**Project Report.**

Introduction.

My application is called, ‘Golf Shot Rating’. There are two main parts to the app.

Firstly the app gives a rating to each shot a golfer hits so that they can see how well they played in each part of their game. This is based on the ‘strokes gained’ statistics that are seen in professional golf. The parts of the game covered are:

* Driving – tee shots hit with the driver.
* Long game - which includes shots hit from distances of 250 – 100 yards from the flag.
* Short game - which includes shots hit from distances of 100 yards from the flag up to the green.
* Putting – shots taken whilst on the green.

Secondly there is a ‘league’ section. In this part the golfer can create or join existing leagues and compete against friends or the general golfing public whether they be in their own club, country or anywhere across the world. The leagues are divided to categories reflecting the parts of the game that are rated by the app, i.e. Driving leagues, long game leagues, short game league and putting leagues. The ratings that a golfer gets after they play a round of golf are added up to give them points in the leagues that they have entered, i.e. a golfer is in a driving league and in their last three rounds their rating for driving the ball are: 2.5, 1.0 and 3.5, their points in a driving league would be ‘7.0’.

My motivation for creating this app comes from two sources.

The first is a desire to enable an amateur golfer to get ‘strokes gained’ statistics for their own game. In the professional world of golf, teams of volunteers use lasers to track distances and input the ‘lie’ (the ‘lie’ is where the ball is, e.g. on the fairway, in a bunker, in the trees, etc.) into databases which enable these golfer to get accurate statistics.

The second source of motivation is from an article written by Andrew Cotter in a golf magazine called ‘Today’s Golfer’ (issue 334). In this article he talks about how the ‘Strava’ app changed recreational running and cycling. He mentions how tracking your performance and then uploading it so that you can see exactly how you did and compare your performance against others, even random strangers, ‘brings a feeling of competition and a sense of satisfaction’. He asks why there isn’t a golfing version of this and finishes by saying that, “any way of making golf more attractive – and interactive – has to be a good thing”.

With these two ideas I decided to create an app that rates golf shots and uses this rating for golf leagues.

Requirements.

The college required that this app had to include 3 out of a list of 4 APIs. The APIs that I choose were:

* Sensors/Locations.

APIs used: android.location.Location;

import android.location.LocationListener;

import android.location.LocationManager;

These APIs are used to get the GPS coordinates of the golfer’s location.

* Database(own creation).

APIs used: android.database.SQLException;

android.database.sqlite.SQLiteOpenHelper;

android.database.sqlite.SQLiteDatabase;

These APIs are used to create a SQLite database to store the data relating to the golfers round.

* Networks (accessing internet).

APIs used: Java.net (classes used: URL, URLConnection, URLEncoder).

