner(
accounts[2]))
    await instance2.functions.requstToLandOwner(id);
    let landInfo = await instance2.functions.landInfoUser(id);
    assert.equal(landInfo[3],accounts[2])
  });
  it("checking␣that␣the␣buyer␣can␣only␣buy,␣if␣the␣request␣is␣approved␣by␣
the␣owner", async() => {
    let id = await simpleStorageInstance.functions.computeId('maharashtra'
,'thane','kalyan',21);
    let state;
    try {
      let instance2 = simpleStorageInstance.connect(provider.getSigner(
accounts[2]))
      await instance2.functions.buyProperty(id,{value : mValue });
      state = false;
    } catch (error) {
      state = true;

    }
    assert(state);
  });
  it("checking␣that␣the␣request␣approval␣works!!", async() => {
    let id = await simpleStorageInstance.functions.computeId('maharashtra'
,'thane','kalyan',21);

    let instance1 = simpleStorageInstance.connect(provider.getSigner(
accounts[1]))
```

```
      let instance2 = simpleStorageInstance.connect(provider.getSigner(
   accounts[2]))
      await instance1.functions.processRequest(id,3);
      let landInfo = await instance2.landInfoUser(id);
      assert.equal(landInfo[4],3)

    })
    it("checking␣that␣the␣buyer␣can␣buy␣the␣property␣and␣the␣ownership␣
   changes", async() => {
      let id = await simpleStorageInstance.functions.computeId('maharashtra'
   ,'thane','kalyan',21);
      let instance2 = simpleStorageInstance.connect(provider.getSigner(
   accounts[2]))
      await instance2.functions.buyProperty(id,{value : mValue});
      let landInfo = await instance2.landInfoUser(id);
      assert.equal(landInfo[0],accounts[2]);

    })

  });

});
```

7.3 Front End

The user when wants to register his/her land on the system it has to be done via admin. The admin registers the user's land on the blockchain network

The following code snippet is in react and it is used by the admin for registration purpose. Only admins can register on this system

7.3.1 Directory Structure of Frontend

7.3.2 Register.js

```
import React,{Component} from 'react';
// import ethersProvider  from "../ethereum/ether";
import { instance } from "../ethereum/factory";
import { accountsList } from "../ethereum/accountsList"
import { number, string } from 'prop-types';
const ethers = require('ethers');
```

```
class Register extends Component {
  constructor(props) {
    super(props);
    this.state = {
        state: '',
        district :'',
        village:'',
        surveyNumber:'',
        CurrentOwner:'',
        marketValue:''

     };
  }
    mySubmitHandler =async (event) => {
    event.preventDefault();
    //alert("You are submitting " + this.state.CurrentOwner);

    // const accounts = ethersProvider.listAccounts();

    console.log("State is ",this.state.state);
    console.log("District is ",this.state.district);
    console.log("Village is ",this.state.village);
    console.log("SurveyNumber is ",this.state.surveyNumber,' ',typeof this.
   state.surveyNumber);
    console.log("CurrentOwner is ",this.state.CurrentOwner);
    console.log("MArketValue is ",this.state.marketValue,' ',typeof this.
   state.marketValue);
    // console.log("This is accountList from there", accountsList);

      const temp1 = this.state.surveyNumber;
      let surveyNumberInt = parseInt(temp1);

      console.log('temp1 is',typeof temp1);
      console.log('surveyNumberInt',surveyNumberInt,'typeOfSurveyNumberInt',
   typeof surveyNumberInt);




      let temp2 = this.state.marketValue;
      console.log('temp2',temp2,'typeof temp2',typeof temp2);

      let landMarketValue = ethers.utils.parseEther(temp2);
      console.log(landMarketValue, typeof landMarketValue);

      let stringMarketValue = landMarketValue.toString()
      console.log('landMarketValue is', landMarketValue.toString())
```

```
    //MarketValue to be passed
    // let tempMarketValue = parseInt(temp2);
    // console.log('tempMarketValue',tempMarketValue);
    // console.log('typeof tempMarketValue',typeof tempMarketValue);


    // console.log("tempSurvey",tempSurvey,"typeof tempSurvey",typeof
 tempSurvey);
  // let propertyId = await instance.computeId(this.state.state, this.
 state.district, this.state.village, temp1);
    let uniqueId = await instance.computeId(this.state.state,this.state.
 district,this.state.village,surveyNumberInt)

    console.log('uniqueId',uniqueId.toNumber());



    const transaction1 = await instance.Registration(this.state.state,
 this.state.district, this.state.village,temp1,
                          this.state.CurrentOwner, landMarketValue,
 uniqueId)

    if(!transaction1){
      console.log("Transaction failed");

    }

    else{
      console.log("Transaction Successful");

    }



};
stateChangeHandler = (event) => {
  this.setState({
      state: event.target.value});
}
```

```
districtChangeHandler = (event) => {
  this.setState({
      district: event.target.value});
}

villageChangeHandler = (event) => {
  this.setState({
      village: event.target.value});
}

surveyNumberChangeHandler = (event) => {
  this.setState({
      surveyNumber: event.target.value});
}

CurrentOwnerChangeHandler = (event) => {
  this.setState({
      CurrentOwner: event.target.value});
}

marketValueChangeHandler = (event) => {
  this.setState({
      marketValue: event.target.value});
}



render() {
  return (
    <form onSubmit={this.mySubmitHandler}>

    <label>state</label>
    <input type='text'
    value={this.state.state}
      onChange={this.stateChangeHandler}/>
    <br/>
    <br/>

    <label>district</label>
    <input type='text' size="50"
    value={this.state.district}
      onChange={this.districtChangeHandler}/>
    <br/>
    <br/>
```

```
      <label>village</label>
      <input type='text' size="50"
      value={this.state.village}
        onChange={this.villageChangeHandler}/>
      <br/>
      <br/>

      <label>surveyNumber</label>
      <input type='text' size="50"
      value={this.state.surveyNumber}
        onChange={this.surveyNumberChangeHandler}/>
      <br/>
      <br/>

      <label>CurrentOwner</label>
      <input type='text' size="50"
      value={this.state.CurrentOwner}
        onChange={this.CurrentOwnerChangeHandler}/>
      <br/>
      <br/>

      <label>marketValue</label>
      <input type='text'
      value={this.state.marketValue} size="50"
        onChange={this.marketValueChangeHandler}/>
      <br/>
      <br/>



      <input
        type='submit'
      />
      </form>
    );
  }
}

export  default Register;
```

7.3.3 Home.js

This page is the home page for every user who have trusted their lands with the blockchain network. The lands belonging to the users are registered here.

This page shows the required details of the land belonging to the users which are feeded by the user at the time of land Registration

```
import React, { Component } from 'react';
// import ethersProvider from '../ethereum/ether';
// import instance from '../ethereum/factory';
import {instance} from '../ethereum/factory';
import  ethersProvider  from "../ethereum/ether";
import {accountsList} from '../ethereum/accountsList';
import { string, any } from 'prop-types';
// var converter = require('hex2dec');


class Home extends Component {

    state={
        accounts : [],
        account : '',
        properties : [],
        propertyInfo : [],
        displayInfo:[],
        count:''
    };


     async componentDidMount(){
         await this.loadAssets()
        // this.loadAssets()
        console.log("Property array: ",this.state.propertyInfo);
    }

     async loadAssets(){

        if(this.state.account == undefined){
            this.setState({
                account : accountsList
            })
        }
        let iproperties = any;
        // iproperties.push(instance.viewAssets());
        iproperties = await instance.viewAssets();

        console.log("This is iproperties it containsid", iproperties);

        this.setState({
            properties : iproperties
        })
```

```
    for(let item of iproperties){
        console.log("In for loop");
        console.log("value of item variable is",typeof item);

         await this.propertyDetails(item.toNumber());
    }
}

async propertyDetails(property){
    let details = await instance.landInfoOwner(property) //Ikde details
mdhe ghetoe value via function of smrtcontct
    // console.log("Property Detials",details);

    let state = true;
    let choiceButton = false;
    let buttonEnabled = details[4];
    if(details[5] == "0x0000000000000000000000000000000000000000"){
        state = false;
    }
    else{
        buttonEnabled = true;
    }

