*Project Report On*

**Title\_of\_project**

*Submitted by*

**Saumya Karri (181ME273)**

**Anshuman Sinha(181EE209)**

**Sathvika.B.Mahesh (181CV141)**

**V SEM B.Tech (IT)**

*Under the guidance of*

**Dr. AnandKumar.M and Ms. Priyadarshini.R**

**Dept of IT, NITK Surathkal**

*in partial fulfillment for the award of the degree*

*of*

**Bachelor of Technology**

in

**Information Technology**

at



**Department of Information Technology National Institute of Technology Karnataka, Surathkal**. ***February 2020***

**Department of Information Technology**

**National Institute of Technology Karnataka, Surathkal Mid Semester Evaluation (February 2020)**

*Course code :* IT 254

*Course Title:* Minor Project

*Title of the Project: Navigetter*

*Details of Project Group*

*Name of the Student Register No. Signature with Date*

1. **Saumya Karri (181ME273)**

2. **Anshuman Sinha(181EE209)**

3. **Sathvika.B.Mahesh (181CV141)**

**Name of Project Guide:**

Signature of the Project Guide:

Place:

Date: 1-12-2020

**INDEX**

1)Abstract

2)Algorithm

3)Screenshots

4)Structure of code

5)Future Scope and Conclusion

**Abstract**

The project, Navigetter, aims to build a smart interface to provide the user with options corresponding to his/her category of destination and then leads the easiest way there according to their means of travel. This project employs the well-established open source geo-navigation software named OpenLayers. Built on the NodeJS framework, once registered, the user receives an automated confirmation email and this module is obtained by using Nodemailer. The elaborate frontend deploys snippets of codes in CSS, JavaScript and Bootstrap.

**Motivation**

One of the leading driving forces behind this project idea was to provide the public a completely open-source platform employing interfaces that promote open-source development.

# Algorithm

**Automated email generation after registration**

The application uses an npm package called Nodemailer. We created a dummy mail-id “[navigetter@gmail.com](mailto:navigetter@gmail.com)” from where the mails are sent. A sendMail function is created which is exported to app2.js. The data that has been entered with the form (contact.html) is also imported into app2.js where it is used as req.body. The sendMail function is called which has arguments from req.body. As soon the function is called, an automated mail is sent to the user. After that the user can go to the home page and use the website.

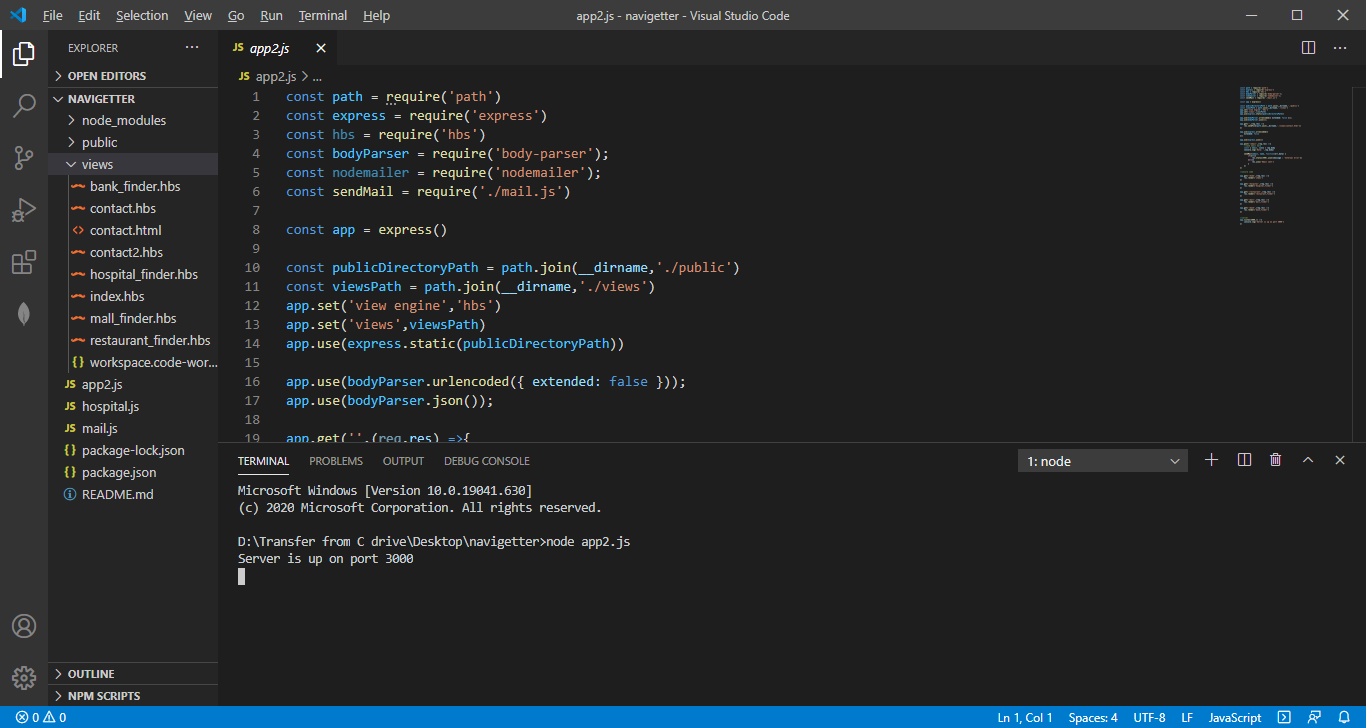
**The geo encoding algorithm in OpenLayers works as follows -**