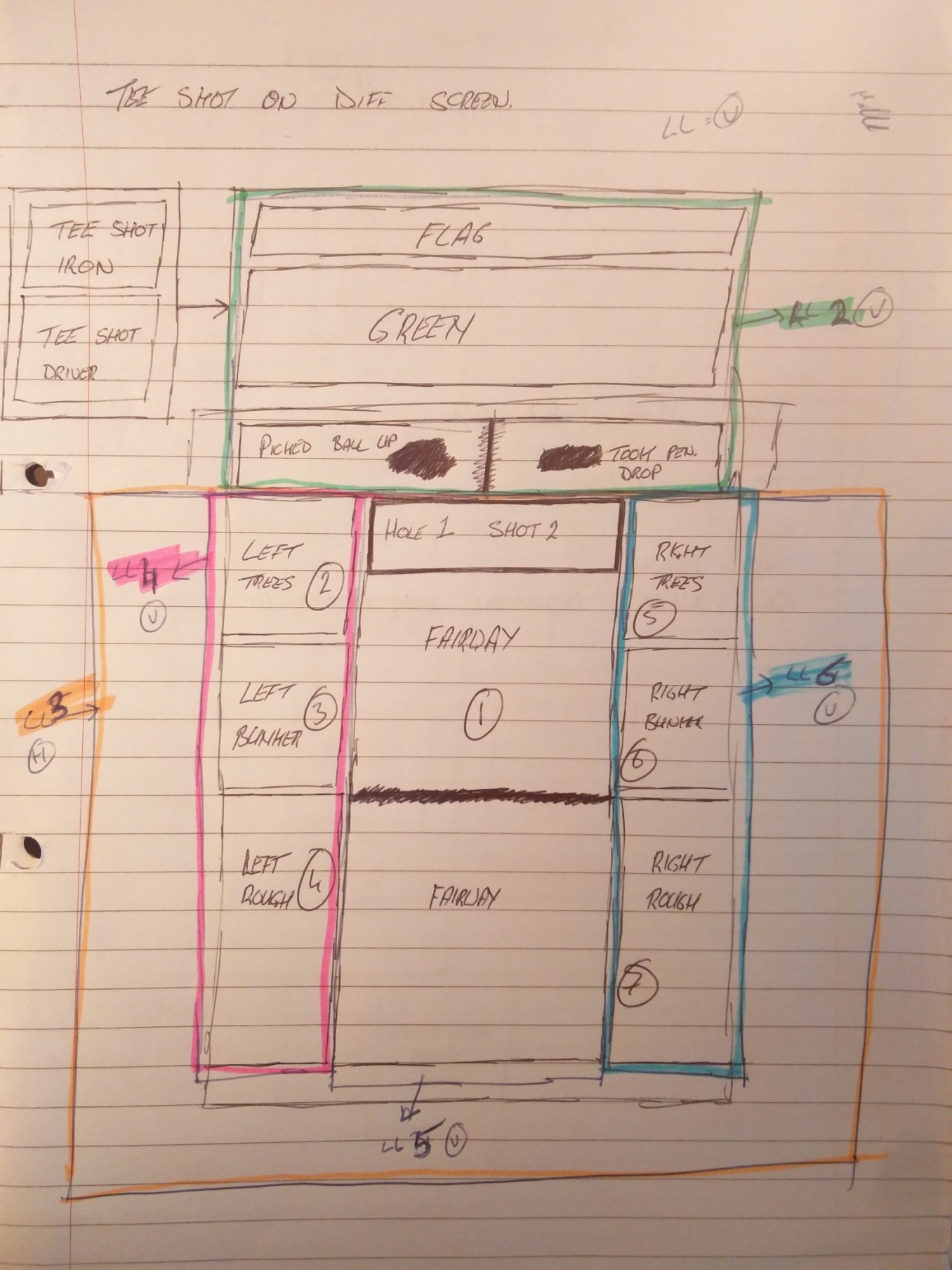
This API was used to send POST and GET requests to a database hosted on the web so as enable the user to register, login, get league information, etc.

The user requirements were that:

* It provides both a summary of each shot on each hole and a summary of the overall round.
* The app is straight forward to use.
* Using it doesn’t slow down the pace of play.
* It provides the ability for golfers to enter specific leagues based on their preference, e.g. an elderly golfer who can’t drive the ball too far can enter a short game league and a putting league.
* You could export a round to a website where they would be saved. They could then be removed from device, saving space. It would also protect against loosing rounds if phone lost or changed.
* An additional requirement that came about after trials was that the app have a distance to the green displayed on the screen

Research and building the app:

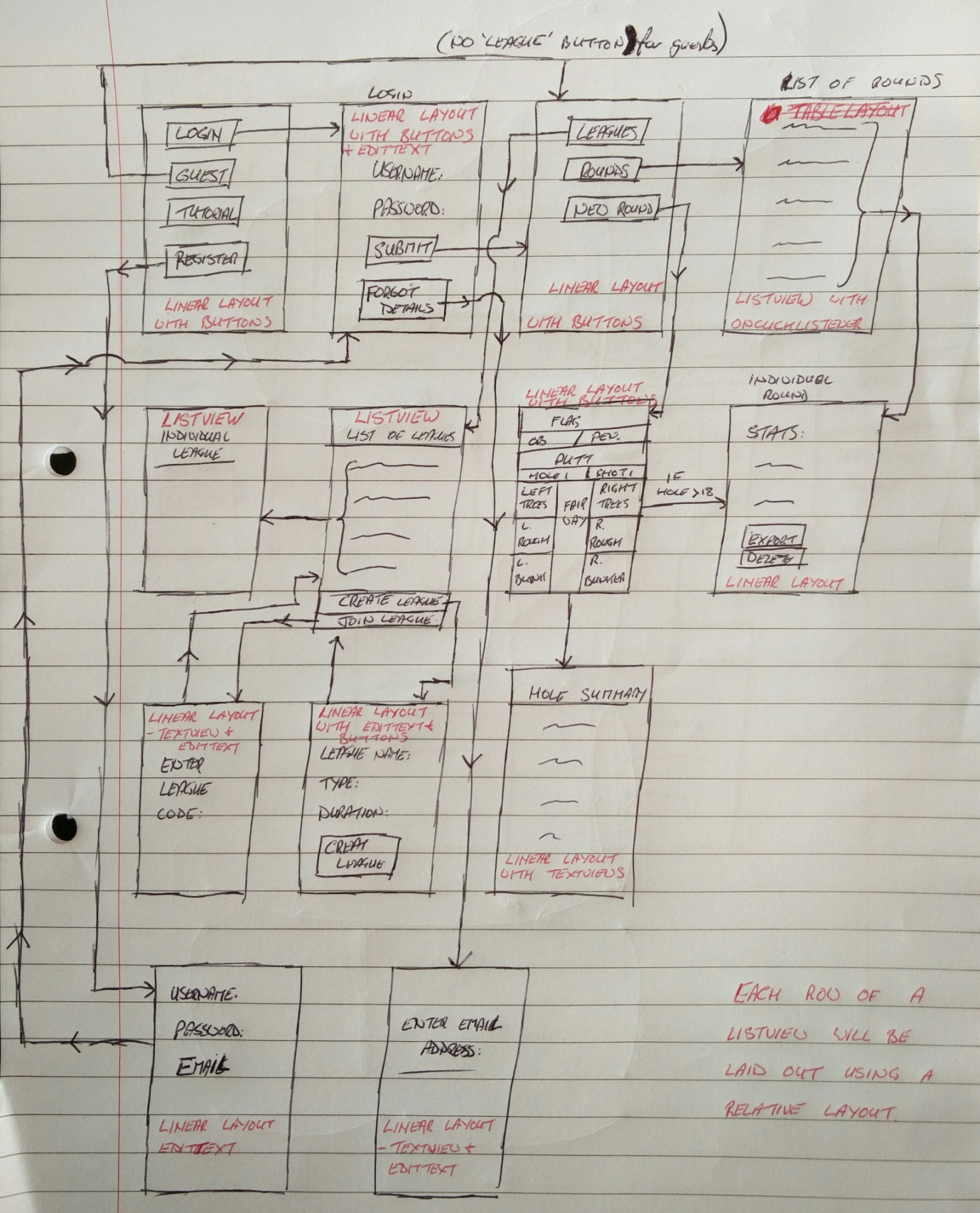
The first thing I did was to draw out a rough sketch of how my app would be navigated, a rough idea of what would be on each screen and the UI components that would be used on each screen. The resultant drawings are here:



Originally I thought about having the tee shots on a different screen. This was to reduce the number of buttons on the main shot input screen. Upon further though I realised that this would be more of a nuisance than a help. The golfers that I asked to try the device were never bothered about the number of buttons on the main screen so I kept all the buttons on the main screen.

The reason for a separate screen to input shots when on the green was to do with the perception that taking out a phone for every putt would be intrusive and possibly slow down the pace of play. To this end shots input on the green are input on a separate screen with a large image of the green on it. The golfer puts the phone in their back pocket (top of the phone down so they don’t click the ‘flag’ icon) and simply puts their thumb in their back pocket and swipes across the screen. The phone vibrates to let the golfer know that a shot has been recorded. Even though this doesn’t save more than a second or two off the time it takes to hit a putt the perception was that it was a big help.

The rest of the original screen layouts were drawn up along with the main input screen and the drawing done is inserted next.



This initial design page has had additions to it over the course of building the app e.g. the ‘Forgot Details’ button on the LOGIN screen and the actual screen to enter an email address to have a new password emailed to the user were added toward the end of the project. Overall the finished app is very close to what I drew here.

The next stage was to research and start testing parts of the app. The research I did for this app used the book, ‘Head first Android development’, websites, and other apps from Adam Porter plus others from the web.

The first thing I did was to research how to get users GPS coordinates. For this I looked at Adam Porters app, ‘LocationGetLocation’. I also looked at various websites including:

* <http://developer.android.com/training/location/receive-location-updates.html>
* <http://stackoverflow.com/questions/6438960/how-to-get-users-gps-coordinates-by-clicking-a-button>
* <https://developer.android.com/training/location/index.html>

Once I had looked at these I created a test app called ‘gps\_test’.

