

Title: MOSBER Coursework Report

Author: Usman Basharat

Date: 6<sup>th</sup> December 2019

Course: COMP-1424 Mobile Application Development

Word Count:5666 (excluding table of content and references)

School of Computing and Mathematical Sciences



## Table of Contents

Table of Contents.....	2
Introduction .....	3
Section 1: Concise Table .....	3
Section 2: Bugs/Weaknesses .....	5
Section 3: Strengths .....	6
Section 4: Implementation .....	6
4.1. Android Application .....	7
Sporting Event Progression .....	12
Search.....	13
JSON .....	14
Search Validation .....	15
Add Event Validation.....	16
Additional Features.....	17
4.2. PhoneGap.....	20
Section 5: Evaluation.....	24
Future Aspects .....	25
References .....	26

## Introduction

In this report, I will be explaining my Mobile Sports Event Reporter (MOBSER) application of how this works. This is two applications that runs on Android in Java and PhoneGap in HTML and JavaScript. Within this report, I will be explaining in a concise table what I have been implemented within both applications. I will be including any bugs and weaknesses that I have implemented. I also will be naming the strengths too. This will include all the screenshots that I have named. Furthermore, I will be including a detailed evaluation of this application including aspects of HCI and security.

## Section 1: Concise Table

<b>Feature</b>	<b>Implementation</b>
<i>a) Enter details of events being reviewed</i>	Fully implemented – Once the user enters the details of the reviewed event, they are numerous required fields that need to be entered such as date, time, name of sport and name of participates. These are the required fields that the user must enter. Date has a DatePicker that enables a pop-up screen of options of the date selection. This is the same for time too. Name of sport has three in a spinner such as Football, Basketball and Ice Hockey. Lastly, name of participates has an EditText that enables the user to type in the participants taking place. All of these are required by the user for this to go to the next stage. The rest of the fields are Location, Email, Prediction and Comment. These are all of the optional fields that the user does not need to be filled in. Validation also has been taken place for empty fields of the required fields. A confirmation box also takes place where it shows the user what has been typed in. User can go back and change anything if necessary.
<i>b) Store, view and delete event details or reset database</i>	Fully implemented – Once the user enters the correct details, a confirmation box has also been inserted once the user is happy with the data. The user can wish to go back if they feel as though something is not right. If this is all happy, once they submit the information, this stores the information. To view this information, this will be in a list view where a loop can constantly update. This is a representation of the database that shows what goes in and what goes out of the database. To edit an event, you must click on the chosen one. There are two options, you can either edit the chosen event; or you can delete the chosen event. To delete all, it takes place where the view is of all events. Once the user has chosen to delete all, this deletes all of the data within the database i.e. resets the database.

c) <i>Enter the score as the sporting event progresses</i>	Fully implemented – The user can wish to enter the score of events as it progresses throughout the event. The user can wish to select the list of events in a spinner. The user can then type in the number of the team that has scored. This updates in the database and the updated score of the current team can be viewed straight away.
d) <i>Search</i>	Fully implemented – the user is able to search throughout the database. A simple form is that the user can type in the desired team and this would filter through the listview and match it with the desired event. Advanced search was attempted by using date-range of the desired event. This filters it through by attempting to select the two date ranges and views this dependant on the search. In addition, the user can also attempt to view the full details of the desired event by simply clicking on it.
e) <i>JSON</i>	Fully implemented – The format for JSON was sent successfully through to the cloud service. This is within the search that is clicked on by sending whatever is within the search. By sending this, the user can see the successful message appear with multiple events being submitted. The user can view this as readable results.
f) <i>PhoneGap platform</i>	Fully implemented – This is similar to the Android version as this has the same features. PhoneGap has the same required fields and optional fields like Android. A confirmation dialog appears if the user is happy. Once submit has been clicked, it goes straight to the database. PhoneGap shows how to store, edit, update, delete, and delete all for all features of this. In addition, view is also showed for this in a message format. Each format can be clicked and edited like the Android application. Validation for this has been inserted too. Any missing required fields does not let the user submit the data, unless the missing field is not empty.
g) <i>Additional</i>	Almost Implemented – This has been partly implemented as I have completed the current location of the user. This can show the user its current location. This has been implemented on the Android platform. In addition, the user can also search for any location by typing it in the search bar and it would locate this on Google Maps. In addition to this, also the user can show the event of a location in Google Maps too. This has been partly completed as PhoneGap additional has not been implemented. However, I do have three additional on Android that have been completed.

## Section 2: Bugs/Weaknesses

Throughout this coursework, I did face numerous of bugs that I felt that I could not fix. One issue that I came across is disabling the keyboard by typing `"android:focusable=false"` within the specific EditText. However, when this was put in to disable the keyboard for advanced search for two dates, it would not be able to pick up the date selected when getting the text. Therefore, this enables me to show the keyboard and make sure that the user has to double click to show the dialog for the date. This happens for both of the keyboards that leaves me with a bug for this section. If this bug had worked, it would enable the keyboard to disappear and prevent any users to type in anything when the date dialog appears. One issue that I do not understand for this that this has been working for when adding a date. However, I do not understand the issue when doing the exact same thing for advanced search date.

Another issue that can be considered as a bug is the same situation at hand is when the menu date is clicked, it would need to be able to click twice to view the options for advanced search date. I have tried to come up with a solution for this to be able to fix by making focusable false, but this does not work. In addition, once the user exits the search page after completing their search and was to return; it would return by clicking the date twice to view the advanced search date. Therefore, users need to be wary of this small issue.

A weakness that I consider is the main interface is very plain and does not look very appealing. I feel that the interface has been neglected and I didn't feel it was necessary at the time to make it appealing. However, I do consider this to be a weak point as every user interface needs to be appealing on the main page.

Another bug that may be considered is I put the hybrid PhoneGap application connected with the Android Application. However, this does not view function very well as I expected. When this is viewed, it enables the interface compact like the Android phone screen. However, the functions are unable to work. For this to work, once the 'Home' link has been clicked, it refers the user to Google Chrome to view the application on the browser. The application needs the browser to work and function, however, when viewed within the specific WebView, it is not enabled. This needs to be revised and considered when trying to merge both applications together.

Another bug that I tried to do was in Advanced when I tried to do Google Maps current location and search. It would have made sense if I had the combination together. However, I was left to separate both since whenever I searched; it was refreshing to the current location. It kept switching to the location and the searched item back and forth. Hence, due to this error, I had to separate both of this.

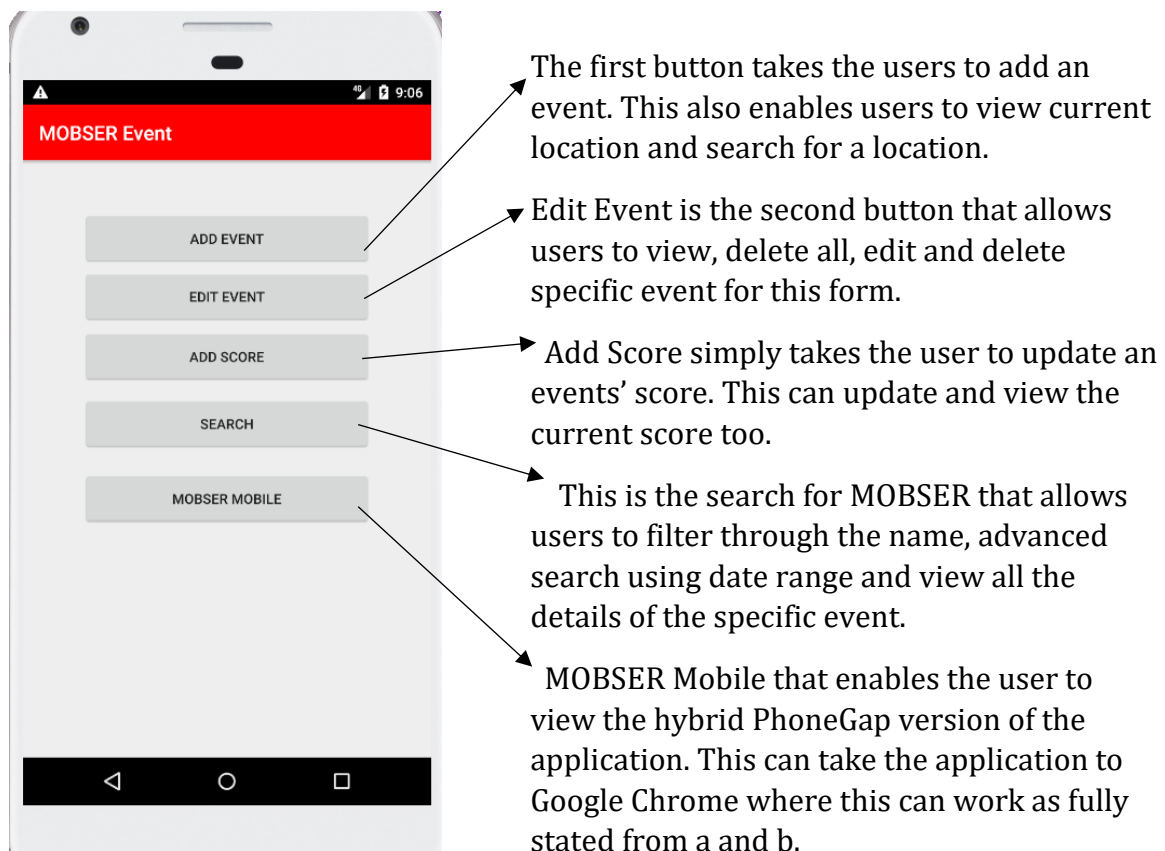
## Section 3: Strengths

I will be stating the strengths that both applications faces. These are the following:

- Database Singleton – this is a pattern that instantiates that restricts the class to be called to only once. This is a pattern that can be used and most commonly used for database classes. I also used this within the database class.
- SQL Injection – is a code technique that prevents hackers from gaining access to the database. This is a good habit to prevent this from happening. This is a strength because both, Android and PhoneGap applications, have SQL Injection.
- Model is separate from onCreate to prevent too much to happen on one class. This so that the all of the model for one class can be spread out into methods that makes it easier for the application to run. This has been complete for all classes for Android. Also, Model class has been used for common methods such as setting a message for easier use.
- Ensuring that the hard-coded strings are referred to using the strings. A good habit to keep in mind and make sure all can update at once too '@string/name'

## Section 4: Implementation

In this section, I will be explaining what I did throughout the report. This will include both Android and PhoneGap applications alongside screenshots of each section from a to g. Below shows the Main Activity which is the main page as soon as the user installs the application.



## 4.1. Android Application

The image shows a mobile application screen titled "MOBSEER Add Event". It contains several input fields with hints: "Click to select date (Required)", "Click to select time (Required)", a spinner menu currently showing "Football", "Teams (Required)", "Location (Optional)" with a red location pin icon, "Email (Optional)", "Comment (Optional)", and "Prediction (Optional)". At the bottom is a grey "ADD EVENT" button and a red location pin icon.

Figure 3 shows the required and optional fields for reviewing an event

The image shows the same "MOBSEER Add Event" form, but now with data entered: the date field contains "5/12/2019", the time field contains "9:39", the spinner menu is set to "Football", and the teams field contains "Arsenal vs Chelsea". The "Location (Optional)" field still has the red location pin icon. The "ADD EVENT" button and location pin icon are at the bottom.

Figure 2 shows the details of entered detailed event.

Figure 1 shows the required and optional fields that are necessary. These are required for the user to be able to fill in. If these are not filled in, an error will appear. Once the user is aware of this, as shown in the hints of each EditText, the user can click on date as shown in Figure 2. This enables the user to choose and select the time and date of the event. If this is not possible, the time allows the user to type by clicking the icon in the left-bottom corner.

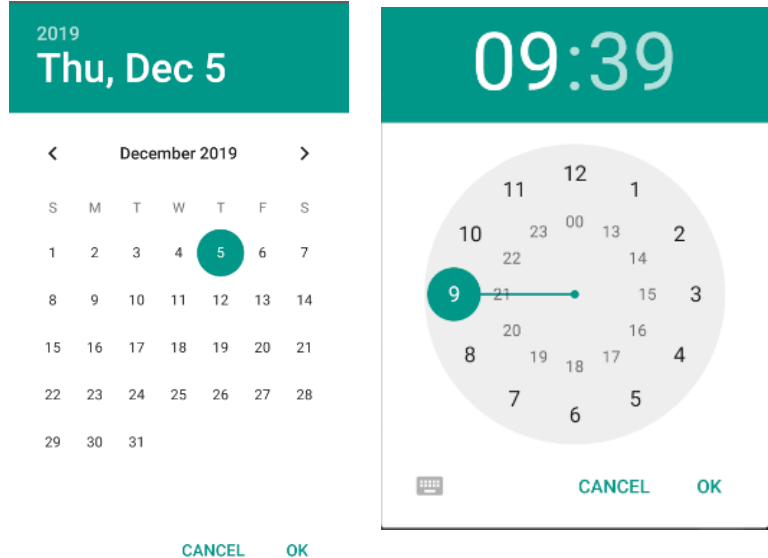
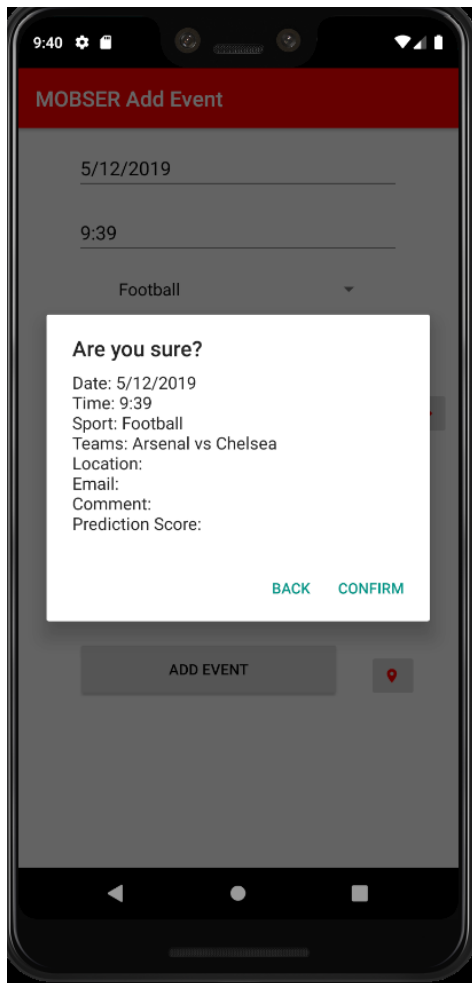


Figure 1 shows the dialogs once date and time has been clicked by the user.

Figure 3 shows this information being entered by the user. This shows that all of the required fields are entered, and the user is able to submit the data by confirming this. A spinner is used for the type of sport. Football is selected as default. This is up to the user to change the relevant sport for the relevant teams that are participating. This can be edited as this would be shown later on in the report. As soon as the user is happy with this, they can click add event and a confirmation box would appear confirming this.



This is a confirmation box using `AlertDialog.Builder` to create this message for the user. This shows what the user has typed into the form. If the user is happy with this, and by clicking confirm, the text typed into the database; it would be sent to the database. This is shown in Figure 6 and Figure 6. Figure 5 shows us the data that has been inserted within SQLite to confirm that all of this information has inserted. Figure 5 shows us that the Toast message that shows after the confirm button has been clicked to let users know that this has been successful.

If the user has checked this information and is not happy with what has been typed in and wishes to change; the user can simply click back. This would enable the user to change the data without it being inserted into the database. Figure 2 would have been shown if this was the case. All of the information that has been typed would not change and can let the user change if necessary.

Figure 4 shows the confirmation box for the user.

```
SELECT * FROM addEvent;
```

Export	ID	DATE	TIME	SPORT	TEAMS	LOCATION	EMAIL	COMMENT	PREDICTION	ACTUALSCORE
1	1	5/12/2019	9:39	Football	Arsenal vs Chelsea	NULL	NULL	NULL	NULL	NULL

Figure 6 shows the data inserted within SQLite Manager

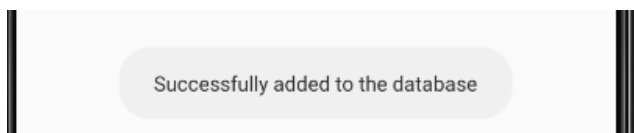


Figure 5 shows the data has been inserted successfully.



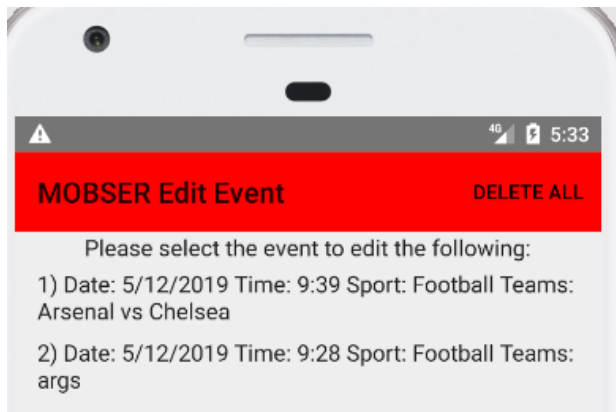


Figure 7 shows a list view of all information in the database.

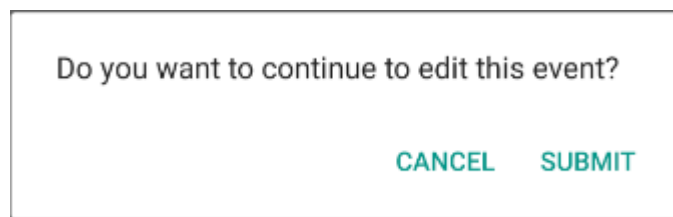


Figure 8 shows the alert message to enable editing for an event

Figure 7 shows us the view of what has been inserted within the database. This shows whether all events that are submitted are successful. Users are advised to use this list to show whether their input is successful or not. As you can see, Figure 2 data has been shown successful in this list. This means that this was successfully shown and can be viewed within this list.

