

Software Development
Ewing, NJ
2019

Table of Contents

Research	2
Project Description	3-4
Plan of Work	5
Documentation	6-18
Self Evaluation & Future Prospects	19
Works Cited	20
Student Copyright Checklist	21

Research

- Problem: Lack of infrastructure to store land records and transactions
 - The poor need to be able to leverage their physical assets
 - Land owned by poor cannot be leveraged if is not recorded
 - This is called “dead capital”
 - There is over \$10 trillion in dead capital around the world
 - Elites have no incentive to change the situation and streamline registration process
 - Land can be stolen if not properly registered
 - Benefits of land registry
 - The poor can use their land to gain more capital, or engage in the economy
 - E.g. credit, loans, mortgages, etc.
 - Government can also gain revenue from property taxes
 - Property taxes compose 0.6% of GDP in developing countries, but 2.2% of GDP in developed countries
- Blockchain
 - Decentralized database
 - Multiple “nodes”, or computers, connect to a blockchain
 - Nodes have wallets, or users of the blockchain
 - Users can transact amongst themselves and register assets
 - Each transaction and registration is recorded on a “block”
 - Each block is connected to the previous with a hash function
 - The transaction is approved when a node validates the hash function
 - The ledger, or list of blocks, is stored on all nodes
 - Multichain - private blockchain
 - Does not allow any computer to connect as a node
 - Transactions are validated through a consensus, so all nodes must find the solution to the hash function
 - Permissioned: each user node has different permissions
 - Benefits of blockchain for land registration
 - Distributed
 - Secure
 - Impermeable
 - Since all nodes have a copy of the blockchain, each node has to be compromised to change the blockchain
 - Allows for transactions between users

Project Description

Asset ownership has always been a global issue, but this is especially true in developing nations. These countries are often plagued by corruption or by a lack of infrastructure. On one hand, countries without the ability to officially record land ownership create dead capital, because it is difficult to utilize a resource with an unknown owner for economic gain. On the other hand, developing countries with these systems often have impermanent records. This increases the risk of the government or other entities to engage in fraud by modifying the records. A robust asset recording system requires permanent records in addition to the ability to conduct asset transactions when two users, the owner and receiver, consent to them. To address the flaws of current land records systems, our group has developed a blockchain-based app which allows users to register and transact land in a secure manner.

Blockchains are decentralized databases that store asset ownership and transaction data. A blockchain network consists of a group of nodes, or computers. These nodes are the interface through which users interact with the blockchain. A user will connect to a node, thus creating a wallet. The user can then register assets in the wallet and transact with other users. Each transaction is recorded in a “block”, containing the transaction information. Each new “block” is “chained” to the previous block, creating a “blockchain”. The blocks are connected to each other through the use of a cryptographic hash function. The data in the block is fed into the function to get a unique value. This unique value is then stored on both the current and the previous block. If the data on a block changes, then the value of the hash function changes, breaking the block. That is the beauty of the blockchain. It is immutable. The data on the blockchain, or the “ledger”, is stored on all nodes. This way, all the data on a blockchain is verified, secure, and decentralized. We used a specific kind of blockchain for our app, called a private blockchain. A private blockchain, like the name suggests, is a chain where nodes cannot connect freely. They must have permission by all the nodes to connect. In addition, each user can be permissioned, allowing certain users to issue and send assets. We used the private blockchain framework Multichain for our app.

We made a JavaScript web app to allow users to connect to the blockchain and execute its functionality. We used a web app framework called Meteor, which automates the web server deployment and allows for the rendering of an HTML JavaScript web app. We used the default Meteor templating engine, called Blaze, to interface between HTML and JavaScript. Our app had 5 use cases: connect to a blockchain, register an asset, list all of a user's assets, transact assets, and display an asset in a map. We integrated the blockchain connection use case with a login function on our app, so a user's wallet was connected to their account in the app. We had a MongoDB database to store users; however, the blockchain data was not stored in the app's database. Rather, we used a MultiChain API call to get blockchain data. We used the Google Maps Node.js Geocoding API for address verification in asset registration, and we used the Meteor Google Maps package for displaying the asset on a map.

We believe that our app can be used to truly solve the problems of land registration and transactions in developing countries. Through the functionality of the blockchain, we could liberate large amounts of dead capital, which is limiting the global economy. Our solution could lift millions of people out of poverty and end the broken system of land titling seen in many places.

Project Requirements

Purpose:

To create a blockchain web app allowing for land registration and transactions.

Audience:

This app is intended for land registrars/governments who wish to set up a secure, decentralized way of recording land data.

Use Cases:

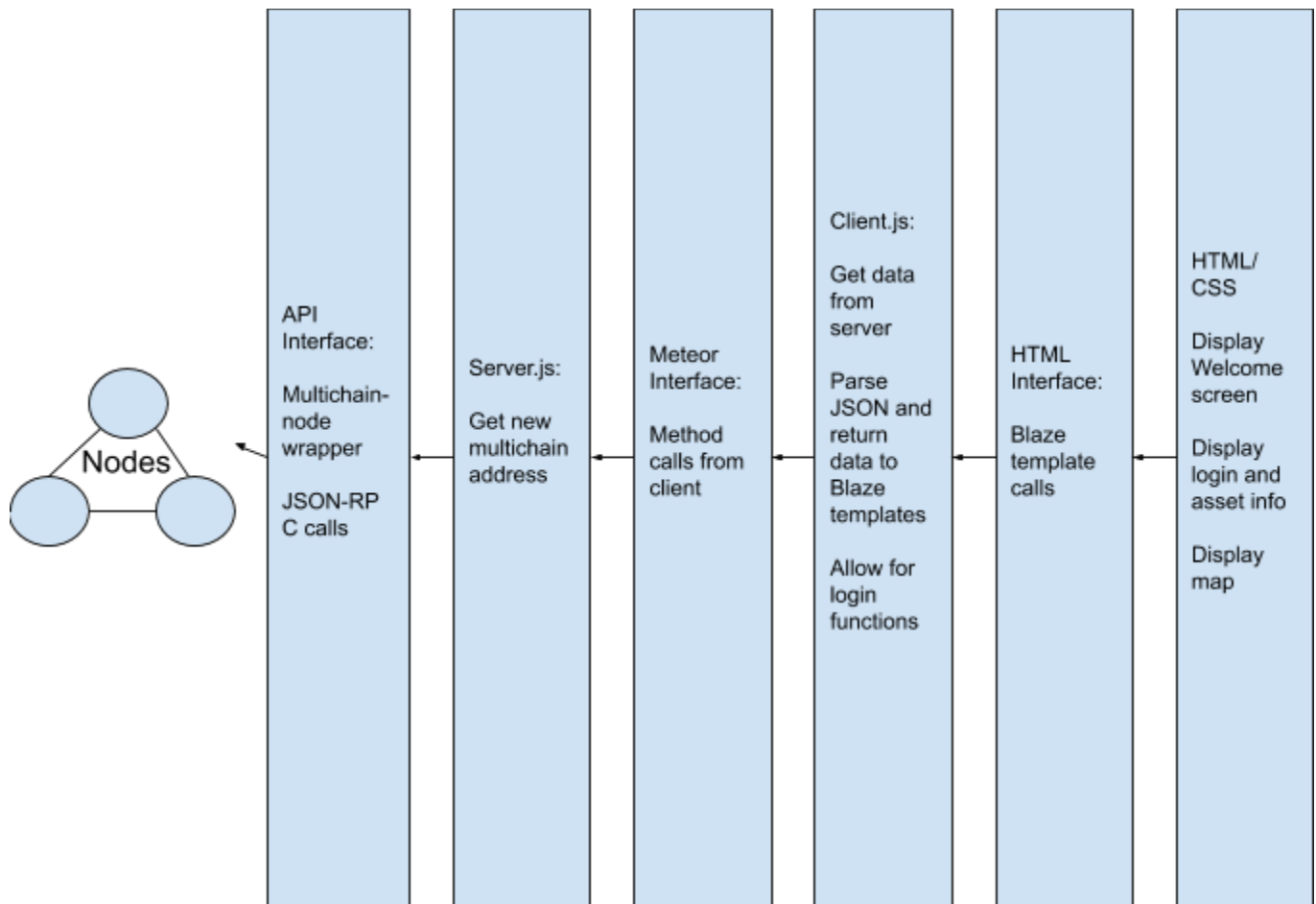
- Allow users to connect to the blockchain
- Allow users to register land assets
- Shows a list of a user's assets
- Show assets on a Google Map
- Allow users to send assets to another user

Software Design:

We created a web app built with the Meteor web app framework

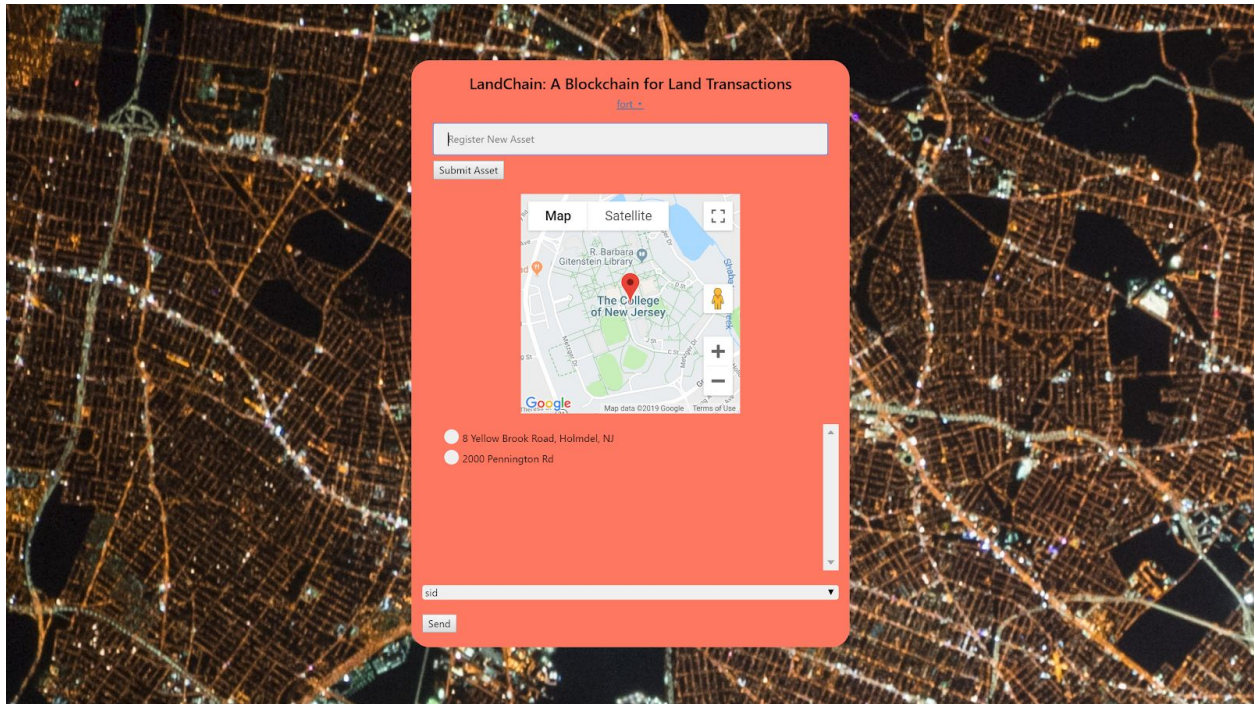
- Client
 - Call server side methods
 - Display results of server side methods through Blaze templates
 - Allow for login and account management through Meteor Account packages
 - Show Google Map
- Server
 - Call Multichain API calls using multichain-node npm package
 - Parse API call returns and return as DOM object to the client

High Level Software Design



Testing

Final code output:



Code screenshots

Multichain-cli

```
Command Prompt - multichaind my_new_chain -daemon
C:\Users\dkuma\AppData\Roaming\MultiChain>cd ../
C:\Users\dkuma\AppData\Roaming>cd ../
C:\Users\dkuma\AppData>cd ../
C:\Users\dkuma>multichaind my_new_chain -daemon
'multichaind' is not recognized as an internal or external command,
operable program or batch file.
C:\Users\dkuma>cd Downloads
C:\Users\dkuma\Downloads>cd multichain
C:\Users\dkuma\Downloads\multichain>multichaind my_new_chain -daemon
MultiChain 1.0.8 Daemon (latest protocol 10011)
Other nodes can connect to this node using:
multichaind my_new_chain@169.254.49.86:4773
This host has multiple IP addresses, so from some networks:
multichaind my_new_chain@192.168.86.141:4773
Listening for API requests on port 4772 (local only - see rpcallowip setting)
Node ready.
```



```
Command Prompt
Microsoft Windows [Version 10.0.17134.706]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\dkuma>cd Downloads
C:\Users\dkuma\Downloads>cd multichain
C:\Users\dkuma\Downloads\multichain>multichain-cli my_new_chain getnewaddress
{"method":"getnewaddress","params":[],"id":"41847590-1555106660","chain_name":"my_new_chain"}
1E1RfhrGPcbP2cvHjM7E8re9Ecc6XxPbvKMU78
C:\Users\dkuma\Downloads\multichain>
```