    let propertyInfo = [...this.state.propertyInfo];
    // propertyInfo.push(property, details[0],details[1],details[2],
details[3].toNumber(),buttonEnabled,details[5],
    //     state,details[6],choiceButton) //**** Ikde Push krtoe
     let displayDetails =propertyInfo.push({
         id:property,
         State:details[0],          //items[0]
         District:details[1],       //items[1]
        Village:details[2],         //items[2]
        SurveyNo:details[3].toNumber(), //items[3]
        isAvailable:buttonEnabled,      //items[4]
        Requester:details[5],
        stateBool:state,                                    //items[5]
        SomeDetail:details[6],          //items[6]
        CButton:choiceButton            //items[7]
     })
     console.log('display details',displayDetails,'type of display
details',typeof displayDetails);



     let bagOfData=any;
     bagOfData = propertyInfo;
```

```
        //  console.log('this is bagOfData',bagOfData);
        //  console.log('');
        //  console.log('');
        //  console.log('displayDetails',displayDetails);
        let displayInfo = [...this.state.displayInfo];
        displayInfo = bagOfData;
        this.setState({
            displayInfo
        })
        this.setState({
            propertyInfo
        })
        // console.log('this.state.propertyInfo',this.state.propertyInfo);


        return true;
    }


    async makeAvailableHandler(id,i){
      let Id = id;
      let index =i;
      let a;
      let propertyInfo;
      propertyInfo = [...this.state.propertyInfo];
      console.log('Here is propInfo',propertyInfo);

      for(a=0;a<10;a++){
          if(index==a){
            // alert('In the console with '+a+' '+'ID is '+Id)
            console.log('In the loop with index ',a,' Id is ',Id);
            console.log('status of the land ',a,' Data is',propertyInfo[a].
isAvailable);
            propertyInfo[a].isAvailable = true;
            await instance.makeAvailable(Id);
          }

      }

      this.setState({
          propertyInfo
      })

    }

    async acceptRequest(id,i){
        let Id = id;
```

```
        let index=i;
        let a;
        let propertyInfo;
        propertyInfo = [...this.state.propertyInfo];
        //    this.propertyInfo[i][9] = true;
        console.log('propertyInfo',propertyInfo);
        for(a=0;a<10;a++){
            if (a==i) {
                propertyInfo[a].CButton=true;
                await instance.processRequest(Id,3)
            }
        }


    }

    async rejectRequest(id,i){
        let Id = id;
        let index=i;
        let a;
        let propertyInfo;
        propertyInfo = [...this.state.propertyInfo];
        //    this.propertyInfo[i][9] = true;
        console.log('propertyInfo',propertyInfo);
        for(a=0;a<10;a++){
            if (a==i) {
                propertyInfo[a].CButton=false;
                await instance.processRequest(Id,3)
            }
        }


    }



render (){
    let details= this.state.propertyInfo;


    let orgsData = details ? (
        details.map((details, i)=>{
            return[
                <div key ={i}>
        <p>Property ID :{details.id}</p>
            <p>state:{details.State}</p>
            <p>district:{details.District}</p>
```

```
            <p>village:{details.Village}</p>
            <p>surveyNumber:{details.SurveyNo}</p>
            {/* <button>make Available</button> */}
            <button onClick={()=>this.makeAvailableHandler(details.id,i)}>
   make Available</button>
            <div>
            {details.stateBool && details.SomeDetail==1 ? (
          <div>
       <p>You have a request from {details.Requester}</p><br/>
             <button onClick={()=>this.acceptRequest(details.id,i)}>Accept
   </button><br/>
             <button onClick={()=>this.rejectRequest(details.id,i)}>Reject
   </button>
            </div>
            ) : (
         <div>
            <p></p>
       </div>
       )}
            </div>
            <hr/>
            <br/>
         </div>
            ];
       })
    ):(
        <div>No data present
        </div>
    )
    return (<div>{orgsData}</div>) ;


};

}

export default Home;
```

7.3.4 Search.js

The buyer who is in search of the lands such that the land details could be trusted will search lands on this application with this page.

This is the page which where the buyer requests the land to the seller and further procedural details are obtained here

```
import React, { Component } from 'react';
import { instance } from '../ethereum/factory';
import { any } from 'prop-types';
import { accountsList } from "../ethereum/accountsList";
const ethers = require('ethers');

class Search extends Component {
  constructor(props) {
    super(props);
    this.state = {
      state: '',
      district: '',
      village: '',
      surveyNumber: '',
      loaded: false,
      availableForRequest: false,
      availableForBuy: false,
      landInfo: [],
      landRequest: [],
      myPropertyId: '',
      buttonOneDisable: false,
      buttonTwoDisable: true
      // propertyId: any,

    };
  }

  mySubmitHandler = async (event) => {
    event.preventDefault();
    try {

      let propertyId = await instance.computeId(this.state.state, this.state
   .district, this.state.village, this.state.surveyNumber)
      console.log('property Id in search', propertyId);
      console.log('survey number', this.state.surveyNumber, ' typeof
    surveynumber', typeof this.state.surveyNumber);

      let Id = propertyId.toNumber();
      console.log('the property Id in number is ', Id);
      this.setState({
        myPropertyId: propertyId
      })


      let landDetails = any;
      landDetails = await instance.landInfoUser(propertyId);
```

```
    console.log('landDetails', landDetails);

    let landInfo = [...this.state.landInfo];
    console.log('landinfo', landInfo);

    landInfo = landDetails;
    console.log('landinfo', landInfo);


    this.setState({
      landInfo
    })


    this.setState({
      loaded: true
    })
    console.log('Setting loaded state this.state.laoded', this.state.
 loaded);   //setting loaded to true


    if ((landDetails[3] == "0x0000000000000000000000000000000000000000")
 && (landDetails[2]) && (landDetails[0] != accountsList)) {
      this.setState({
        availableForRequest: true
      })
    }
    if (landDetails[3] == await accountsList && landDetails[4] == 3) {
      console.log('In buy wala condition');

     await this.setState({
        availableForBuy: true
      })
      console.log('Setting buy state to true',this.state.availableForBuy);

    }

  } catch (error) {
    console.log("error:", error);

  }

}

async makeAvailable(id) {
  try {
    let landDetailsRequest = await instance.requstToLandOwner(id);
```

```
    console.log('landDetailsRequest', landDetailsRequest);
    // let landRequest = [...this.state.landRequest]
    // landRequest = landDetailsRequest;
    // console.log('This is a land request',landRequest);
    // this.setState({
    //   landRequest
    // })
    this.setState({
      buttonOneDisable: true
    })
  } catch (error) {
    console.log('error', error);

  }
}

async buyProperty(id){
  let landInfo = [...this.state.landInfo];
  // console.log('landinfo', landInfo);
  // let mValue = parseInt(landInfo[1]);
  // console.log('mValue',mValue);
  // mValue = (mValue/10);
  // let StringValue=  mValue.toString();
  // console.log("mValue:",StringValue)
  let mValue = landInfo[1].toString()
  console.log('landInfo[1]',landInfo[1]);
  let pValue = parseInt(mValue);
  console.log('pValue is ',pValue,' typeof pValue ',typeof pValue);
  pValue +=(pValue/10)
  let StringValue = pValue.toString()
   console.log('StringValue', StringValue);

  let bNumb = ethers.utils.parseEther(StringValue);
  console.log('-------->This is parseether',bNumb);


  try {
    landInfo = await instance.buyProperty(id, {
                                            gasLimit: 33600,
                                            value: landInfo[1]
                                          })
    this.setState({
      buttonTwoDisable : true
    })
  } catch (error) {
    console.log('---------->',error);
```

```
  }




}

stateChangeHandler = (event) => {
  this.setState({
    state: event.target.value
  });
}

districtChangeHandler = (event) => {
  this.setState({
    district: event.target.value
  });
}

villageChangeHandler = (event) => {
  this.setState({
    village: event.target.value
  });
}

surveyNumberChangeHandler = (event) => {
  this.setState({
    surveyNumber: event.target.value
  });
}

render() {

  let loadedValue = this.state.loaded;
  let landInfo = [...this.state.landInfo];
  let tid = this.state.myPropertyId;

  return (
    <div>

      <div>
        <form onSubmit={this.mySubmitHandler}>

          <label>state</label>
          <input type='text'
            value={this.state.state}
```

```
              onChange={this.stateChangeHandler} />
            <br />
            <br />

            <label>district</label>
            <input type='text' size="50"
              value={this.state.district}
              onChange={this.districtChangeHandler} />
            <br />
            <br />

            <label>village</label>
            <input type='text' size="50"
              value={this.state.village}
              onChange={this.villageChangeHandler} />
            <br />
            <br />

            <label>surveyNumber</label>
            <input type='text' size="50"
              value={this.state.surveyNumber}
              onChange={this.surveyNumberChangeHandler} />
            <br />
            <br />

