“I’M A TOURIST” ANDROID APPLICATION REPORT

SEM2220 Assignment 1

Robert Mouncer
rdm10@aber.ac.uk

Introduction

This report is a write up for the first Mobile Solutions assignment. The assignment was to create an android application using *jQuery Mobile, HTML* and *JavaScript*. The application would then be built using *PhoneGap*. The *android* application needed to be able to store visits on user request. The visit would include notes, a photo from the phone's camera, the location information and the date/time.

A *PhoneGap* project and an APK file was available to us to help start the process of implementation. I based most of my implementation from the *PhoneGap* project provided by Chris Loftus.

The report will include the following:

1. How I went about implementing the assignment.
2. Problems that I encountered.
3. Evidence of the application working.
4. Evaluation – including what I have learnt.

Process of Implementation

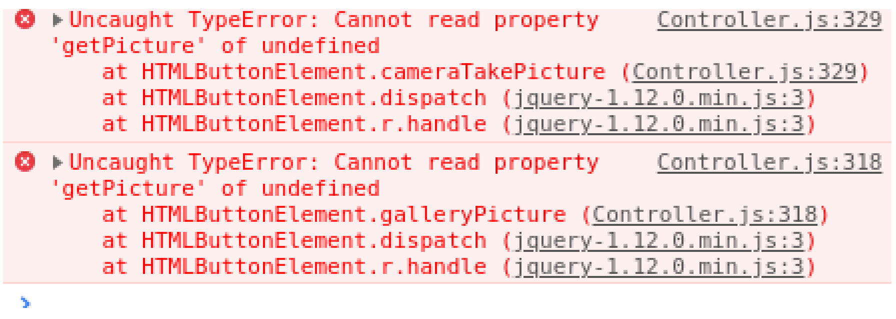
After reading the assignment specification I began to look at the provided code to make sense of it. I found the code and comments very informative as I have very little experience in web development. Once I had a rough idea of the conference app implementation, I began striping it down, removing *JavaScript* code and *HTML*. I then had a bare application with very limited functionality including events that would trigger on page changes.

I began by creating the visit form that the user would fill in when submitting a visit. The form was easy to create. Using the *jQuery Mobile* documentation. As the specification didn't specify that the picture or notes were required, or if anything was required, for a visit I decided to add another input parameter for the user - "Title". This meant that notes and/or an image were optional. Once I had the form in place, I needed to be able to create an image using an android phones camera.

I was surprised at how easy it was to implement the functionality for the Camera. The *Cordova-plugin-camera* online documentation led me through the installation of the plugin to the *PhoneGap* project. Within the documentation there were enough examples of how to use the plugin with *JavaScript*. Using the examples, I was able to successfully implement the camera functionality and display the image on the webpage.

A problem I found was that the camera didn't allow functionality to the gallery, this wasn't a requirement in the specification, but I decided to add this additional feature using an additional button.

Access to the gallery used the same functions as gaining access to the camera, the only difference being that the options passed to the plugin function (".getPicture()") required different parameters. When selecting images from the gallery I noticed that the images would often rotate 90 degrees. This was when the images chosen were a ".Jpg" file. After researching the cause of this, I found that the camera plugin has an option to return the image in its correct orientation. I found this an unusual option to require but the documentation mentioned the reason as an "android quirk".



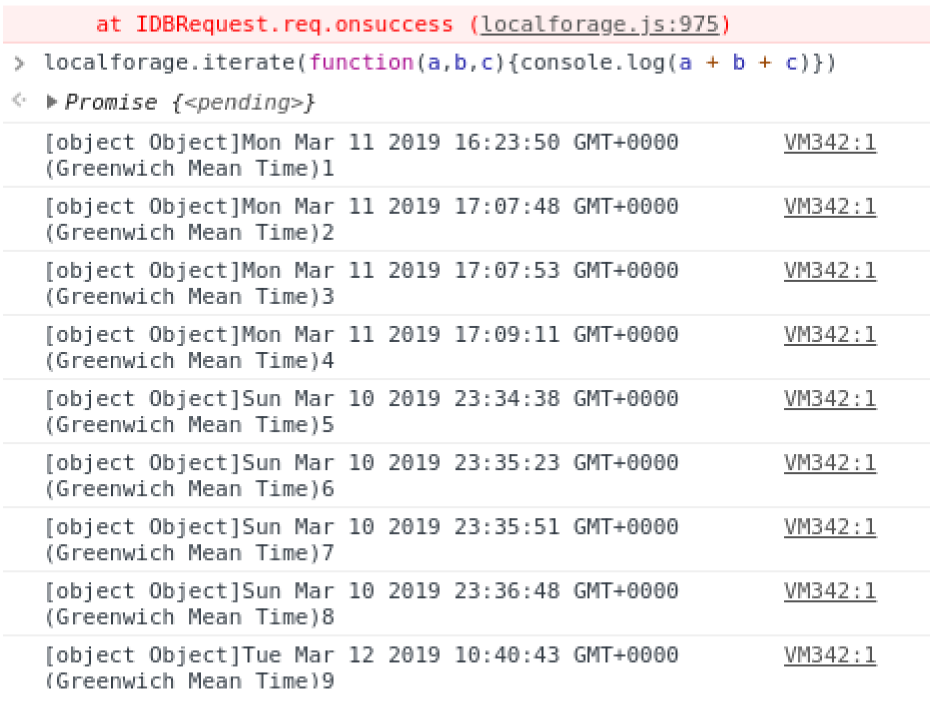
```
▶Uncaught TypeError: Cannot read property    Controller.js:329
'getPicture' of undefined
    at HTMLButtonElement.cameraTakePicture (Controller.js:329)
    at HTMLButtonElement.dispatch (jquery-1.12.0.min.js:3)
    at HTMLButtonElement.r.handle (jquery-1.12.0.min.js:3)
▶Uncaught TypeError: Cannot read property    Controller.js:318
'getPicture' of undefined
    at HTMLButtonElement.galleryPicture (Controller.js:318)
    at HTMLButtonElement.dispatch (jquery-1.12.0.min.js:3)
    at HTMLButtonElement.r.handle (jquery-1.12.0.min.js:3)
>
```