1. It can detect the user's current location and return the respective geographical coordinates.
2. Using this, we refer to our database and find 50 closest locations (hospitals/restaurants/malls/banks)
3. Once we have information on these detected locations, they can be displayed on the map in and around the current location.
4. On clicking on either of the locations, the app is devised to show the shortest route from the current location to the destination (in km/miles; decided in the code).
5. The user can also choose their means of conveyance (walking/ car/ cycling), according to which the routes may modify as per the optimized route.

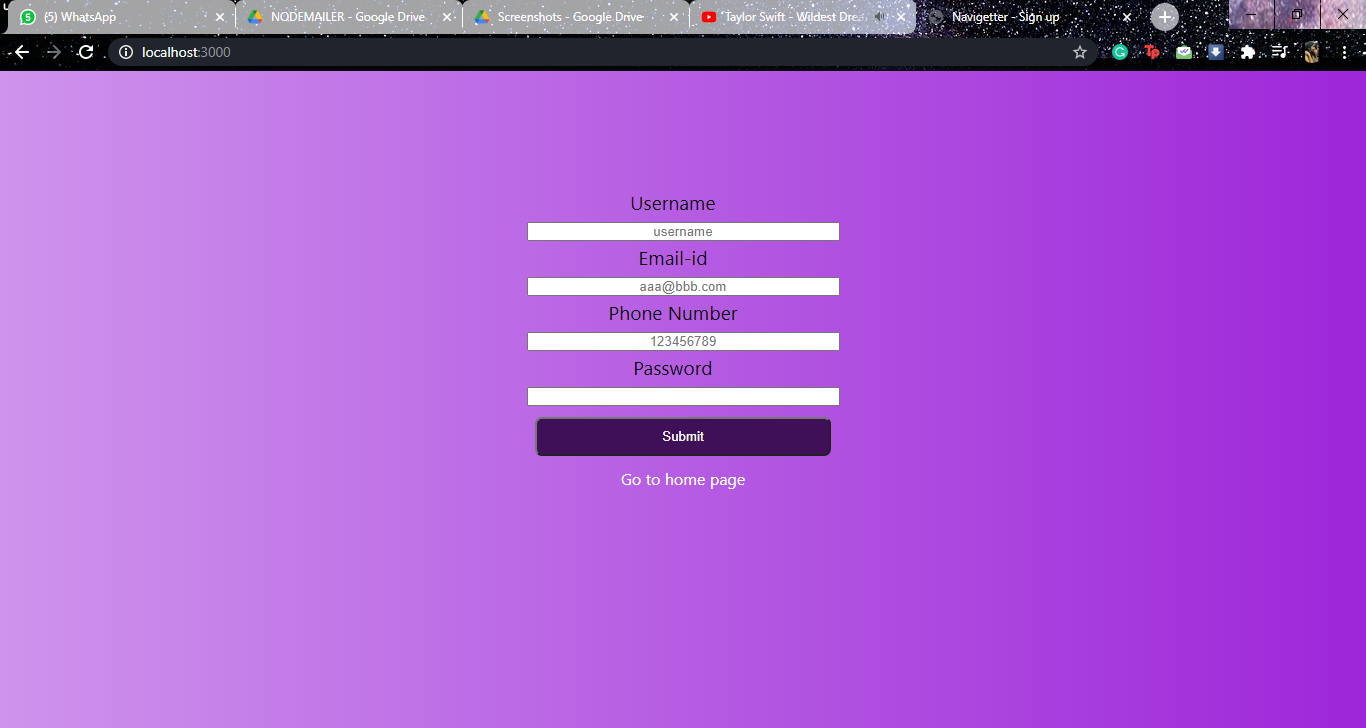
# 

# Screenshots

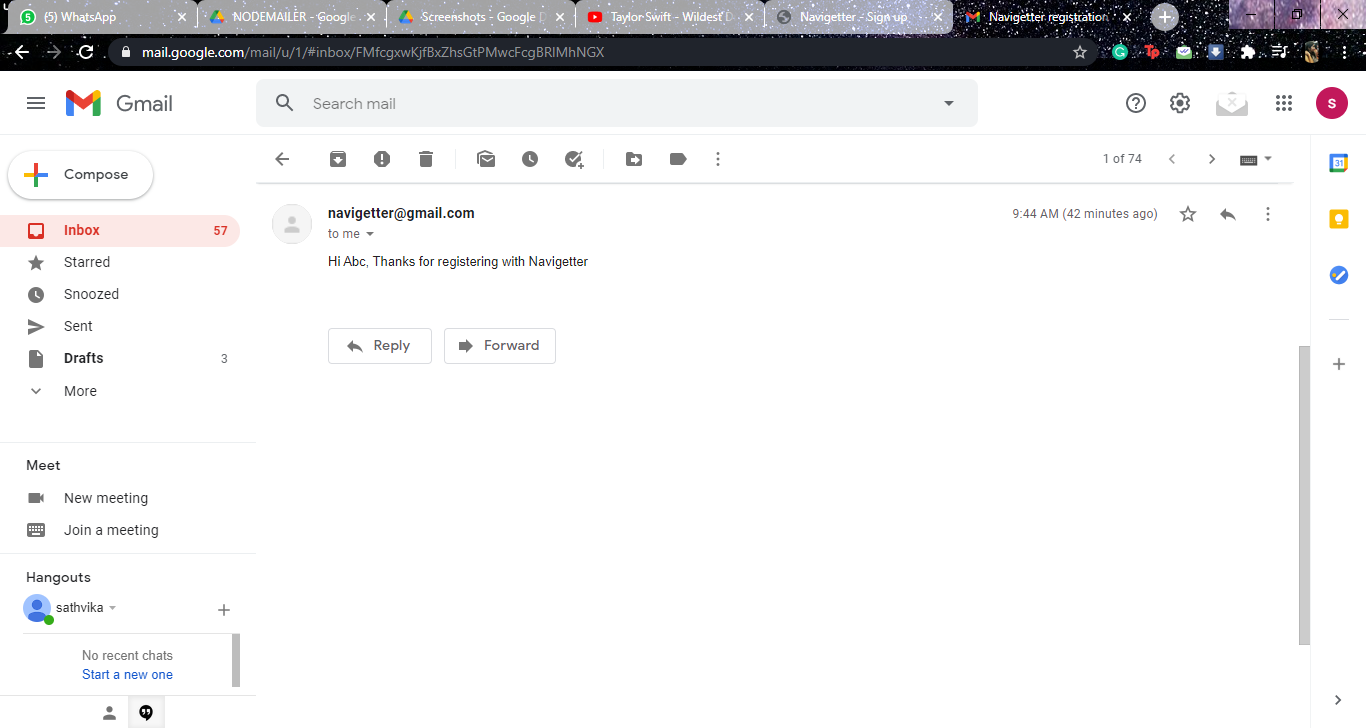
Main file app2.js which runs on localhost:3000