            <input type='submit' />
          </form>
        </div>
        <div>
          {loadedValue ? (

            <div>
              <p>Property Id is : {tid.toNumber()}</p>
              <p>Account: {landInfo[0]}</p><br />
              <p>Market value : {landInfo[1].toString()}</p><br />
            </div>
          ) : (
              <div>
                <p></p>
              </div>
            )}
            {/*Making available for request */}
          {this.state.availableForRequest ? (

<div>
  <button onClick={()=>this.makeAvailable(tid.toNumber())}>RequestForLand</
   button>
```

```
</div>
) : (
  <div>
    {/* <button disabled='false'>RequestForLand</button> */}
    <p></p>
  </div>

)}

{/*Making available for Buy */}
{this.state.availableForBuy ? (

<div>
  <button onClick={()=>this.buyProperty(tid.toNumber())}>Buy Property</
   button>
</div>
) : (
  <div>
    <p></p>
  </div>


)}

        </div>


      </div>

    );
  }
}

export default Search;
```

7.3.5 Navigation.js

This page is the navigation page which is required navigating through the home and search tabs. Admins will require the /register functionality within this to get to registration

```
import React from "react";
import { NavLink } from 'react-router-dom';

const Navigation = () => {
  return (
    <div>
```

```
      <NavLink to="/">Home</NavLink><br/><br/>
      <NavLink to="/search">Search</NavLink><br></br>
      <NavLink to="/register"></NavLink><br></br>

    </div>
  );
};

export default Navigation;
```

7.4 Connecting With Ethereum

7.4.1 accountList.js

This functionality is used to get the account number that is the hexadecimal account address of the account which is currently getting used in metamask.

```
import ethersProvider from '../ethereum/ether';

// eslint-disable-next-line no-undef
export let accountsList = ethersProvider.listAccounts();

// var secondAccountList = [];
// secondAccountList = ethersProvider.listAccounts();
// console.log('list',secondAccountList);
//this is accountList
//testcommit
console.log('accountsOfMetamask',accountsList);
```

7.4.2 ether.js

```
const ethers = require('ethers');

let ethersProvider;

if(typeof window!='undefined' && typeof window.web3!=='undefined'&& typeof
   window.ethereum!=='undefined')
{
   window.ethereum.enable()

    //const ethersProvider = new ethers.providers.Web3Provider(window.web3.
   currentProvider);
```

```
  //  var ethersProvider = new ethers.providers.Web3Provider(web3.
    currentProvider, ethers.providers.networks.ropsten);
    ethersProvider = new ethers.providers.Web3Provider(window.web3.
    currentProvider);

     console.log("Metamask is connected");


}

export default ethersProvider;
```

7.4.3 factory.js

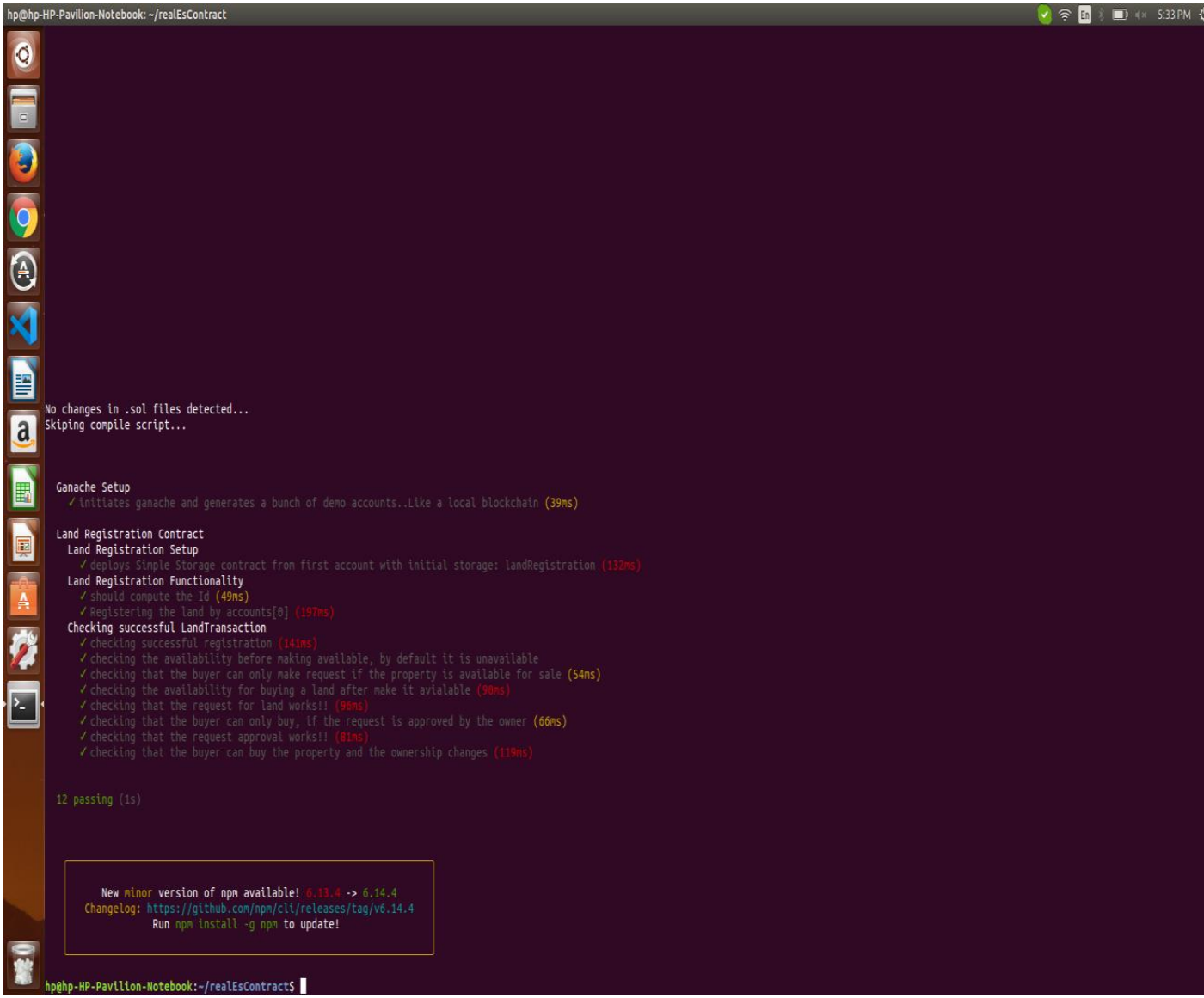
The required contract instance is obtained within this code. The instance is further used for performing operations on blockchain.

```
import Web3 from 'web3';
import ethersProvider from './ether';
import { abiCT } from "./contractTools";

const ethers = require('ethers');
const address = '0xac7eba938a3bfb754cd7915b05454e9471b9f3b7'; //kovan
    address

const metamaskSigner = ethersProvider.getSigner();