HTML:

```
<head>
  <title>simple-todos</title>
</head>
<template name="map">
  <div class="map-container">
    {{ googleMap name="map" options=mapOptions }}
  </div>
</template>
<body>
  <div class = "container">
    <header>
      <h1>LandChain: A Blockchain for Land Transactions</h1>
      {{ loginButtons }}
      {{#if currentUser}}
      <form class="issuenew">
        <input type="text" name="text" placeholder="Register New Asset" />
      </form>
      {{#if returnForm}}
      <form class="issueasset">
        <input type="button" value="Submit Asset"/>
      </form>
      {{/if}}
      {{#if errorReturn}}
      Asset Registration Error!
      {{/if}}
      {{ map }}
      {{else}}
      <div class="paragraph">
        <p>Welcome to LandChain, a blockchain web app for land registration. Start off by creating an account</p>
      </div>
      {{/if}}
    </header>
    {{#if currentUser}}
    <div class="list-group">
      {{#each counter}}
      <input type="radio" name="asset_button" value="{{name}}" class="form-radio">{{name}}<br>
      {{/each}}
    </div>
    <div class="send">
      <form class="transactasset">
        <select name="user">
          {{#each returnAllUsers}}
          <option value="username">{{username}}</option>
          {{/each}}
        </select>
        <input type="submit" value="Send">
      </form>
    </div>
    {{/if}}
  </div>
```

JavaScript-Client

```

import { Template } from 'meteor/templating';
import { ReactiveVar } from 'meteor/reactive-var';
import { Meteor } from 'meteor/meteor';
import { Accounts } from 'meteor/accounts-base';

import './main.html';
Meteor.startup(function() {
  GoogleMaps.load({key: "AizaSyDK_WE6H2MK6KUwKXWj5_GKNk7PI7Rg1yc"});
});
Session.set('location', {"lat":100, "lng":100});
Session.set('submit', false);
Session.set('error', false);
Accounts.ui.config({
  passwordSignupFields: 'USERNAME_ONLY',
});
Accounts.onLogin(function() {
  Meteor.call('getNew', (err, result) => {
    console.log(err);
    console.log("My result:", result);
    Session.set('q', result);
  });
});
Accounts.onLogout(function() {
  Session.set('q', null);
});

```

```

Template.body.onCreated(function helloOnCreated() {
  Meteor.call('getNew', (err, result) => {
    console.log(err);
    console.log("My result:", result);
    Session.set('q', result);
  });
  return Session.get('q');
});
Template.map.onCreated(function() {
  var self = this;

  GoogleMaps.ready('map', function(map) {
    var marker;
    self.autorun(function() {
      var latlng = Session.get('location');
      if (!latlng)
        return;
      if (!marker) {
        marker = new google.maps.Marker({
          position: new google.maps.LatLng(latlng.lat, latlng.lng),
          map: map.instance
        });
      }
      else {
        marker.setPosition(latlng);
      }
      map.instance.setCenter(marker.getPosition());
      map.instance.setZoom(15);
    });
  });
});
Template.map.helpers({
  mapOptions: function() {
    var latlng = Session.get('location');
    // Initialize the map once we have the LatLng.
    if (GoogleMaps.loaded() && latlng) {
      return {
        center: new google.maps.LatLng(latlng.lat, latlng.lng),
        zoom: 15
      };
    }
  }
});

```

```

Template.body.helpers({
  counter() {
    return Session.get('q');
  },
  returnAllUsers() {
    return Meteor.users.find();
  },
  returnForm() {
    return Session.get('submit');
  },
  errorReturn() {
    return Session.get('error');
  }
});

```

```

Template.body.events({
  'submit .issuenew'(val) {
    val.preventDefault();

    const target = val.target;
    text = target.text.value;
    Meteor.call('geoLocate', text, (err, result) => {
      console.log("Error", err);
      if(!err) {
        console.log(result.geometry.location);
        Session.set('location', result.geometry.location);
        Session.set('asset', text);
        Session.set('submit', true);
        Session.set('error', false);
      }
    });
    target.text.value='';
  },
  'click .issueasset'() {
    console.log(Session.get('asset'));
    Meteor.call('issueNew', Session.get('asset'), (err, res) =>{
      if(err) {
        Session.set('error', true);
      }
      else {
        Session.set('error', false);
      }
    });
    Meteor.call('getNew', (err, result) => {
      console.log(err);
      console.log("My result:", result);
      Session.set('q', result);
    });
    Session.set('submit', false);
  },
  'click .list-group'(input) {
    console.log("yeet");
    const target = input.target;
    const text = target.value;
    console.log(text);
    Session.set('z', text);
    Meteor.call('geoLocate', text, (err, result) => {
      console.log("Error", err);
      if(!err) {
        console.log(err);
        console.log("My result:", result.lat, result.lng);
        Session.set('location', result.geometry.location);
      }
    });
  },
  'submit .transactasset'(val) {
    val.preventDefault();
    const target = val.target;
    const number = target.user.selectedIndex;
    const text = target.user.options[number].text;
    console.log(text);
    Meteor.call('transact', Session.get('z'), text, (err, res) => {
      console.log(res);
    });
  }
});

