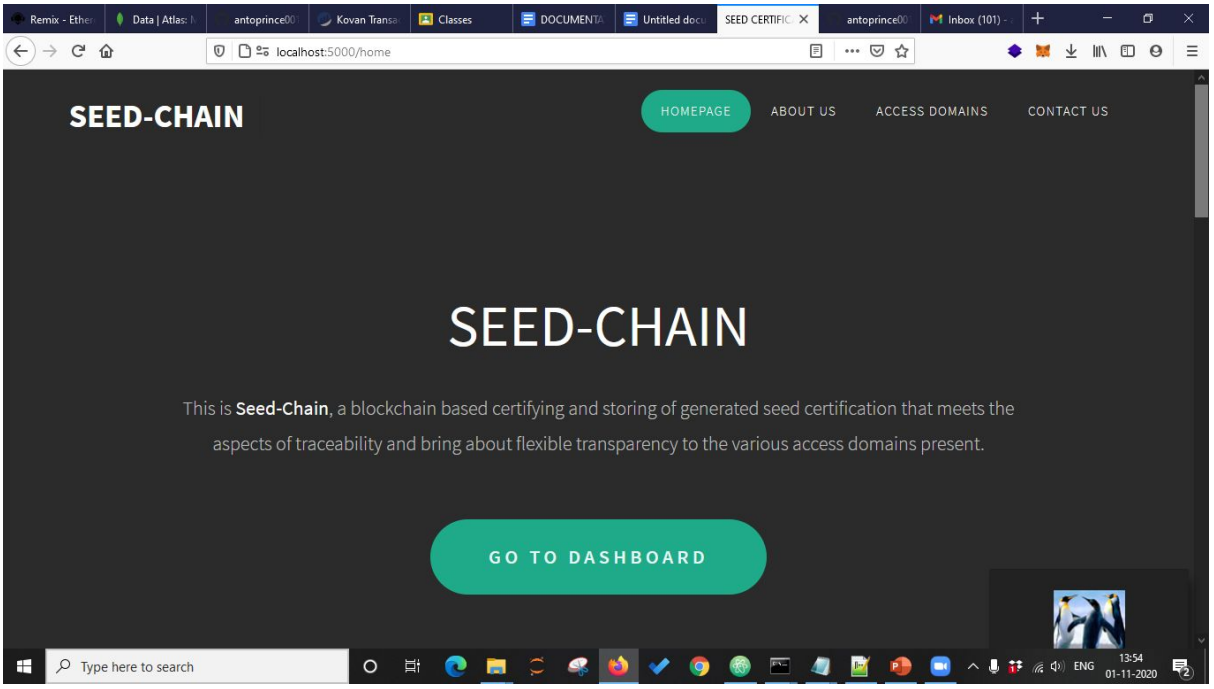
SEED CHAIN - Blockchain based seed supply chain system

Seed supply chain is a complex ecosystem involving various stakeholders. Seed quality issues have demanded a need for an effective traceability solution. Several important criteria such as origin of the seed, stages in production, conformance to quality standards such as genetic purity, germination rate etc. are to be validated.



PROPOSED SYSTEM:

The proposal aims to pilot the seed supply chain using block-chain to track its entire supply movement– from seed aggregators, distributors, retailers to farmers. The move to implement block chain is aimed at bringing transparency, authenticity and stopping spurious and low-quality seeds from entering the market.

Our project can be found at the git link:

https://github.com/antoprince001/seedCertification

INSTALLATION PROCEDURE:

Create a basic react application . Install the create-react-app package by running the following command

npm i -g create-react-app

use create-react-app package to create a new react application.

create-react-app seedChain

Install the required dependencies such as express, jwt, multer.

The project holds both the client application (front end) and the server application (backend), there will be node modules in two different places. First run npm install from the root. Then run npm run-script install-all from the root.

