Documentation

Objective

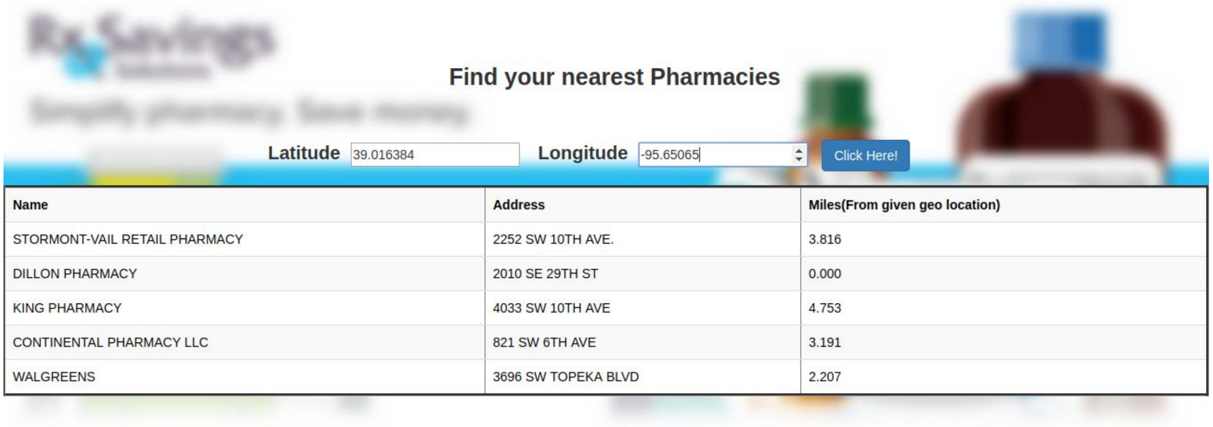
Create a very basic API using tools and languages of your choice. The API will have a single endpoint/method with a single function.

Implemented a Mean Stack Application where user enters the location of latitude, longitude and gets the results of nearest pharmacies to the given location.

Steps to Reproduce for Output

1. Open the folder in any IDE i.e webstorm.
2. Create an account in mlab.After that, create a new collection and import the csv file.The instructions are explained below in the document.
3. Replace all the details of mlab like collection name, database name in the server code.
4. Run the server.js file in it.It will be running at localhost address with 8081 port.
5. Run the html page from the westorm.
6. Give the values in the text fields on the ui.
7. Click on the button click here to get the result.The result will be as follows.

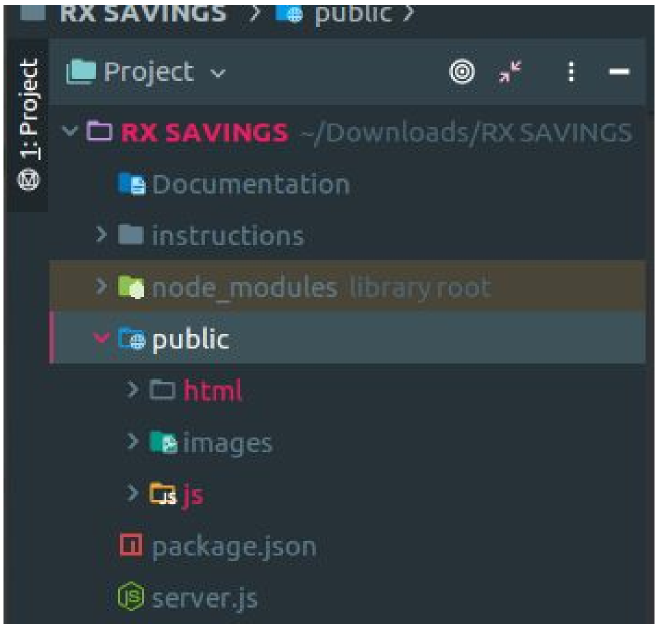
   Output



Name	Address	Miles(From given geo location)
STORMONT-VAIL RETAIL PHARMACY	2252 SW 10TH AVE.	3.816
DILLON PHARMACY	2010 SE 29TH ST	0.000
KING PHARMACY	4033 SW 10TH AVE	4.753
CONTINENTAL PHARMACY LLC	821 SW 6TH AVE	3.191
WALGREENS	3696 SW TOPEKA BLVD	2.207