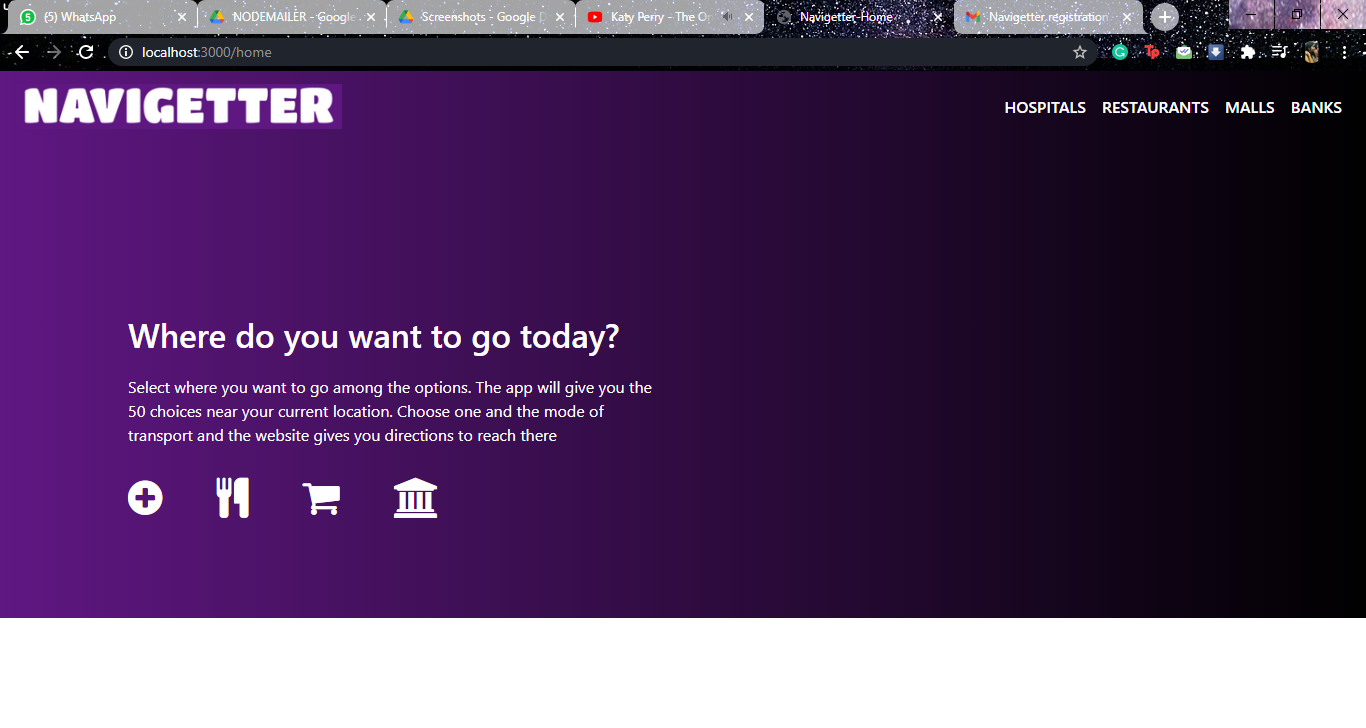


The registration form for the website

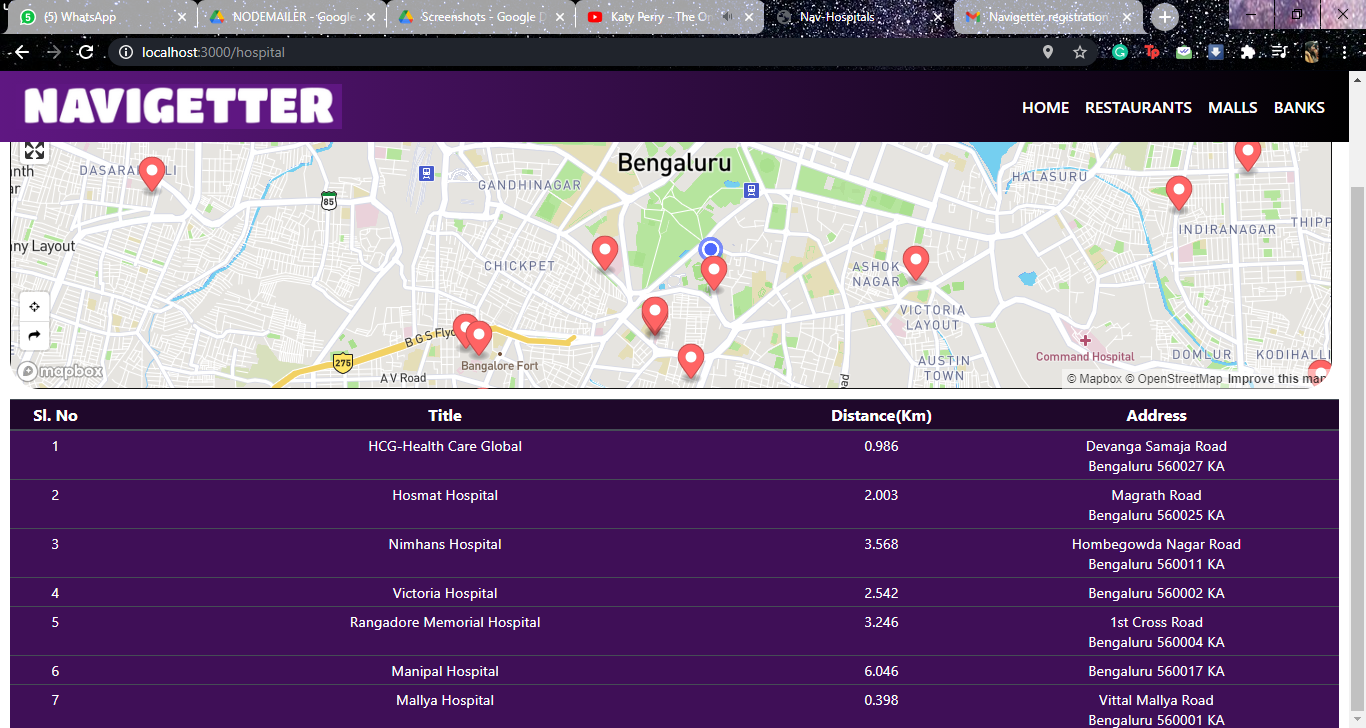


