MOBILE USER EXPERIENCE

Travellify

NAME	NET ID	GITHUB ID	EMAIL ID
Careena Braganza	cmb499	cmb499	cmb499@scarletmail.rutgers.edu
Vishalsingh Hajeri	vsh15	vsh15	vsh15@scarletmail.rutgers.edu
Vikti Desai	vdd23	viktidesai	vdd23@scarletmail.rutgers.edu
Aneesh Abhyankar	ana85	AneeshAbhyankar	ana85@scarletmail.rutgers.edu

TABLE OF CONTENTS

Travellify

1. Abstract

Many trips are ruined because of over planning and we aim to make an app that allows friends to think less and enjoy more. This would be an app for people who love to make impromptu decisions while on a road trip. To this end, we build an application which allows the user to plan his trip in a two-step process: Select his trip by specifying source and destination and get recommendations of spots he can visit by making small detours. We employ the Google Maps API to find route for the user and then recommend him spots on the way based on images other users have contributed. We call these locations 'hotspots'. We use MIT's 'gridLatLong' PHP script to essentially divide the earth into a grid of any resolution. This helps to classify photographs clicked at a distance close to each other as the same place. We use MySQL database hosted online to store all the images clicked by the user as well as the image classification information described above. We also give the user a choice to make contributions to the application by uploading photographs, which will be the basis of making recommendations. We utilize the clarifai API to suggest tags to the user that he can associate with the photograph he clicks before uploading, to further classify similar kinds of images.

2. Main Components of Application Architecture

2.1) MySQL Database:

The data that we are dealing with is user uploaded photographs, tags associated with each image, geographical location of each image, cluster information of images so that multiple images in close geographical proximity are clustered together, etc. Since this data is interlinked or interdependent, we chose the MySQL as the means to store data. The tables in our database include images, image_details, place_details, etc.

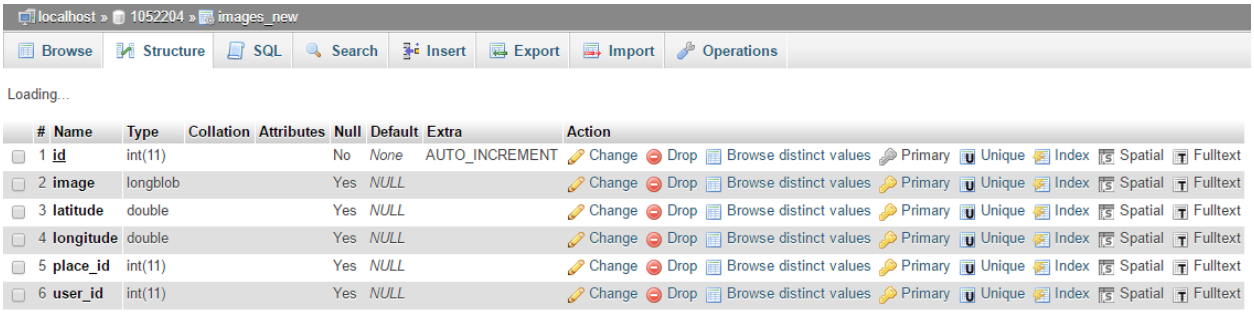

Figure 1: Structure of 'images' table

The above figure shows the structure of images table which contains the image in long BLOB format, the latitude and longitude associated with it, the place_id which is the cluster information

as explained above and the user_id which identifies the user who uploaded that image. The place_id is basically an integer associated with the grid in which that latitude-longitude pair lies. All the latitude-longitude pairs in close proximity will be given the same place_id. This is done by the gridLatLong php script by MIT. It essentially divides the earth's surface into a grid of defined resolution, in our case 20,000. Thus, all the geographical points lying in a particular grid square are given the same place_id. The script accepts latitude and longitude of a place as inputs and gives a place_id as output.