Implementation

Folder Structure

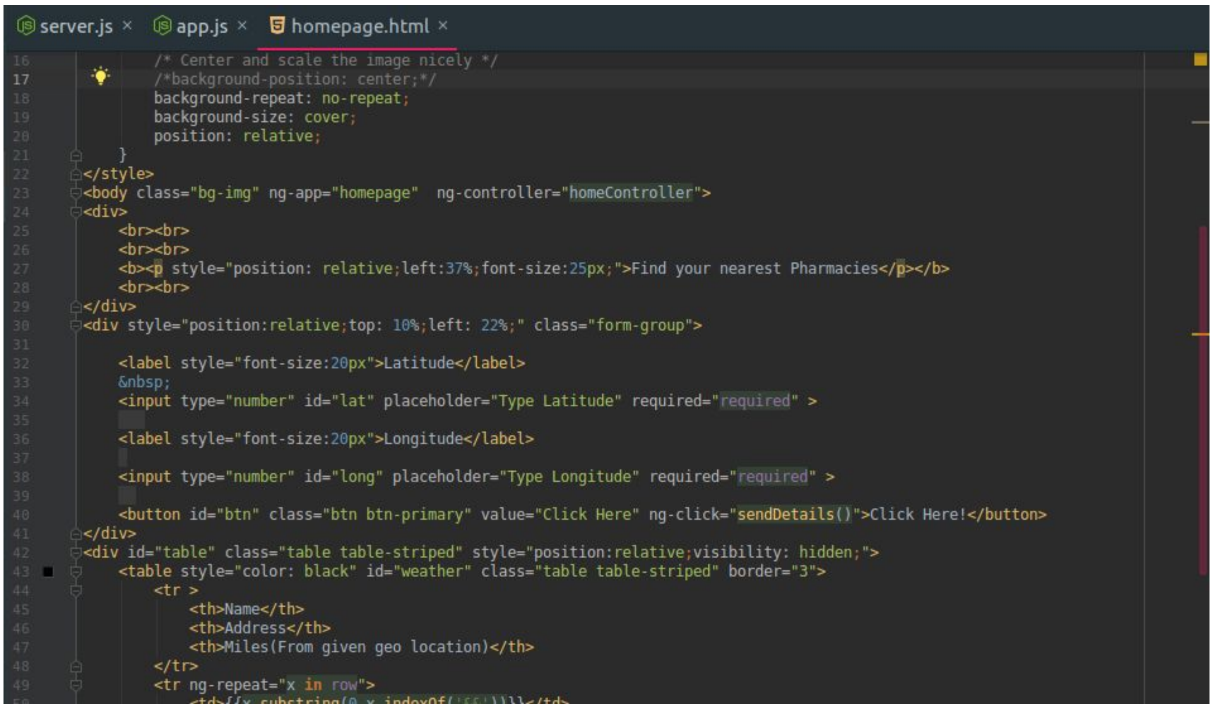


```
RX SAVINGS > public >
Project
RX SAVINGS ~/Downloads/RX SAVINGS
    Documentation
  > instructions
  > node_modules library root
  v public
    > html
    > images
    > js
    package.json
    server.js
```

1. The server code i.e. it has the API implementation and database code.
2. In the node modules package it has all the packages required for the project.
3. In public folder it has 3 sub folders “html, images, js”.
4. All the html files are stored in html folder.
5. The image used for the background is stored in the folder.
6. The controller codefile is stored in the js folder.

Html File

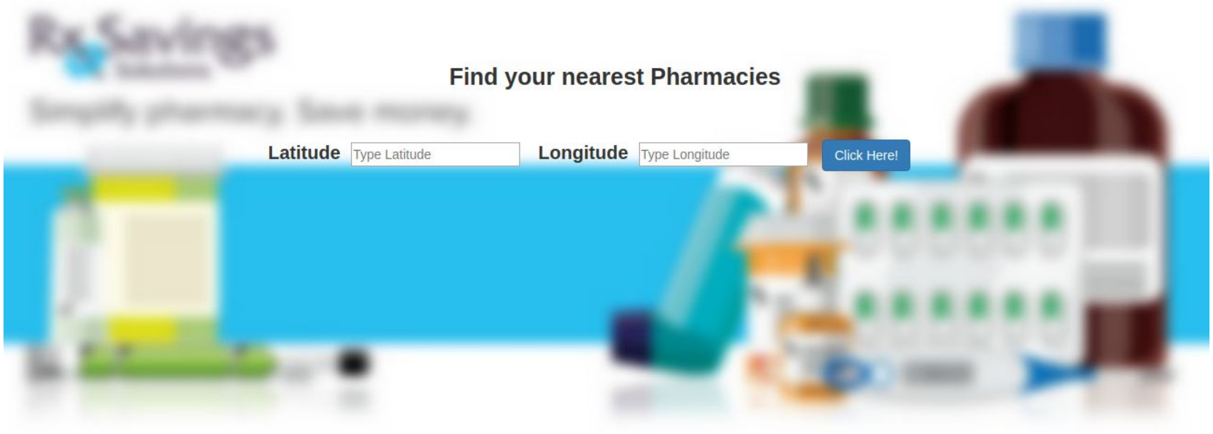
Code Snippet



```
        /* Center and scale the image nicely */
        /*background-position: center;*/
        background-repeat: no-repeat;
        background-size: cover;
        position: relative;
    }
</style>
<body class="bg-img" ng-app="homepage"  ng-controller="homeController">
<div>
    <br><br>
    <br><br>
    <b><p style="position: relative;left:37%;font-size:25px;">Find your nearest Pharmacies</p></b>
    <br><br>
</div>
<div style="position:relative;top: 10%;left: 22%;" class="form-group">

    <label style="font-size:20px">Latitude</label>
    &nbsp;
    <input type="number" id="lat" placeholder="Type Latitude" required="required" >

    <label style="font-size:20px">Longitude</label>

    <input type="number" id="long" placeholder="Type Longitude" required="required" >

    <button id="btn" class="btn btn-primary" value="Click Here" ng-click="sendDetails()">Click Here!</button>
</div>
<div id="table" class="table table-striped" style="position:relative;visibility: hidden;">
    <table style="color: black" id="weather" class="table table-striped" border="3">
        <tr >
            <th>Name</th>
            <th>Address</th>
            <th>Miles(From given geo location)</th>
        </tr>
        <tr ng-repeat="x in row">
```