The automated mail that the user receives on the registered mail id

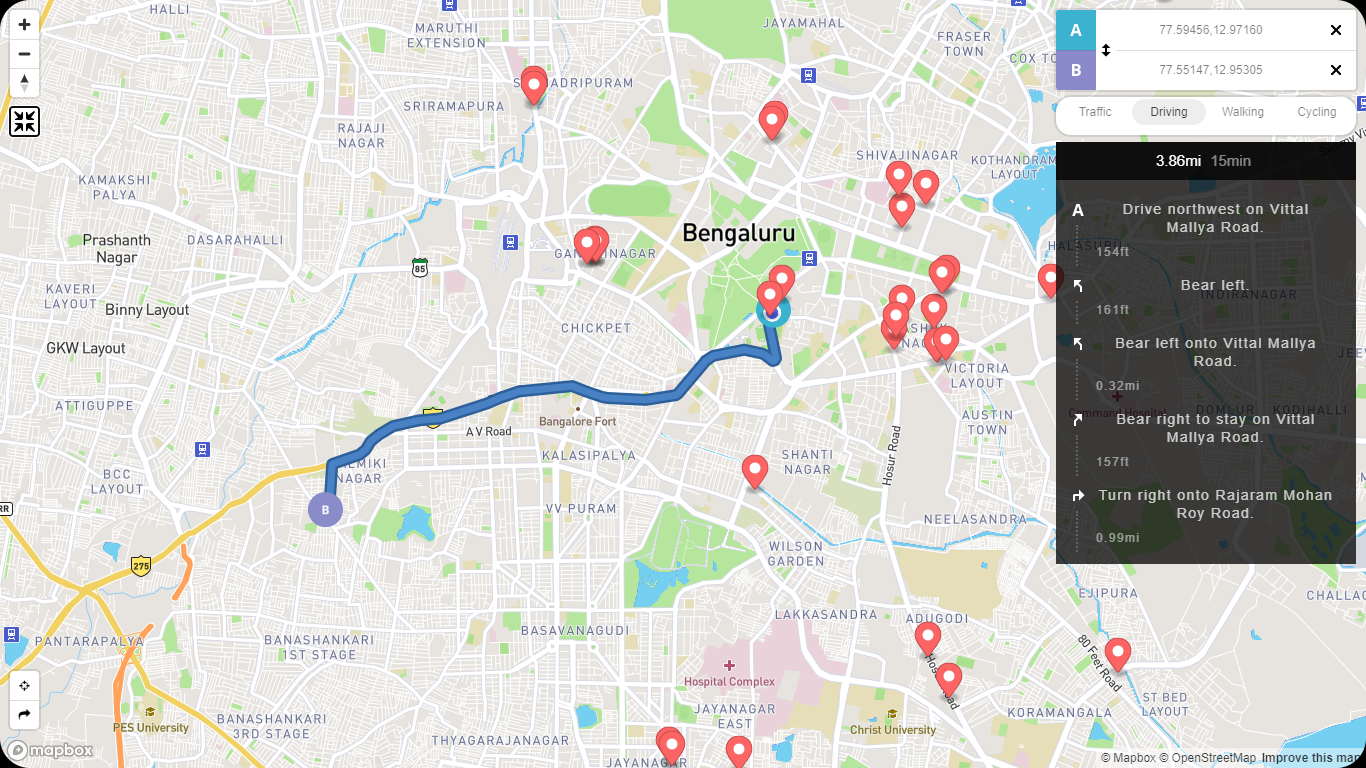