Figure 8 shows us a message of whether the user would like to continue to edit the information. This is viewed once the user has clicked on the specific one that has been wished. Once the submit has been clicked, it enables the user to view full information of the specific event. Figure 9 shows the specific data being changed.

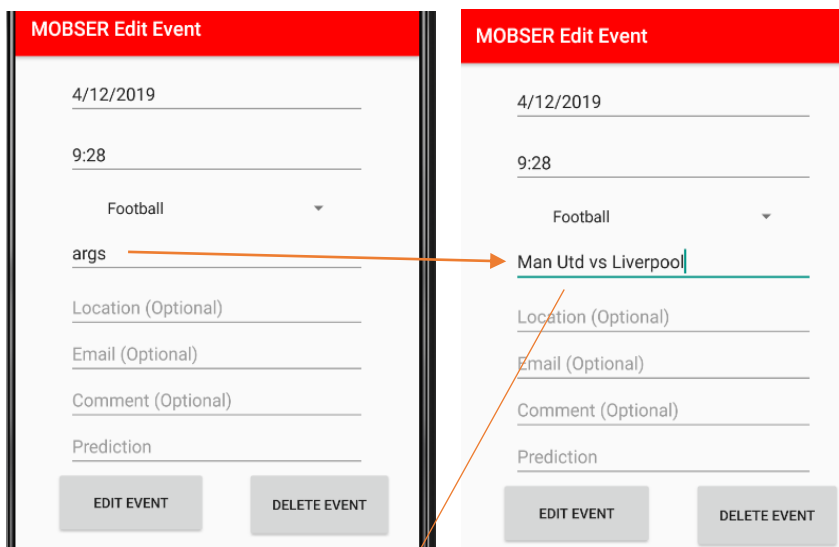


Figure 10 shows the information of this data and what has been changed.

Referring to Figure 9 and Figure 10, this shows us that the edited event has been successful within the database. I can change this from 'args' to Man Utd vs Liverpool. This is important to change as users can make a mistake as shown. This can be a chance to change what has been typed in. Also below shows us in SQLite Manager that this matches the both of the list view.

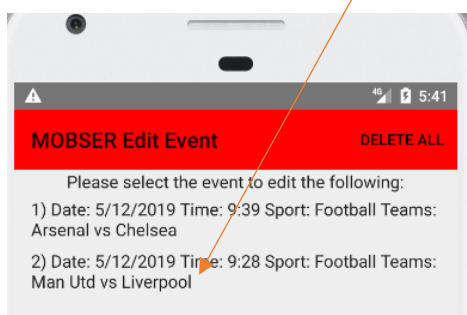


Figure 9 shows us that this has updated.

```
SELECT * FROM addEvent;
```

Export	ID	DATE	TIME	SPORT	TEAMS	LOCATION	EMAIL	COMMENT	PREDICTION	ACTUALSCORE
1	1	5/12/2019	9:39	Football	Arsenal vs Chelsea	NULL	NULL	NULL	NULL	NULL
2	2	5/12/2019	9:28	Football	Man Utd vs Liverpool	NULL	NULL	NULL	NULL	NULL

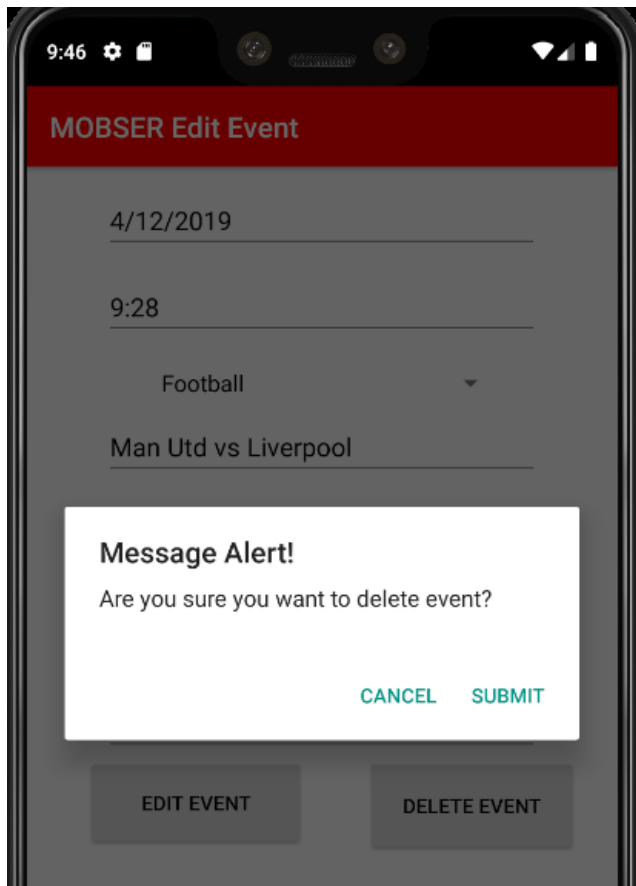


Figure 11 shows the process of deleting a specific event.

Figure 11 shows us the similar way to update, it is the same to delete. The process is the same to get to this stage. You have to click the one you wish you want to edit. In this case, it's Man Utd vs Liverpool game. Figure 8 shows this message. If you want to delete, simply click the delete event and this enables this to be deleted. Figure 12 and Figure 13 both show us that once delete has been submitted, this is deleted from the database. You can compare this to Figure 9 database and it shows that it has been deleted.

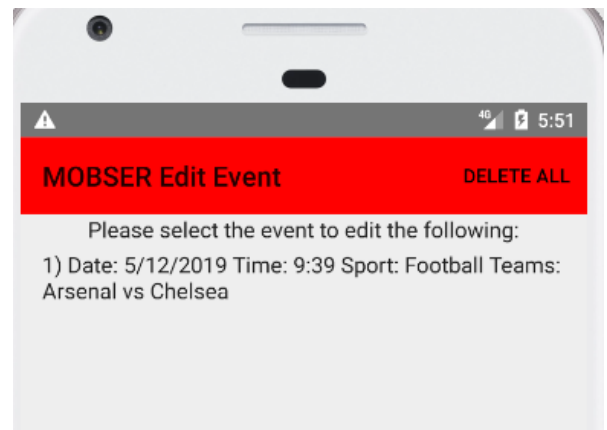


Figure 12 shows the list view that the delete was successful.

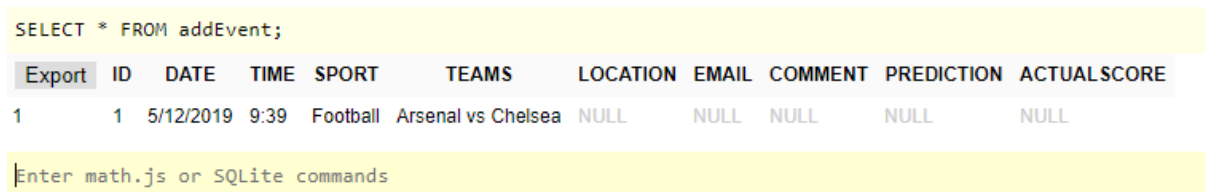


Figure 13 shows us that this was successful within SQLite Manager.

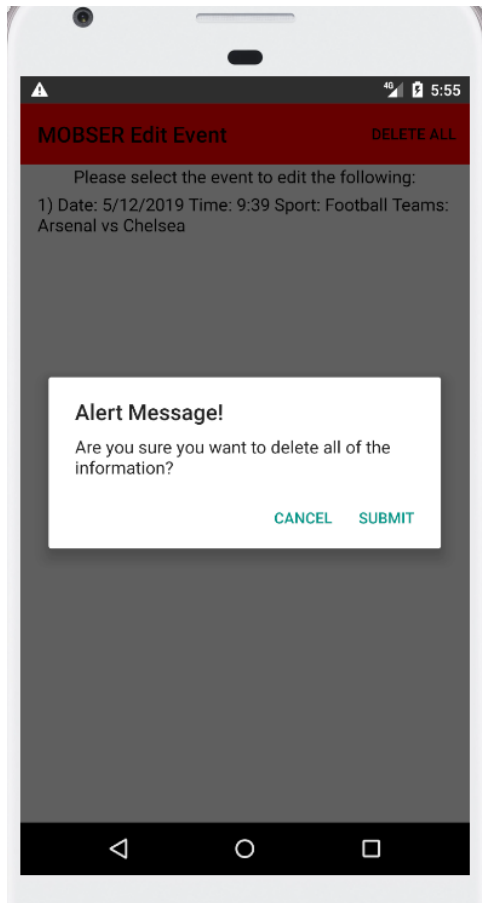


Figure 14 shows that a confirmation box appears to confirm to delete all from the database.

Figure 14 shows us that the user has an option to delete all from the database. Exactly like deleting and editing information, the user has a confirmation dialog to confirm that they are certain to delete all information from the database. This works the same as the others. Once submit has been clicked, this deletes all information from the database. Figure 15 shows us that all of the information is cleared, and this lets the user know by stating this in a toast.



Figure 17 shows that all of the content has been cleared.

```
SELECT * FROM addEvent;
```

empty output

Figure 16 shows that the database is empty.

