I'm A Tourist Web App
Alex Roan – alr16

Implementation

To begin development of the app, I started with the conference_example project provided on blackboard. I approached the project by incrementally adding features using the existing framework provided. There were six main pieces of functionality which I incrementally implemented, they were as follows:

1. Adding a Visits table in the database

2. Camera functionality

3. Saving new visits to the database

4. Displaying thumbnails within the visits list

5. Map functionality

6. Displaying details of a visit when clicked on in the list

Before implementing these features, I added two tabs to the bottom of the app: Visits and Camera. The Visits tab used the same icon as the Session tab and the Camera tab used the same icon as the Home page tab. During the development stages, I kept the other articles and tabs that were not needed for the assignment (Home and Sessions) for testing purposes. These were later removed when most features were completed. In the Controller, in the page change event call, new options were added for the new article pages.

The first task of adding a Visits table to the database was not particularly difficult to implement as there was plenty of sample code implementing other tables in the project. The Visits table comprised of seven fields: _id, title, notes, latitude, longitude, datetaken, imagepath. The _id field was implemented as a self-incrementing identity field, the rest as text fields.

As well as creating the table, methods in the DataContext class needed to be implemented to access the data from the new table. I added queryVisits which selects all fields from the visits table in order of the datetaken field. processVisitsList was also added to the DataContext class to initiate the query with a renderer function as a parameter to render the results to the screen.

In the Controller class, renderVisitsList method was added which acts as the renderer to processVisitsList to display the Visits from the table. This is called when the Visits page is displayed. Initially, I did not implement the thumbnail image functionality due to the test data not having actual file paths to real images.

The next task was to implement camera functionality. Initially, I added a HTML5 input element to the camera article page which worked when accessing the site on a mobile web browser. However, when I tested this on a built phonegap app, the camera was not opening. This led me to investigate phonegap plugins.

Looking into phonegap specific plugins, the camera plugin under org.apache.cordova.camera enabled me to open the camera when loaded onto an android device. However, this phonegap specific plugin meant that I had to implement a separate procedure for browsers, in the cases where phonegap was not supported.

On the camera article page, the user is shown a single button labelled: "Open Camera". When this is clicked, the function checks whether phonegap is supported. If not, then getDesktopPhoto is called, otherwise navigator.camera.getPicture is called.

getDesktopPhoto runs a video stream onto the page directly from the default camera input on the device. A button to capture a photo is displayed and when clicked, a still image from the video stream is captured and displayed in an image element below it. From this point, the user can enter a title and some notes about the image, and save the visit.

navigator.camera.getPicture opens the default camera input on the device and upon success, the captured image is displayed on the screen. From here the user can add a title and notes and save the visit should they click the save button.

The save procedure first moves the image taken from its temporary path, to a file path within the device's storage. This is done by using the phonegap specific plugin: org.apache.cordova.file. It then takes the title and notes values from the page, retrieves the geolocation of the device from navigator.geolocation.getCurrentPosition, the current date time and the new file path and saves these fields in the Visits table.

To save new records to the visits table, I added a procedure in the DataContext class called insertNewVisit. This function takes all the fields needed other than the _id field, to add a new record to the table.

The fourth task of displaying a thumbnail for each of the visits in the visit table was not a large task to accomplish, but needed more testing within the phonegap app environment. This meant that I had to upload many revisions of the code to the phonegap website in order to test the app on my mobile device. Naturally, this was a fairly drawn out process but eventually ended in success.

In the renderVisitsList function in the controller, I simply added an <img> tag with the source set as the image path from the record. Initially, the trouble was that the file path was not being recorded as an absolute path for the saved image. Instead, it would save the file path from the devices default directory path. This meant that when the thumbnail tried to load, the device could not find the file. To remedy this, I added “file:///storage.emulated/0” to the beginning of the file path when saving the new record. This meant that the absolute file path was available when reading the record from the database on android devices. The ideal solution was to retrieve the default absolute path for the specific file system that the app was running but I could not get this to work, which is why I opted to implement the android specific path.