When I was happy that I had the correct GPS coordinates I needed to save them into a database. As android apps have a SQLite database this was what I researched. The research for this was mainly done through the book, ‘Head first android development’. They build an app and part of it uses a SQLite database. This was the starting point for my testing of SQLite databases. I created the test app ‘db\_test’ from the book and also from websites including:

* <http://instinctcoder.com/android-studio-sqlite-database-example/>
* <http://stackoverflow.com/questions/18370219/how-to-use-adb-in-android-studio-to-view-an-sqlite-db>
* <http://developer.android.com/tools/help/sqlite3.html>
* <http://www.101apps.co.za/index.php/articles/using-a-sqlite-database-in-android.html>
* <http://mrbool.com/how-to-insert-data-into-a-sqlite-database-in-android/28895>
* <http://www.techotopia.com/index.php/An_Android_Studio_SQLite_Database_Tutorial>

The ‘db\_test’ app had the initial tables that I would need for my finished app, i.e. Shot, round, stroke\_baseline and putt\_baseline. As the putt and stroke baseline tables were lookup tables that would have static data I populated these tables with the required data. The functionality of this app was very basic, you clicked a button and it inserted a static piece of data into the database.

With the testing done for the GPS coordinates and the SQLite database I rebuilt my ‘gps\_test’ app under the name MYGPS1 and added in the SQLite part from my ‘db\_test’ app. I added in different buttons to represent what the ‘lie’ of the golf ball was e.g. fairway, bunker, etc. And tested that the GPS coordinates and the ‘lie’ was entered correctly.

I then added in a user confirmation dialogue screen to get a user to confirmation when a button is clicked. The website I used to help with this code was:

<http://stackoverflow.com/questions/22424064/creating-simple-confirmation-dialog-on-button-press-android>

This went well and at this point I had an app where you could click a button and it would record your GPS coordinates and ‘lie ‘in the SQLite database.

After I had this done I had the coordinates for each shot and for the ‘flag’ which would be the reference point against which all distances would be measured. To calculate the distances between GPS coordinates I used the ‘haversine’ formula which I got from:

<http://www.movable-type.co.uk/scripts/latlong.html>

I also used the website: <http://www.csgnetwork.com/gpsdistcalc.html> to verify distance calculations.

I added this formula to the app and tested that it would return the correct distance in yards (distances in golf are normally measured in yards). I then tested that the distances would be put in the SQLite database correctly.

Next I had to calculate the scores for each shot. This was the most time consuming part of the app. To calculate the score you first need to calculate the distance for each shot to the ‘flag’. You then need to lookup the corresponding score in either the stroke\_baseline or the putt\_baseline table depending on where the shot was hit from. This score is really a difficulty rating for the shot. Once you have this difficulty rating for each shot you need to subtract the rating for the following shot from the present shot. E.g. shot 1 has a rating of 4.9 and shot 2 has a rating of 3.8. You subtract 3.8 from 4.9 leaving you 1.1. AS it took you 1 shot to get from shot 1 to shot 2 you subtract this from the score giving you 1.1 – 1 = 0.1. This is the final score for the first shot. This must be done for each shot and the final scores recorded in the SQLite database.

To make these calculations I first had to find out how many shots were taken on the present hole. This figure minus 1 (I subtracted 1 as the ‘flag’ coordinates don’t represent a shot)was used to initialise arrays in which figures would be held. Then each shots distance and lie was looked up in the relevant baseline table and the ‘difficulty’ rating retrieved. These ratings were put into an array. I then looped through the array taking the score from each position for the previous positions score and subtracting 1 from the resulting score. This final score was then recorded in the SQLite database in the row to which it belonged.

I then created a summary page that would list the shots and their scores. I decided to in the ability to see the shots and scores from all holes if the user turned the device in to landscape orientation. This caused me some problems as I used two xml layouts for this with one tableLayout in the portrait xml and 18 tableLayouts (one for each of the 18 holes) in the landscape layout. When I tested this out the app crashed and the logcat showed that the java file was looking for all 18 tableLayouts but in portrait mode they weren’t there. My way to fix this was to have all tableLayouts in both portrait and landscape xml’s but in portrait mode the 18 tableLayouts for the landscape mode have a height of 0dp. Then in the landscape xml the tableLayout for the portrait mode has a height of 0dp.

