Final Report

Matty Christopher – 130080943

Programme of study: Computer Science G400

Supervisor: Dr Anne Hsu

Project Title: Quiz Maker App

Date: 24/04/2016

Table of Contents

Abstract:

The project that I have chosen is Dr Anne Hsu's "*Project B: Game for prompting students to generate self-study quizzes*". The projects objective is to create an app in which users can create quizzes either on their own or as part of a group, users can then play quizzes created by them or other users. The project required the creation of an app which I created on the android platform. Through research of existing apps I found that none of the existing apps tested met the project requirements as none of the existing apps give the user the option to create questions in groups. The development process included creating a prototype in which feedback was gotten which was then used to improve the prototype for final release. The development process and findings is documented in the final report. The final app release meets all the initial requirements set out along with extra features that I learnt from testing which would improve the app more.

1. Introduction:

The project that I have chosen is Dr Anne Hsu's "*Project B: Game for prompting students to generate self-study quizzes".* The app allows users to create study material in the form of quizzes which they can create on their own or as groups to have an alternative form of study. As more and more people are getting top grades it is important that students get the best grades that they can possibly get which requires lots of work and effort. Therefore students must learn in ways which work for them through using different forms of study such as video tutorials, reading around a subject or topic, writing notes etc. With the rise of Smartphone's students have another resource to study from through various apps, games and the internet therefore making a self-study app would be easily accessible to many students. With the help of my app I hope that students will be able to learn in an alternative way to other resources while having fun on their own or with friends.

1.1 Aims:

The aim of this project is to create a quiz app for android devices in which users can create questions on their own or as part of groups and to upload them to a database server. The app will allow users to play the quizzes created by themselves and others, each question can be graded on how helpful the questions and answers were to the user. The purpose of the app is to help users revise for upcoming exams or improve their knowledge on certain topics by allowing them to play and create with friends to create another fun way of learning.

2. Background Review:

2.1: Quizzes in education:

As the aim of the project is to create a self-study quiz application I decided to research whether quizzes have a positive impact on study.

Quizzes help to test a user's current knowledge of a topic to see what areas they lack knowledge in- "*Great way to be exposed to the subtleties of a topic that you don't know*"[1] this allows you to pinpoint and focus on particular sections of a topic that you feel may need more study. There have been various studies by researchers which aim to measure whether there have been meaningful benefits to quizzes in study. A study carried out in 2013 at Al al-Bayt University in Jordan compared two sets of students on the same course-one set was taught in a typical lecture style while the other was taught using quiz games as one of the forms of educating. The study found that of the two groups that the students in the quiz format on average significantly had better post-test and retention results.

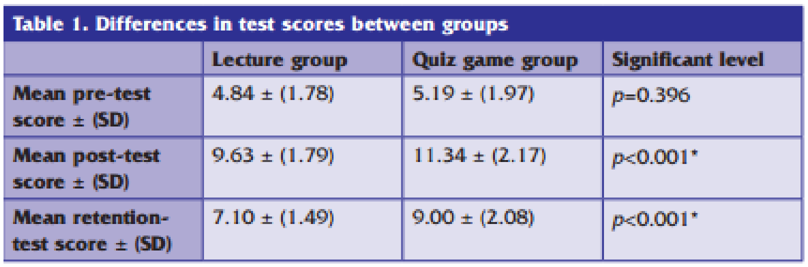


Table 1. Differences in test scores between groups

	Lecture group	Quiz game group	Significant level
Mean pre-test score ± (SD)	4.84 ± (1.78)	5.19 ± (1.97)	p=0.396
Mean post-test score ± (SD)	9.63 ± (1.79)	11.34 ± (2.17)	p<0.001*
Mean retention-test score ± (SD)	7.10 ± (1.49)	9.00 ± (2.08)	p<0.001*

Figure 1: results of study comparing lecture format to quiz style teaching format[2]

This data is similar to another study which measured whether quizzes enhance existing learning resources. It found that quizzes improved Learning Object quality and that quizzes can easily be created and added by teachers to enhance study. On average overall scores for Learning objects with quizzes were 10-30% higher.[3]

Another study at the University of Louisiana found that labs grades were significantly better when trailing class discussion quizzes compared to labs which didn't utilize class quizzes. These quizzes were found to engage students more and consequently help them get a better grasp of topics.[4]

2.2 Android Review:

Android is an operating system designed for portable devices such as Smartphone's and tablets which is developed and maintained by Google. Android devices dominate the market with 82.8% of the mobile market share as of quarter 2 of 2015 [5]. Android applications are created using Java and run on virtual machines. Android uses Android RunTime which is the sucessor to Dalvik which is the virtual machine in which apps are run on[6], this is used to turn the apps instruction set into native instructions for the device to execute.

Android's software stack is split up into 5 layers: The kernel and low level tools - which handle input and output requests, it executes processes and tasks run by the user. Native libraries - which are optimized to device optimized native code to reduce usage of CPU and memory. Android Runtime-contains Dalvik virtual machine which is the interpreter for byte code instructions, the framework layer -provides abstractions of underlying libraries. The application layer is where apps which can be used by the user are installed.[7]

To develop the app I will need to read through the Android Studio documentation as I will be using the Android studio IDE. This will be needed as there are many different classes with lots of methods, so I will need to read up on classes that I may need and how to implement their methods. I will need to do this throughout the development of my project as there will be classes that will help my implement specific parts of the app which will save time compared to having to come up with a unique solution.