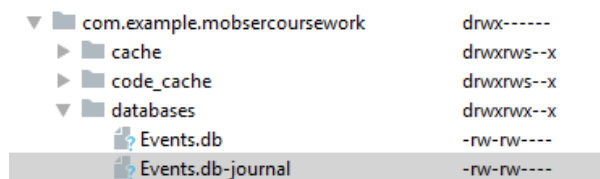


Figure 15 shows the steps how to view this within SQLite Manager.

Figure 16 and Figure 1 both show that this has the view of list view matches what has been typed into the database. The reason that I have showed the steps match throughout all of this is to prove that these updates similar within the database. Figure 15 shows the steps that I took to access the database that has been created within the database. This can be viewed by using the following :

**Device File Explorer → Data → Data → com.example.mobsercoursework → databases**

This can be downloaded and dropped down to the extension of SQLite Manager and this shows these tables of what is inside the database.

## Sporting Event Progression

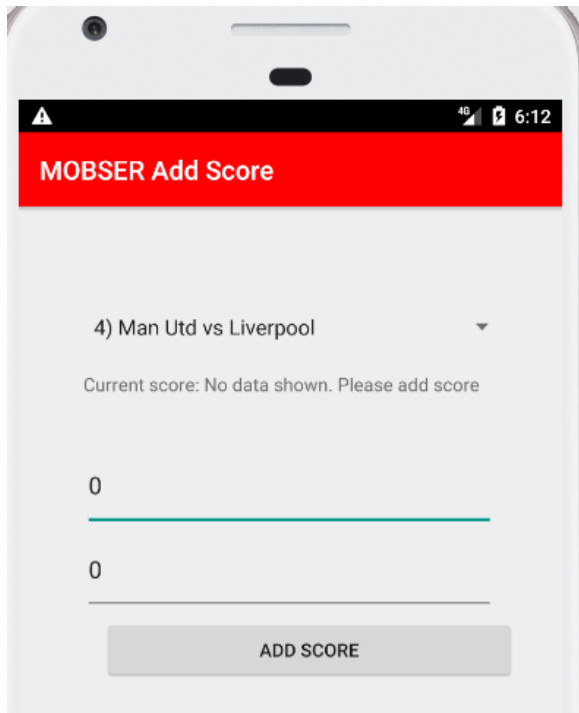


Figure 18 shows the process of add score.

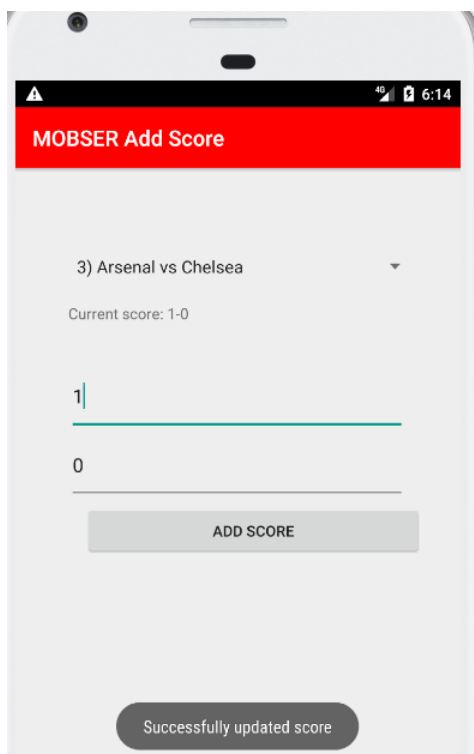


Figure 21 shows that the score is updated.

Figure 19 shows us the interface of how the score would look like if it has not been updated. It tells the user that the score is not updated, and it needs to be updated. Figure 20 updates the score by putting 1-0 to Arsenal. This shows the user the score has been updated within the database. Figure 18 shows the filter within each score. For example, if the user was to switch matches to check the current score, they would need to click Man Utd vs Liverpool, and it would display Figure 19 saying that no data has been inserted. If this was updated, it would show the relevant score. This is important to keep each score different to show the differentiation of each score. It cannot be muddled up. Users would want to be informed about each score. This filters it for the user. Figure 20 shows us that this is the same within the database in SQLite Manager.

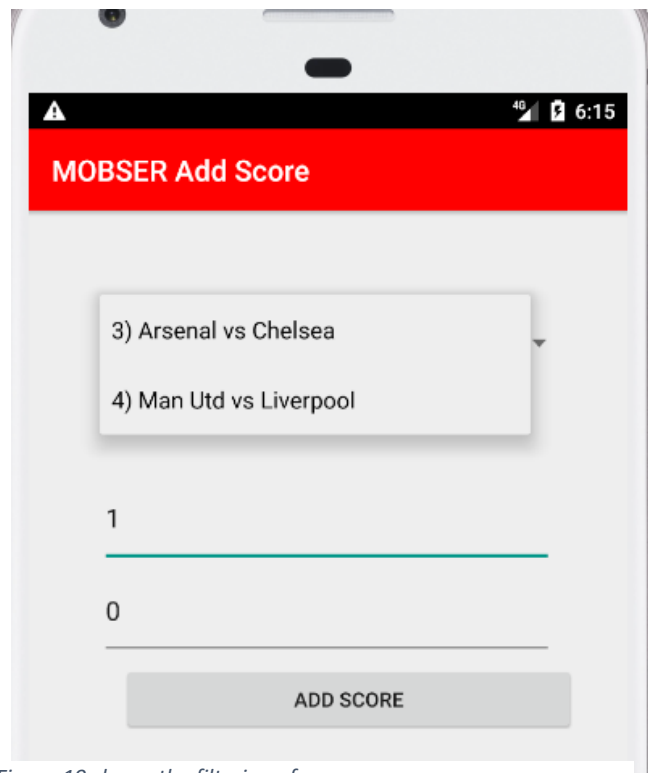


Figure 19 shows the filtering of score.

```
SELECT * FROM addEvent;
```

Export	ID	DATE	TIME	SPORT	TEAMS	LOCATION	EMAIL	COMMENT	PREDICTION	ACTUALSCORE
1	3	5/12/2019	9:39	Football	Arsenal vs Chelsea	NULL	NULL	NULL	NULL	1-0
2	4	5/12/2019	9:28	Football	Man Utd vs Liverpool	NULL	NULL	NULL	NULL	NULL

Figure 20 shows that this has been updated within the database.

## Search

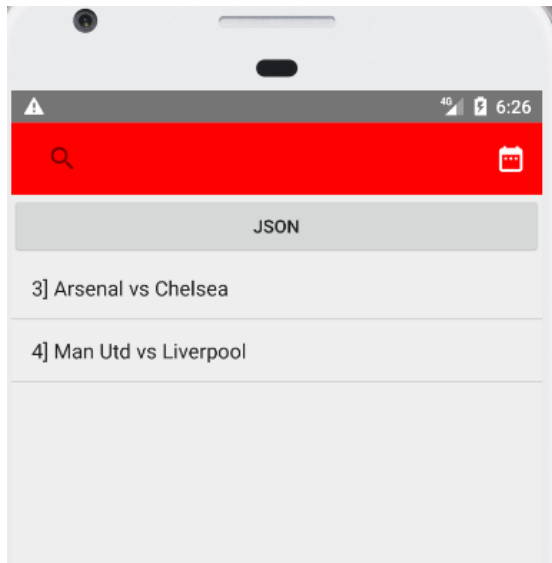


Figure 22 shows the search list view.

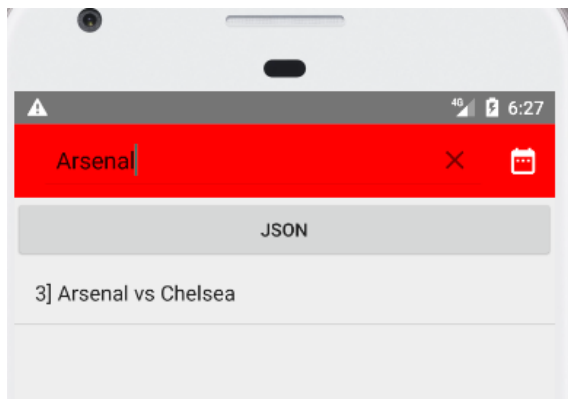


Figure 24 shows the filter for name of search

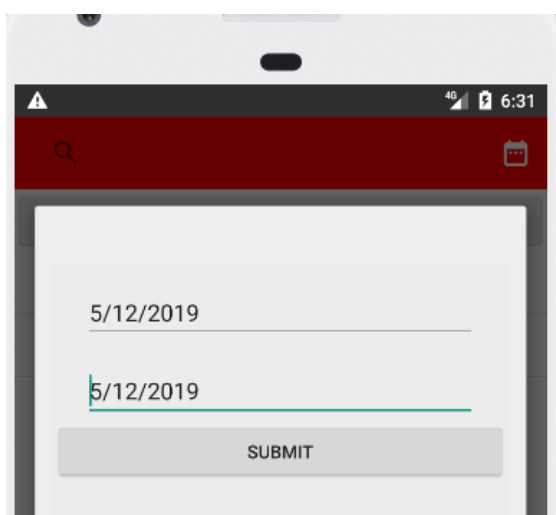


Figure 25 shows us the date range selection for search.