OVERVIEW OF FILE STRUCTURE

Frontend - Holds the client application

- public - This holds all of our static files
- src
    - assets - This folder holds assets such as images, docs, and fonts
    - components - This folder holds all of the different components that will make up our views
    - views - These represent a unique page on the website i.e. Home or About. These are still normal react components.
    - index.js - This is what renders the react app by rendering App.js
    - package.json - Defines npm behaviors and packages for the client

Backend - Holds the server application

- config - This holds our configuration files, like mongoDB uri
- models - This holds all of our data models
- routes - This holds all of our HTTP to URL path associations for each unique url
- tests - This holds all of our server tests that we have defined

- server.js - Defines npm behaviors and packages for the client

package.json - Defines npm behaviors like the scripts defined in the next section of the README

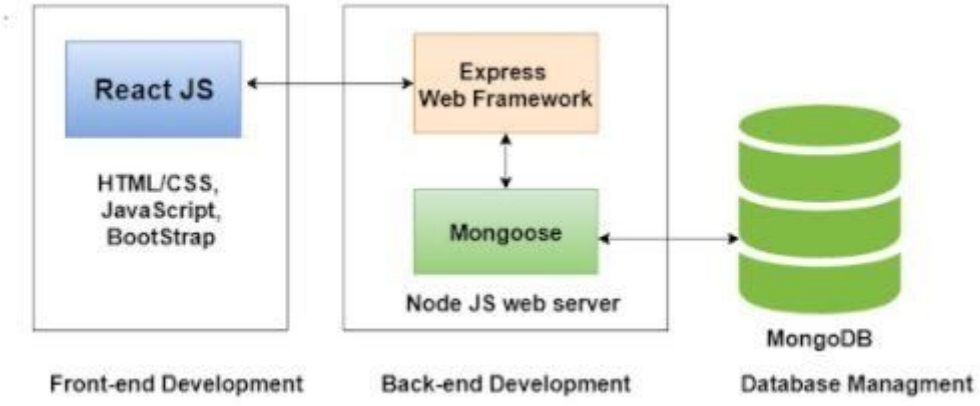
npm run build

Builds the app for production to the build folder.

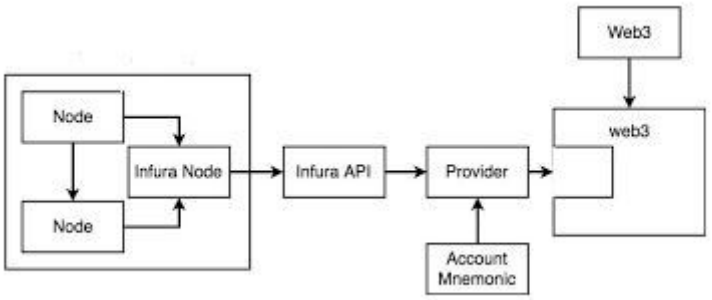
It correctly bundles React in production mode and optimizes the build for the best performance.

DEVELOPMENT PLATFORM:

The MERN stack is obviously used to create an end to end application. React is used to render the front end components. Node js is used to communicate with the backend



BLOCKCHAIN ETHEREUM



Infura is an external api provider that gives access to maintained ethereum nodes.
Each user after registering in infura can opt for an api url that corresponds to their assigned node.
Our implementation is done in Kovan Testnet which is a proof of Authority based network.
Web3.py module is utilized to make the necessary rpc calls for which the module acts as a wrapper.

DATA SCHEMA:

```
LotNumber : {
        type: String,
        required : true,
        unique : true,
        trim : true,
},

owner :{
        type: String,
        required: true
},

crop: {
```

```
        type: String,
        required: true
},

variety : {
        type : String
},

SourceTagNo : {
type : String
},

SourceClass : {
type : String
},

DestinationClass : {
type: String
},

SourceQuantity : {
type: String
},

SourceDateOfIssue : {
type: String
},



spaName: {
type: String
},

dateOfIssue: {
type: String
},
```

```
sourceStorehouse : {
type: String
},

destiStorehouse : {
type: String
},

sgName : {
type: String
},

sgId : {
type: String
},

FinYear : {
type: String
},

Season : {
type: String
},

landRecordsKhatano : {
type: String
},

landRecordsPlotno : {
type : String
},

landRecordsArea : {
type: String
},
```

```
cropRegCode : {
type: String
},

SppName : {
type: String
},

SppID : {
type: String
},


TotalQuantityProcessed : {
type: String
},

processingDate : {
type: String
},

verificationDate: {
type: String
},

SampleSecretCode : {
type: String
},

SamplePassed : {
type: String
},

SampleTestDate : {
type: String
},
```

```
CertificateNo : {
type: String
},

CerticateDate : {
type: String
},

TagSeris : {
type: String
},

TagIssuedRangeFrom : {
type : String
},

TagIssuedRangeTo : {
type: String
},

NoOfTagsIssued : {
type: String
},

CertificateValidityInMonth : {
type: String
}
```