export const instance = new ethers.Contract(address, abiCT, metamaskSigner);
```

Chapter 8

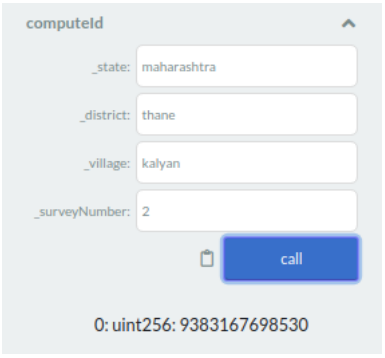
Test Cases

8.0.1 Testing On terminal

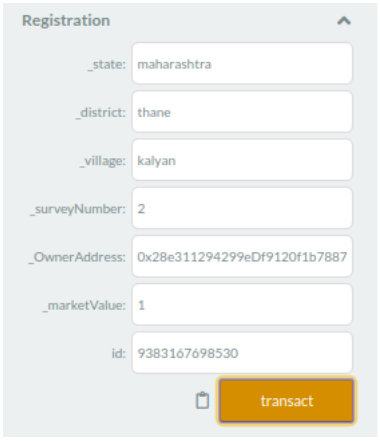


8.0.2 Testing On Remix IDE

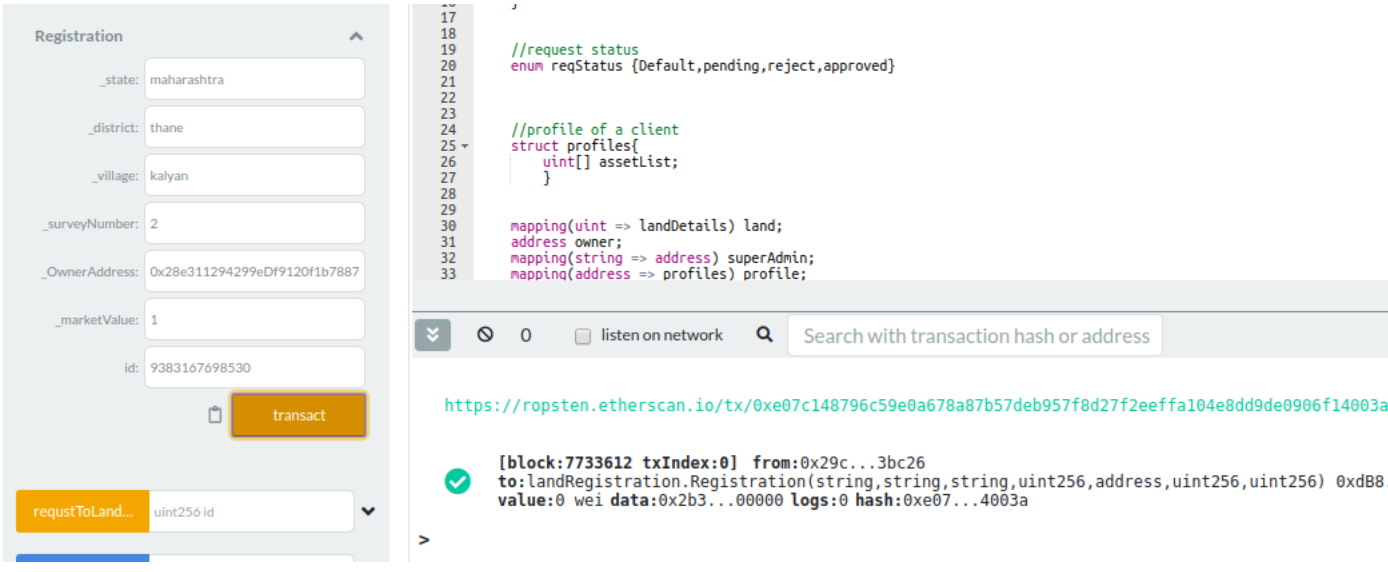
computeID



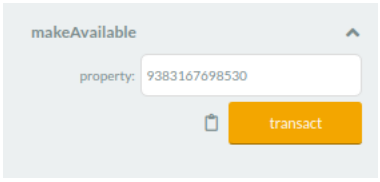
For Computing ID



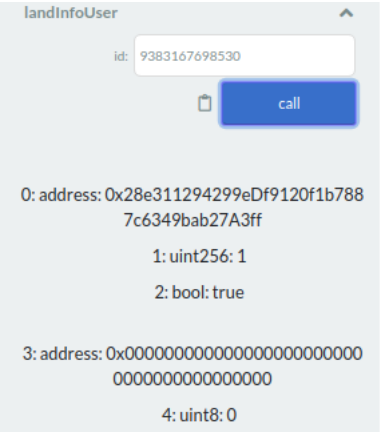
Testing Registration



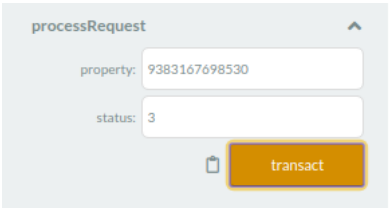
Registration Confirmed Transaction



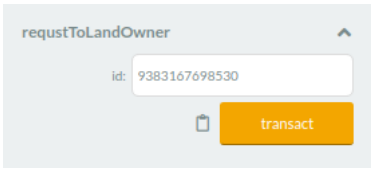
Testing Make Available



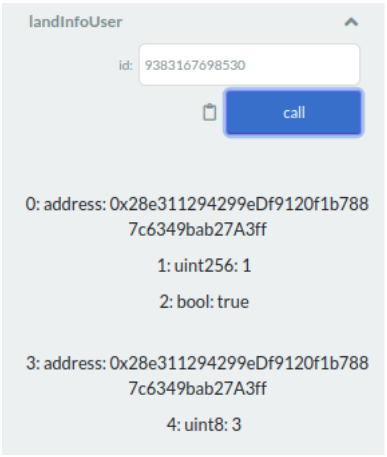
Before Transfering Ownership



Testing Request Processing Function



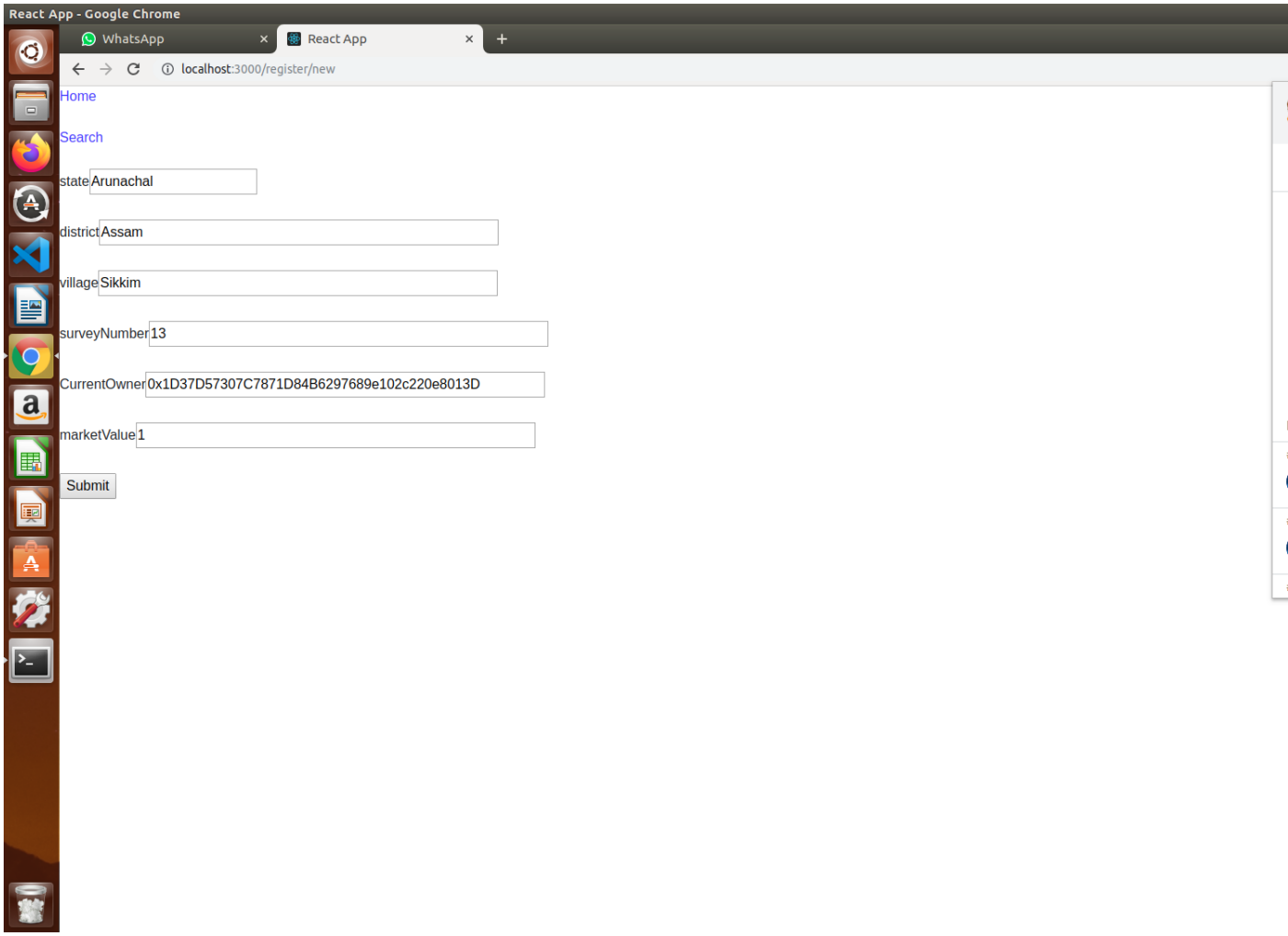
Testing Request Via Id



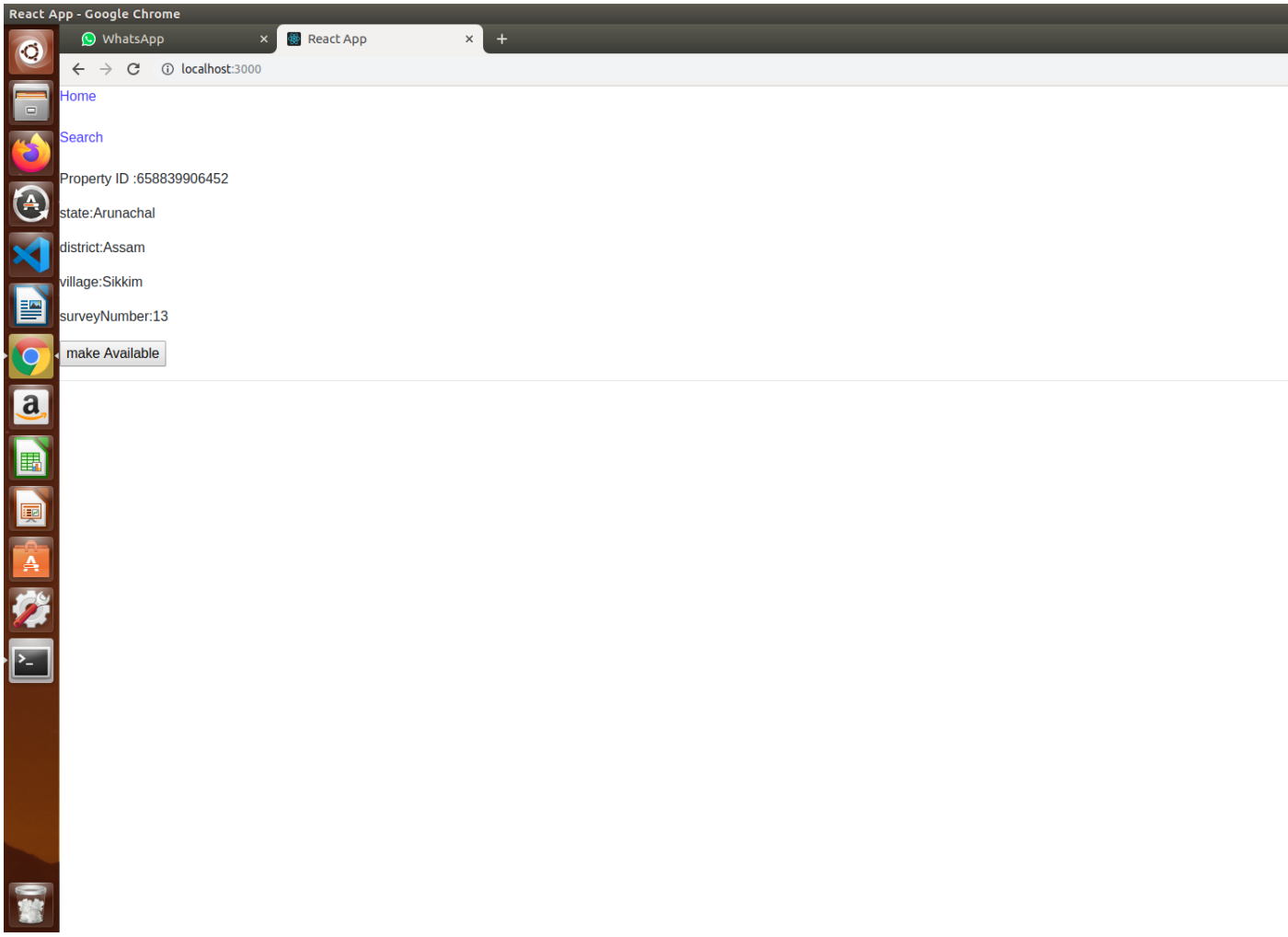
After Transfering Ownership

Chapter 9

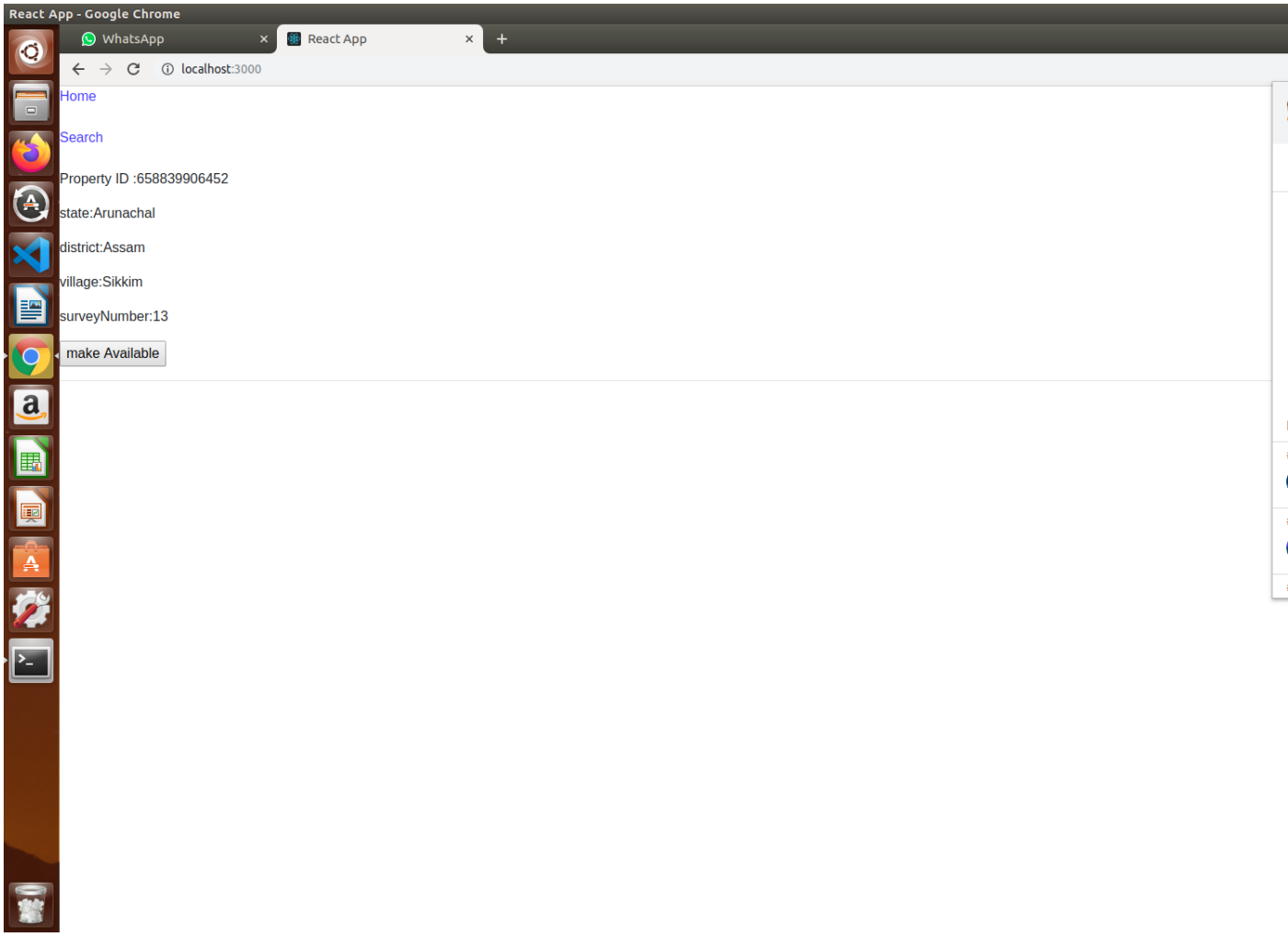
Result