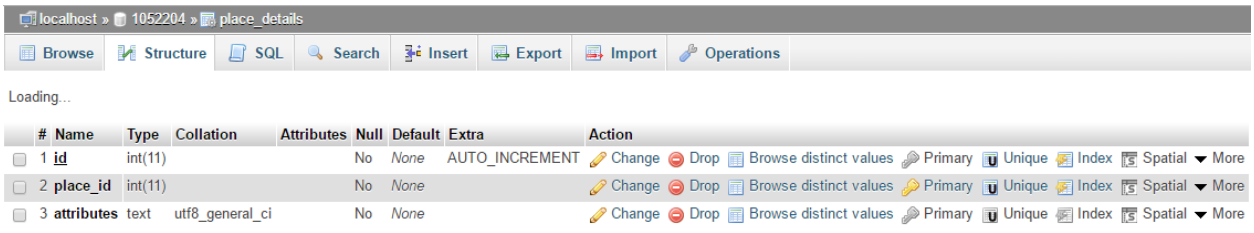

Figure 2: Structure of 'place_details' table

The figure above shows the structure of 'place_details' table, which includes place_id and attributes, i.e. tags the user assigns to the photos. Several tags are recommended to the user by the clarifai API, from which he chooses a few.

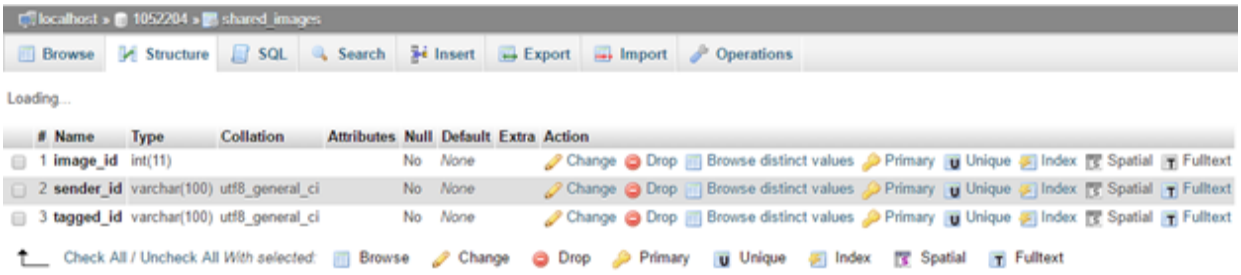

Figure 3: Schema of 'shared_images table

The figure above shows the structure of shared_images table, which includes image_id, sender_id and tagged_id. This table summarizes the image sharing information, about which user tagged which user/users in which images. A query is run to show the user which images he was tagged in.

2.2) Clarifai API for Image Recognition:

We have used the Clarifai API for Image Recognition and corresponding tag recommendations. Since one of the features of the application allows the user to retrieve images based on tags we thought of using the Clarifai API. It allows us to recognize images by suggesting corresponding tags sorted in order of relevance as determined by the API. With the aid of the documentation provided by Clarifai for android app development, we integrated the API usage in our application. Whenever the user uploads an image, tags would be suggested by the API which the user can then select and store the image in the database along with the tags. These tags are then used while retrieving images while doing a tag-based location search.

For using the Clarifai API, we need an *access token, clarifai key* and *clarifai secret* on using the Clarifai client library for Android. Images are uploaded to Clarifai API and tag recommendations are retrieved. The recommendations are displayed as a grid of buttons from which the user can multi-select upto 5 tags. The images are then uploaded to the database and retrieved based on these tags using a select query. The user is given an option to upload an existing image or click a new one. The image is uploaded by associating the current location with it.

2.3) Google Map Directions API:

Google provides Directions API free of cost with a limit of 2500 API calls per day

Direction API was used to perform following:

- API provides the liberty of specifying origin and destination as text strings (e.g. "Chicago, IL" or "New Brunswick, NJ"), or as latitude/longitude coordinates, or as place IDs. We took a decision to retrieve only Destination String inputted by the user and we automatically detect his current location and use it as origin.
- Time to reach the destination and distance upto origin are calculated and updated in the Activity.
- API provides a feature to return multi-part directions using a series of waypoints.We have utilised this functionality by giving the user the ability to add hotspots to his trip.