Figure 1 - PhoneGap functions failing in browser on PC

As the plugin required to be ran on a *PhoneGap* application, the gallery or camera feature could not be tested using a browser with development tools as shown in Figure 1. This slowed down development as I needed to tweak options slightly and rebuild a few times to test on an android device. Once this was working, if I wasn't using it as a *PhoneGap* application, on gallery or camera click I would use a random image from online.

The data that was entered needed to be stored. The specification mentioned that a *WebSQL API* could be used, though an *IndexDB* solution could also be used. I decided to choose *IndexDB* for additional flair marks. I spent many hours trying to set up the local datastore using the online documentation and online guides with no success. While searching for solutions I found a wrapper for *IndexDB* called *LocalForage*, a key-value pair solution. After failure trying to get *IndexDB* to work, I managed to get *LocalForage* working within an hour. My main reason for using the *IndexDB* wrapper, *LocalForage*, is because I couldn't get the standalone *IndexDB* to work.

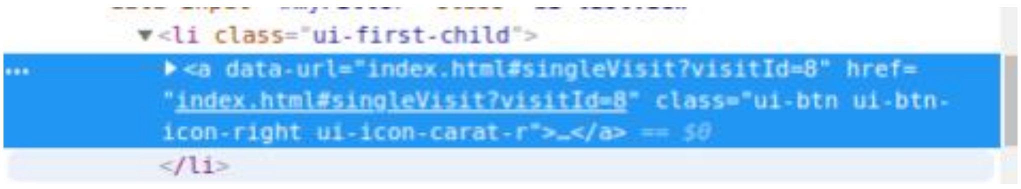
Once I realised how simple LocalForage was to use, I was able to save the text data entered within local storage. As the image was returned as a base64 string, this was also very easy to store and retrieve.



```
    at IDBRequest.req.onsuccess (localforage.js:975)
> localforage.iterate(function(a,b,c){console.log(a + b + c)})
< ▶Promise {<pending>}
  [object Object]Mon Mar 11 2019 16:23:50 GMT+0000      VM342:1
  (Greenwich Mean Time)1
  [object Object]Mon Mar 11 2019 17:07:48 GMT+0000      VM342:1
  (Greenwich Mean Time)2
  [object Object]Mon Mar 11 2019 17:07:53 GMT+0000      VM342:1
  (Greenwich Mean Time)3
  [object Object]Mon Mar 11 2019 17:09:11 GMT+0000      VM342:1
  (Greenwich Mean Time)4
  [object Object]Sun Mar 10 2019 23:34:38 GMT+0000      VM342:1
  (Greenwich Mean Time)5
  [object Object]Sun Mar 10 2019 23:35:23 GMT+0000      VM342:1
  (Greenwich Mean Time)6
  [object Object]Sun Mar 10 2019 23:35:51 GMT+0000      VM342:1
  (Greenwich Mean Time)7
  [object Object]Sun Mar 10 2019 23:36:48 GMT+0000      VM342:1
  (Greenwich Mean Time)8
  [object Object]Tue Mar 12 2019 10:40:43 GMT+0000      VM342:1
  (Greenwich Mean Time)9
```

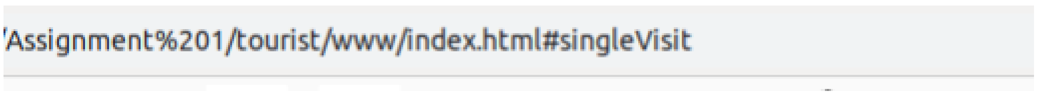
Figure 2- LocalForage Iterate function working in the developer console