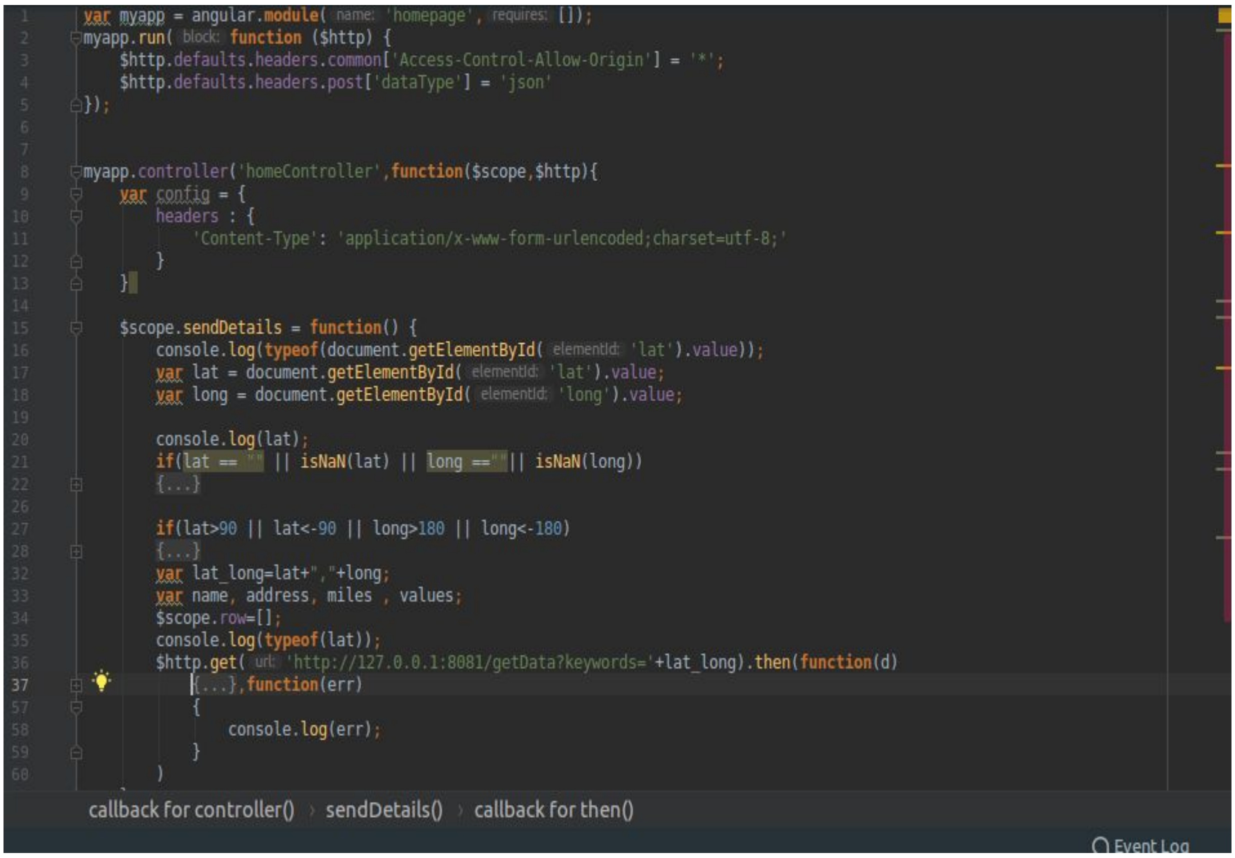
1. Here the code is written in Html5, the code has 2 input which takes only numbers, and a button where on clicking you will be getting the result.
2. The html page is designed with the help of Html,CSS, Bootstrap.