Markers:

Markers identify locations on the map. The default marker uses a standard icon, common to the Google Maps look and feel. Markers are objects of type Marker, and are added to the map with the GoogleMap.addMarker(markerOptions) method.

We have designed Markers to be interactive. All the Markers throughout the app receive click events by default, and are often used with event listeners to bring up info windows.

We have tried and tested various markers properties like the icon's color, image, snippet , alpha value, InfoWindow popup and visibility to enhance User Experience and Application design.

2.4) Google Places API:

The Google Places API for Android provides app with rich information about places, including the place's name and address, the geographical location specified as latitude/longitude coordinates, the type of place (such as night club, pet store, museum), and more. We utilised this API by passing the user’s current location, place type and radius value to be searched around the user’s location. API returns a list of places of specified type which we populate on the map.

We have provided the user with the option of exploring from following place types:

- Restaurant

- Liquor Store
- Gas Station
- Hospitals

OnClick event on any of the place Marker takes User to a new Activity which provides detailed information about that place like Address, Rating and whether it is currently open or closed.

2.5) Zoom the View Animation:

We utilised a google library to implement a touch-to-zoom animation in our activity. In the DownloadImage Activity we are populating all the images in a grid view. This library is useful to animate a view from a thumbnail to a full-size image that fills the screen. We took this decision to use this library as it saves us the time of creating and maintaining a whole new activity just to view the enlarged image. Also, this feature is favourable from user point of view as they won't be switching screens from one activity to another so often.

2.6) OAuthentication :

OAuth is an open standard for authorization, commonly used as a way for Internet users to authorize websites or applications to access their information on other websites but without giving them the passwords.This mechanism is used, for example, by Google, Facebook, Microsoft, Twitter, etc to permit the users to share information about their accounts with third party applications or websites.

Generally, OAuth provides to clients a "secure delegated access" to server resources on behalf of a resource owner. It specifies a process for resource owners to authorize third-party access to their server resources without sharing their credentials. Designed specifically to work with Hypertext Transfer Protocol (HTTP), OAuth essentially allows access tokens to be issued to third-party clients by an authorization server, with the approval of the resource owner. The third party then uses the access token to access the protected resources hosted by the resource server

For user sign-in and maintaining user-profiles, we use Google's OAuthentication in our application since every Android user has a Google account. To implement the OAuth in Android, the *GoogleApiClient* class was used. We used the *GoogleSignInOption* class to configure Google sign and request user-information. A sign-in button (of class *SignInButton*) was added to the layout and inflated in the Main Activity with corresponding on-click listeners for implementing user-authentication.

3. Functionality and Design

3.1) Signing In :

OAuthentication is used for signing in to the application as specified earlier.

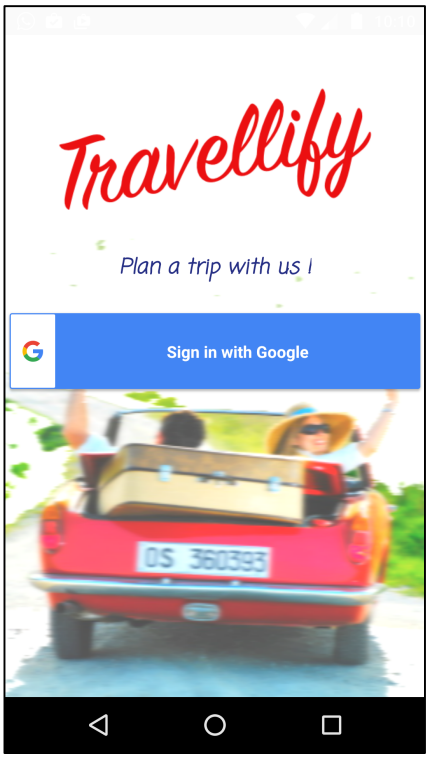

Figure 3: Login Page

3.2) Finding *Hotspots* on the route:

The application asks the user to enter the source and destination the user wishes to go to. Like mentioned earlier, we have used the Google Map Directions API and Google Places API for our maps activity. In addition to the we integrated the GridLatLong algorithm to our application.The algorithm, which is in the form of a PHP script, divides the earth's surface into numerical global grid reference of X by X , X being any whole number, typically 1000. The dimensions we chose were 20000*20000 for increasing the accuracy of finding locations. With the aid Google Places API and the GridLatLong algorithm the application provides the user with a path from the source to the destination and the nearby "hotspots" on this route. The GridLatLong algorithm facilitated in easy and quick look-up of the hotspots on the route fetched. Hotspots are popular locations along and parallel to the user's route where the user might wish to stop by during his/her journey.