Figure 22 shows us a list view of the event names. This can be filtered if the user types in the first few letters of the event name. For example, Figure 24 shows us that if you type in Arsenal, the relevant event appears in front of you.

I made some amendments to this by changing the date of Arsenal vs Chelsea game to 4<sup>th</sup> December 2019 to show the relevant search. Figure 25 shows us that by typing the date range specific, you can show the results by Figure 23 shows us. Only one result was found as I stated I changed one of the results to yesterday to prove that this search query works. Once you click on each of the event, it can show you all of the information in detail of what contains in it.

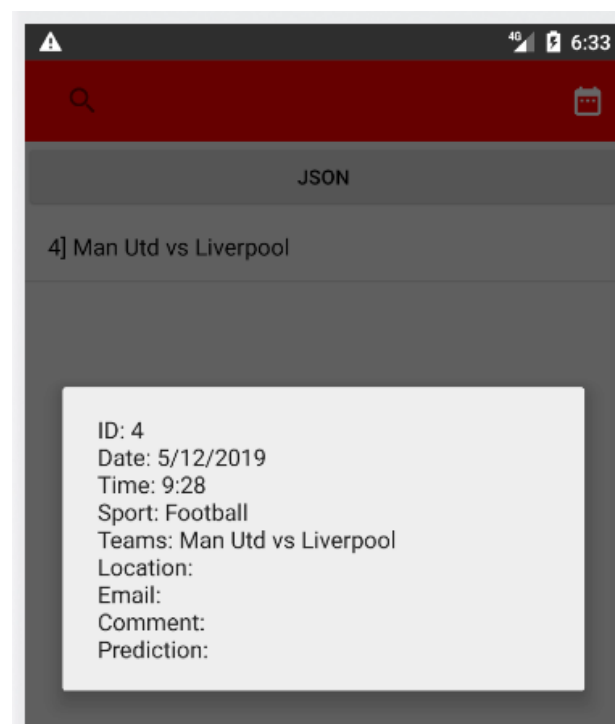


Figure 23 shows the results of the advanced search.

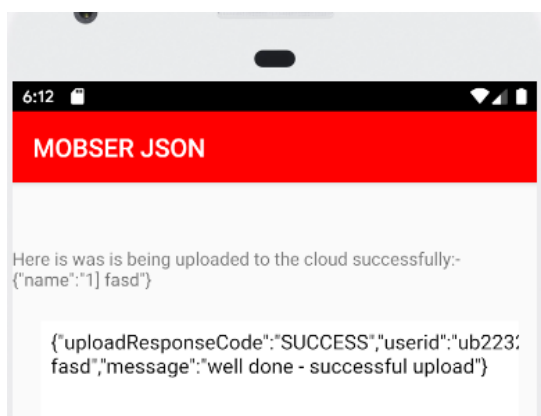
## JSON



Figure 26 shows the list view of what is going to be sent to the cloud for JSON



Figure 27 shows us the message of what has been sent



This shows the results of what has been uploaded for the user to see.

Figure 26 shows us the list of what is going to be sent to the cloud. This has been done by sending it to (<http://gillwindallapp1.appspot.com/madservlet>). This is a web service and this should display a message dependant on its success or not. Figure 27 shows us that this has been successful. Figure 27 also shows its flexibility by inserting its 2 events into the cloud. This has been recognised as a response as it shows the number of what it contains as it states 2. The format is important for the user to be readable to show the user the success of its upload by stating, "well done – successful upload".

## Search Validation

Referring to Figure 28 and Figure 29, this both shows the validation that has been done for both advanced search dates. If the user has left it empty, a Toast is shown to the user letting them know that the date is empty for the field. The designated field is shown by a `setError` stating which of the fields are empty. Once both of these dates have been entered, this then disables the `setError` and checks that there is something in the field. This enables the user to submit the data that is given and show the relevant data. This is for an empty validation for search.

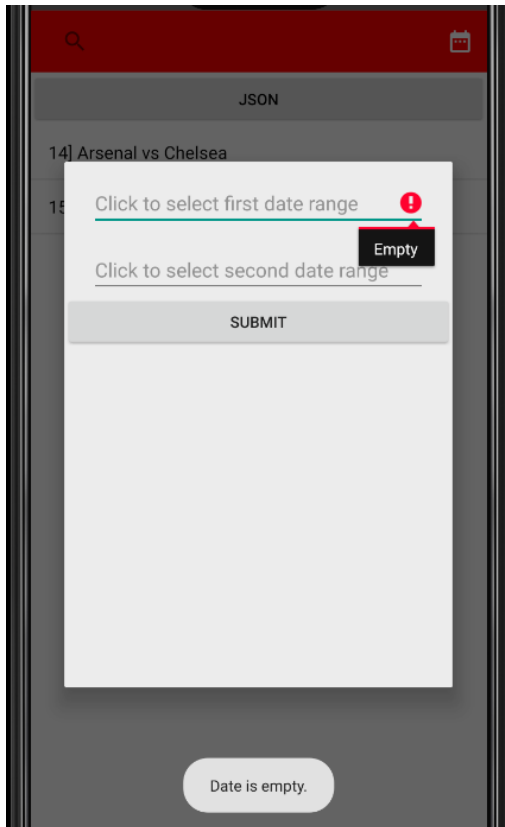


Figure 29 shows the validation of advanced search

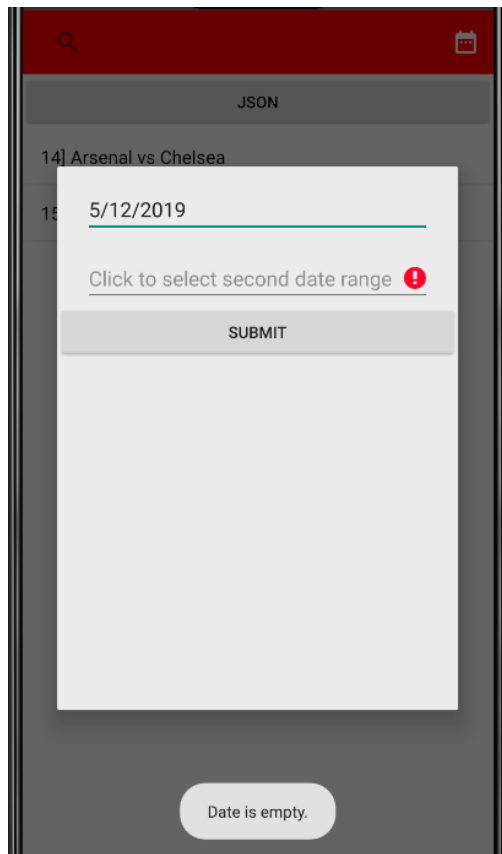


Figure 28 shows the validation of date search

### Add Event Validation

This is a validation for adding an event. These are required fields that must be entered to add the event specified. Figure 30 shows us that a date is a required field. This must be entered. However, if the user submits the button without entering any of these required fields an error occurs letting the user know that this is a required field. This is the same for Figure 31 and Figure 32 too.

Figure 33 shows us the result that once these required fields have been filled then they would be no errors within these fields. However, the optional fields have validation too. This makes it validated by whatever the user types in, it enables an error message to occur. However, as this is an optional field, it is not required; it does not stop the user entering this field. All of these fields shown in Figure 34 are the same for the rest of the optional fields. All of these prevent unnecessary keys to be entered. Email has an email validation. Prediction has a hyphen to be allowed.

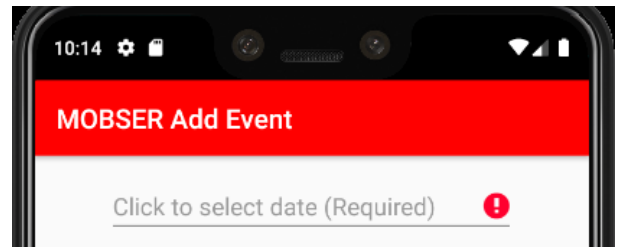


Figure 30 shows the date validation.

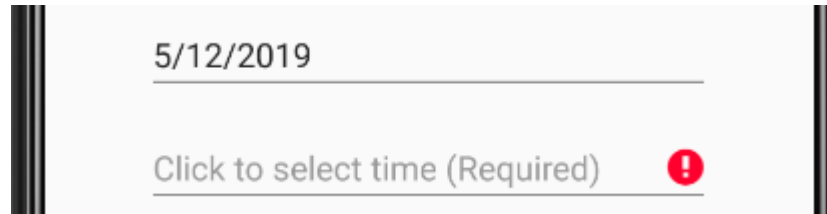


Figure 32 shows the error for time

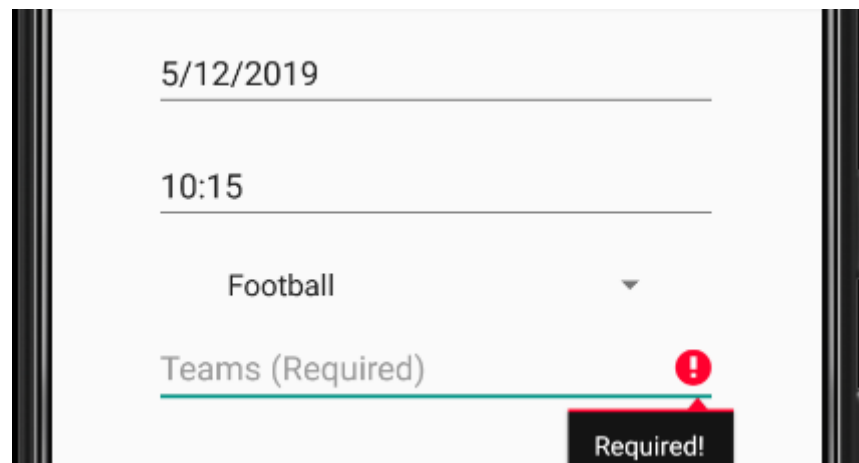


Figure 31 shows the required field for teams.