The fifth task was map functionality. There was a lot of online support regarding this task due to the popularity of Google Maps and their APIs. I realised early on that I’d probably have to call the processVisitsList function within DataContext to retrieve the full list of Visits from the database. Therefore, when the map page is loaded, dataContext.processVisitsList(print_map); is called, where print_map is the function that deals with displaying the map on the screen. I also realised that navigator.geolocation.getCurrentPosition would need to be used within the function to be able to

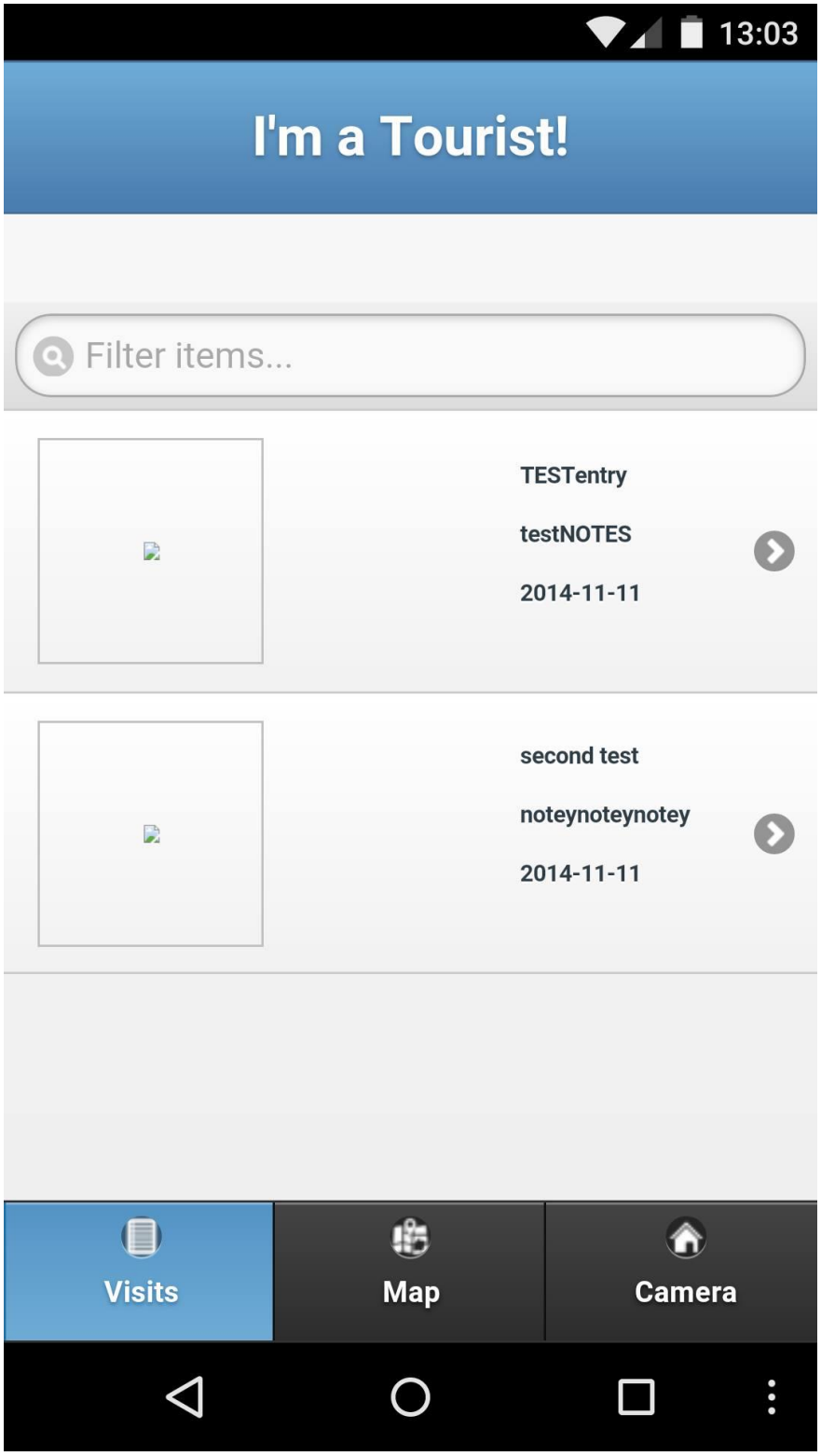
centre the map on the devices current position, leading to implementing embedded functions within the success criteria.

I included a google map api script within the index page which could deal with map requests. Using this, I used a stackoverflow example [1] of how to place multiple markers on a map, and centred it using the devices geolocation.