To view the visits that had been stored locally, I adapted a function that Chris Loftus provided for the Conference application. The provided code retrieved an array and iterated over each item to build a view to render. This was very easy to adapt for the visits as *LocalForage* provides a function to iterate over every item in the datastore. As the items in the datastore are stored in the order they are created, this would iterate over them in date order, newest at the top. An anchor was added with the key id in the URL as a parameter.



```
▼<li class="ui-first-child">
   ▶<a data-url="index.html#singleVisit?visitId=8" href=
   "index.html#singleVisit?visitId=8" class="ui-btn ui-btn-
   icon-right ui-icon-carat-r">…</a> == $0
</li>
```

Figure 3- Anchor link showing the full address



```
'Assignment%201/tourist/www/index.html#singleVisit
```

Figure 4- Once clicking on the anchor link, the URL that the user would be directed to

A problem that I encountered was that the URL created for the anchor would be shortened upon click. Figure 3 shows that the anchor contains "?vistitId=8" but upon clicking on the anchor the URL was shortened to not contain this URL parameter, as shown in Figure 4. I noticed if the link was opened in a new tab, or the link was copied, the correct URL would load. I incorporated this into the anchor with "target=_blank". This wasn't the ideal solution, but it was the only option I could get to work.

After creating the page to show the single visit, I decided to add a delete button. This took advantage of the simplicity of LocalForage. It was very easy to implement.

I noticed in the configuration file that Chris Loftus had left the default icon and splash screen tags. I created a quick logo and used an online generator (https://www.resource-generator.com) to create the logo for different screen sizes and to output the tags needed within the config file. The splash screen doesn't quite fit the screen on the android device I was testing on, shown in Figure 8.

The map was very easy to implement as the online documentation and examples provided by Google were very easy to follow and were very intuitive. I added an information window to show the users location onload of the map. Adding the markers to the map was also easy as I used the iterate function provided by LocalForage to add a marker for each visit. Google also had online documentation and examples for information windows that displayed when a marker was clicked.

Flair

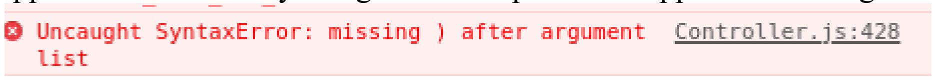
This section will include the additional functionality that was included in order to gain flair marks as per the specification.

1. Requirement of the Title Attribute of a visit.
2. The use of the IndexDB wrapper – LocalForage.
3. Ability to add photos that already exist on the phone, not just photos taken from the camera. Both the camera and gallery functionality use the Cordova-Plugin-Camera.

4. When a user submits their visit, the latitude and longitude is used for geocode reversing with the Google Maps API. This allows to pinpoint the address that the image was taken and is then displayed to the user when they are viewing a single visit.
5. Delete functionality of a single visit.
6. I have taken advantage of the ability to add icons and splash screens to the application.
7. The use of information windows:
   - To show the user the location they are in when they load the map tab.
   - To show information about the visit.

Testing

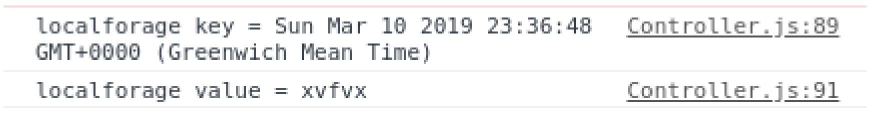
Testing for the application was mostly using the developer tools suppled with Google Chrome.



```
Uncaught SyntaxError: missing ) after argument   Controller.js:428
list
```

Figure 5- Error shown in Developer console

When a new feature was added I would firstly check the console for any errors, as shown in Figure 1. If there were errors, these would need to be addressed first. The developer console was very useful as it would display errors for brackets out of place, which I found myself mistakenly doing as shown in Figure 5.



```
localforage key = Sun Mar 10 2019 23:36:48   Controller.js:89
GMT+0000 (Greenwich Mean Time)
localforage value = xvfvx                    Controller.js:91
```

Figure 6 - console.log () in action

I found myself using the "console.log()" function frequently to ensure that the correct values were being retrieved for a visit as shown in FIGURE.
When adapting the CSS for items displayed, I would us the styles tab in the developer console to test and adjust the view.

App Screenshots

All screenshots were completed on my own personal android phone. This was because the android emulator would constantly crash on my PC.