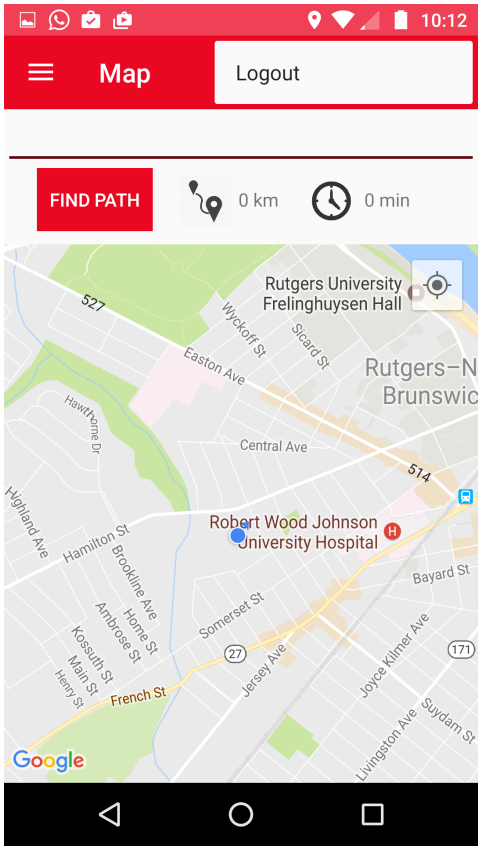

Figure 4: Maps Activity

3.3) Fetching Images on Clicking location markers:

Another highlight of the application is that when a user clicks on one of these hotspot markers, all images clicked or uploaded at that location through the application are retrieved from the MySQL database. Images are classified based on place-id provided by GridLatLong algorithm. All the images from the database which match the place-id are retrieved from the database server. These lists of retrieved images are stored in an ArrayList of Images and it is then populated in a Grid View. The user can zoom-in on the images by clicking on them. This will allow the user to visually look-up a hotspot and decide whether to go this place. We provide the user with the option of adding this hotspot in his journey and when the user clicks on "Add to Trip" button we modify the Start to destination path in the maps activity by adding the hotspot as the waypoint in the journey. We also update the total miles and time to travel variables.

Figure 5: Images retrieved on clicking the location markers

3.4) Finding Nearby popular locations:

Additionally, at any given point during the user's journey, he/she can search for nearby locations by category. The categories include gas stations, restaurants, liquor stores, hospitals. Thus, when a user clicks on one of these categories, the map is marked by these locations for a radius ranging from 0.5 km to 5 km depending on the category. For instance, a user can search for restaurants with a radius of 0.5 km and for hospitals and restaurants within a radius of 5 km. Consequently, the variable specifying the radii for searching locations was set to 500 to 5000 in the application code. This application feature was achieved using Google Places API. OnClick Listeners set on these markers will take the user to a new activity which provides details of the clicked place.