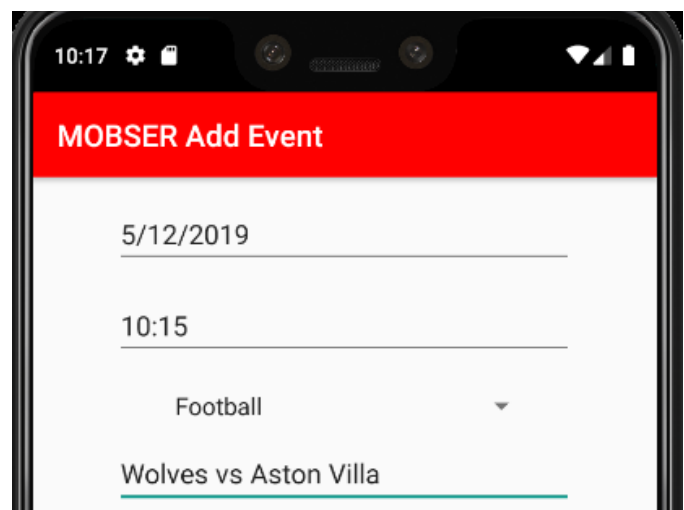


Figure 34 shows the result.

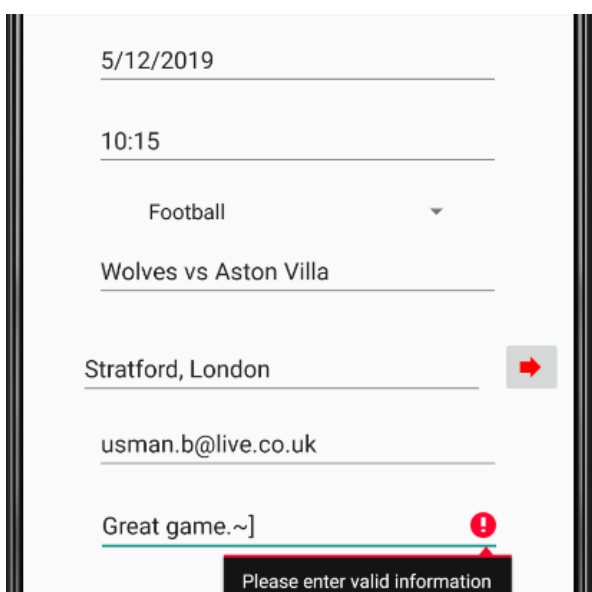


Figure 33 shows the optional fields have validation too.



## Additional Features

### Current Location Google Maps



Figure 36 shows the emulator's location

Therefore, I did further test to test my current location on my phone. I downloaded the current application's APK and downloaded this within my phone to test this. Figure 36 shows the current location that was tested on my current phone.

Please note that refresh button is for the Google Maps refresh its current location. The marker on the maps tells us the current location on Google Maps.

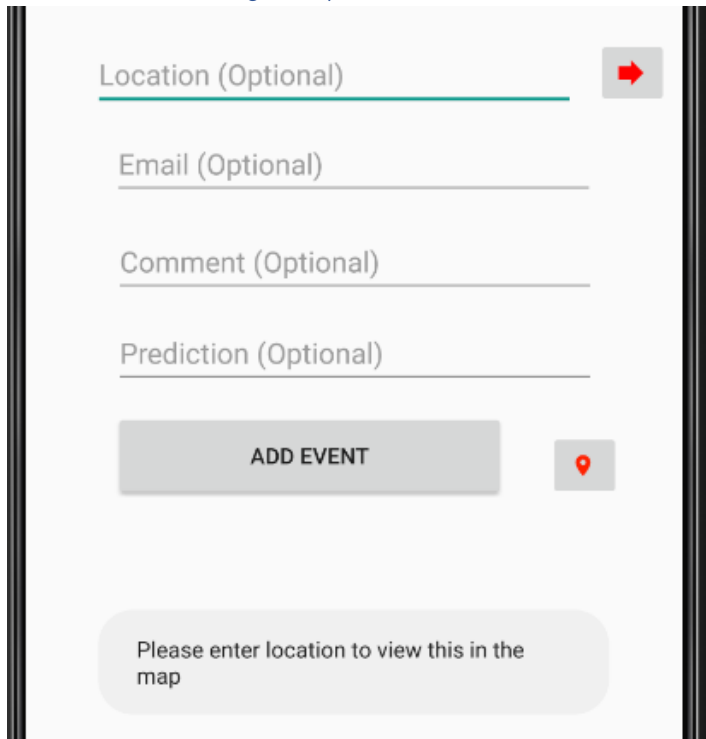
This was enabled my Android has already given an example of Google Maps in an activity ready as a template. I used this template and added my unique API. Once this was enabled, I was able to start doing location for this

Please refer yourself to Figure 2 as this shows the Add Event template. You can see a location Image Button. This shows the user its current location. Please do not hesitate as the current location shown by Google Maps represents the Emulator's location that is located in America. To view this location, the user must enable location and permission within their phone to view this. In addition, the user must press refresh to enable the current location.



Figure 35 shows the current location on my personal phone.

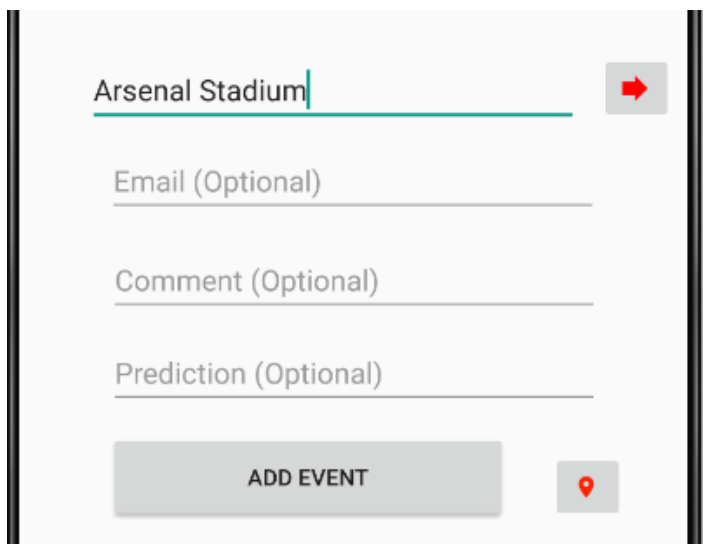
### Search Location Google Maps



The form contains four input fields: 'Location (Optional)', 'Email (Optional)', 'Comment (Optional)', and 'Prediction (Optional)'. The 'Location (Optional)' field is highlighted with a red border and a red arrow icon to its right. Below the fields is a grey 'ADD EVENT' button with a red location pin icon to its right. At the bottom, a light grey rounded rectangle contains the text 'Please enter location to view this in the map'.

Figure 37 shows the validation for the optional field. By clicking the arrow button, and the field is empty; a message is displayed as shown. Figure 38 and Figure 39 both show the result once the button has been clicked. This shows the result on Google Maps of the typed in field. If the user would like to see another location, they would need to type it in to see. Figure 40 would show this for you. Also, this gets the location straight away. Users has to be aware to enable location and permission for this to happen.

Figure 37 shows the validation for the field.



The form is identical to Figure 37, but the 'Location (Optional)' field now contains the text 'Arsenal Stadium'.