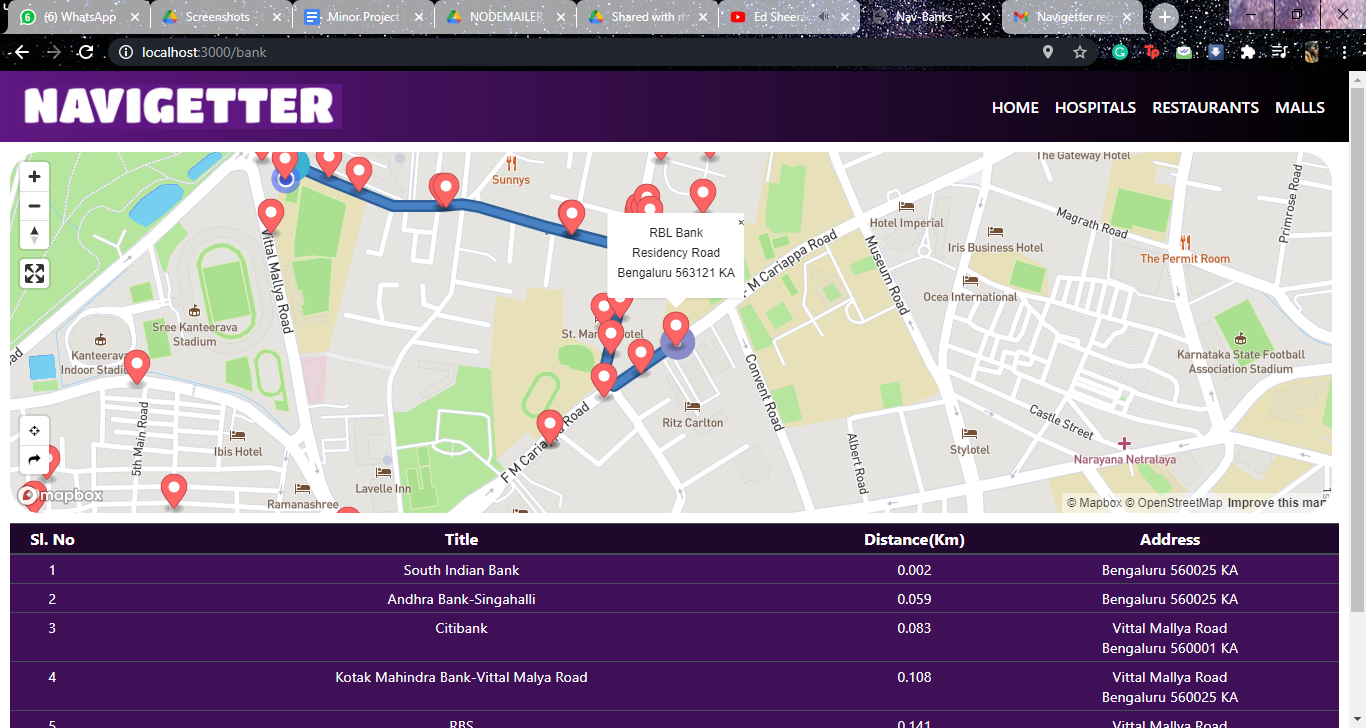
The user can go to the home page after registering. Screenshot of the home page is shown below