The sixth and final main piece of functionality implemented was the ability to display the full details of a visit by clicking on it within the visits list page. This proved slightly difficult as for some reason, I could not add event listeners to the list items within the Visits page. Instead, I changed the renderVisitsList procedure to display an onclick attribute within the <li> elements pointing to a method called list_item_clicked, with the parameter as the _id field of the visit. When clicked, list_item_clicked would take the _id number parameter, query the database and display the full information about the visit, including displaying a large photo of the image captured during saving the visit. I added an extra article to the index page called “visitdetails” to achieve this. I didn’t not add a new tab to the footer however, because this page would only be accessed by clicking on a visit list item.

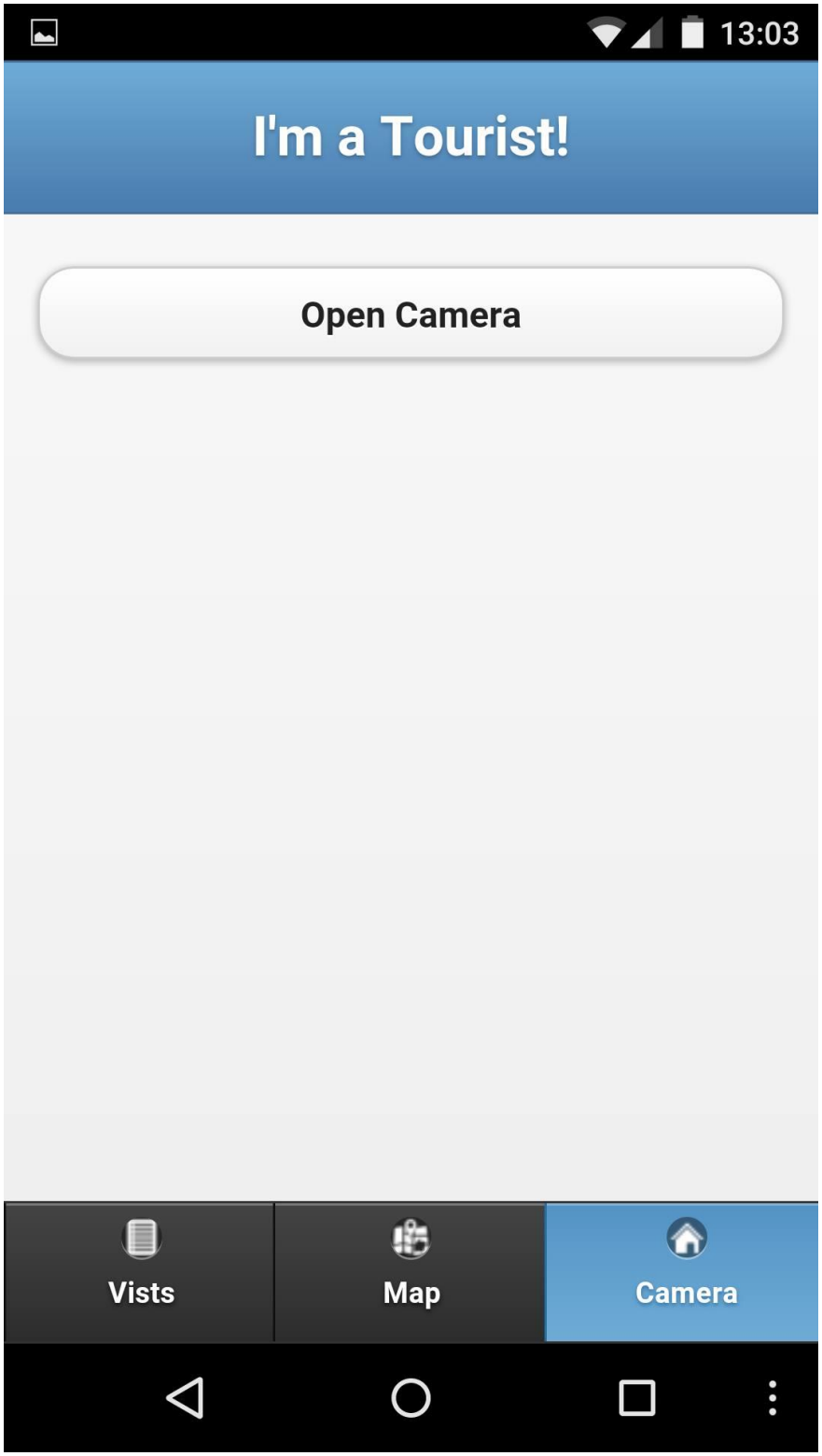
I believe that all of the main functionality was implemented to a good standard in this project. There is some room for improvement in making the app identical throughout all devices and browsers, but for its main purpose as a phonegap app it performs very well and as required. I therefore predict this project to receive a mark near 65%.