```

JavaScript-Server

```
import { Meteor } from 'meteor/meteor';
import { Promise } from 'meteor/promise';
import { Accounts } from 'meteor/accounts-base';
import { Mongo } from 'meteor/mongo';

const connection = {
  port: 4772,
  host: '127.0.0.1',
  user: "multichainrpc",
  pass: "Bot53WZ4bJ1iaDxUHyjtuUyq5MEtdMYdFKM8HtoAqic"
};

const googleMapsClient = require('@google/maps').createClient({
  key: "AIzaSyDK_WEGH2MK6KUwOXWj5_GKNk7PI7Rg1yc",
  Promise: Promise
});
Accounts.onCreateUser((options, user) => {
  const new_address = Promise.await(multichain.getNewAddress());
  //console.log(new_address);
  user.multichain_address = new_address;
  return user;
})
const multichain = require('multichain-node')(connection);
Meteor.methods({
  getNew() {
    //console.log(Meteor.user());
    if(!Meteor.userId()) {
      throw new Meteor.Error(401, 'you must be logged in!');
    }
    //console.log(Meteor.user().multichain_address);
    const mine = Promise.await(multichain.getAddressBalances({minconf: 0, address: Meteor.user().multichain_address}));
    // console.log(mine);
    return mine;
  },
  issueNew(name) {
    console.log(Meteor.user());
    if(!Meteor.userId()) {
      throw new Meteor.Error(401, 'you must be logged in!');
    }
    const newAsset = Promise.await(multichain.issue({address: Meteor.user().multichain_address, asset: name, qty: 1, units: 1, details: {hello: "world"}}));
    return newAsset;
  },
  transact(assetName, user) {
    //console.log("User", user);
    user_object = Accounts.findUserByUsername(user);
    user_address=user_object.multichain_address;
    //console.log("User", user_address);
    // console.log("Asset", assetName);
    multichain.sendAssetFrom({from: Meteor.user().multichain_address, to: user_address, asset: assetName, qty: 1}, (err, tx) => {
      // console.log(tx);
      // console.log(err);
    })
  },
  geoLocate(assetAddress) {
    const myLatLng = Promise.await(googleMapsClient.geocode({address: assetAddress}).asPromise());
    console.log(myLatLng.json.results[0].geometry.location);
    return myLatLng.json.results[0];
  },
  getAsset(asset) {
    const assetInfo = Promise.await(multichain.getAssetInfo({address: asset}).asPromise());
    return assetInfo;
  }
})
Meteor.startup(() => {
  // code to run on server at startup
});
```

CSS Screenshot:

```
body {
  font-family: "Segoe UI", Arial;
  background: blanchetalmond url("new-york-city-1030778.jpg") no-repeat scroll;
  background-attachment: fixed;

  position: absolute;
  top: 0;
  bottom: 0;
  left: 0;
  right: 0;

  padding: 0;
  margin: 0;
  font-size: 14px;
  color:black;
}
header {
  padding: 20px 15px 15px 15px;

  position: relative;
}
h1 {
  font-size: 1.5em;

  margin: 0;

  margin-bottom: 10px;

  text-align: center;
}
.container {
  max-width: 600px;

  min-height: 100;

  background: blanchetalmond;
  position: absolute;
  left: 50%;
  top: 50%;
  transform: translate(-50%, -50%);
  border-radius: 25px;
  background-color: #FE7860;
}
.paragraph {
  margin-top: 20px;
  margin-bottom: 20px;
}
select {
  width: 100%;
  margin-bottom: 20px;
  border: none;
  border-radius: 4px;
  background-color: #f1f1f1;
}
input[type=text] {
  width: 100%;
  margin: 8px 0;
  margin-top: 20px;
  padding: 12px 20px;
  border: none;
  border-radius: 4px;
  background-color: #f1f1f1;
}
```

```
ul {
  margin: 0;

  padding: 0;

  background: white;
}
#login-buttons {
  text-align: center;
  margin-top: 70;
  margin-bottom: 70;
  display: block;
}
.map-container {
  margin-left: auto;
  margin-right: auto;
  margin-top: 20px;
  width: 300px;
  max-width: 100%;
  height: 300px;
}
.list-group {
  min-height: 200px;
  height: 200px;
  padding-left: 35px;
  margin-bottom: 12px;
  max-height: auto;
  overflow: hidden;
  overflow-y: scroll;
}
.form-radio
{
  -webkit-appearance: none;
  -moz-appearance: none;
  appearance: none;
  display: inline-block;
  position: relative;
  background-color: #f1f1f1;
  color: #666;
  top: 3px;
  right: 5px;
  height: 20px;
  width: 20px;
  border: 0;
  border-radius: 20px;
  cursor: pointer;
  margin-right: 7px;
  outline: none;
  padding-right: 15px;
}
.form-radio:checked::before
{
  position: absolute;
  font: 13px/1 'Open Sans', sans-serif;
  left: 6px;
  top: 3px;
  content: '\02143';
  transform: rotate(40deg);
}
.form-radio:hover
{
  background-color: #f7f7f7;
}
.form-radio:checked
{
  background-color: #f1f1f1;
}
```


Github Screenshot

<> Code 🔔 Issues 0 🔗 Pull requests 1 📁 Projects 0 📊 Insights ⚙️ Settings


Google maps integ #1 Edit






Merged sidsrivastava001 merged 5 commits into `master` from `GoogleMapsInteg` 43 seconds ago


🗨️ Conversation 0 🔗 Commits 5 📄 Checks 0 📄 Files changed 5 +203 -25

 sidsrivastava001 commented a minute ago

Merging google maps integration

 sidsrivastava001 added some commits 20 minutes ago

-  `client\main.css` 9406c09
-  `client\main.html` 3142758
-  `client\main.js` 5a84a95
-  `public\new-york-city-1030778.jpg` 96b0c01
-  `server\main.js` c11c730