The user can choose between hospitals,restaurants,malls and banks. Say he choses hospital. He’ll be redirected to the hospital page where the website will ask for location permission. Once provided, he’ll get the top 50 results as shown



The user is required to choose one hospital. Once that is done, he has to choose the mode of transport. The application shows him the shortest and the most convenient path according to the mode of transport. The user is also given directions for the same.





**Structure of code**

MVC being used

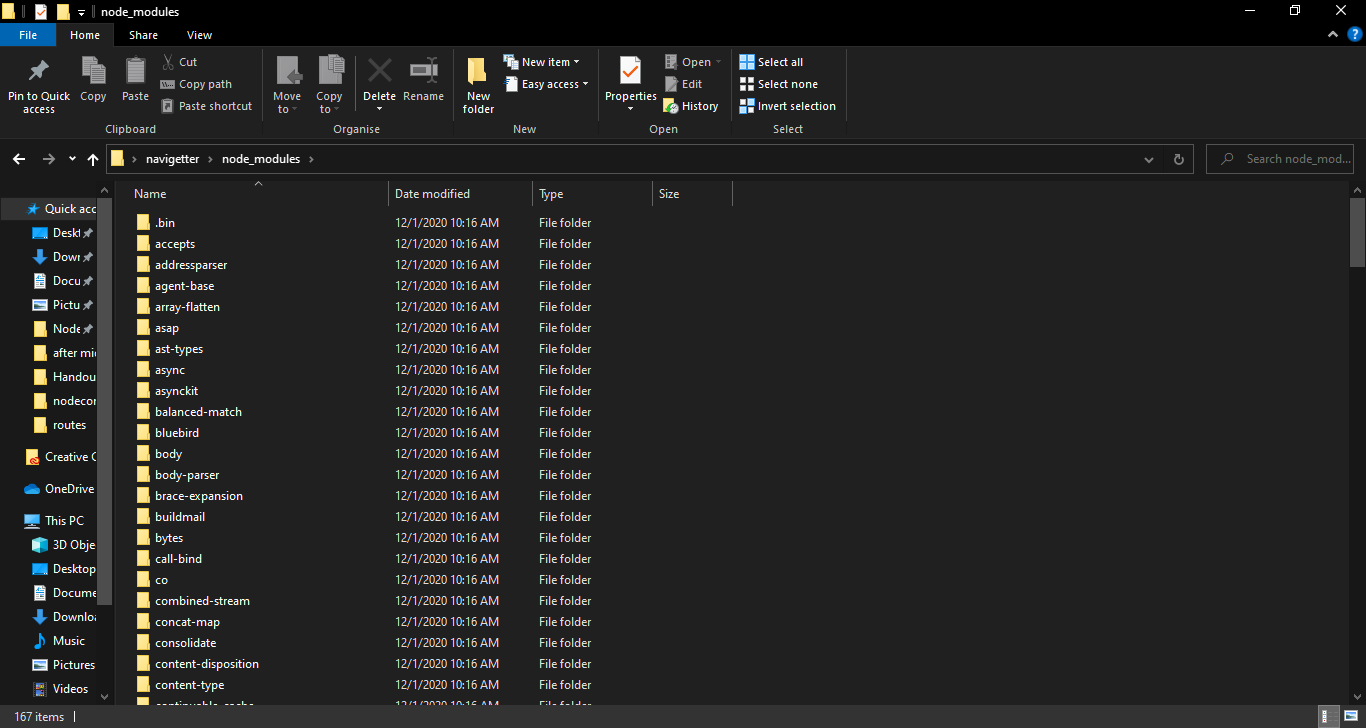