Testing & Example Screenshots

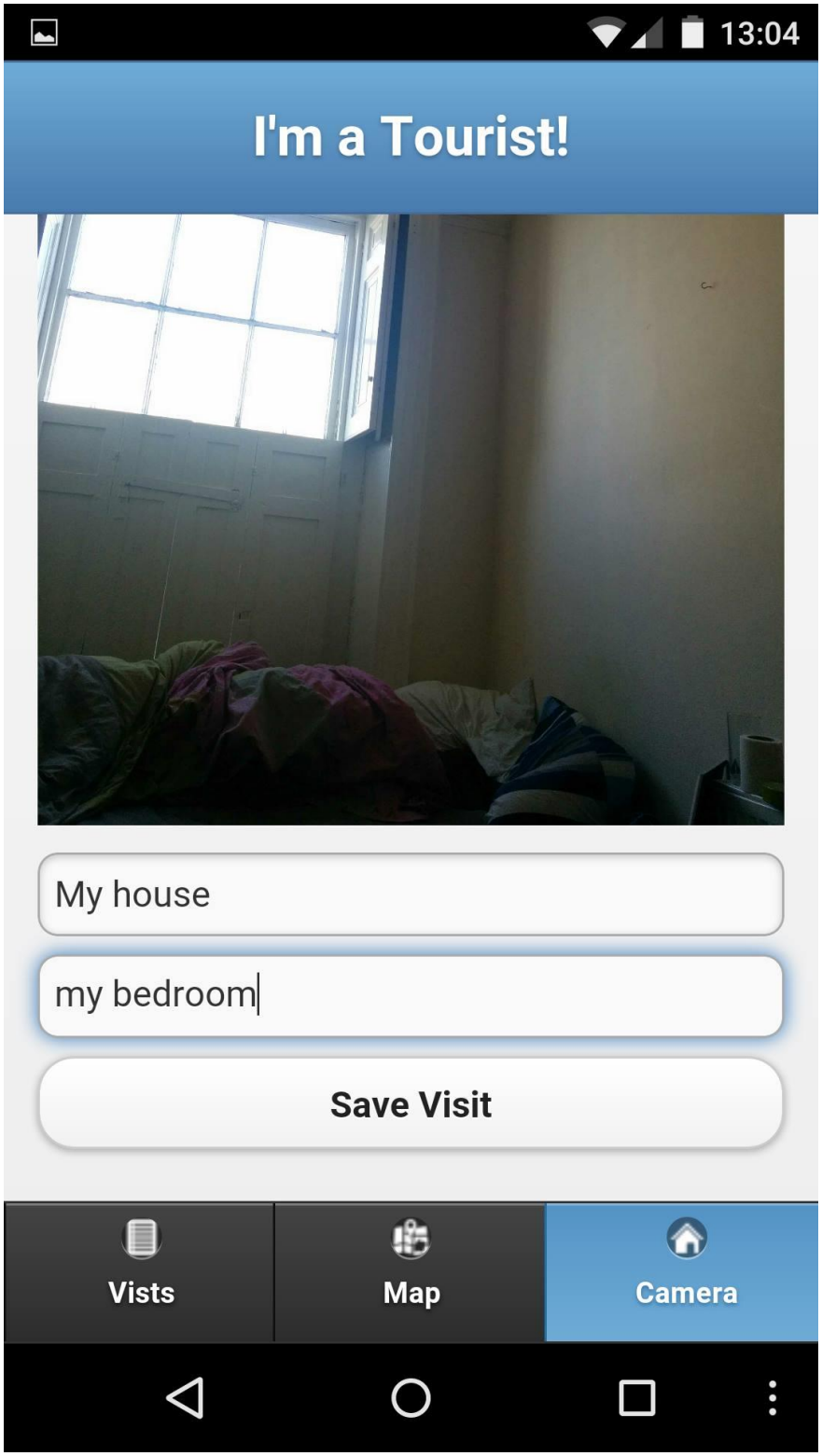


This first image displays what the user sees upon opening the app. The two test visits that are shown are inserted into the visits table as test data when the database is initialised. Obviously they would not be present in the finished app.