Figure 39 shows the user has entered the location of the event

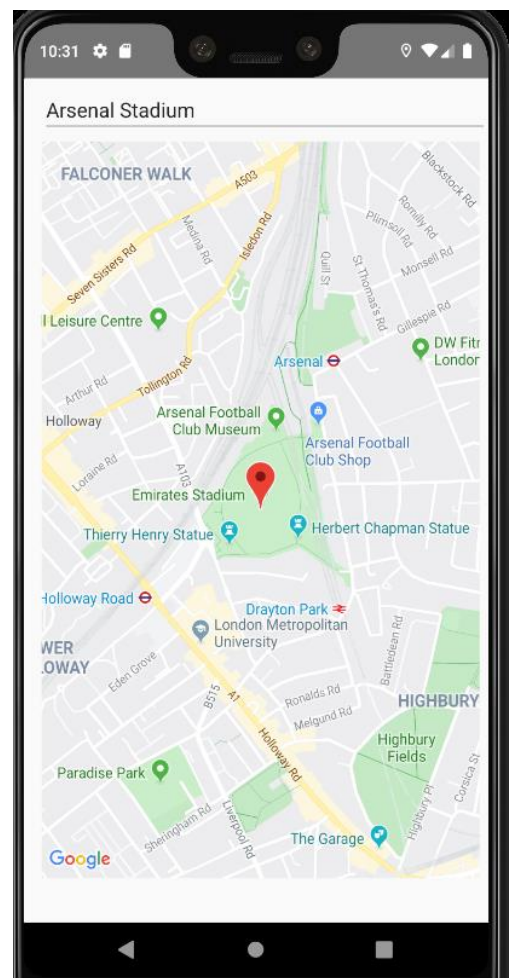
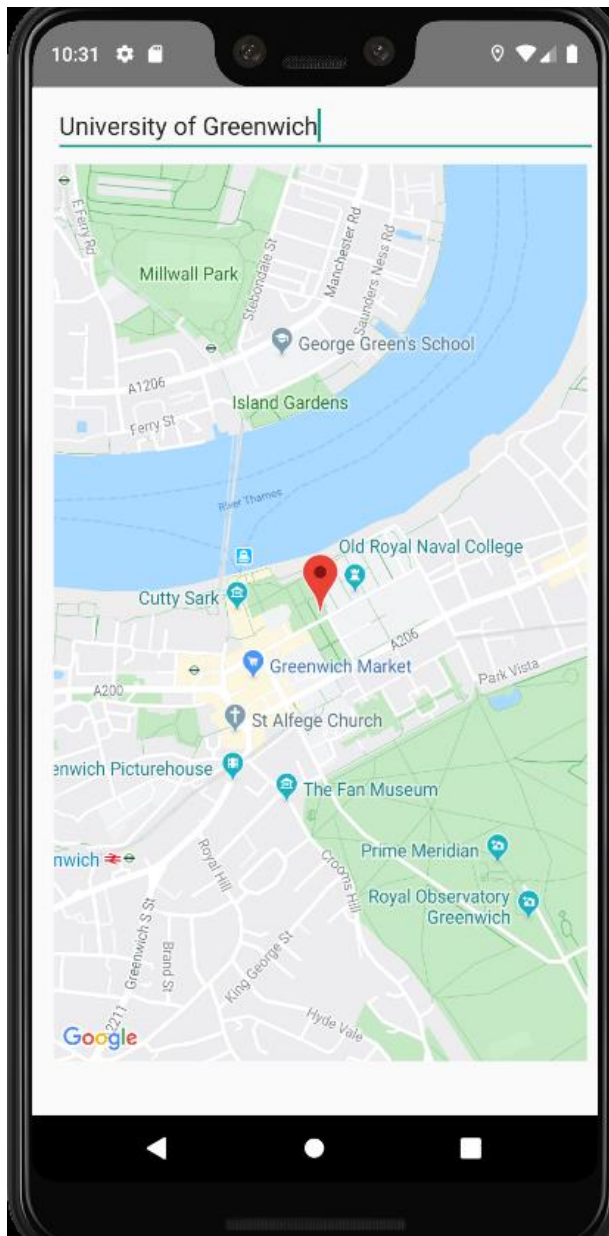


Figure 38 shows the location typed in.



Referring to Figure 40, this shows that the user can enable to search for a location of this event. This is to prove that not only the user can display the location of an event, the user can search for a location too. Figure 40 shows that I have typed in University of Greenwich, and as displayed on the maps; it shows the university. As stated, as soon as the search has typed; it would enable the location to keep refreshing for what has been typed in. If something that Google Maps has been typed in that is not recognised, a Toast message would appear saying that is unable to search for this.

Figure 40 shows the search for Google Maps.

4.2. PhoneGap

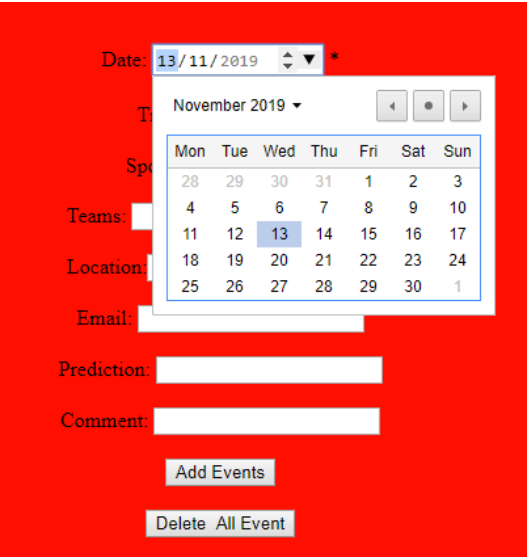


Figure 42 shows the options of date picker in Javascript.

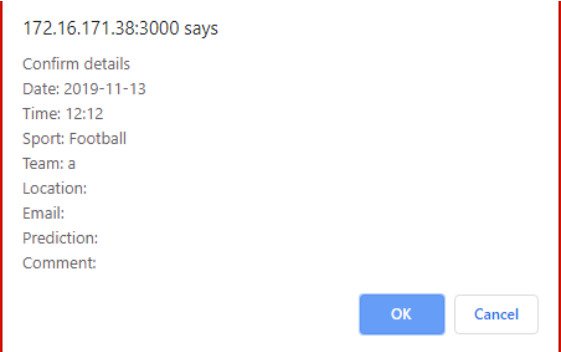


Figure 41 shows a confirmation box

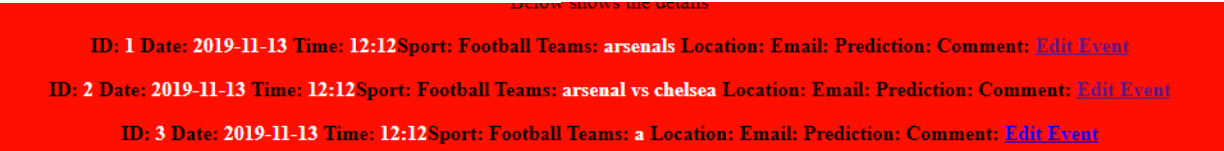


Figure 43 shows list of the events.

ID	ID	date	time	sport	team	location	email	prediction	comment
1	1	2019-11-13	12:12	Football	arsenals				
2	2	2019-11-13	12:12	Football	arsenal vs chelsea				
3	3	2019-11-13	12:12	Football	a				

Figure 44 shows the Web SQL database

Figure 41 and Figure 42 both shows the process of what the user must do in order to add an event. This is similar to Android as the user must go through the required fields as it has a star next to it. This shows that the user cannot enter an event as soon as this is done. Figure 41 shows the confirmation box of what has been typed in. Figure 43 shows that this has been inserted into the database and this is ready to edit/delete the event as shown.

Figure 44 shows the Web SQL inserted within the database. This matches Figure 43 by confirming that Figure 41 has been inserted within the database. Figure 43 also shows us that the data that is inserted is white. This is to show no confusion and enable the entered data to be brighter and clearer to see.

Figure 45 shows us that this can be done by clicking the Edit Event link shown in Figure 43. By clicking this, this collects all information from that row including the ID and puts it in the URL. By doing this, it can be read by using Windows Search Location in JavaScript. By getting all this information, you can rewrite each input by what it has on each row. Enabling this is shown in Figure 45. Once this has been done, the user can type in the specific detail to change. Figure 46 shows the message that this has been done. Figure 47 shows us that this has been updated in the list view message. Figure 48 shows all of this process has been successfully completed within the database.

172.16.171.38:3000/edit.html?id=3&&date=2019-11-13&&time=12:12&&sport=Football&&team=a&&location=&&email=&&prediction=

Date: 13/11/2019

Time: 12:12

Sport: Football

Teams: afs

Location:

Email:

Prediction:

Comment:

Edit Events

Delete Event

Figure 45 shows the user can edit this item.

172.16.171.38:3000 says

Successfully updated

OK

Figure 46 shows that the user has successfully updated this.

Below shows the details

ID: 1	Date: 2019-11-13	Time: 12:12	Sport: Football	Teams: arsenal	Location:	Email:	Prediction:	Comment: <a href="#">Edit Event</a>
ID: 2	Date: 2019-11-13	Time: 12:12	Sport: Football	Teams: arsenal vs chelsea	Location:	Email:	Prediction:	Comment: <a href="#">Edit Event</a>
ID: 3	Date: 2019-11-13	Time: 12:12	Sport: Football	Teams: afs	Location:	Email:	Prediction:	Comment: <a href="#">Edit Event</a>

Figure 47 shows that this has updated in the list message view too.

ID	date	time	sport	team	location	email	prediction	comment
1	2019-11-13	12:12	Football	arsenals				
2	2019-11-13	12:12	Football	arsenal vs chelsea				
3	2019-11-13	12:12	Football	afs				

Figure 48 shows the database updated in Web SQL.

172.16.171.38:3000 says  
Are you sure you would like to delete?

OK
Cancel

Sport: Football

Teams:

Location:

Email:

Prediction:

Comment:

Edit Events

Delete Event

Figure 49 shows that the user can delete the specific event.