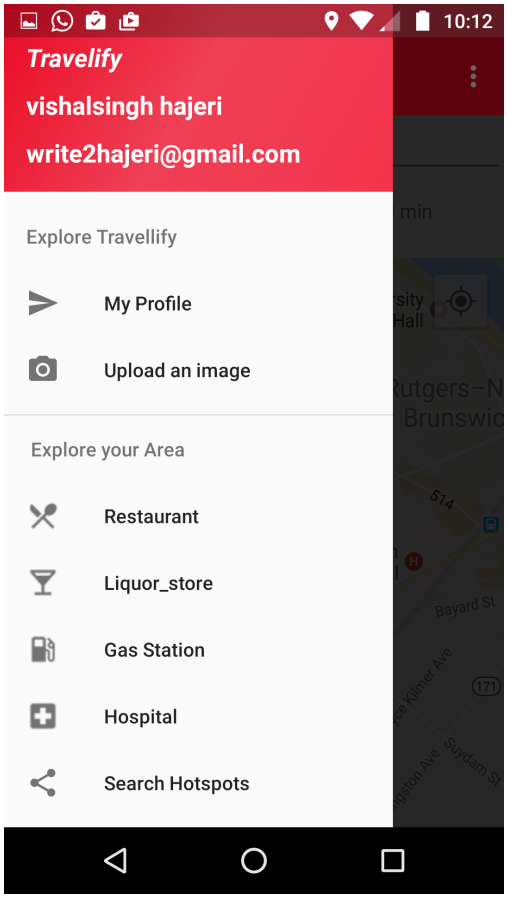

Figure 6: Navigation bar Options displaying location categories

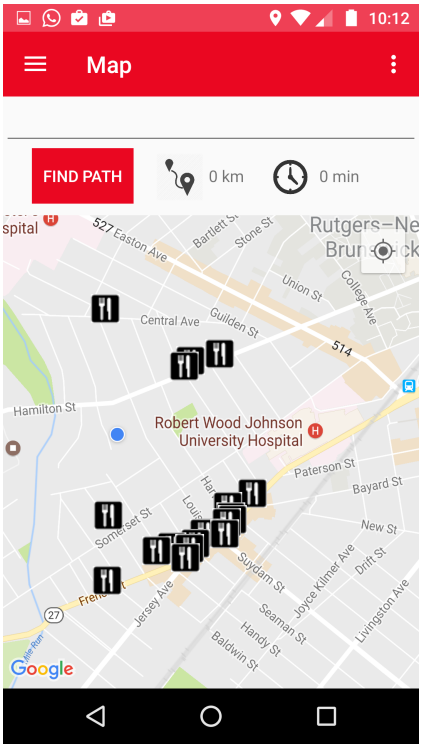

Figure 7: Showing all nearby Restaurants

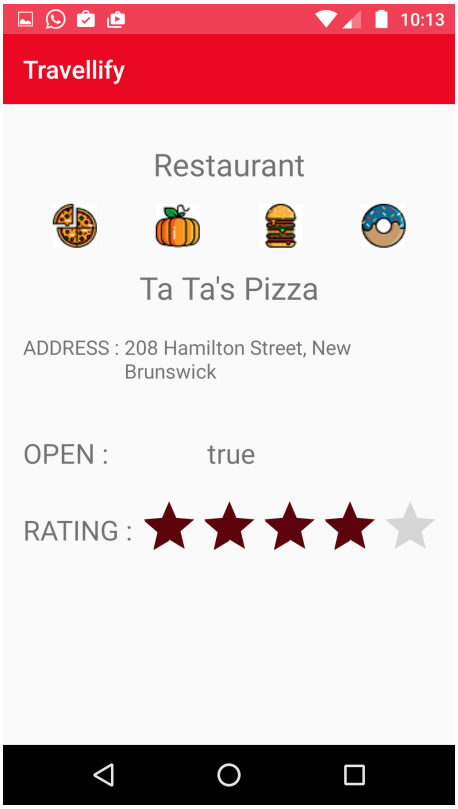

Figure 8: Restaurant Details displayed on clicking the restaurant location marker

3.5) Upload Image:

The upload image is an Async task which runs in the background without interfering with the UI thread. In this, we make an Http POST request, with the image (long BLOB format), latitude, longitude (of the current location), user_id and attributes. Attributes are the tags recommended by the Clarifai API. Like mentioned earlier, the user can store upto 5 of these tags. The php script that receives these variables stores the image in the images table and the place and attributes in the places_details table.