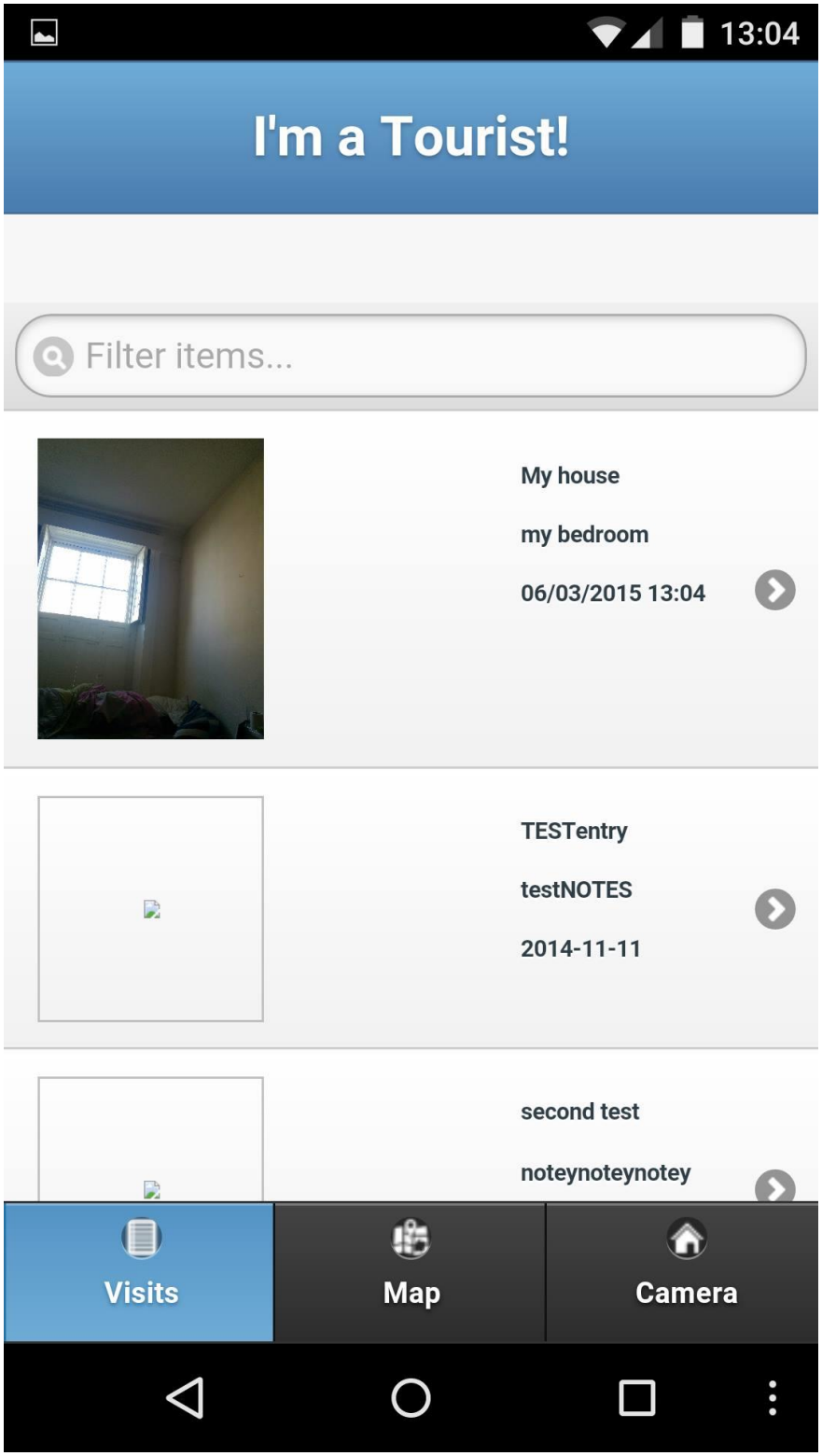
This second image displays the map tab. As you can see one of the test visits is displayed in the map