**FRONTEND :Bootstrap,HTML,CSS**

**BACKEND: MapboxAPI**

**ENVIRONMENT: NodeJS**

**Folders**

**1)Node modules and npm packages**

****

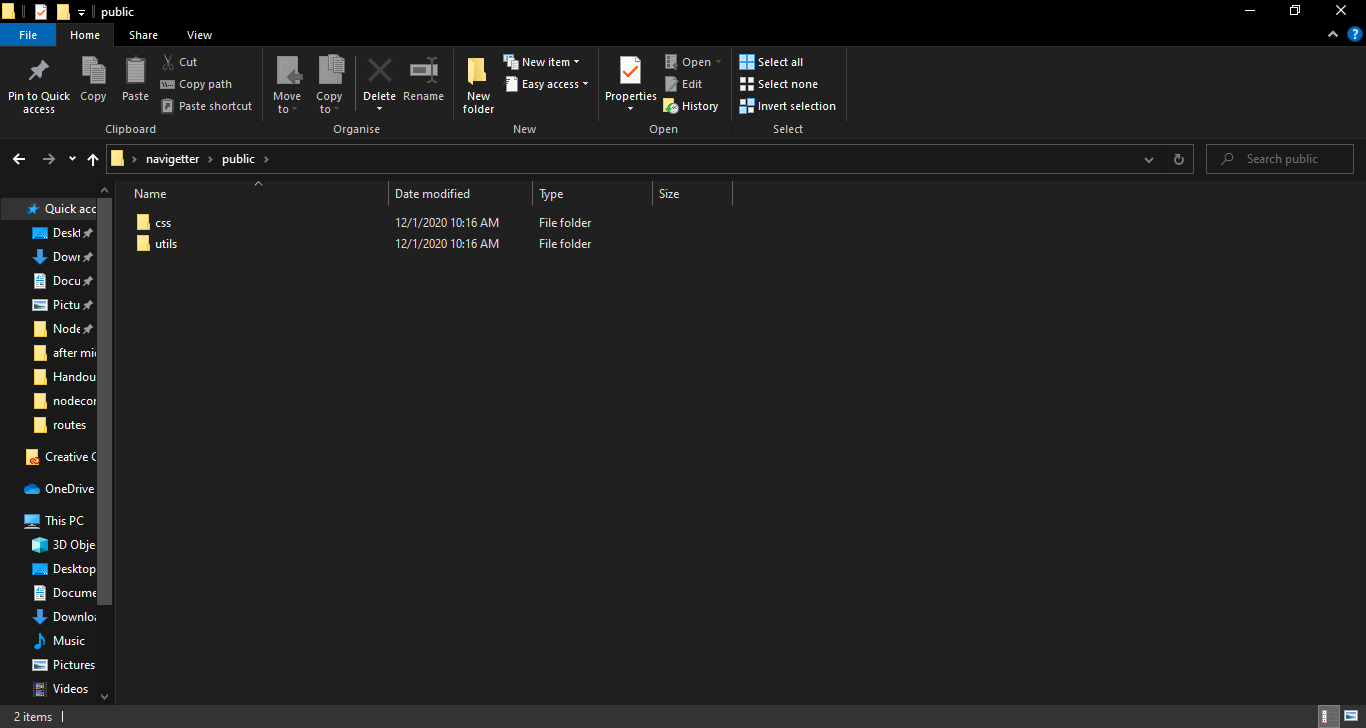
Packages are dropped into the node\_module under the prefix. When installing locally, this means that you can require(“packagename”) to load its main module or require(“packagename/lib/path/to/…..”) to load other modules

The important npm packages that have been used for the project are :

* Express
* Hbs
* BodyParser
* Nodemailer etc.

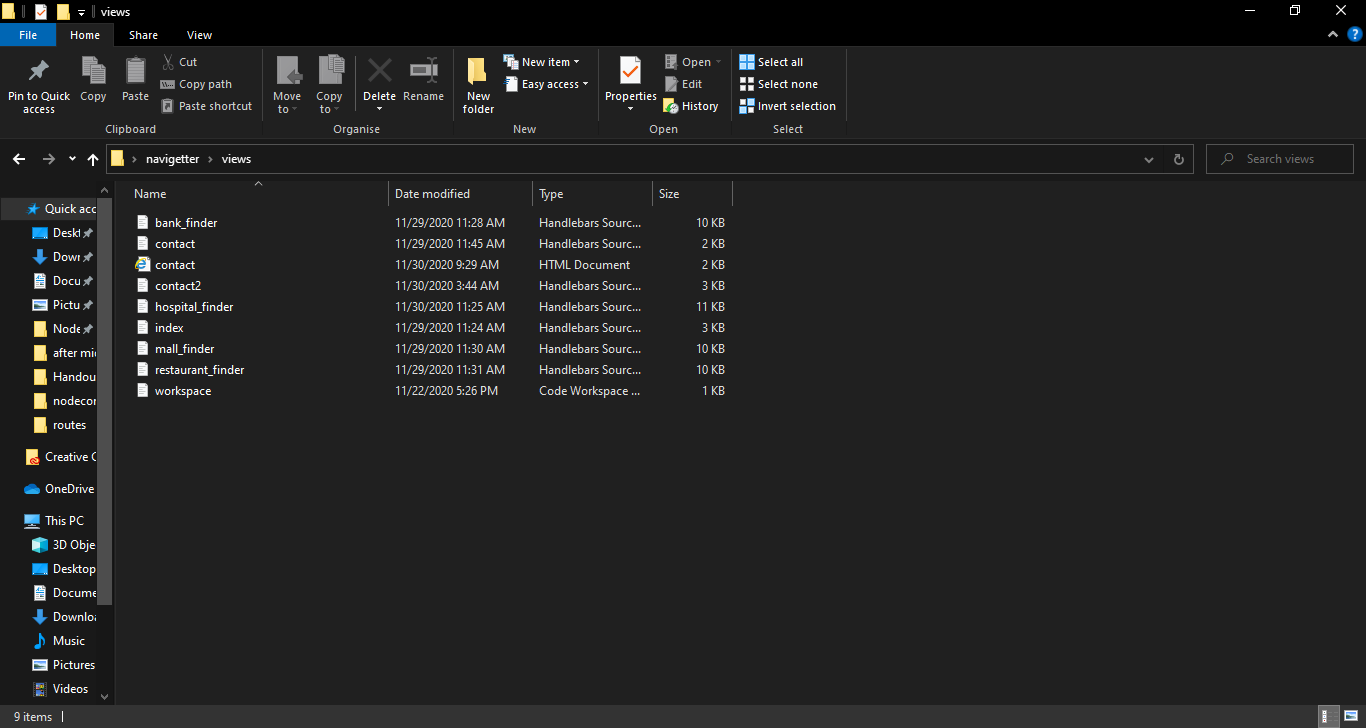
The handlebars(hbs) view engine and MVC architecture.