Output



Controller

Code snippet



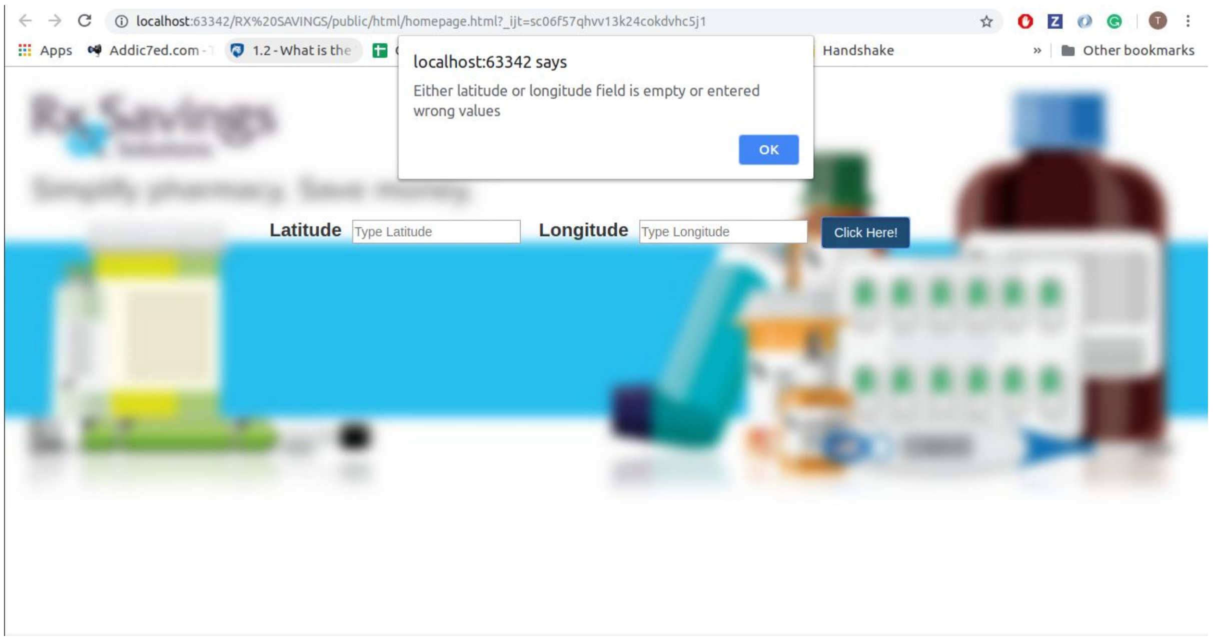
```
var myapp = angular.module( name: 'homepage', requires: []);
myapp.run( block: function ($http) {
    $http.defaults.headers.common['Access-Control-Allow-Origin'] = '*';
    $http.defaults.headers.post['dataType'] = 'json'
});


myapp.controller('homeController',function($scope,$http){
    var config = {
        headers : {
            'Content-Type': 'application/x-www-form-urlencoded;charset=utf-8;'
        }
    }

    $scope.sendDetails = function() {
        console.log(typeof(document.getElementById( elementId: 'lat').value));
        var lat = document.getElementById( elementId: 'lat').value;
        var long = document.getElementById( elementId: 'long').value;

        console.log(lat);
        if(lat == "" || isNaN(lat) || long ==""|| isNaN(long))
        {...}

        if(lat>90 || lat<-90 || long>180 || long<-180)
        {...}
        var lat_long=lat+","+long;
        var name, address, miles , values;
        $scope.row=[];
        console.log(typeof(lat));
        $http.get( url: 'http://127.0.0.1:8081/getData?keywords='+lat_long).then(function(d)
            {...},function(err)
            {
                console.log(err);
            }
        )
```

callback for controller() › sendDetails() › callback for then()