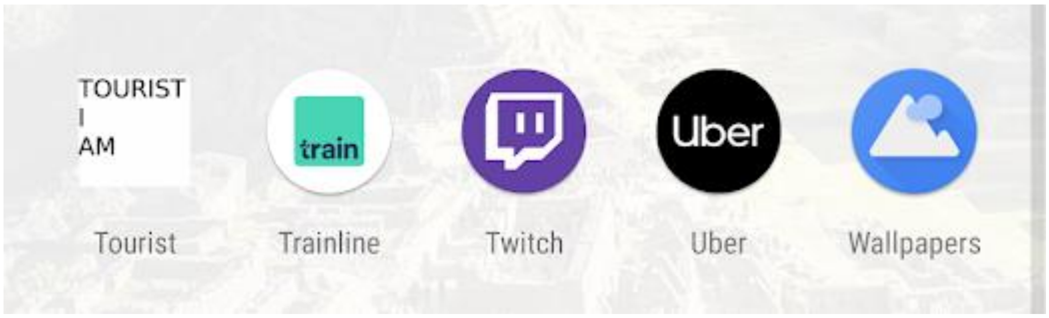


Figure 7 - Application Icon

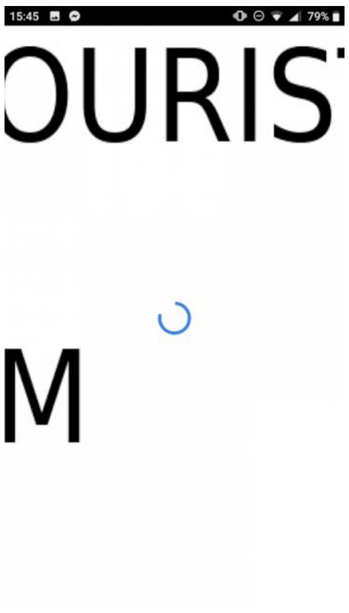
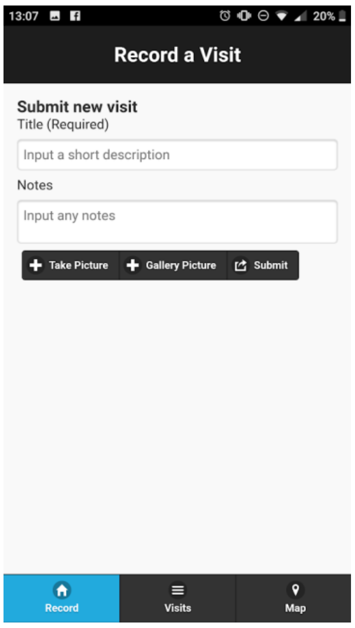


Figure 8 - Splash Screen not working as it should.


13:07 20%

Record a Visit

Submit new visit

Title (Required)

Input a short description

Notes

Input any notes

Figure 9- Empty form - This is where visits would be submitted.

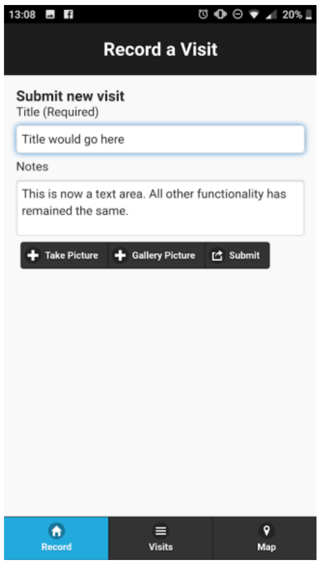


Figure 10 - form with text area. Filled in.

After testing the application on an actual device, I changed the notes section into a *textarea*. This is shown in Figure 9 and Figure 10. The following screenshots do not show it as a *textarea*. This doesn't change any other functionality.