 sidsrivastava001 merged commit `05a8eca` into `master` 43 seconds ago Revert

Reviewers ⚙️

No reviews

Assignees ⚙️

No one—assign yourself

Labels ⚙️

None yet

Projects ⚙️

None yet











Milestone ⚙️

No milestone

Notifications

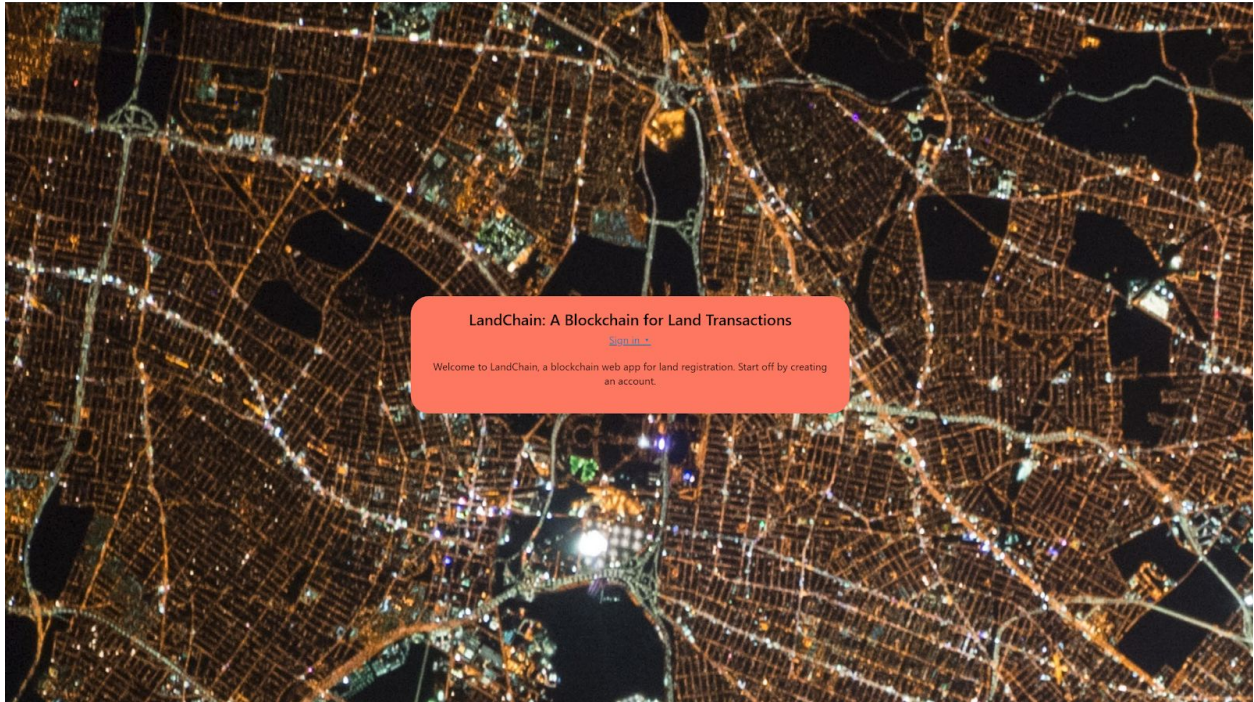
🔔 Unsubscribe

You're receiving notifications because you modified the open/close state.