Event Log

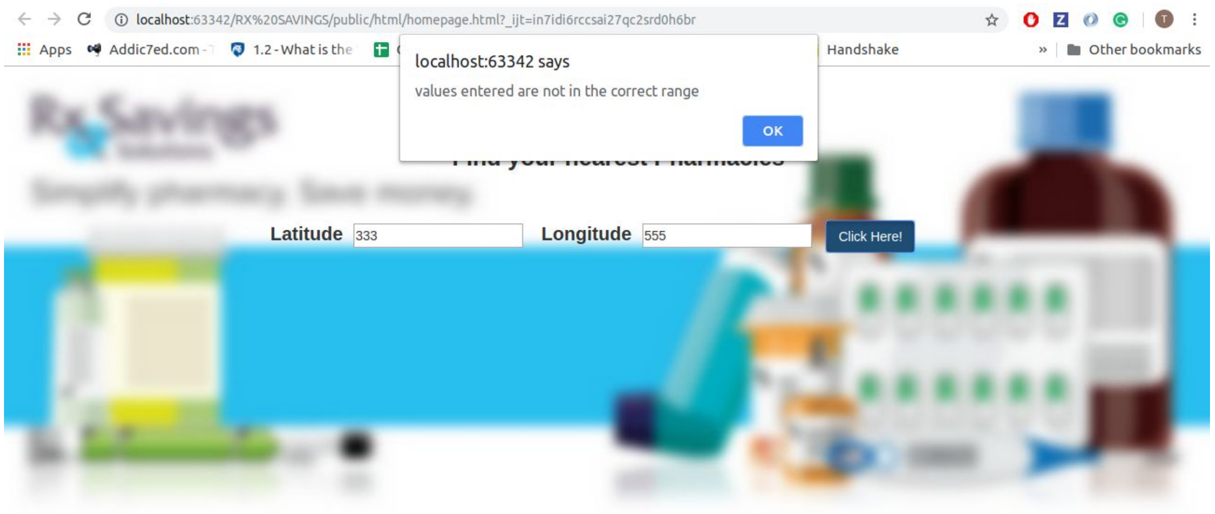
1. Here in this controller the sendDetails method is triggered on the button click.
2. We will take the latitude and longitude from the UI which is entered by the user.
3. Validations are done on the values like, the values should not be null or the text box.
4. The values of latitude and longitude should be in the given range of -90 to 90 and -180 to 180.
5. The Values are sent to the server and fetching of the nearest pharmacies and miles calculation is done in server.

Outputs


localhost:63342 says

Either latitude or longitude field is empty or entered wrong values

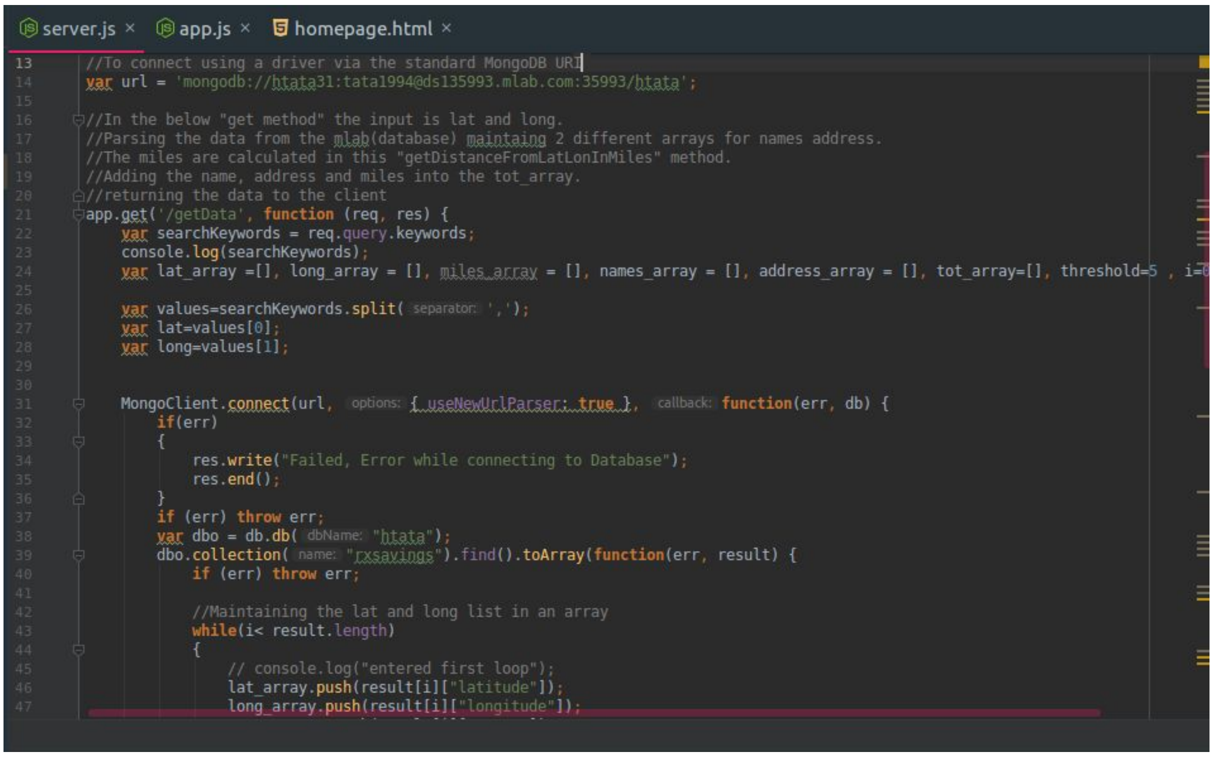
When the values are entered are empty, then the alert is popped up.


localhost:63342 says

values entered are not in the correct range

When the values are entered out of range, then this alert is popped up.