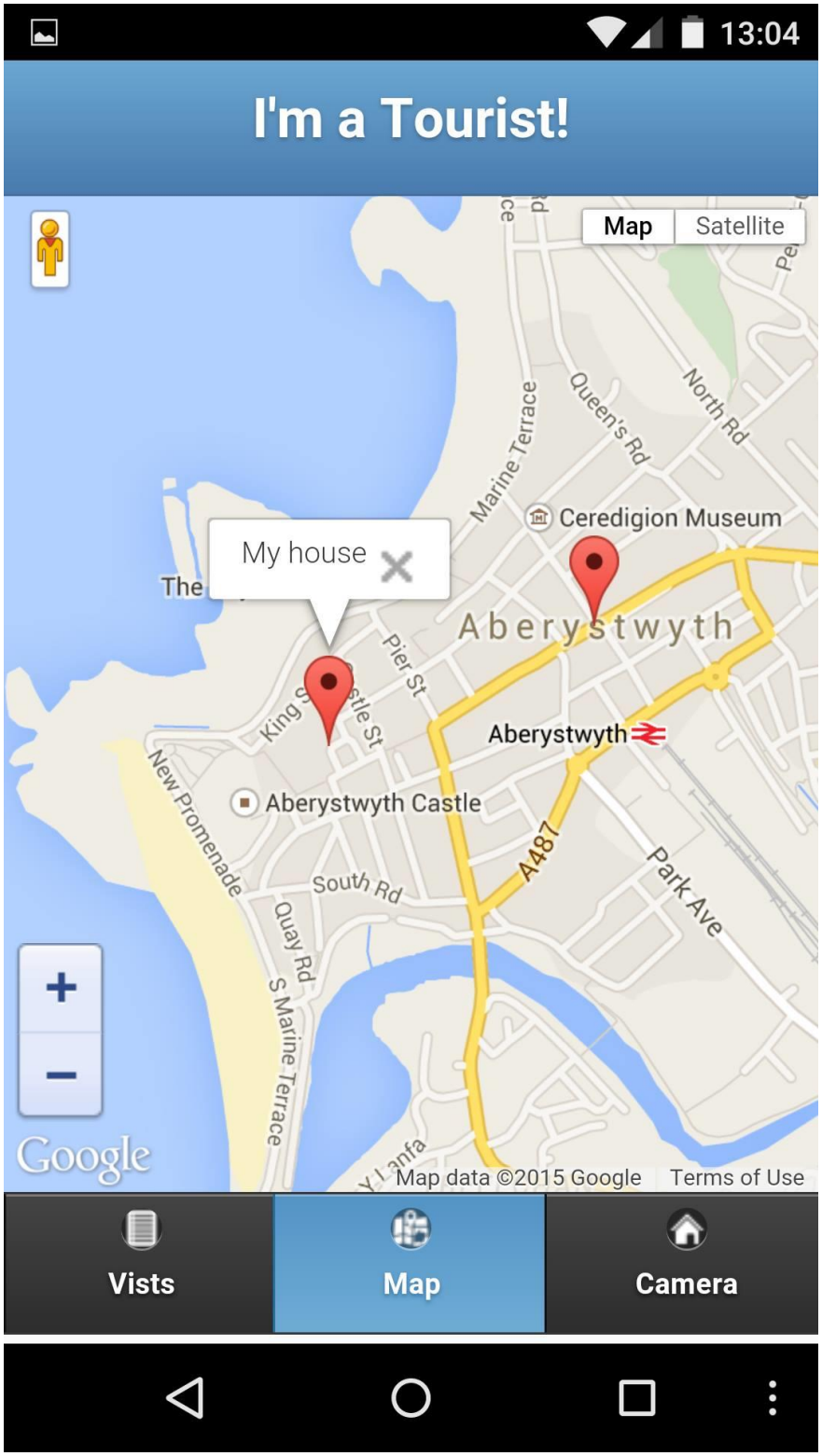
The third image shows the camera page. The button at the top opens the default camera for the device, and on completion displays it.