**2)Public**

****

Contains all CSS and util part responsible for appearance on browser

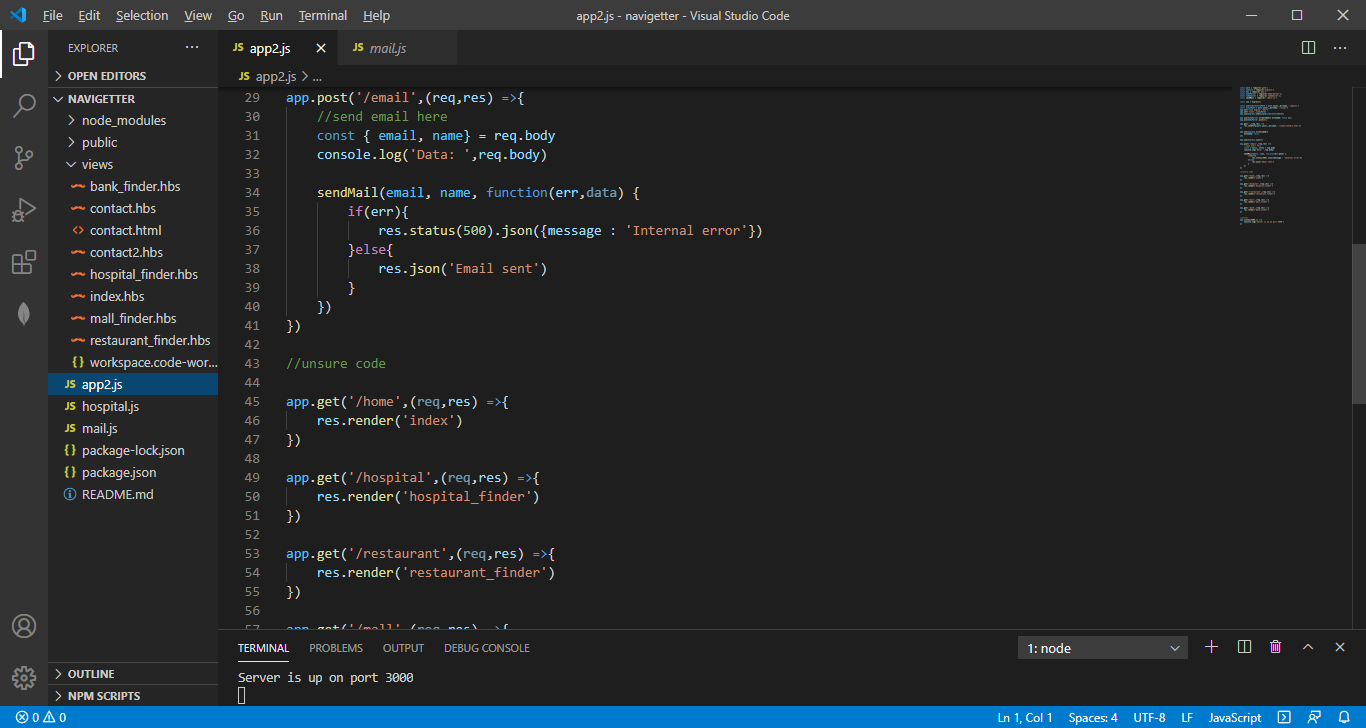
**3)Views**

****

Contains the code which finds nearest hospital, restaurant,mall and banks

**4)App2**

App2.js is the file which runs on the localhost. It contains paths to all the finder files and the nodemailer file(mail.js). App2.js extracts the username and email entered in the contact.html file as req.body and uses it in the automated mail.



# Future Scope

1. Handy to help assist citizens to nearest hospital in case of an accident
2. Android app like AarogyaSetu
3. Text to Speech conversion so replica of google maps navigation
4. Geocode the location entered by the user and convert it to latitude and longitude and used by our application

**Conclusion**

While this application is a useful tool for reaching the required hospital,restaurants,malls and banks, there’s a lot of scope for improvement. For example, a user on the go will find an android app more convenient. The application can be further modified to differentiate between banks and ATMs and the types of restaurants and hospitals according to speciality. We can also add test and pathological centers.