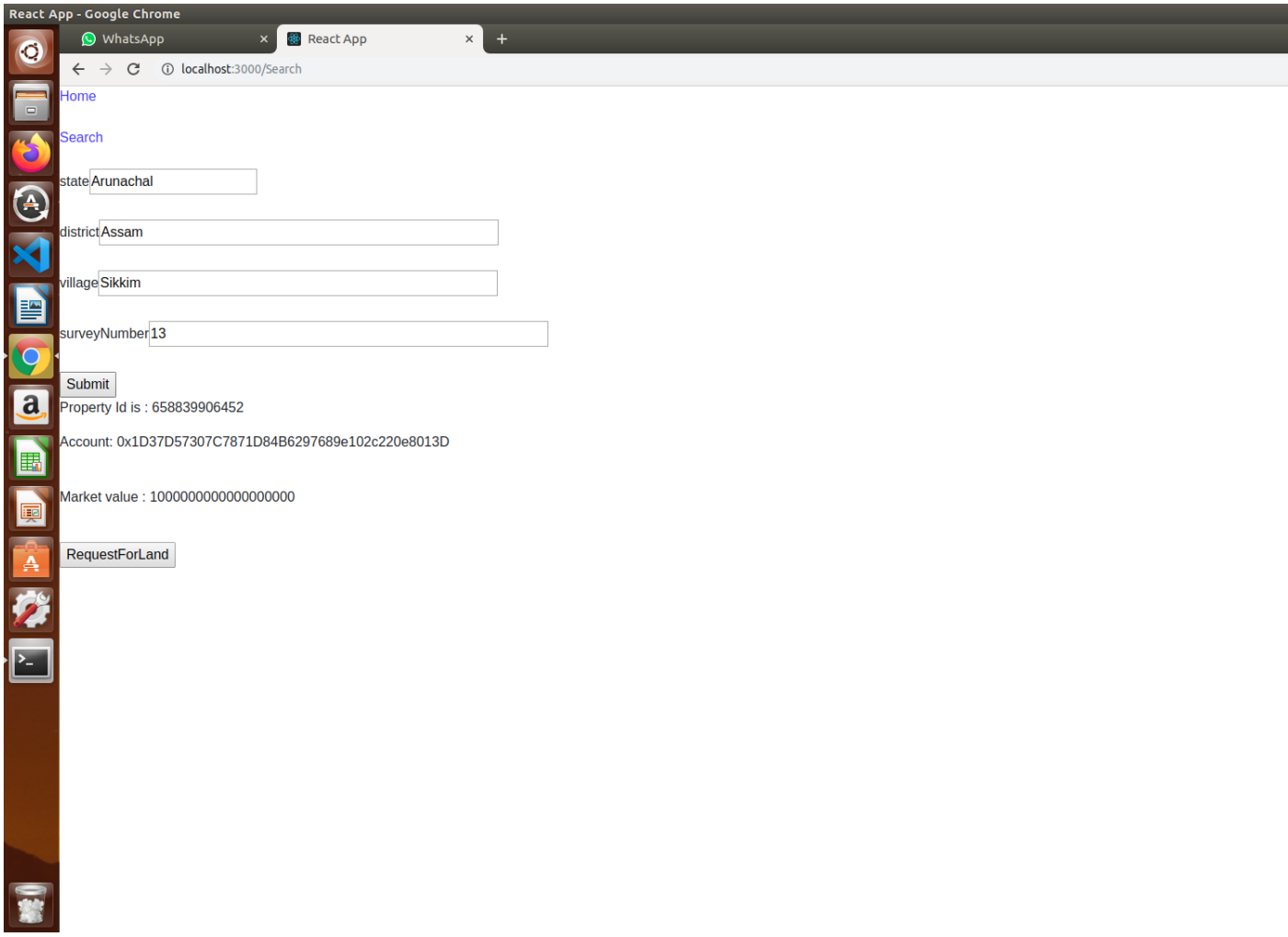
Registration



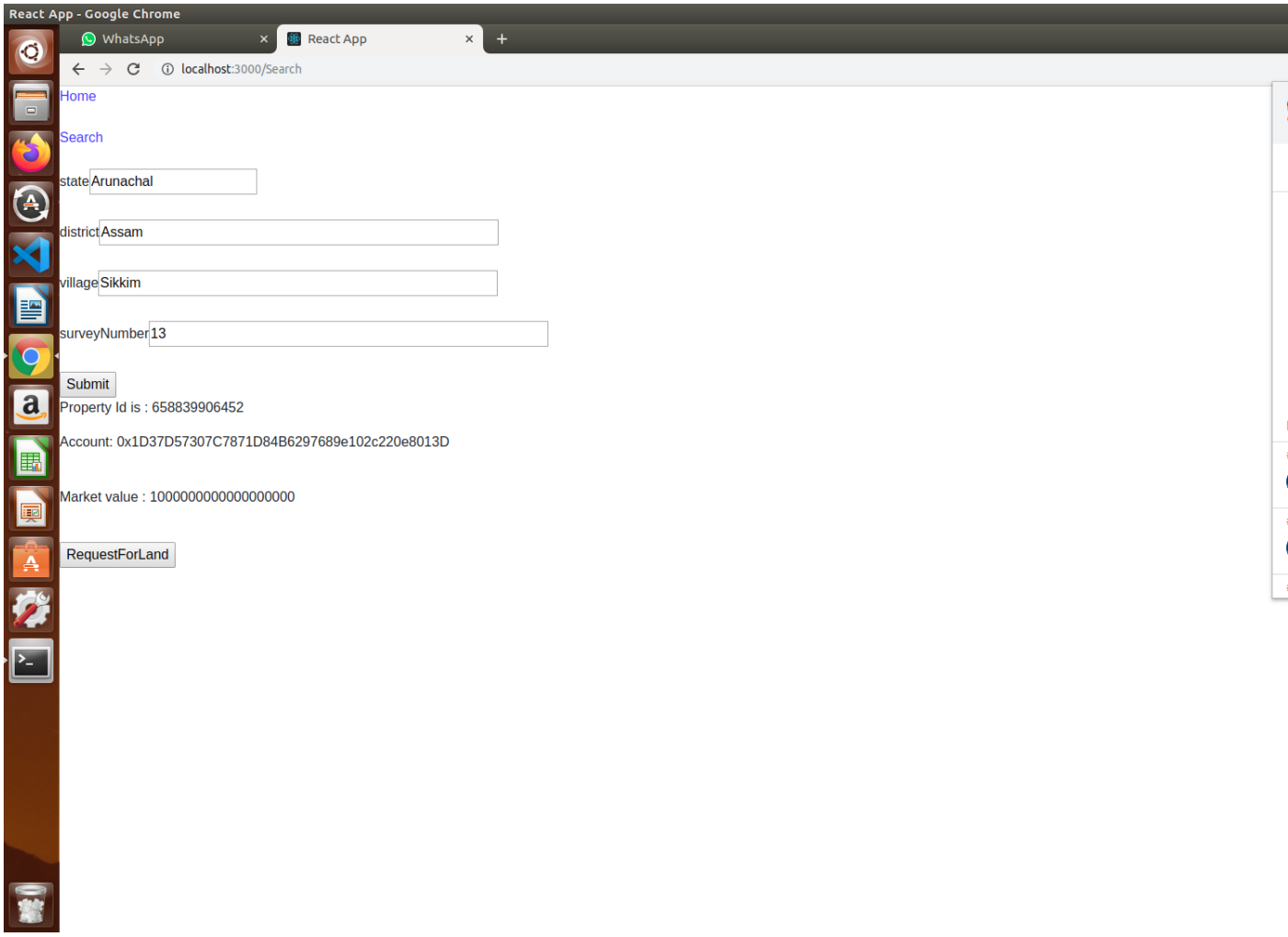
Displaying Registered Land



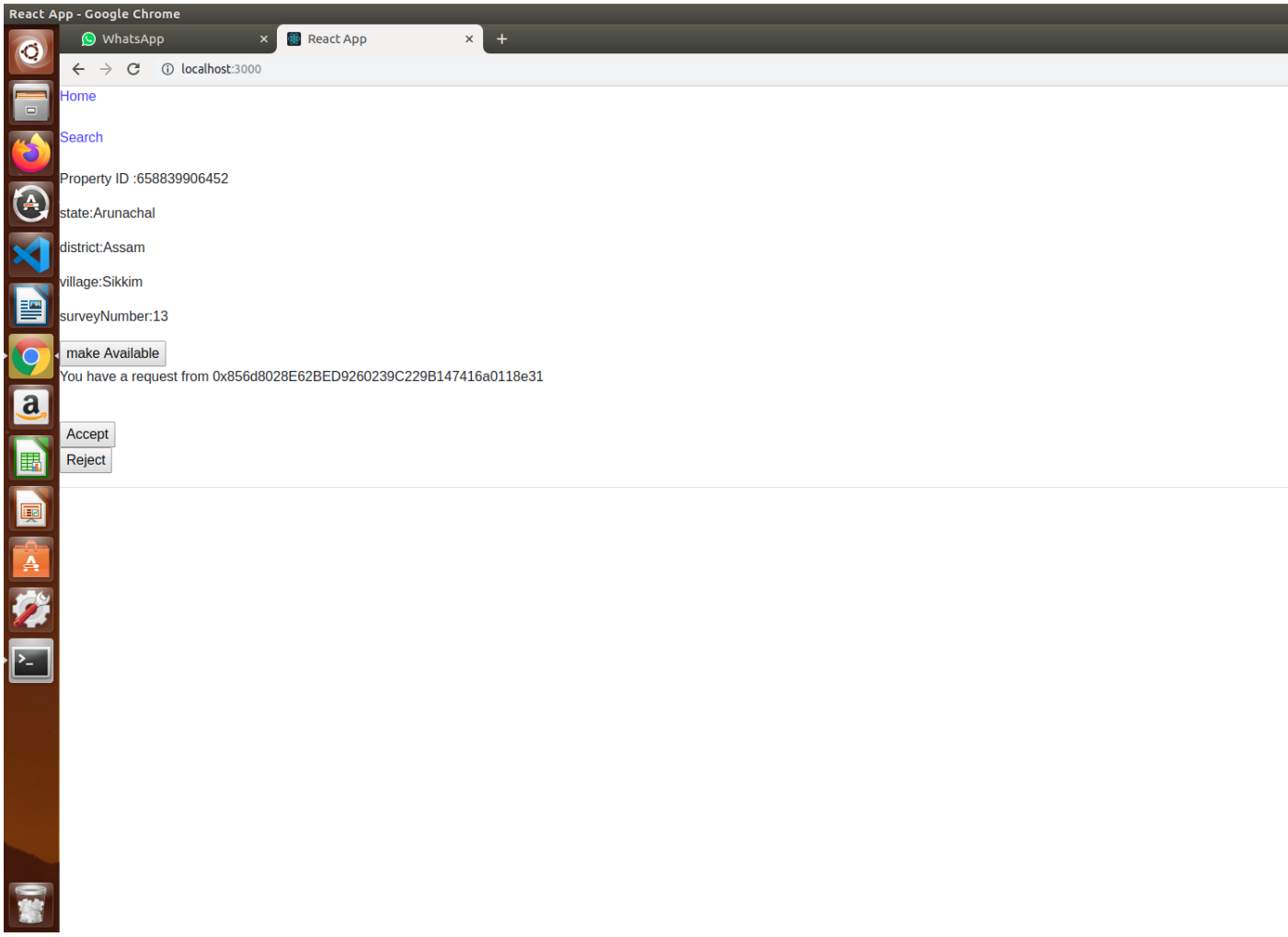
Making the land available for sale



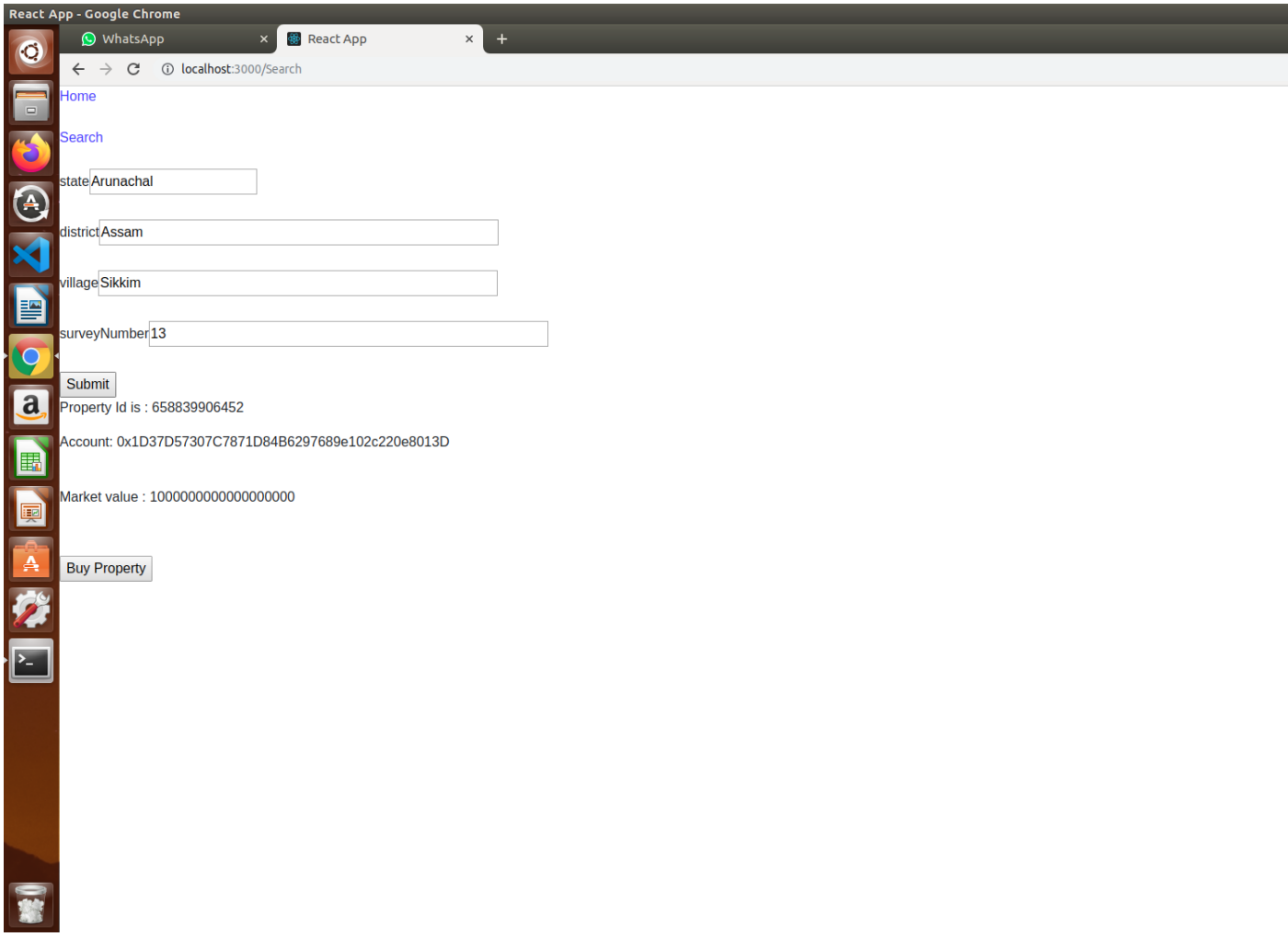
The Buyer Searches For the Land



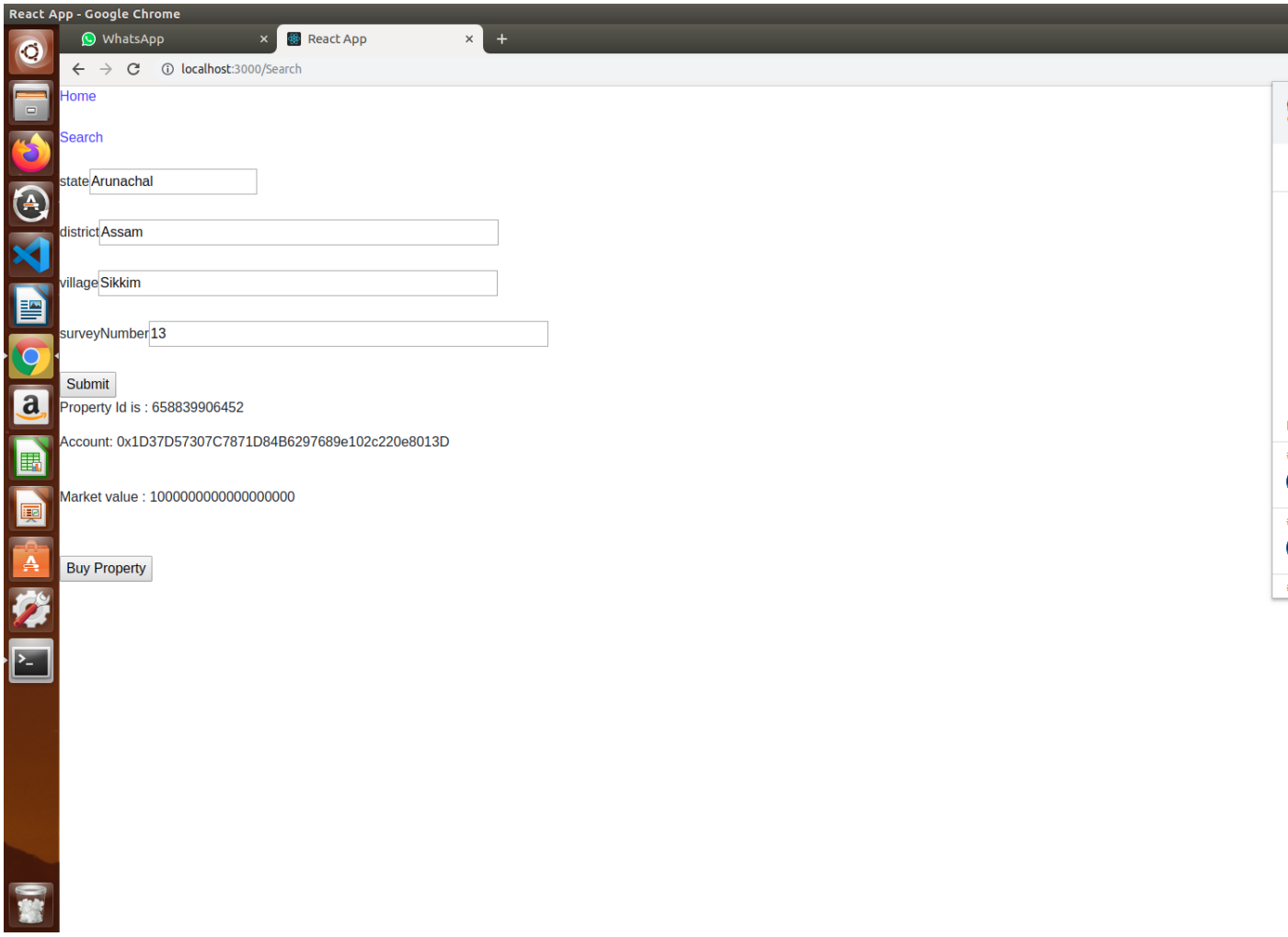
The buyer Requests for the land



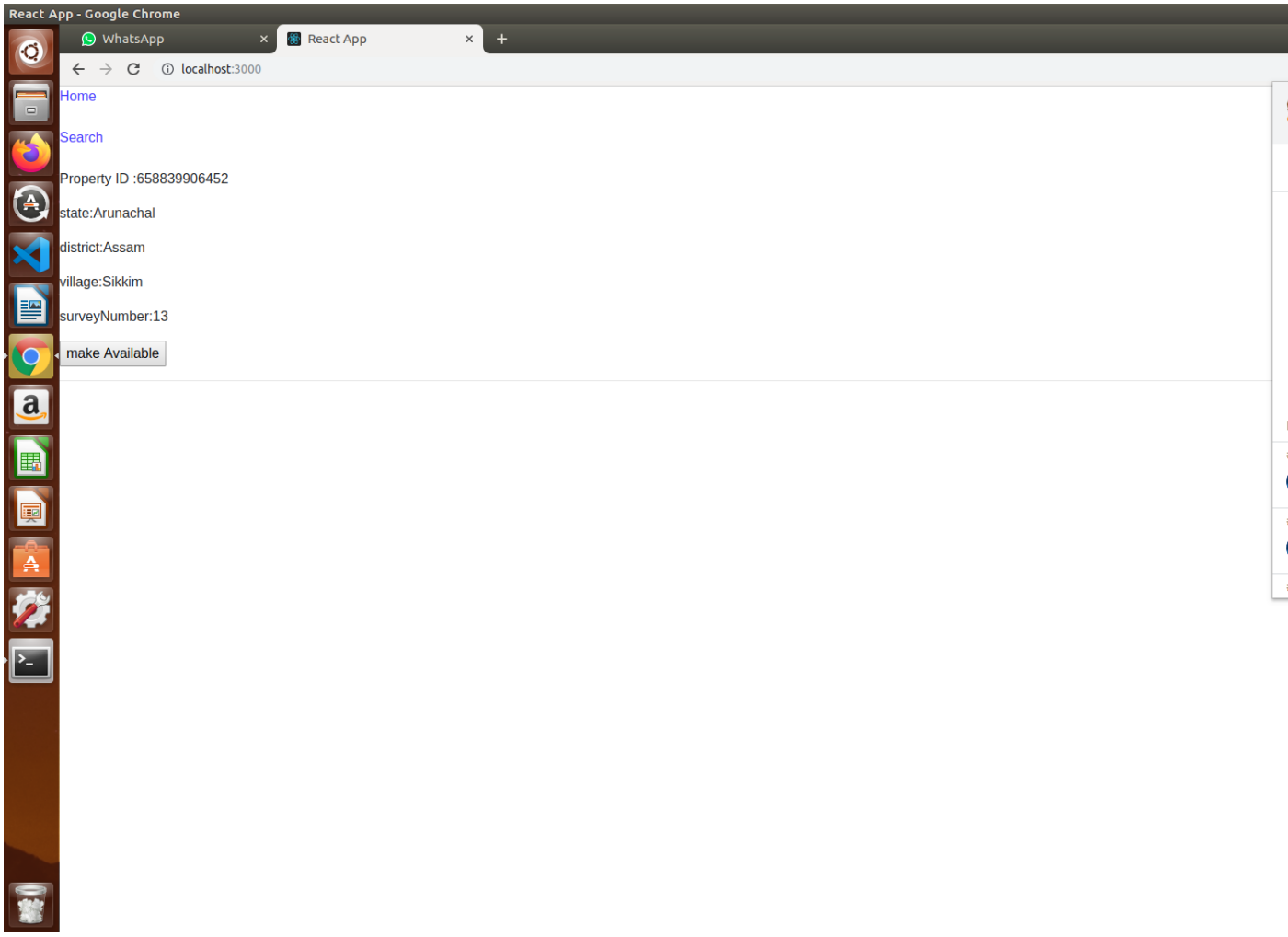
The Seller Decides to Accept Or Reject



If Accepted This Buy Button will appear



After clicking buy the transfer will happen



After the transfer the seller will have land on his account

Chapter 10

Conclusions and Future Scope

Blockchain is a forum where there can be almost all abuse and fraud Removed, and everything may be legal. The real estate initiative aims to solve the issue Issue regarding land registration and transfer of ownership. The project reveals Eliminating barristers and other intermediaries.