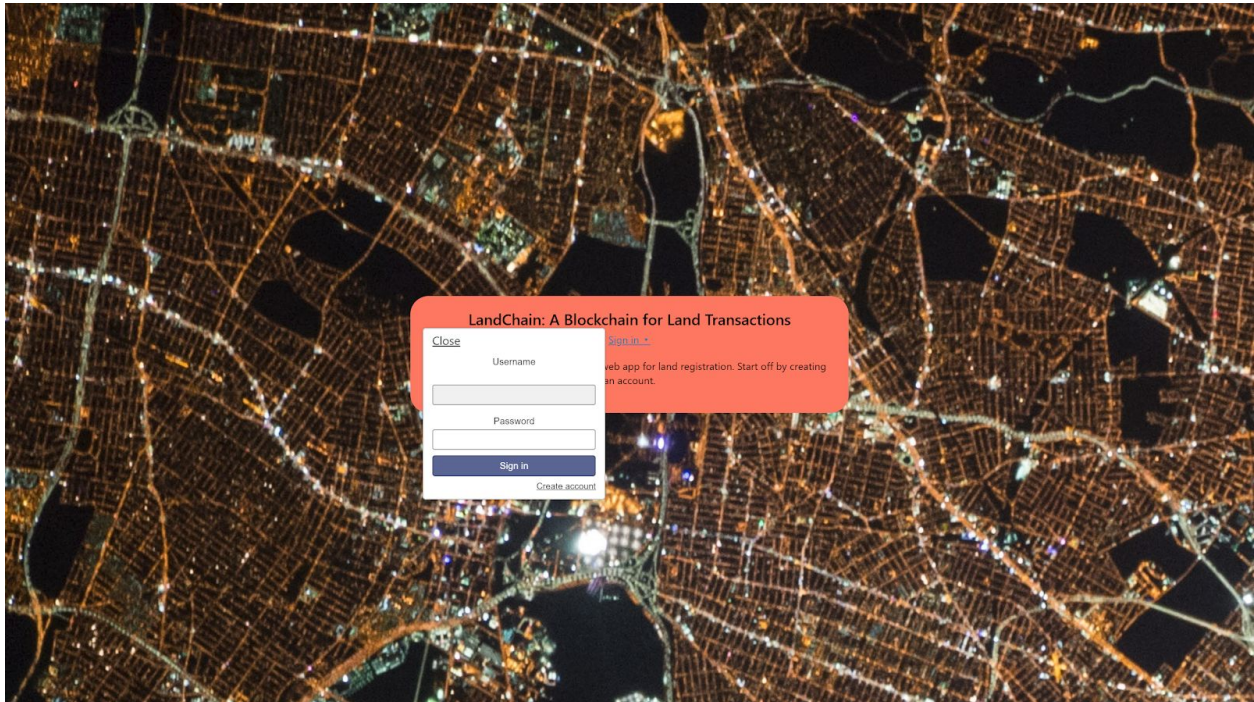
 Write Preview **AA B i**         

End User Documentation

1. Open up the web app url.

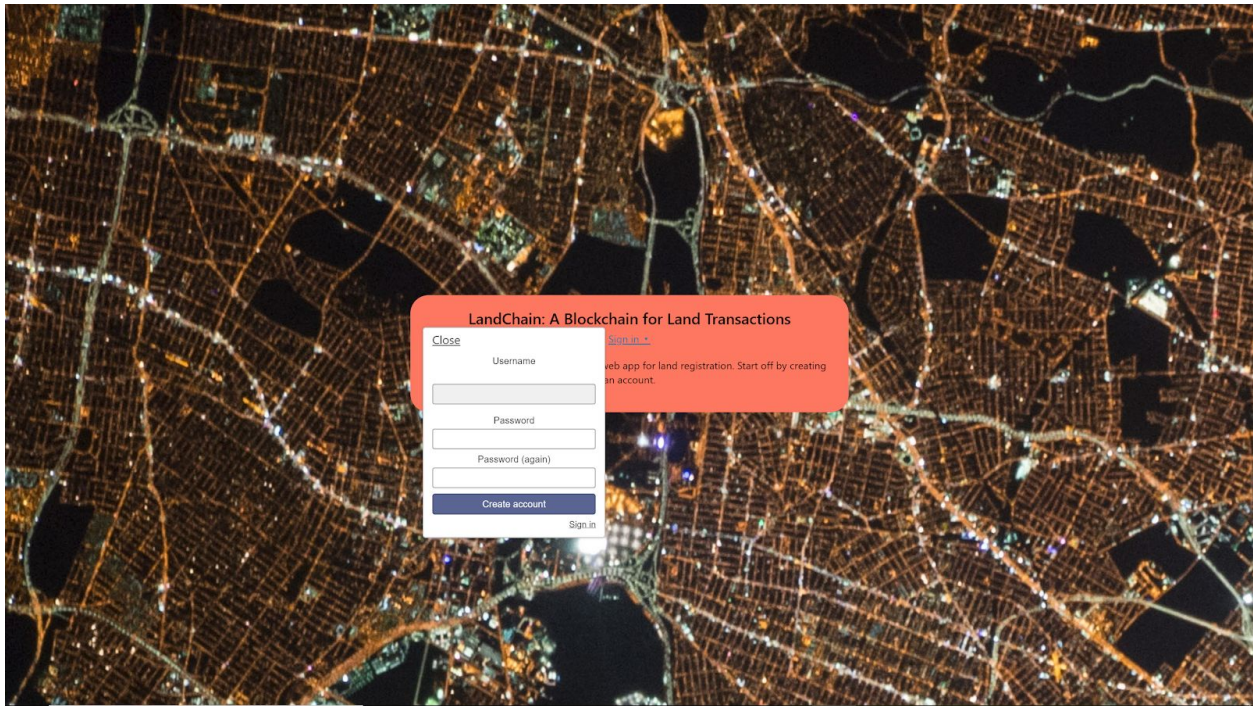


2. Click the Sign In button.



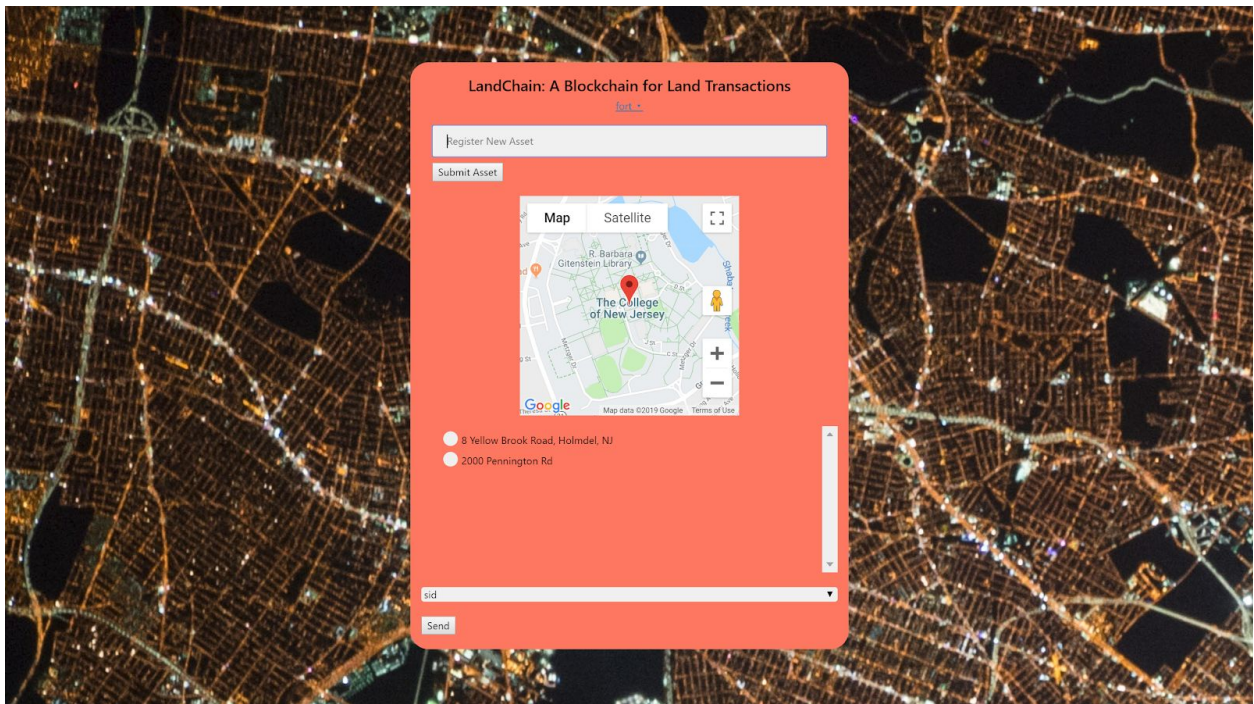
3. If your account exists, sign in using your credentials.
4. If not, click the create account button.

a. Enter a username and password.