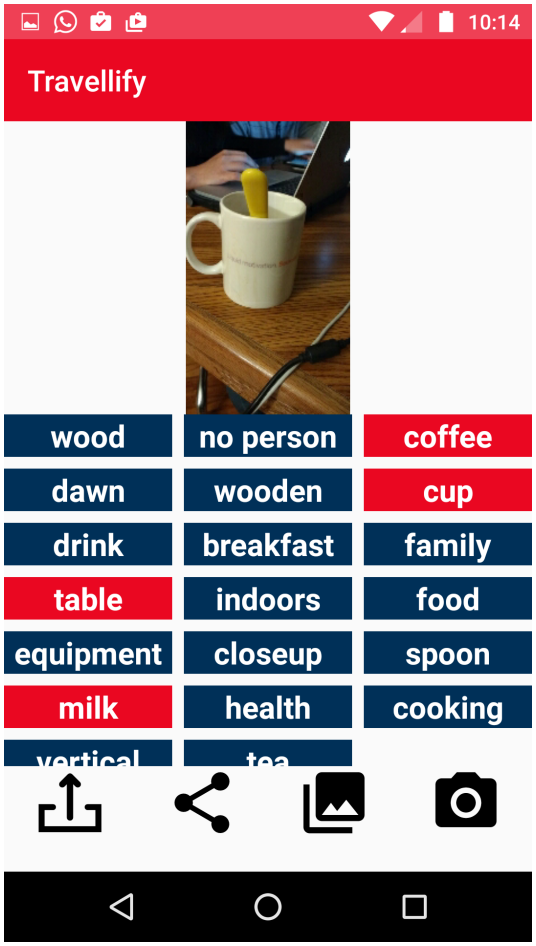

Figure 9: Image Clicked and corresponding tags recommended by Clarifai

3.6) Download Image:

The download image is also an Async task. It makes a GET http request to the server to download the image, which it converts back to Bitmap before displaying.

3.7) Search Images by tag:

A user can search image by tag. This means that whenever a user enters a tag, e.g. “beach” all images whose attributes match the tag are retrieved from the MySQL database. These tags in the first place were saved by the various users during image-upload on recommendation by the Clarifai API.

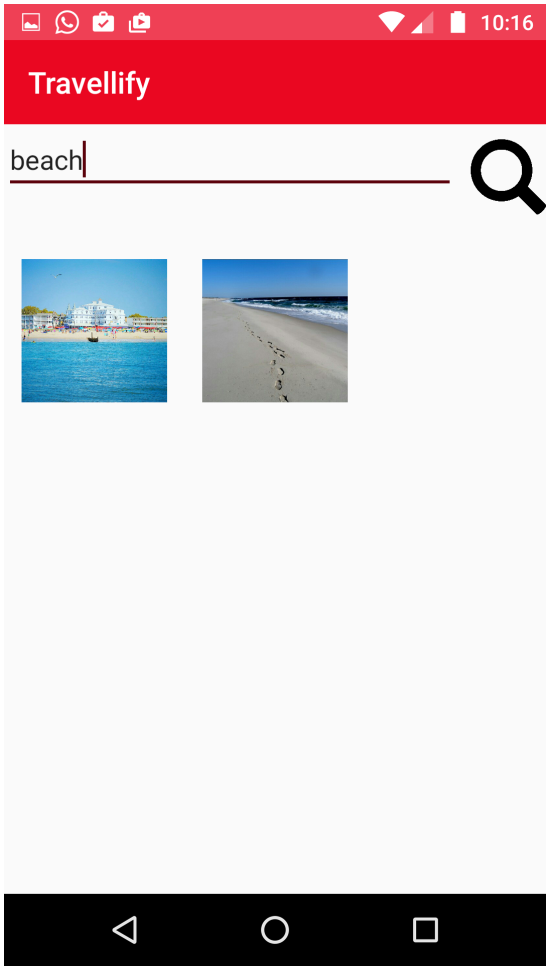

Figure 10: Searching Image by tag

3.8) Tagging friends:

The user can tag friends while uploading images. A search bar for finding friends is provided in this activity where the user must enter the username of the friend they wish to tag. The search bar provides suggestions for auto-completion.