I then put in a piece of code to take the user to a ‘round summary’ screen if the hole counter reached 18. This page is the same as the hole summary but the statistics here a for the whole round.

At this point I put in a button that the user clicked if they wanted to export their round data to a web database.

Before data could be exported I needed to get the app to check that there was a network connection. My research for this brought me to this page:

<http://developer.android.com/training/basics/network-ops/connecting.html>

After I had written the code to check to check for a network connection I found this website to help connect to a web based MySql database:

<http://www.tutorialspoint.com/android/android_php_mysql.htm>

From this website I created the test app, ‘TestMYSQLCon’. I created the database and the relevant PHP files and now I could send data to my MySql database. I then researched how to put in a ‘progress dialogue box’ as it could take a few seconds to export the data. I did some research and got the relevant information from this website:

<http://stackoverflow.com/questions/4538338/progressdialog-in-asynctask/4538370#4538370>

When I had this working I added it into the MYGPS1 app.

At this point I created a user registration and login system. This was needed as, when users exported their data it would need to be marked with a userID to differentiate one users data from another’s. I added in the database tables and I modified the code for exporting the round data to enable usernames and password to be inserted and retrieved from the database. The code to validate unique usernames was done by PHP scripts. It was later on that I realised that I had forgotten to add in the functionality to get new passwords if a user forgot theirs. I then added in the new column in the database to accept email addresses to which new passwords would be sent. This caused a problem later on. The usernames and passwords that were registered before the email column was added had no email addresses associated with them. When I added in the email column into the database if didn’t set it to be ‘unique’ and it therefore let all the original usernames have no password in their email column. Then when I was testing the ‘forgot login details’ part of the app, if I clicked on ‘send new details’ without entering an email address, ALL the original users without an email address had their passwords updated with the same password. I fixed this by putting email address into columns where there were none and making the email column unique. As the users were registering in a central database the userIDs were all unique.

Once this was done when a user logged in the app retrieved their unique userID and put it into the userID column of the round data. Therefore when the round data was exported it had a unique userID with it and all data could be associated with the correct user.

Now that each user had a unique ID I created the leagues. First of all I created the database tables and added in a few leagues and associated users with some of the leagues. For the app to retrieve the information I felt that the code I had used to send data to the database would be too hard to modify to retrieve the amount of information that would be needed, especially if a user was a member of a lot of leagues. Researching this I found these websites:

* <http://www.geeks.gallery/retrieve-data-from-mysql-database-using-php-and-displaying-it-by-tableview-in-android/>
* <http://www.mysamplecode.com/2012/07/android-populate-listview-from-json.html>
* <http://mobilesiri.com/json-parsing-in-android-using-android-studio/>

From the last website I downloaded the code and complied the app, ‘JSON-Parsing-in-Android.’ I then created my own testing app called, ‘JsonTest’. This I used to test retrieving information about leagues and individual leagues from the web based MySql database. Once I had this testing done I added the code into the MYGPS1 app.

At this point a user could register and login. They could play a round of golf and get ‘strokes gained’ statistics. They could join or create leagues and then export their data to a web based database which would update their league points.

I then went about adding in code to enable a user to select their golf course from a list of courses near their location. This was done by getting a database of golf courses and their GPS coordinates and adding it to my own web bases database. Then the app would send the users location and a PHP script would retrieve the top 10 location s nearest the users location. This information was sent back in Json format and was displayed in a ListView for the user to select.

Upon using the app it became apparent that if it could give the user the distance to the centre of the green it would be a big help. Other golfers liked the app but if they were already using an app to tell them distances to the green then it became too much of a hassle to keep changing apps. Another advantage of a distance calculator was that it encouraged golfer to take their phone out for every shot. On occasion a golfer would forget to take out their phone and record their shot.