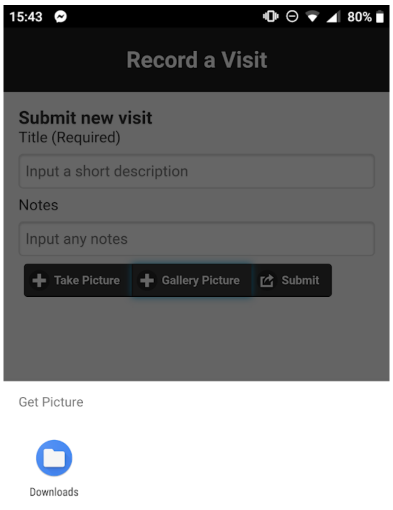


Figure 11- Selecting the "Gallery Picture" button

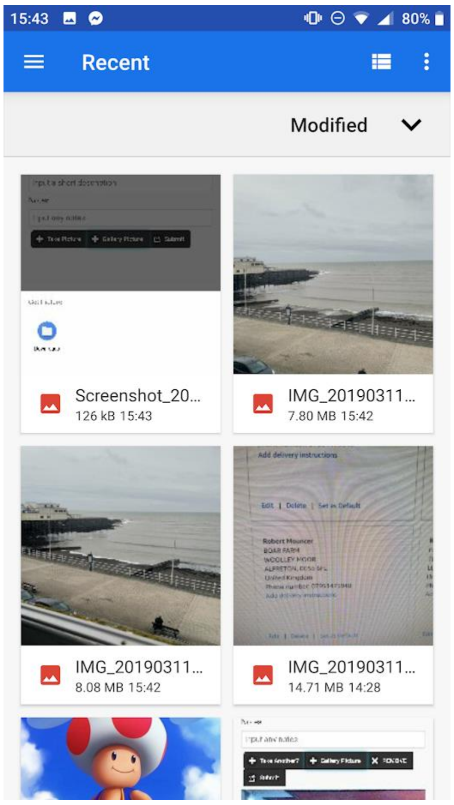


Figure 12 - A picture can be selected from the phone's files

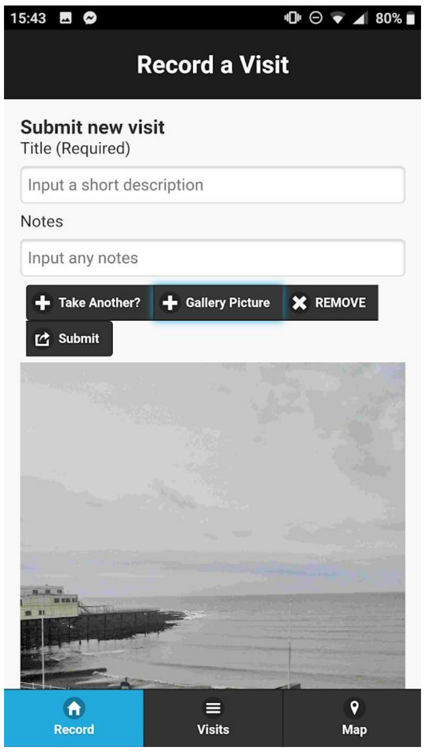


Figure 13 - Choosing a phone would then show it on the submit visit page

The fields were then filled in and submitted.

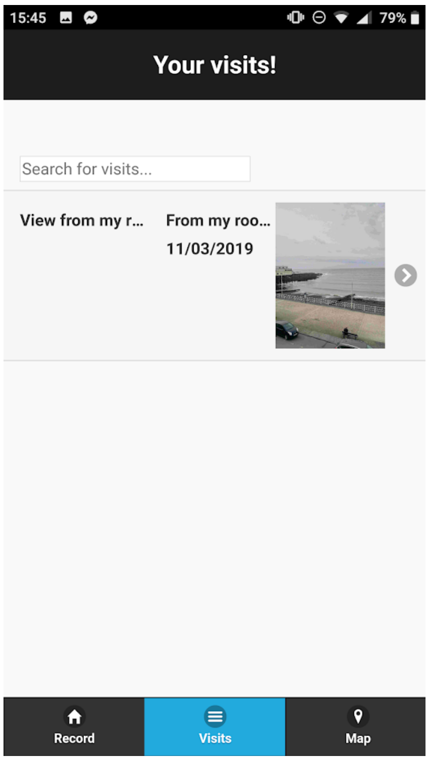
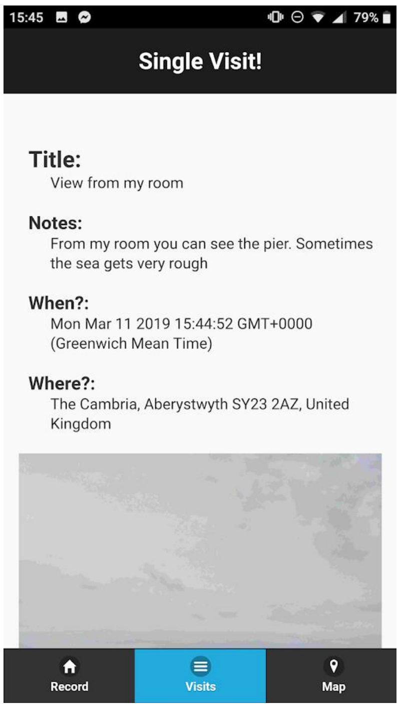


Figure 14 - The visit is then shown on the visits tab



Title:
View from my room

Notes:
From my room you can see the pier. Sometimes the sea gets very rough

When?:
Mon Mar 11 2019 15:44:52 GMT+0000 (Greenwich Mean Time)

