Search

Doctor Search

DoctorSearch Client Side

This page is used to search any doctor around any zip or address.

The user location should be enabled on pop up request to find the directions to the destination.

**Screens**

1. Search Screen

2. Result Screen

3. Doctor Details

4. Direction Screen

**API Used:**

1. Google map api for Javascript to plot the geo coordinates on the map

2. Google direction api to find the route to the destination

**Search Screen**

- Keyword(Optional): The user can enter any doctor name to be searched.

- Miles(Optional): Defines the distance range to which the locations has to be present.

- Category(Optional): Searches for the required category of the doctor. Default is All categories

- Nearby Address(Optional): Decides the location around which the doctor location should be searched.

If not mentioned, the user's location coordinates will be considered.

- Search Button: On click of the button, the results will be fetched an displayed

**Result Screen**

- Displays results as a list, ordered according to the increasing distance from the user.

- Has ónclick listener's to the items in the list, that would show the Doctor Details screen on the click.

- New Search Button: Clears the result screen and takes to Search Screen form.

**Doctor Detail**

- Displays the deatails of the doctor clicked.

- Onclick, takes to the directions to the doctor location

**Direction Screen**

- Displays the direction from user location(A) to the clicked destination(B)

- Displays the step by step direction and the distance

- Has 4 categories of directions: Walking, Driving, Transit and Bicycling

- Back Button: takes to the doctor result screen.

**Doctor Search Server Side**

Uses Google's API and a database of doctors from http://nppes.viva-it.com/NPI\_Files.html to display a list

of doctors and their details based on the user location, zip code and doctor's category.

**Classes Used:**

1. DoctorSearch.java

2. DoctorObject.java

3. DoctorResult.java

4. Mysql.java

5. MysqlPopulate.java

**API's Used**

1. Google geocode API to fetch the geo coordinates of the doctor's address

**DoctorSearch.java**

- A servlet that gets the parameters fromt he client and bundles in to a DoctorOjbect.

- This object is passed to the getDoctorInfo() method of the DoctorInterface interface.

- Sorts the returned Result object based on the distance from the source

- Converts to JSON using Google's gson API and sends the result to the client.

**DoctorObject.java**

- Extends Result.java object

- An object to assign each result returned from the Database as object.

**DoctorResult.java**

- Implements the DoctorInterface to fetch the results to write to the client

- Creates a database connection using Mysql.java

- Gets the result set by calling getResultSet()

- Iterates through the result set and builds a list of DoctorObjects

- Fetches the geocordinates for each of the results using Google geocode API

- Bundles into a list and returns the list

**Mysql.java**

- Based on the DoctorObject values, the query is prepared, fetched from the database.

- A result set is returned

**MysqlPopulate.java**

- Populates the file obtained at http://nppes.viva-it.com/NPI\_Files.html into the tables in the database

**General Search**

**General Search Client Side**

This page is used to search any location around any zip or address.

The user location should be enabled on pop up request to find the directions to the destination.

**Screens**

1. Search Screen

2. Result Screen

3. Direction Screen

**API Used:**

1. Google map api for Javascript to plot the geo coordinates on the map

2. Google direction api to find the route to the destination

**Search Screen**

- Keyword(Mandatory): The user can enter any keyword to be searched.

- Miles(Optional): Defines the distance range to which the locations has to be present.

- Nearby Address(Optional): Decides the location around which the keyword location should be searched.

If not mentioned, the user's location coordinates will be considered.

- Search Button: On click of the button, the results will be fetched an displayed

**Result Screen**

- Displays results as a list, ordered according to the increasing distance from the user.

- Has ónclick listener's to the items in the list, that would show the direction screen on the click.

- New Search Button: Clears the result screen and takes to Search Screen form.

- Map displays the items on the map, with blue icon being the user's location.

**Direction Screen**

- Displays the direction from user location(A) to the clicked destination(B)

- Displays the step by step direction and the distance

- Has 4 categories of directions: Walking, Driving, Transit and Bicycling

**General Search Server Side**

Uses google API's to get the location co ordinates based on the zip code or address.

Searches the keyword and returns the JSON response to the client that includes all

the details of the locations near the keyword.

**Classes used:**

1. LocationSearch.java

2. GoogleResult.java

3. Process.java

4. GoogleObject.java

5. Result.java

6. GeocodeOjbect.java

**API's Used:**

1. Google's nearby search API to get all the results based on keyword

2. Google's location detail API to get Phone and Rating for a geo location

3. Google's Geocode API to get the geo codes for an address or a zip code

**LocationSearch.java**

- Servlet that takes input GET parameters from the client side

- Calls the getCommonResult() of GoogleResult using it's instance.

- Sorts the returned Result object based on the distance from the source

- Converts to JSON using Google's gson API and sends the result to the client.

**GoogleResult.java**

- Implements the CommonResult Interface and implements the method getCommonResult()

- Constructs the URL based on the parameters obtained from the client side to for the

Google's nearbysearch API.

- Calls getJsonString() from the Process.java class to get the Google results as JSON

- Converts the JSON to GoogleObject by calling getGoogleObject and returns a List of Result object

**Process.java**

- A utility class to compute the distance between the two lat-lng sources

- Also gets the string out put given a url

**GoogleObject.java**

- An object defined to store Google response result

**Result.java**

- An object that stores parameters to be sent tot he client.

- Ultimately, the returned result to the client is a list of this object

or a subclassed object packed as JSON.

**GeocodeObject.java**

- Object to store the results obtained from the google Geocode API.