As the app already retrieved a golfers coordinates it was just a case of getting the coordinates of the green. As I didn’t have these I had to write code to enable the user to ‘tag’ a green when they came to it. I put a column in the database called ‘pins’. This would hold a comma separated string of coordinates for the location of the greens. Then when the app retrieved the top 10 nearest golf courses it also retrieved the coordinates of the greens. If this column was empty then the course had not been tagged and the user was presented with a screen asking them if they wished to ‘tag’ the green. If they did then on the ‘green’ screen they would get a ‘target’ icon which they would click on when they were in the centre of the green. The location would be stored in shared\_preferences and when the round was over all the coordinates were put into a comma separated string and exported to the web based database. Then when they or any other user play that golf course the app measures the distance from the user’s location to the centre of the green using the ‘haversine’ formula and displays it on the screen.

Challenges:

The issue of polling for GPS coordinates was one I had to think about a lot. Most apps that get locations get the most accurate coordinates and then keep those coordinates until either a certain amount of time has passed or the distance the device has moved is beyond a certain distance. Upon using the app in a round of golf it was very hard to define a set of time and / or distance figures that could be used. One golfer hit a ball which only moved about 20 yards. They immediately walked forward and hit it again within seconds of their first hit. Upon discussion with other golfers they could see golfers hitting shots within 3 seconds of each other and shots that were only a few feet apart. To this end the polling figures used were, 2 seconds for time and 5 meters for distance.

Calculating the final score for each shot hit was the biggest problem. Keeping track of the different variables and arrays along with calculating array lengths and retrieving data from the SQLite database was a tough job. Every time I looked at it from a distance it seemed like it should be an easy task but when I got into it, it became more difficult to find my way through. Breaking the process down in to small testable parts was the best way to accomplish this part of the app.

Keeping track of a logged in users ID was initially a problem. When I first wrote the code I passed the UserID from one activity to another via the ‘Intents’. In some activities the userID wasn’t used but I had to pass the Id anyway as, when the user goes on to another screen that does need the ID the ‘passing’ activity needs the ID just so it can pass it on. I decided to change from pasing the ID via ‘Intents’ to making the userID a public variable and accessing it via: *class.variableName*. This worked but if there was any problem with the app (i.e. I encountered an error that didn’t crash the app but reset it and reset the userID to 0.) then it was unreliable. Being able to fix all errors and make sure that there were no more errors in the future was unrealistic so I changed my approach and put the userID in shared\_preferences. This way I could always access the current users ID from any activity when the app was running.

When a league was created I needed the creator of the league to be a member of the league. In the database there is a table for users, one for leagues and one for league members. So when a league was created the ‘league’ table would be updated and then the league members table needed to be updated with the userID and leagueID. The problem was that I didn’t know the leagueID until the league was created. Initially I tried to solve this via PHP scripts. The user would create the league. This was done by a call to a PHP script. The PHP script would then retrieve the leagueID from the ‘NEW’ table. (The ‘NEW’ table is MySql’s version of SQLs ‘INSERTED’ table.) Simply retrieving the latest league to be created would not work as there was a possibility that concurrent users could create league at the same time and this would then return the wrong league. This would then insert the leagueID along with the userID in to the league members table. I got this working but it seemed a bit cumbersome and I eventually turned to using a ‘trigger’ on the ‘league’ table. When there was an ‘insert’ in the league table the trigger was activated. The trigger got the userID and leagueID and inserted them in to the league members table without any need of the PHP script.

When the app was retrieving a lot of information from the web database I put the data in Json format. To then display this information in a ListView the research I did always showed the use of HashMaps along with ArrayLists. When I first looked at the code it seemed over complicated and confusing. If I needed to modify it in any way I needed to understand how it all worked. This took a bit of time and more research:

<http://javarevisited.blogspot.ie/2011/02/how-hashmap-works-in-java.html>

When I wrote the code for ‘tagging’ the green I thought the process would be simple. I would put in a clickable ImageView that, when clicked would put the coordinates into an array which I would access and export at the end of the round. Unfortunately every time the app went to the next ‘hole’ it reinitialised the array and wiped the data that was in it. I tried to create and initialise the array in the NewRound activity and access the array via: *NewRound.ArrayName* but the array was still wiping all the data. I never found a way to use arrays for this part and instead I had to put the coordinates for each green in shared\_preferences.

Testing and verification:

The testing was in two parts. The first part was the testing of individual parts of the app and I did this as I was building the app. I would research a certain part and then build a test app to get it working as I needed. The main test apps I created were:

Gps\_testing .

Db\_test.