Where?:
The Cambria, Aberystwyth SY23 2AZ, United Kingdom

Figure 15- The single visit is shown with details when clicked on

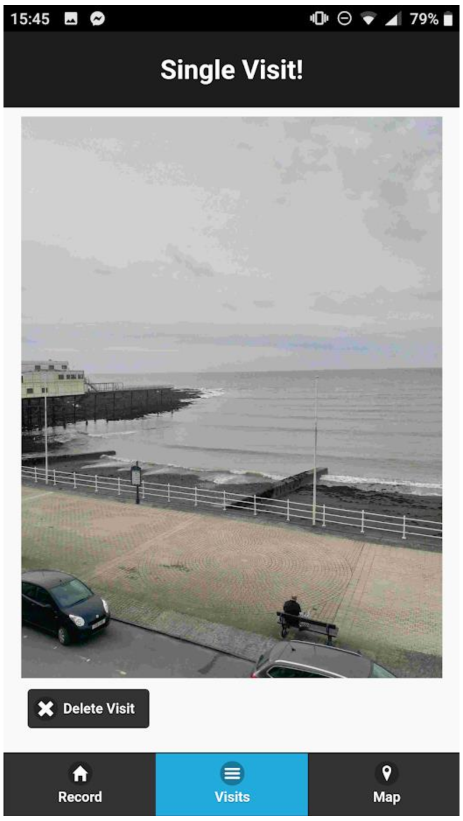
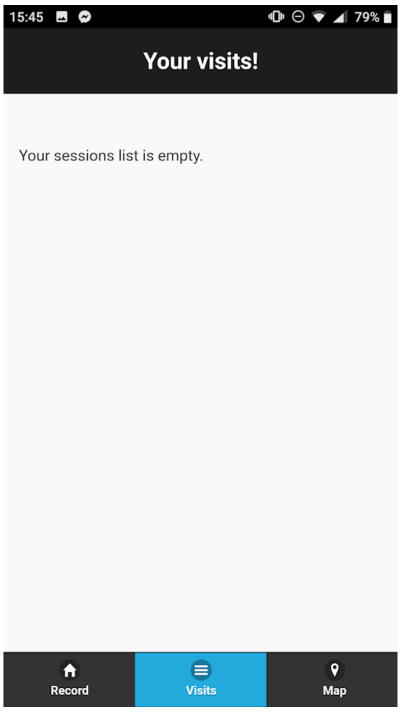


Figure 16 - Scrolling down shows the whole image and the delete button

The delete button was then pressed.



Your sessions list is empty.

Figure 17 - View when no visits are shown

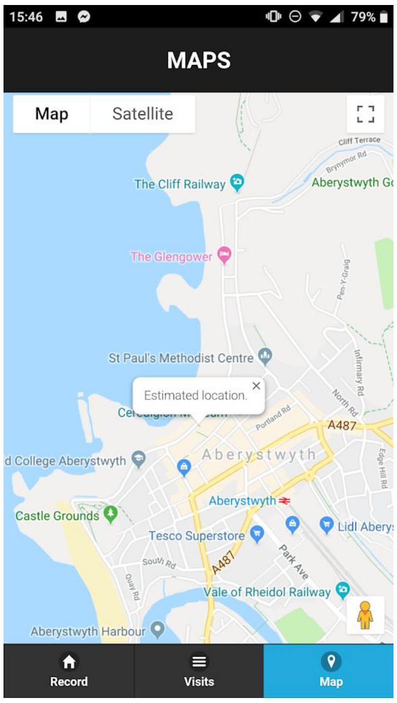
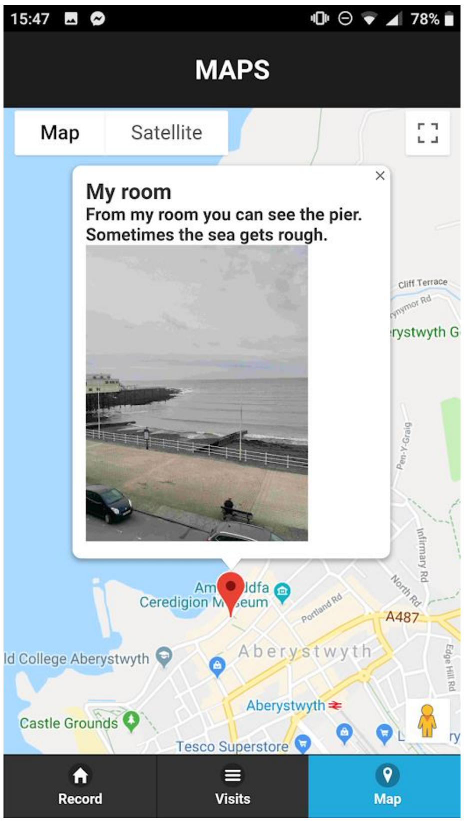


Figure 18 - Location shown on map load



My room
From my room you can see the pier.
Sometimes the sea gets rough.

Figure 19 - When clicking on the marker an information window is visible.