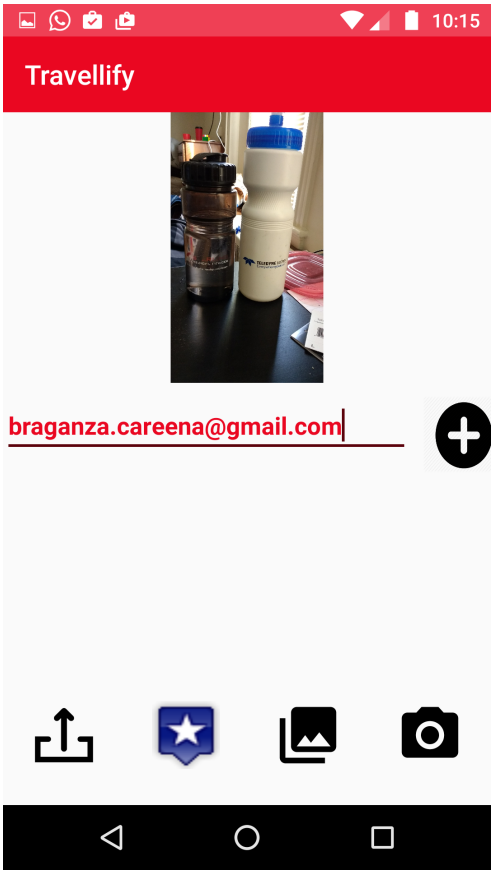

Figure 11: Tagging a friend on a picture

3.9) Maintaining a user profile:

Users can have their profile where they can view their profile picture, username and the images they have uploaded and images they are tagged in. This is done by querying user information from the MySQL database.

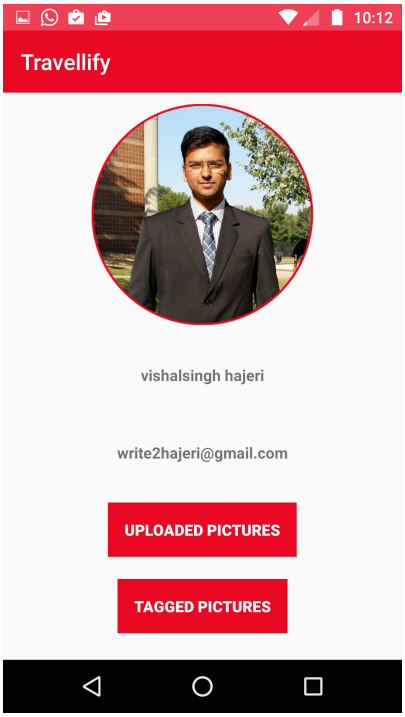

Figure 12: Example of a user profile

4. Contributions:

Task No.	Task Name	Duration	Start	Finish	Assigned To
1	Project Topic Research	3d	09/26/16	09/28/16	Aneesh, Careena
2	Literature Survey	2d	09/29/16	09/30/16	Vikti
3	Exploring Google API	2d	09/30/16	10/03/16	Vishal, Vikti
4	Map Activity with Routes	5d	09/30/16	10/06/16	Vishal, Careena
5	Places API and Hotspots	5d	10/09/16	10/13/16	Vishal
6	Grid Latitude Longitude Algorithm	3d	10/07/16	10/11/16	Aneesh
7	Search by HotSpot / Tags	4d	10/27/16	11/01/16	Careena, Aneesh
8	Google oAuth - Account Management	3d	11/16/16	11/18/16	Careena
9	User Profiling - Uploads	3d	11/28/16	11/30/16	Vikti, Vishal
10	User Profiling - Shared Images	5d	11/28/16	12/02/16	Careena, Aneesh
11	Navigation bar and other Activities	12d	10/12/16	10/27/16	Vishal, Careena
12	Database Schema Design	3d	09/30/16	10/04/16	Aneesh, Vikti
13	Server Setup	2d	09/29/16	09/30/16	Aneesh
14	MySQL DB Integration and PHP Scripts	30d	10/20/16	11/30/16	Aneesh, Careena
15	Integrating Clarifai with Application	4d	11/22/16	11/25/16	Vikti, Careena
16	Testing and Debugging	3d	12/01/16	12/05/16	ALL
17	UI enhancement	3d	12/06/16	12/08/16	Vikti, Vishal