TestMYSQLCon

JsonTest

With each of these I could try and configure the code so that I got what I needed. I would try different inputs and view the results. In some cases I would restrict users input options e.g. a spinner with specific values for user’s handicap input. And in others I would write code to deal with unwanted input e.g. prepared statement to stop SQL injection attacks.

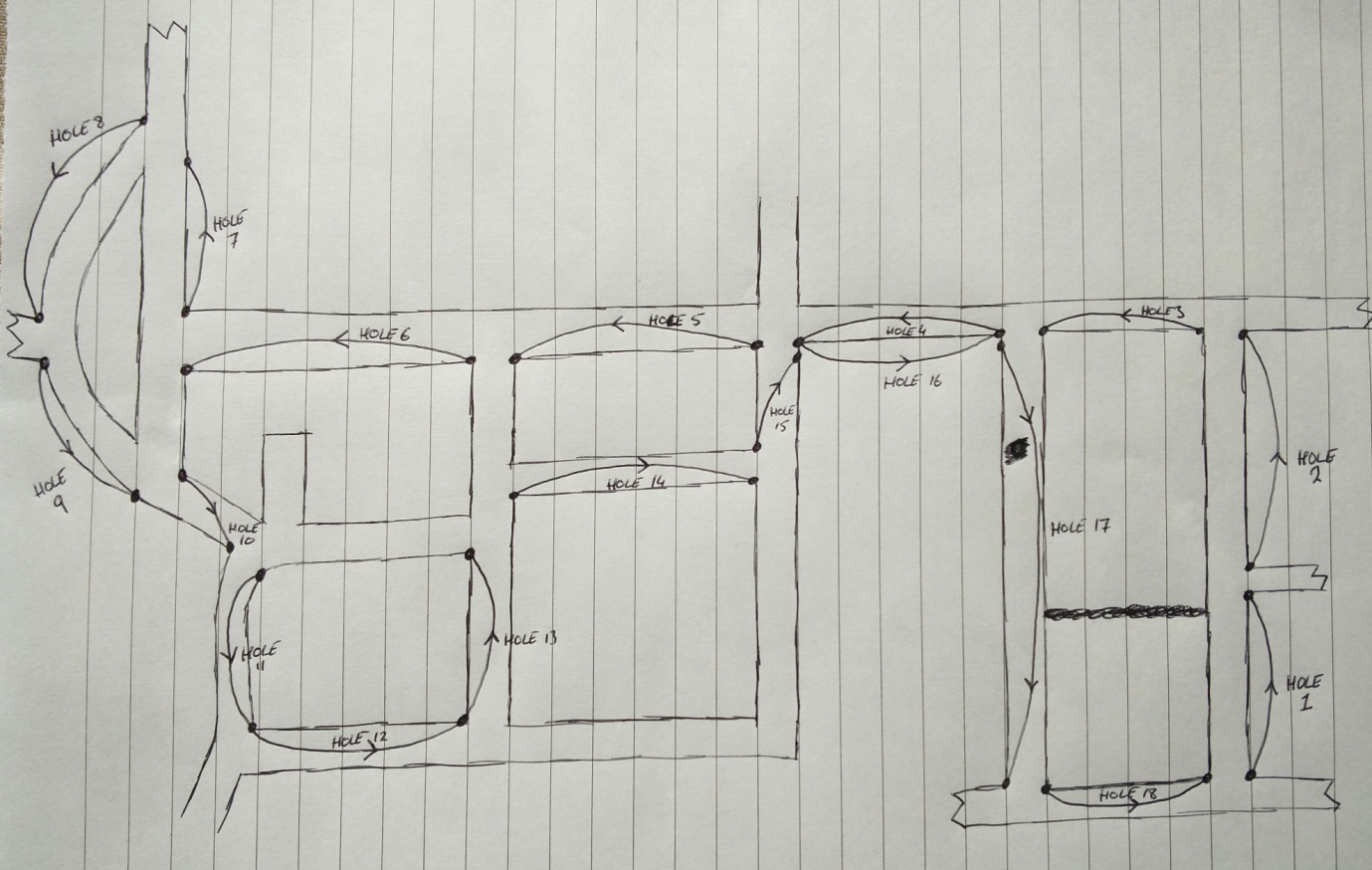
When this testing was done I used the ‘use cases’ to create a ‘one stop test list’. Here I created a sequence of actions that covered as much of the apps functionality as I could think of.