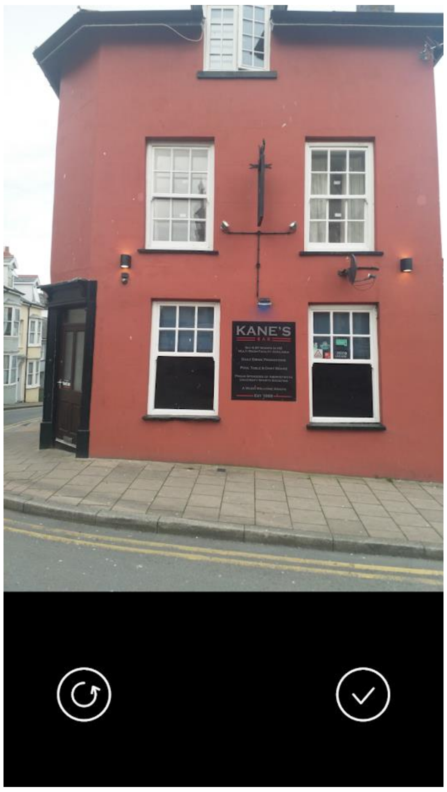
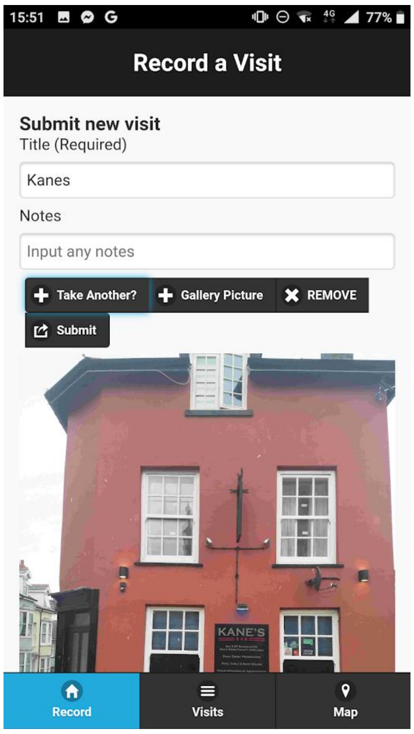


Figure 20 - Selecting the camera shows the camera screen. This is after the picture has been taken.



15:51 77%

Record a Visit

Submit new visit

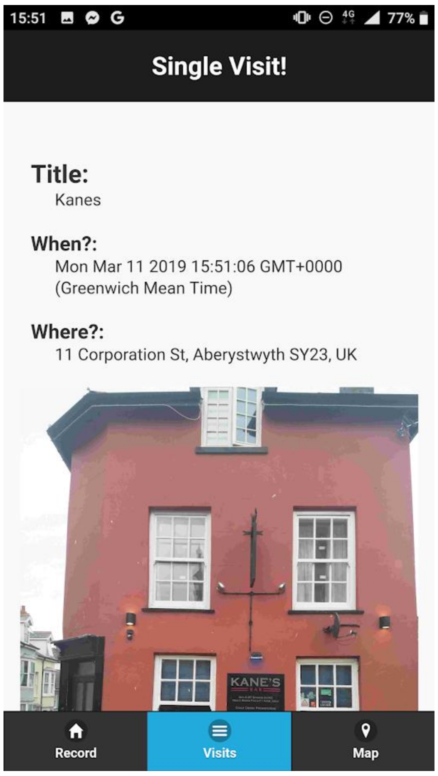
Title (Required)

Kanes

Notes

Input any notes

Figure 21- Camera submission with no notes (only title).


15:51 77%

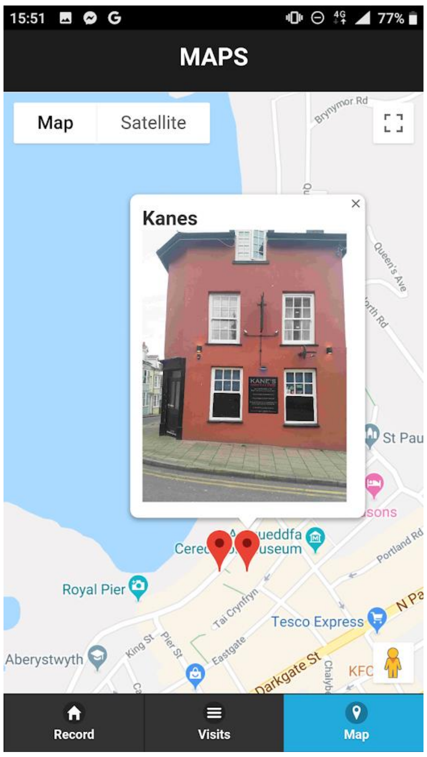
Single Visit!

Title:
Kanes

When?:
Mon Mar 11 2019 15:51:06 GMT+0000 (Greenwich Mean Time)

Where?:
11 Corporation St, Aberystwyth SY23, UK

Figure 22 - Single visit view of visit with no notes.


15:51 77%

MAPS

Figure 23 - Map showing submission.