In the case of potential Scope rather than using a new wallet as a metamask Could be created to reduce the problems of a standard connecting solution and thus enhancing the total security of the application. The next big thing might be registering flats based on their respective address on the blockchain network.

Bibliography

[1] Nathan Shedroff “Self-Managing Real Estate”, IEEE 2018.

[2] Shuai Wang , Liwei Ouyang, Yong Yuan , Senior Member, IEEE, Xiaochun Ni, Xuan Han, and Fei-Yue Wang ”Blockchain-Enabled Smart Contracts: Architecture, Applications, and Future Trends”

[3] Alex Norta, Chad Fernandez, Stefan Hickmott ”Commercial Property Tokenizing With Smart Contracts”

Some Websites

- https://tecadmin.net/install-latest-nodejs-npm-on-ubuntu/
- https://www.zeolearn.com/magazine/setup-react-ubuntu

Appendices

Information related to the necessary installations

Appendix-A: Install NodeJS

Commands are as follows

sudo apt-get update

Installing curl for data transfer
sudo apt-get install curl

Adding PPA to our system using curl
curl -sL https://deb.nodesource.com/setup$_1$4.$x|sudo - Ebash -$

$NowinstallingNodeJs$
$sudoapt - getinstallnodejs$

$CheckingNodeJsVersion$
$node - v$

$CheckingNodePackageManager(npm)Version$
$npm - v$

Appendix-B: Install ReactJS

Installing ReactJs Via terminal

Using npm to install create-react-app
npm install -g create-react-app

Checking the create-react-app version
create-react-app –version

Creating a new project
npx create-react-app my$_p roject$

$cdmy_project$

$npmstart$

Acknowledgement

We have great pleasure in presenting the report on **Real Estate On Blockchain** We take this opportunity to express our sincere thanks towards our guide **Prof. Sachin Malave** Department of Computer Engineering, APSIT thane for providing the technical guidelines and suggestions regarding line of work. We would like to express our gratitude towards his constant encouragement, support and guidance through the development of project.

We thank **Prof. Sachin Malave** Head of Department,Computer Engineering , APSIT for his encouragement during progress meeting and providing guidelines to write this report.

We thank **Prof. Amol Kalugade** BE project co-ordinator, Department of Computer Engineering, APSIT for being encouraging throughout the course and for guidance.

We also thank the entire staff of APSIT for their invaluable help rendered during the course of this work. We wish to express our deep gratitude towards all our colleagues of APSIT for their encouragement.

Siddhant Bhadsavale: 16102038:
Himanshu Malhotra: 17202006:

Chapter 11

Gantt Chart

Gantt Chart