Server

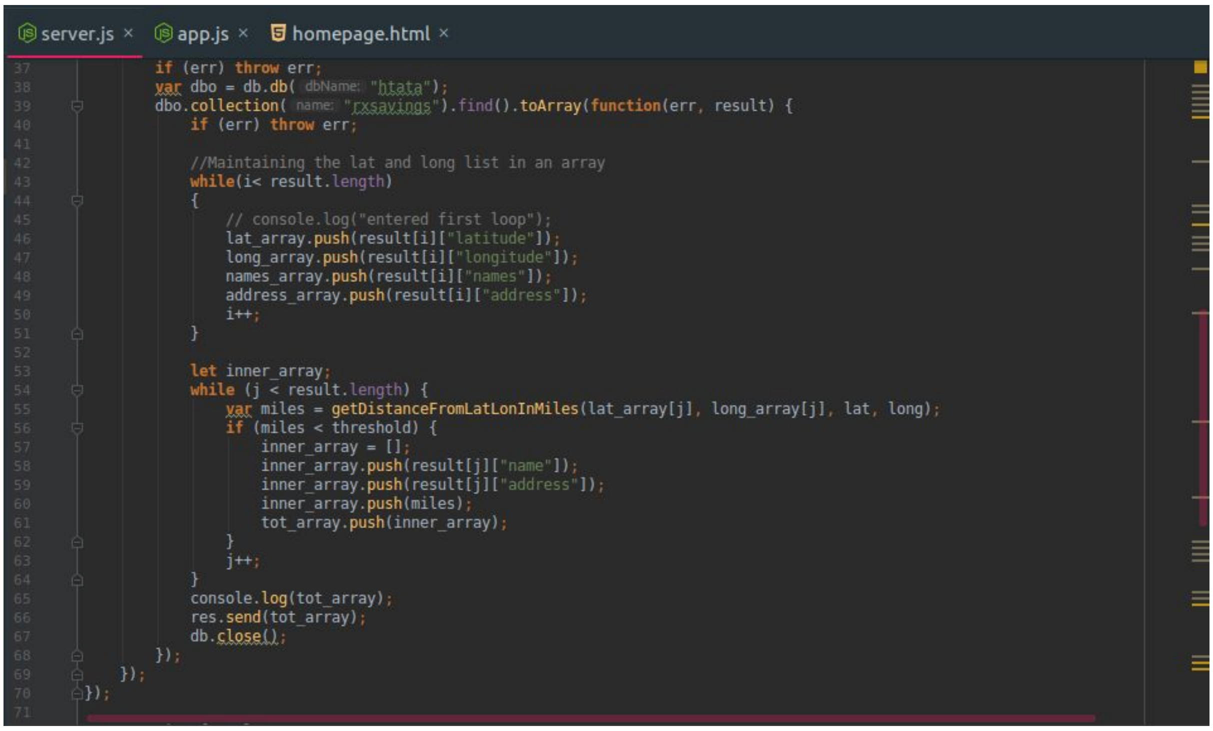


server.js × app.js × homepage.html ×

```
//To connect using a driver via the standard MongoDB URI
var url = 'mongodb://htata31:tata1994@ds135993.mlab.com:35993/htata';

//In the below "get method" the input is lat and long.
//Parsing the data from the mlab(database) maintaing 2 different arrays for names address.
//The miles are calculated in this "getDistanceFromLatLonInMiles" method.
//Adding the name, address and miles into the tot_array.
//returning the data to the client
app.get('/getData', function (req, res) {
    var searchKeywords = req.query.keywords;
    console.log(searchKeywords);
    var lat_array =[], long_array = [], miles_array = [], names_array = [], address_array = [], tot_array=[], threshold=5 , i=

    var values=searchKeywords.split( separator: ',');
    var lat=values[0];
    var long=values[1];


    MongoClient.connect(url,  options: { useNewUrlParser: true },  callback: function(err, db) {
        if(err)
        {
            res.write("Failed, Error while connecting to Database");
            res.end();
        }
        if (err) throw err;
        var dbo = db.db( dbName: "htata");
        dbo.collection( name: "rxsavings").find().toArray(function(err, result) {
            if (err) throw err;

            //Maintaining the lat and long list in an array
            while(i< result.length)
            {
                // console.log("entered first loop");
                lat_array.push(result[i]["latitude"]);
                long_array.push(result[i]["longitude"]);
```

1. First keep your dataset i.e. “csv file” in mlab.It is a Database as a service for Mongodb.
2. Import the csv file into mlab with the following command “`mongoimport -h <host> -d <database> -c <collection name> -u <user> -p <password> --file <input .csv file> --type csv --headerline.`”.