Figure 49 shows us that the user can choose to delete the specific event. This can be completed by clicking Edit Event shown in Figure 50. Once this has been clicked, it would show two options. For this, it has been deleted within the database. Figure 50 and Figure 51 both correlate and prove that this has been deleted from the database by specific ID. This works similarly as edit. This gets the ID from the URL as all this information is gained from the Edit Event link that is passed to the URL.

Below shows the details

ID: 1 Date: 2019-11-13 Time: 12:12 Sport: Football Teams: **arsenals** Location: Email: Prediction: Comment: [Edit Event](#)

ID: 2 Date: 2019-11-13 Time: 12:12 Sport: Football Teams: **arsenal vs chelsea** Location: Email: Prediction: Comment: [Edit Event](#)

Figure 50 shows that this has been successfully updated and deleted in the database.

ID	date	time	sport	team	location	email	prediction	comment
1	2019-11-13	12:12	Football	arsenals				
2	2019-11-13	12:12	Football	arsenal vs chelsea				

Figure 51 shows this in the database

Referring to Figure 52, this shows us that the user has an option to delete all information from the database. This can be done on the same page as the button once the button has been clicked, it shows a confirmation dialog. Once all of this information has been deleted, it is cleared from the database as Figure 53 is shown.

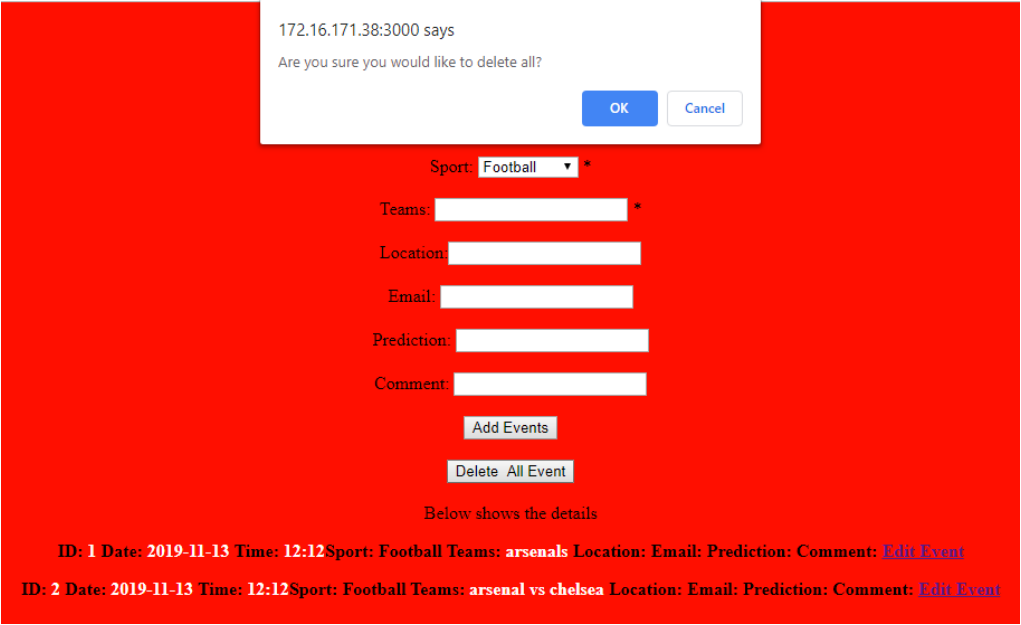
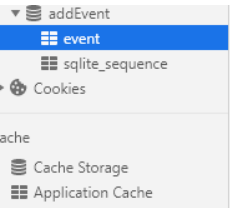


Figure 52 shows that the user has an option to delete everything from the database.



The "event" table is empty.

Figure 53 shows that the database has been cleared.



## Section 5: Evaluation

During this section, I will be thoroughly evaluating Android and PhoneGap applications of issues that could improve. Aspects that will be discussing are the following human computer interactions (HCI), run applications on different sizes and any improvements that are suggested for future aspects.

Throughout this implementation stage, bugs do occur for any application. These bugs were identified and found an alternative solution if this was not necessary. All this information was stated in Section 2. It is impossible for an application to be faultless, as many applications have bugs/weaknesses that do occur.

Matt Jones describes Human Computer Interactions (HCI) as a “study of how people interact with computers and to what extent computers are or are not developed for successful interaction with human beings” (Jones, 2019). The goal of HCI is to understand the factors of user interactions, develop tools and techniques and simply putting people first. This is a huge part of developing an application. A strong part that I would like to highlight throughout this course is the fact of using different usage of tools that are available to select. For example, Android application has a usage of menus, image buttons, spinners, and list views. However, as stated in Section 2 as a weakness; this must improve for the Main Activity of the android application. Section 4 highlights the Main Activity that is used for this application. This Main Activity only has four buttons that simply direct users from page to page. Due to lack of time and consistency, I would like to point out an improvement for this. This would have been better if these buttons were directed as a side-menu. This could leave the blank page as an information page for users to understand the viewpoint of the application. This could be mentioned the same interface for PhoneGap. If you refer yourself to Figure 42, you can see that the interface is all on two pages. Index.html is for the add event and Edit.html is for updating and deleting events. This is an aspect that could be improved by separating both interfaces and making a main activity page.

Android has various of screen compatibility that allows screen sizes to run differently on different screen types. By default, Android resizes to fit the current screen. However, anything that is hard-coded, it would stretch the view of the format (Android Developers, 2019). The android application does use different regimes of dominantly linear layouts and constraints layout. Linear layouts simply arrange the layout in horizontally or vertically in one single row. Constraints has its flexibility by moving it to any position. However, having done various of tests; the ability to change a range of screens is very limited. One positive that can be highlighted is the MOBSE Mobile for PhoneGap that is used within Android. This converts the browser format to a mobile format that the user can see. However, this was seen as a bug that has been stated that the application is non-functional by using JavaScript. This has been verified and tested by using Google Chrome and the page converts the browser format and simply works too. In terms of page format of bigger screens; the format stays intact. However, when a smaller screen the format is more merged together. It is still functional. However, some of the buttons are missing. This was tested in the Emulator by using smaller screens. This would need to be an improvement as for smaller screens; it would still need to be functional.



McAfee have discussed various of cloud security issues that concerns users of uploading to cloud. One main issue they mention is *incomplete control over who can access sensitive data*. This is due to by uploading information to the cloud, it is a shared platform that can be accessed. Therefore, it is advised that sensitive data should not be uploaded to the cloud for this reason (McAfee, 2019). This application can be accessed by numerous of users as the target reach of this application is wide across two platforms. Sensitive data needs to be addressed on JSON when the information is uploaded. However, I have made sure that no sensitive information such as email address is not uploaded. The only information that has been uploaded to the cloud, as seen on Figure 27, the event date is uploaded to the cloud. For JSON, it says for users to be able to view the message in a format that is readable. I feel as though this was a rushed attempt, but I realised that this format of letting the user know very late at this stage. This was an attempt, but I feel as though this could be improved by having a toast of what has been sent and if the message was successful.

SQL Injection is a most commonly a web hacking technique that is a placement of malicious code in SQL by web input. This is usually complete by users to try to get access to information that usually is restricted of access. Therefore, a technique to prevent SQL Injection is to use Prepared Statements that disables the use of this and prevents it from happening. This has been highlighted within the strengths of Section 3, but even its importance; it is important to underline this again. This has been done for both applications of Android and PhoneGap to prevent this from happening.

### Future Aspects

The last aspects of any future enhancements of live deployment. I do understand that tweaks need to be made to both applications. However, for future references, this application can be made public by uploading this application to Google Play. This would enable to upload the application to Google Play where this can be live for users to download. Another future development can be is to put a main server on for PhoneGap. Each time PhoneGap has been deployed, the server is different and its only accessed locally. This can be changed, so that it can be made public for users to see. One last aspect in for Android that can be a suggestion is for Google Maps is to provide directions from current location to directions to be searched. This would enable users to search for the event if they wish to attend the event in the near future.

## References

1. Android Developers. (2019). *Screen compatibility overview | Android Developers*. [online] Available at: [https://developer.android.com/guide/practices/screens\\_support](https://developer.android.com/guide/practices/screens_support) [Accessed 6 Dec. 2019].
2. Jones, M. (2019). *Introduction to HCI*. [online] Cs.bham.ac.uk. Available at: <https://www.cs.bham.ac.uk/~rx/Teaching/HCI%20II/intro.html> [Accessed 6 Dec. 2019].
3. McAfee. (2019). *Security Issues in Cloud Computing | McAfee*. [online] Available at: <https://www.mcafee.com/enterprise/en-gb/security-awareness/cloud/security-issues-in-cloud-computing.html> [Accessed 6 Dec. 2019].
4. W3Schools. (2019). *SQL Injection*. [online] Available at: [https://www.w3schools.com/sql/sql\\_injection.asp](https://www.w3schools.com/sql/sql_injection.asp) [Accessed 6 Dec. 2019].
5. Wolf, M. (2019). *University of Greenwich*. [online] Moodlecurrent.gre.ac.uk. Available at: <https://moodlecurrent.gre.ac.uk/mod/resource/view.php?id=942500> [Accessed 5 Dec. 2019].