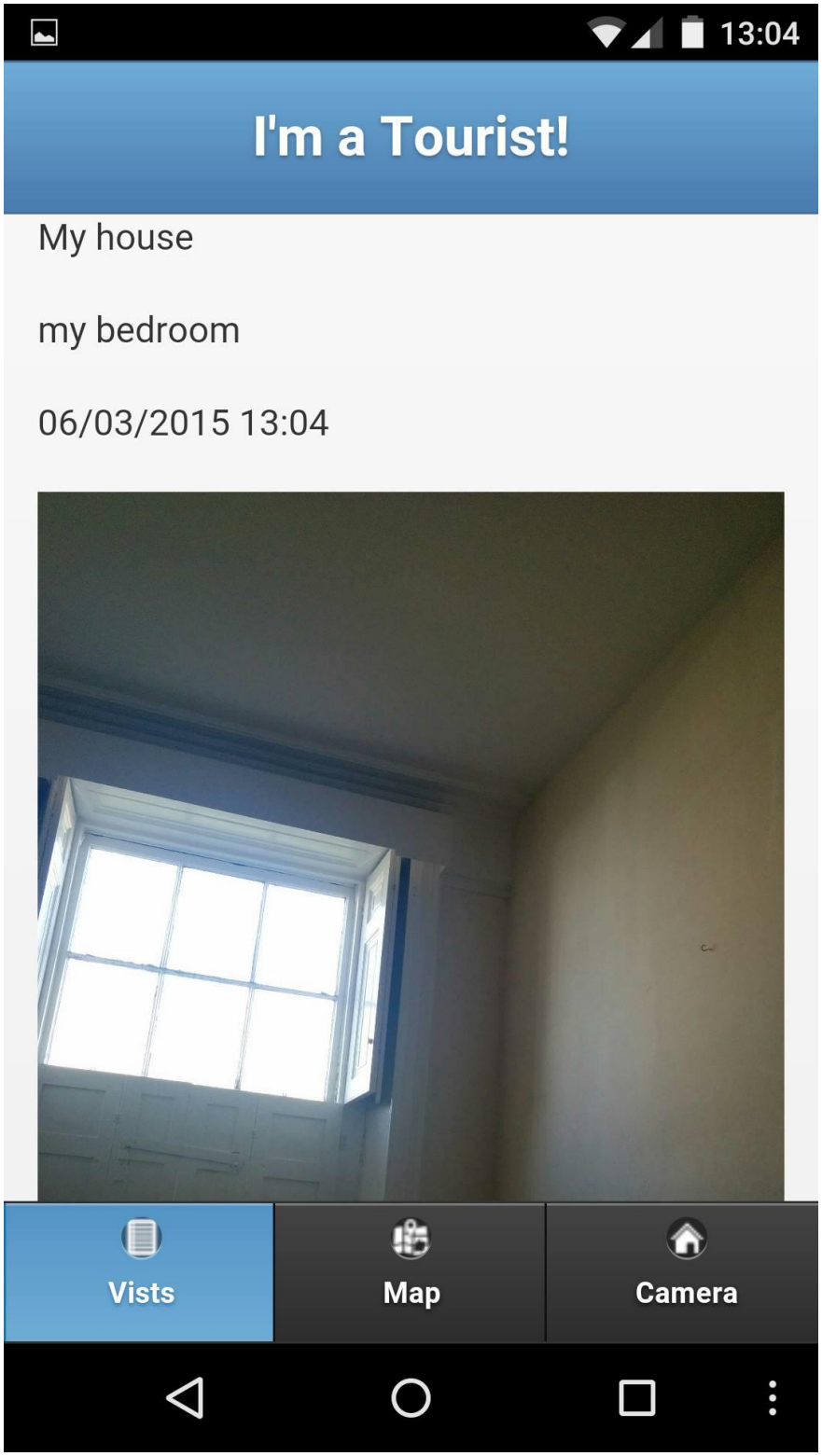
This fourth image shows the image that has been taken, two input fields for Title and Notes, and a save visit button. The save Visit button saves the visit using the location, date time, entry fields and image.



The fifth image is an example of the new visits list now that another visit has been saved. As you can see, a thumbnail of the image is provided



The sixth image shows the map feature with the new visit included.



The seventh image shows the screen which is displayed when the user clicks on a specific visit. The full image is displayed, and its details.

References

[1] - http://stackoverflow.com/questions/3059044/google-maps-js-api-v3-simple-multiple-marker-example - Multiple Markers on Google Maps API