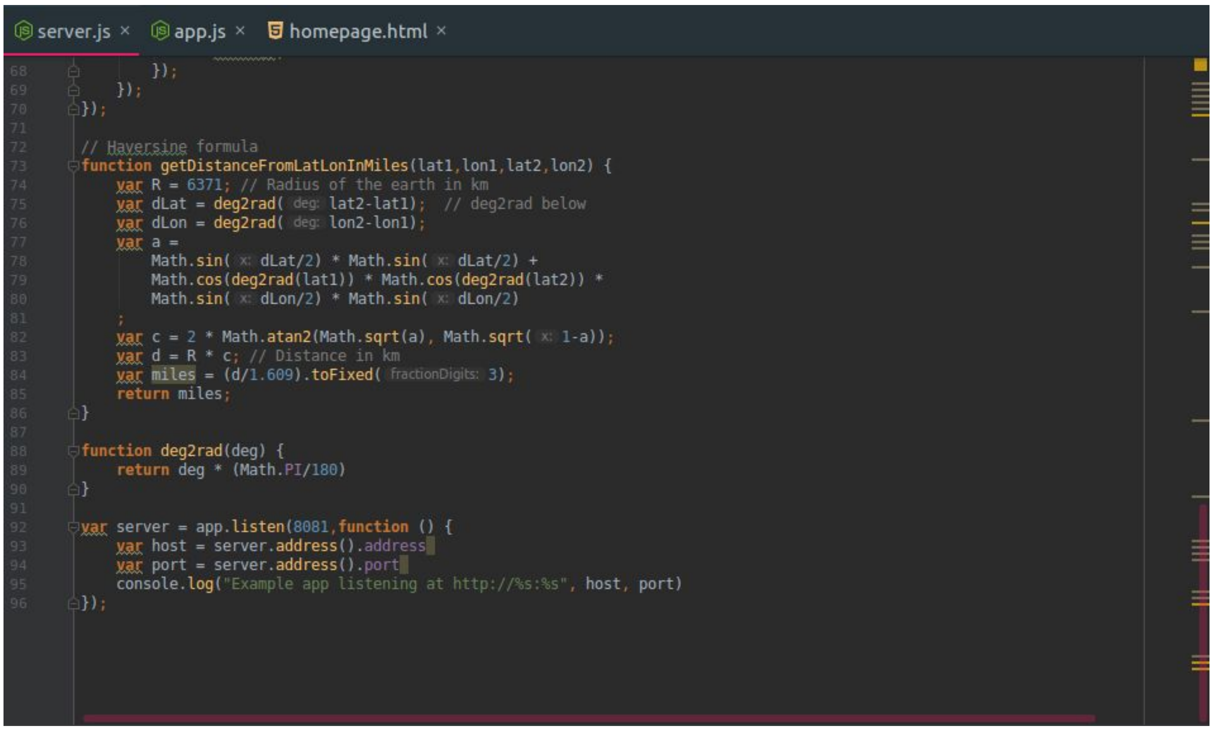
3. Above in the code accessing the collection “rxsavings” in the database “htata” an fetching all the documents in it.



server.js × app.js × homepage.html ×

```
            if (err) throw err;
            var dbo = db.db( dbName: "htata");
            dbo.collection( name: "rxsavings").find().toArray(function(err, result) {
                if (err) throw err;

                //Maintaining the lat and long list in an array
                while(i< result.length)
                {
                    // console.log("entered first loop");
                    lat_array.push(result[i]["latitude"]);
                    long_array.push(result[i]["longitude"]);
                    names_array.push(result[i]["names"]);
                    address_array.push(result[i]["address"]);
                    i++;
                }

                let inner_array;
                while (j < result.length) {
                    var miles = getDistanceFromLatLonInMiles(lat_array[j], long_array[j], lat, long);
                    if (miles < threshold) {
                        inner_array = [];
                        inner_array.push(result[j]["name"]);
                        inner_array.push(result[j]["address"]);
                        inner_array.push(miles);
                        tot_array.push(inner_array);
                    }
                    j++;
                }
                console.log(tot_array);
                res.send(tot_array);
                db.close();
            });
        });
});
```