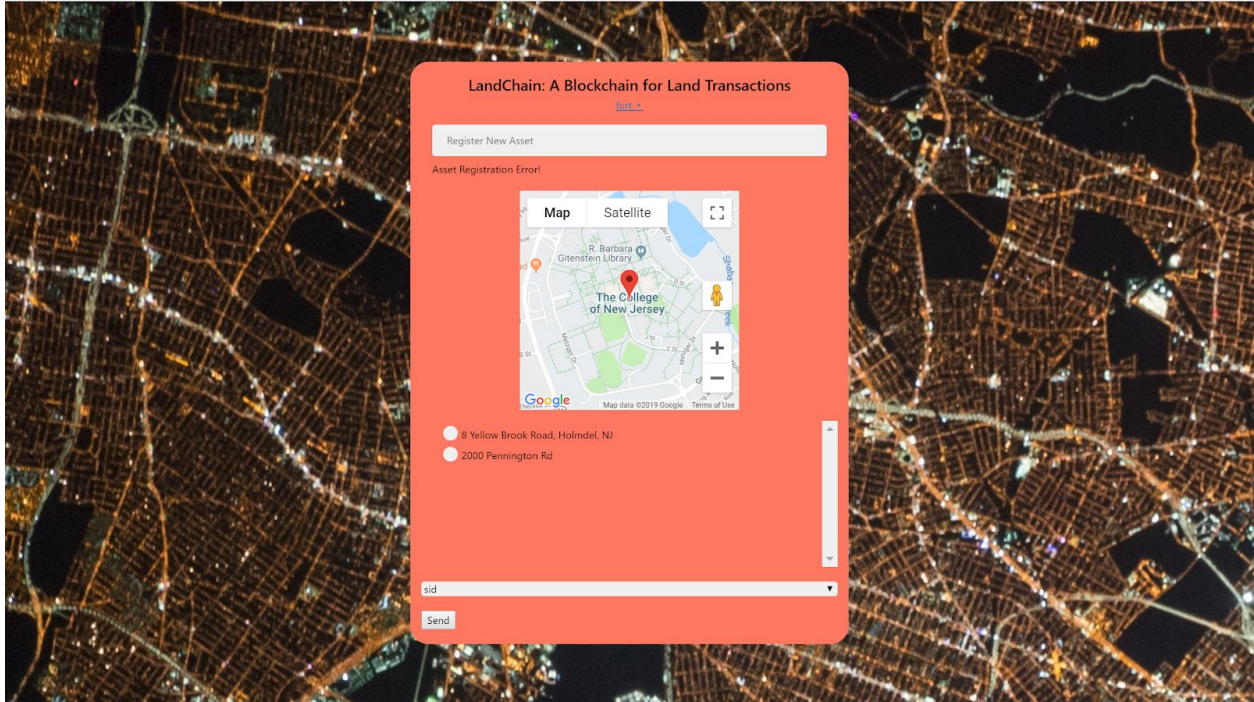


5. To register an asset, enter in an address in your country's format.

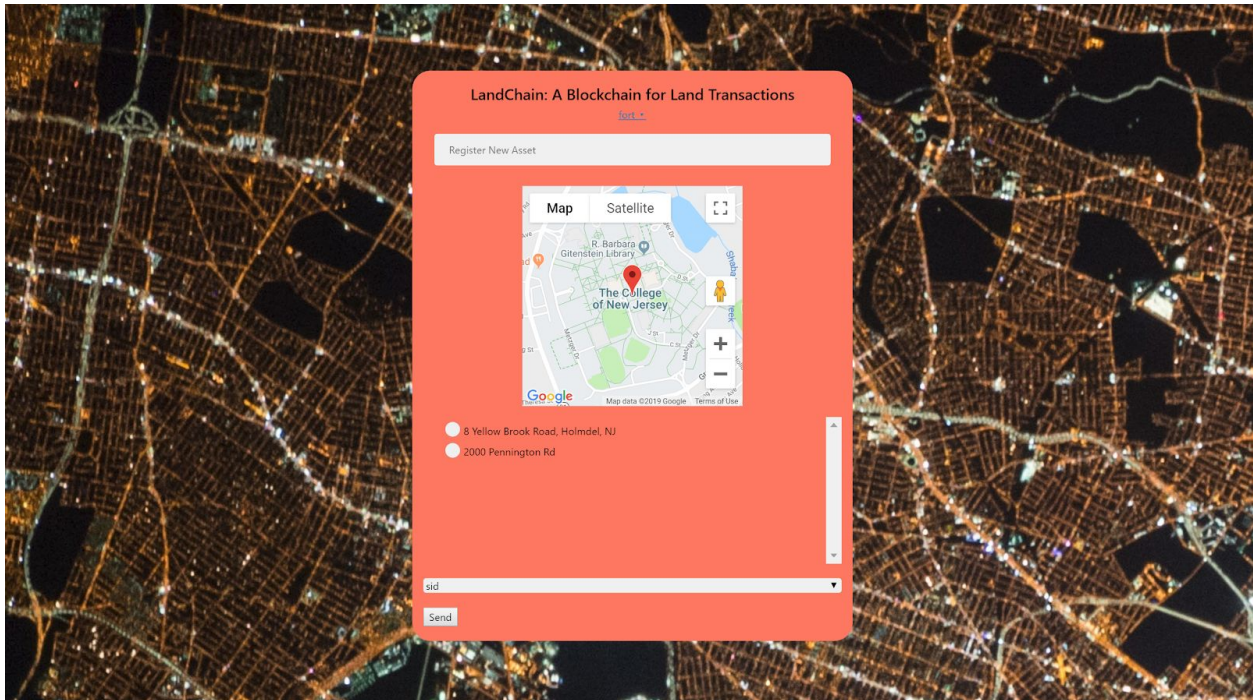
a. If the asset exists, a submit asset button will appear. Click that button.



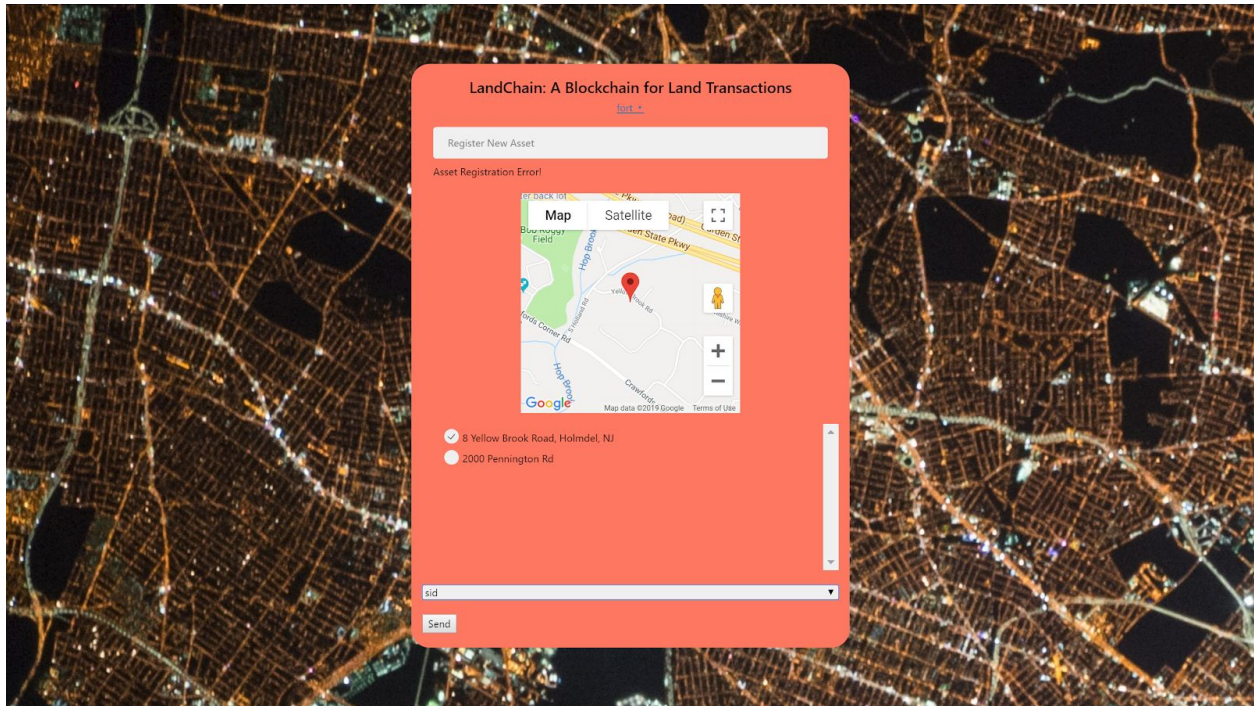
b. If asset registration fails, an error message will be displayed.