The list is:

1. Register as: user1. Password: password. Email: yahoo@yahoo.co.uk
2. Login as user1
3. View ‘rounds’ – should not be any rounds.
4. View ‘leagues’ – should not be any leagues.
5. Create a new league called: user1 league. Type: driving. Duration: 2 months.
6. View created league – check that name is correct, code is correct and days left are correct.
7. Go to New Round Screen.
8. Select a golf course that has not been ‘tagged’.
9. Enter a handicap and click ‘continue’.
10. Does an option to ‘tag’ the greens show up?
11. \*\*Play round of golf. \*\*
12. Check SQLite database that all shots and round data are correct.
13. Export scores.
14. Check the database. That all data exported successfully.
15. View league to see that score showed up.
16. Logout.
17. Register as: user2. Password: password. Email: yahoo@yahoo.co.uk
18. Password is the same as user1 so check that action not allowed and correct warning displayed.
19. Register as: user1. Password: password. Email: gmail@gmail.co.uk
20. User name has already been used so check that action is not allowed and correct warning is displayed.
21. Register as: user2. Password: password. Email: gmail@gmail.co.uk
22. Login as user2.
23. View rounds – there should be none.
24. View leagues – there should be none.
25. Join league – enter code for user1 league.
26. View league – user2 should be in league along with user1.
27. Enter code for a nonexistent league.
28. Action should not be allowed and correct warning displayed.
29. Click New Round
30. Select course that user1 tagged.
31. \*\*Play round a check that distance to greens show up and are correct. \*\*
32. Export round.
33. Check that scores show up in league.
34. Logout.
35. Click ‘Continue as guest’.
36. Check that ‘leagues’ button is disabled and greyed out.
37. Login as user1 but click on ‘forgot login details’.
38. Enter password and check that username and new password are sent to email.
39. Login as user1 using new password.
40. Go to leagues – check that user2 score is in league.
41. Click New Round.
42. Select course that has not been tagged.
43. Does option to tag course show up.
44. Select ‘no I will not tag course’.
45. \*\*Play round – icon to tag greens should not show up. \*\*

\*\*

As this is a golf app, part of the testing needs to be done on a golf course. It would have been impractical to have to go to a golf course, test the app, find a problem, go home, fix the problem, go back to the golf course, test the app, etc. So I decided to create an imaginary golf course using the roads where I live as fairways:



This meant that I could test, fix and retest the app numerous times a day. It also meant that I could predefine the shots that I would take and the coordinates that I would take them from and so have an expected output against which to measure the results.

I then used test list to create a test report which used predefined inputs where possible and measured the apps output against them. The Test report is submitted as a separate document.

Evaluation:

As the app uses the API’s listed in the ‘requirements’ section of this document, the app meets the colleges requirements.

The app also meets the user requirements and an additional requirement of a distance to green display was also met.

I am fairly happy with the app. The graphics on the ShotInputScreen could be a lot better. Also I wanted to have an icon beside each round in the list of rounds page that would be either orange or green, to let the user know if they had already exported the round or not. I was unable to get the ID of the ImageView in the ListView (I tried the getView() method but I just couldn’t get it to work).

If there was a version 2 I would change the ShotInputScreen to have an image of a golf hole on the screen. The user could click on parts of the image to record their ‘lie’. The image would be along the lines of:



This is just an idea of the image that could be used. If put into the app the image would need areas that the user could click on with ease so the tee boxes, bunkers, etc would need to be increased greatly. I research the idea of clickable areas on an image and it seems relatively straight forward but as my graphic design ability is worse than useless I decided that I would leave it alone.

In the current version of the app all shots are exported when a round is exported. Only the round summary is needed for the leagues but I exported all shots as, in the future, analysis of strokes gained for specific handicap levels could be calculated.

Conclusion:

I found the experience of creating my own app both good and bad. I learnt a lot about android development and found that it was something I really enjoyed. I also found that I got frustrated by my own lack of coding knowledge. I had ideas that I wanted to implement but either I couldn’t or they took a long time to simple things. As I spent more and more time coding I found that my productivity greatly increased and if I needed to change anything in the app I could do it relatively quickly. Even adding in the distance to green calculator wasn’t too much bother.