CS 5412 - Assignment 1: Real-Time Restaurant Geolocation Service with Node.js

Name: Prashanth Basappa NetID: pb476

A simple real-time web application that will determine and show the locations of the user's input address directly on a map along with the nearest restaurants extracted from a large CSV file located within a particular radius. For this purpose I am using Node.js and MongoDB for the server side and the HTML5 for front end. Node.js is an asynchronous web server which is built on the Google V8 JavaScript engine and is a perfect solution as a back-end for real-time apps. My app will let users see addresses on the map with the help of the MongoDB which is queried on the zipcode of the restaurants for a real-time data channel. Our example will work in all modern browsers that support the HTML5.

Installing node

We need to install node.js. We can get pre-compiled Node.js binaries for several platforms from the download section of the official website: http://nodejs.org/download. After the installation is complete you will get access to the node package manager (npm). Also install a utility like nodemon that will keep an eye on your files and you won't need to restart your server after every change:

```
npm install nodemon -g
```

"-g" means that it will be installed globally and accessible from every node repo.
Async is a utility module which provides straight-forward, powerful functions for working with asynchronous JavaScript. Although originally designed for use with Node.js and installable via *`npm install async`*, it can also be used directly in the browser.

This webapp uses:

- Google Maps API
- HTML/CSS
- NodeJS
- MongoDB

Extra Credit(GUI):The HTML

The “index.hbs” is in our src/views directory. For rendering our map on the page we will use map canvas from google. The styles.css file is in public/stylesheets folder.

Server side:

Now we are ready to start with the back-end of our app. The index.js file is in the routes folder. It has the entire coding logic for the assignment. It takes in input for the html page that is either LatLong pair or the user’s address. Then run “nodemon www”.
Now go to **localhost:4000** in your browser.

For geocoding and reverse geocoding I used Google API. I also used the node-geocoder package(https://www.npmjs.com/package/node-geocoder) was done writing a script. Restaurant database: Importing the restaurant csv file to the mongoDB is done by the script which is in README.txt.

Cloud - based resources The Google Maps API is used to:

- Place a personal location marker on the map
- Set the location in terms of latitude and longitude
- Allow a user to input a radius from the user’s input address
- Display the restaurants in that radius.
- Draw a circle to indicate the radius

Cloud based resources used: Google's Geocoding API
https://developers.google.com/maps/documentation/geocoding/

Disadvantages of Google Maps API Web Services:

- Sending too many requests per day.(If you exceed the usage limits you will get an OVER_QUERY_LIMIT status code as a response.)
- Sending requests too fast, i.e. too many requests per second.
- Sending requests too fast for too long or otherwise abusing the web service.
- Exceeding other usage limits, e.g. points per request in the Elevation API.

Design Decisions:

1. I decided to use NodeJS along with MongoDB as NodeJS can be used on both front end and back end. It also has a wrapper around MongoDB.
2. MongoDB was chosen because the geocoding information retrieved from the Google API is stored in JSON format and using the zip code to retrieve the restaurants would be much more faster and powerful. The running time is shown in the queries.txt file.

Potential Enhancements:

1. Displaying the Roadmap distance from the user's address to the nearest restaurant among multiple users in real time.
2. Displaying multiple markers was not implemented and I am currently working on it.
3. Caching the addresses of nearby restaurants and using it for future searches.
4. Using geocoding services that does not have a QUERY_LIMIT and scaling out the distance calculation on multiple servers.