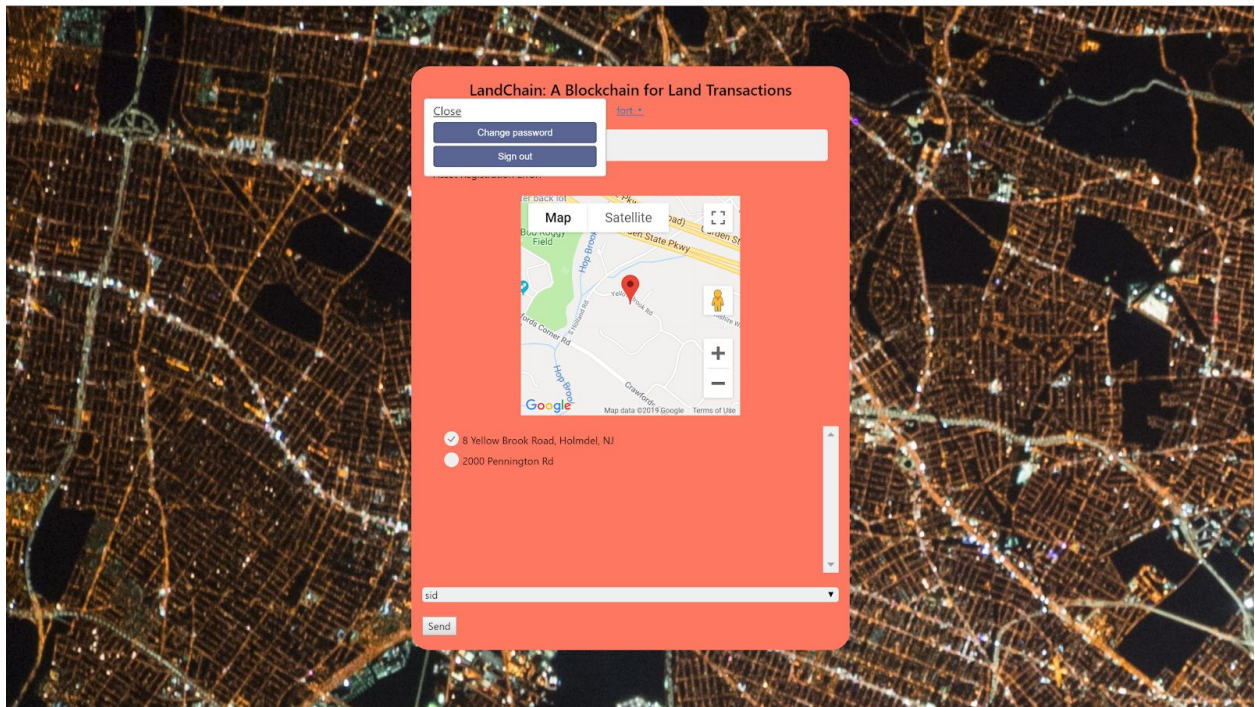
6. To see your registered assets, navigate the list of assets under the map.



7. To send an asset, use the radio buttons on the side of the assets to select an asset to send.



- a. Click the drop-down menu of users and select a user.
 - b. Click the send button.
8. To log out, click the user button and click Logout.



Self Evaluation and Future Prospects

2084165 - Although there were many components of our work that can be improved upon, I feel general satisfaction with what we accomplished in the past few months. Our team managed time well, and worked consistently since early December. All members were able to actively participate in multiple components of the development of our project.

2084164 - During the brainstorming process of our project, many unspoken expectations were set forth. At first I was unsure of how well our actual project would eventually live up to my visions. Now, looking back, I am very pleased with how well it turned out. Not only were we able to have our original goals completed, but we were able to consider future improvements as well. Our team as a whole worked really hard in the past few months to fulfill our duties and make the project a success.

2084163 - Personally, I contributed a great amount of work to this project, and feel gratified to see the fruition of my efforts. The amount of work I put into this project justifies my expectations for the rest of the team, and I am happy to announce that the others were also able to do their part and show results. I am very proud of what I have accomplished these past few months, and how well I was able to communicate with the rest of the team to get it done.

2084162 - This was a project that required lots of cooperation among ourselves. We gave up numerous lunch periods and hours over the weekend in order to ensure that this would be a success. Although this proved difficult at first, it was well worth the sacrifice. This was a great learning experience for us all; not only was I able to gain more knowledge in terms of software development, but I was also able to grow as a person. I acquired insight into core values, and what it truly means to work with others.

2084161 - What a test of time management and work ethic - I was pushed harder than almost any other project previously. My typical schedule is very laid back, so it was definitely a change. However, I do believe it was for the good; perhaps this is what dedication feels like. In conclusion, it was nice to be able to motivate myself and dedicate effort towards a single project for once. TSA brought me out of the mindless cycle I was once trapped in, awakening the long forgotten emotions of joy and pride in my work once again.

2084160 - Finally, we are nearing the end of the tunnel. I can see the light, and hope to soon bask in the glory of our creation. This TSA project was a fun experience, and I wish to try it again in the future, maybe with the same team. I feel that our team worked really well together, and completed our project successfully.

Future Prospects: Although our project did rise to our expectations, there were several aspects of the project that could have been improved or built upon. Firstly, our land registration labeling utilizes known addresses via Google Maps's API. However, this exempts unknown plots of land from being labelled, and in order to better this project, we would need to implement a four-coordinate system of registration. It would also be beneficial to public and private key functionality. Furthermore, modifying the blockchain parameters to give users more options would also be useful.

Works Cited

- Flows, Capital. "Property Rights Of The Poor Need To Be Recognized In Developing Countries." *Forbes*, Forbes Magazine, 8 Jan. 2015, www.forbes.com/sites/realspin/2015/01/08/property-rights-of-the-poor-need-to-be-recognized-in-developing-countries/#26d8f28e4cf2.
- "Getting Started with MultiChain." *Open Source Blockchain Platform*, www.multichain.com/getting-started/.
- "Land." *World Bank*, www.worldbank.org/en/topic/land.
- "Meteor." *Meteor Blog RSS*, www.meteor.com/tutorials/blaze/creating-an-app.
- "MultiChain JSON-RPC API Commands." *Open Source Blockchain Platform*, www.multichain.com/developers/json-rpc-api/.
- Soto, Hernando de. *The Mystery of Capital: Why Capitalism Triumphs in the West and Fails Everywhere Else*. CDE, 2001.
- Tapscott, Don. "How the Blockchain Is Changing Money and Business." *TED*, www.ted.com/talks/don_tapscott_how_the_blockchain_is_changing_money_and_business?language=en.