4. Here in the first loop I am getting all the latitude, longitude , names and address from the database and storing it into different arrays.
5. The input lat- long values and the lat-long values from the database are sent to the “getDistanceFromLatLonInMiles” method which calculates the distance between the 2 lat-long points.



server.js × app.js × homepage.html ×

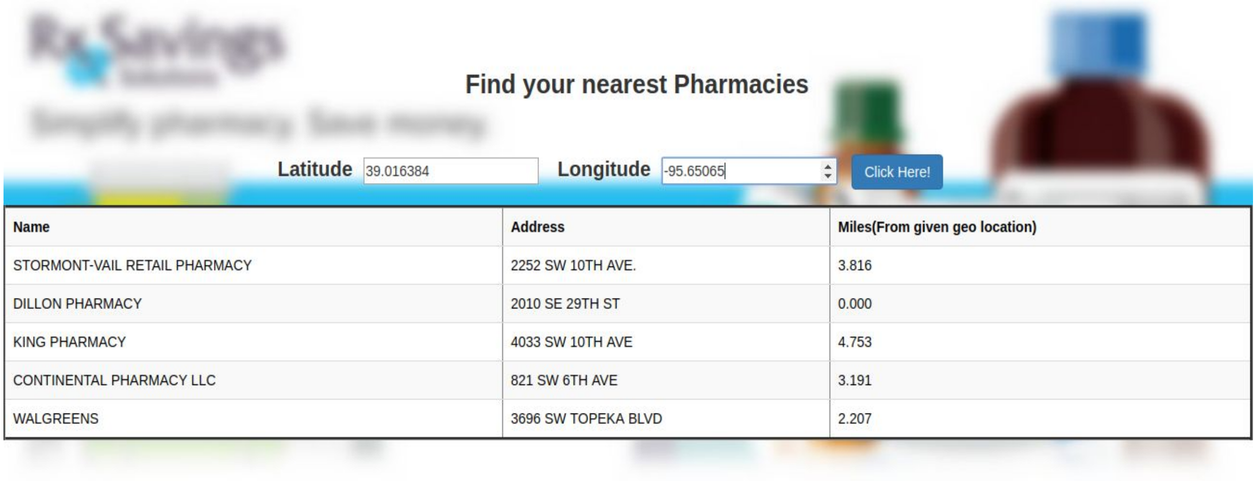
```
            });
        });
});

// Haversine formula
function getDistanceFromLatLonInMiles(lat1,lon1,lat2,lon2) {
    var R = 6371; // Radius of the earth in km
    var dLat = deg2rad( deg: lat2-lat1);  // deg2rad below
    var dLon = deg2rad( deg: lon2-lon1);
    var a =
        Math.sin( x: dLat/2) * Math.sin( x: dLat/2) +
        Math.cos(deg2rad(lat1)) * Math.cos(deg2rad(lat2)) *
        Math.sin( x: dLon/2) * Math.sin( x: dLon/2)
    ;
    var c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt( x: 1-a));
    var d = R * c; // Distance in km
    var miles = (d/1.609).toFixed( fractionDigits: 3);
    return miles;
}

function deg2rad(deg) {
    return deg * (Math.PI/180)
}

var server = app.listen(8081,function () {
    var host = server.address().address
    var port = server.address().port
    console.log("Example app listening at http://%s:%s", host, port)
});
```

6. The above algorithm logic is “Haversine Formula” which gives the straight distance between 2 lat-long points.
7. To get the distance according to the travelling by means of car , it can be achieved with apis namely google maps API and matrix API (“https://graphhopper.com/api/1/docs/matrix/”).
8. A threshold is initialised with the value 5.
9. The miles which is the output of the first loop is sent to the second loop where if the value is less than the threshold then the miles, name, address are stored in an array and sent to the client.
10. In the controller these array is parsed and sent in the format as the html understands.

Output

With the given lat and long position it will display the nearest pharmacies which are available in the database which displays

1. Name
2. Address
3. Miles



Name	Address	Miles(From given geo location)
STORMONT-VAIL RETAIL PHARMACY	2252 SW 10TH AVE.	3.816
DILLON PHARMACY	2010 SE 29TH ST	0.000
KING PHARMACY	4033 SW 10TH AVE	4.753
CONTINENTAL PHARMACY LLC	821 SW 6TH AVE	3.191
WALGREENS	3696 SW TOPEKA BLVD	2.207

Note: This application can be deployed into cloud with the help of “heroku”.

References

1.https://stackoverflow.com/questions/27928/calculate-distance-between-two-latitude-longitude-points-haversine-formula

2.https://docs.mlab.com/
3.https://nodejs.org/en/docs/