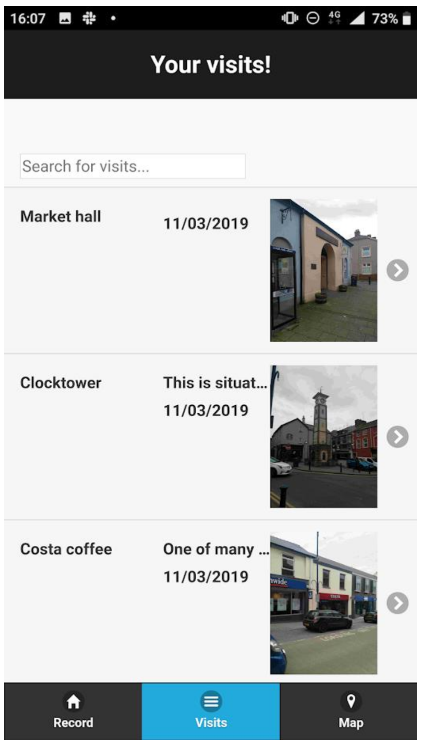


Figure 24 - Multiple submissions from Aberystwyth Town

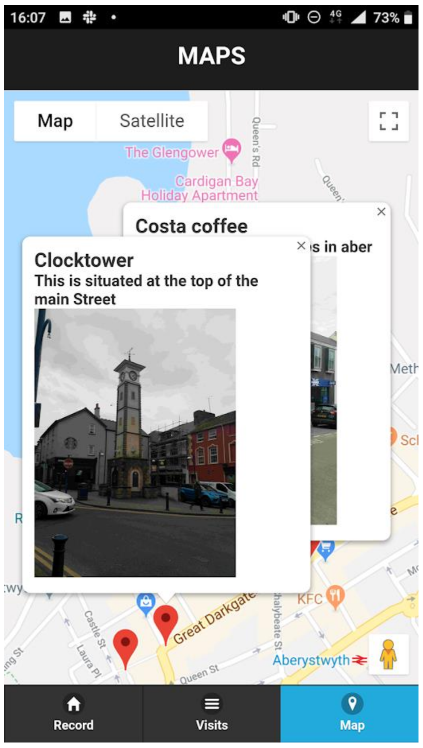


Figure 25 - Multiple information windows.

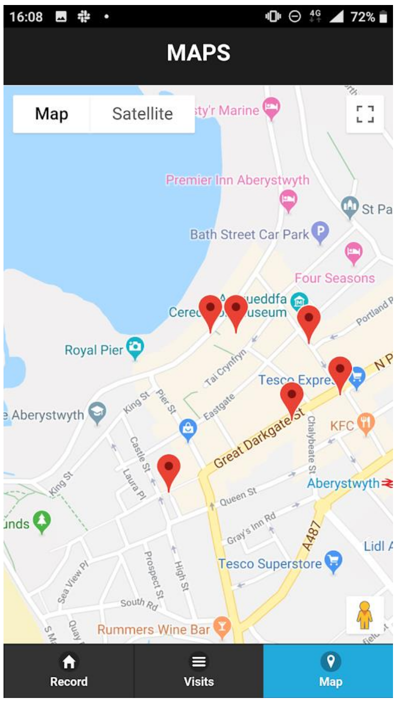
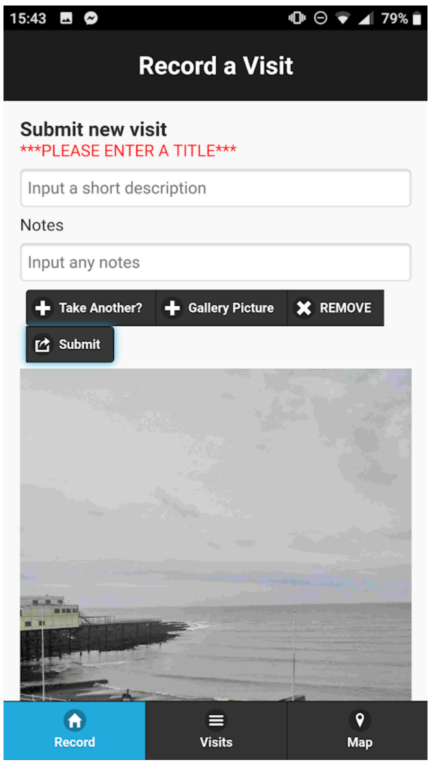


Figure 26 – Multiple markers.



Submit new visit

PLEASE ENTER A TITLE

Input a short description

Notes

Input any notes

Figure 27- Error shown when no title is entered.

Evaluation

Overall, I believe that the assignment was a success. I managed to get all the functionality working with additional features. Though the implementation may not be the best solution to the problem, it still is fully functional.

I found the assignment enhanced my web development skills which previously had been lacking. I learnt:

- How to use the *jQuery Mobile API*.
- How to test web applications using the web development tools.
- How to build hybrid web apps with the use of *PhoneGap*.
- How to use a local datastore for web apps.

I found the benefit of using Google Chrome Developer tools very useful and aided development. Each new feature that was implemented was tested using this before a collection of features were tested on a device. I would use these tools again.

The mark I would award myself for the various sections are shown in the table below.

Section	**Self-awarded mark**	**Reasoning**
Documentation	55-60%	I am aware that my literacy skills are not my strongest, though I feel the documentation covers everything required by the specification.
Implementation (record visit function)	70-75%	I found this easy to implement and covers all requirements from the specification. The view of the application could be improved.
Implementation (list visits function)	60-65%	Covers all requirements from the specification. The link to each visit opens a new tab on a PC when testing but when running on a mobile device, I believe it just refreshes the application.
Implementation (map function)	75-80%	This was very easy to implement. All functionality required is covered.
Testing	55-60%	I have shown a small amount of testing.
Flair	65-70%	I implemented a range of different features that weren't required in the specification. These features were easy to implement.