2.3 Client Server model:

"The Client- server model is a system that performs both the functions of client and server so as to promote the sharing of information between them. It allows many users to have access to the same database at the same time, and the database will store much information.[8]"

Client-server architecture is composed of; the application server, the database server and the device. One of the main architectures is the 3-tier client-server system architecture which involves the database server and the client's device along with an application server. This allows having one server being able to interact with many clients, the application server is used as middleware which processes the data passed from the client or database[8].

As my project requires the use of a database server I need to use a database system that provides use of a server based on this model, therefore I will use MySQL as I am already familiar with it. The user will interact with the database server through the "front end" on the user's device running the app and the data will be handled on the server "back end" which will then communicate the data back to the application.

In Android the database server cannot be directly interacted with by the application, only through web server interaction[9]. The web server must be used in order to do database operations such as inserting and retrieving data by using Http connections to connect to PHP scripts stored on the web server which then retrieve or enter data to/from the database. The data format of this can be in the JSON format. JSON is a lightweight format for exchanging data which is able to parse data up to 100 times faster than XML[10].

2.4 Human Computer Interaction:

Human-Computer Interaction is the interaction between the user and a device. The ultimate goal of Human-Computer Interaction Research is exploring how to design the computer to help people complete the necessary tasks more safely and efficiently [11].

Some of the main principles of creating a good interface are:

Strive for consistency: The psychological vulnerability and memory of user has a direct impact on the efficiency of the control of the product, consistency helps reduce the memory of the user and helps to reduce confusion when using an interface.

Reduce memory load: Humans are certainly more efficient in carrying out tasks that require less memory burden, long or short term. Keeping the user's short-term memory load light is of particular importance with regard to the interface's role as a quick and easy guidance to the completion of the task [12]. Reduced memory load helps to relieve user pressure allowing users to make less errors which can make users complete tasks more efficiently.

Know the user: This states that the interaction of the system and its interface should be geared towards the needs and capabilities of the target audience of the proposed system. By studying the target audience and collecting data it will allow designers to create the right interface to improve user experience.

Human computer interaction is important when creating an interface as following these principles will allow users to have the best possible experience when using the interface. This will help to make the user want to keep on using the interface and will allow users to be able to use the interface correctly.

2.5 Review of Existing Apps:

1. Genius Quiz (Andre Birnfeld):

The genius quiz app is an app in which you must correctly answer a series of questions in the quickest possible time, you are given three lives and when they are gone the game ends. It offers a variety of question types such as simple puzzles, riddles and reaction questions.

Pros:

- The app can be very challenging on first use
- offers different language options "English" and "Portuguese"
- Has a leader board with the top 40 times
- nice variety of questions
- allows user to turn off background music

Cons:

- There is only one mode
- The leader board doesn’t work as all the times are 00:00:00
- Little replay ability as all the questions are the same on replays so you can memorise them quickly.

2. QuizUp (Plain vanilla games corp.):

QuizUp is a multiplayer trivia quiz in which you can choose topics to play against other players and you gain points for each answer you correctly answer, the winner is determined by how many points each user has after 7 questions. It is one of the biggest trivia apps on the Google play store and has over 1 million downloads and won various awards.

Pros:

- real time play against other players from around the world
- offers over 20 different categories to choose from
- excellent presentation and layout

- fast and responsive
- can search and chat to other players
- offers 6 different languages
- can see opponents answer once the question has been completed

Cons:

- cannot play on your own would be better if there was a single player or practice mode as well
- can take a while to search for an opponent
- if you answer quickly you must wait for your opponent to move on
- To create your own topic with questions you must do it on their website, cannot be done locally on the app

3. General knowledge quiz (Ingenify):

Ingenify general knowledge quiz is a quiz app in which the user is given 20 random questions and gets points for each correct answer, you can get a maximum of 600 points, after each question the app explains the answer to the user so they can improve their knowledge.

Pros:

- after each question a short extract explaining the correct answer appears
- correct answer lights up green and incorrect answers red
- keeps track of how many you correctly answered which shows the user at the end of the quiz
- Allows the user to change the colour scheme between black and white
- questions are different each time improving replay ability

Cons:

- Some of the buttons on the homepage for new features don't seem to work (if not implemented they should have a message pop up) as on pressing an error occurs saying to reconnect device
- When completing the quiz, the text dialog to enter a username appears in German even though the rest of the app is in English
- There is no check for duplicate usernames so you cannot tell if duplicate usernames relate to the same person
- the maximum score you can get is 600 if you answer 20/20

4. Millionaire Quiz (Fehencke Apps):

This app is based on the TV show "Who wants to be a Millionaire" where the user has 3 power plays to reach £1 million, you can choose whether they want to give up and take the money or carry on, if you guess incorrectly then you either win nothing or you gain money from checkpoints passed.

Pros:

- based on the TV show "Who wants to be a Millionaire" and the layout matches the TV show
- The user is given a 30 second time limit to put them under pressure
- Have 3 power plays similar to the TV show in which you can only use them once
- can choose difficultly level
- Rotating device changes the layout between simple and layout based on TV show

Cons:

- Annoying pop up ads appear after every question
- The local highscore leader board doesn't work correctly shows other users who should only appear on the online leader board.
- Online leader board doesn't work

5. Capitals Quiz (Paridae):

Pros:

- Offers a learning mode which doesn't count towards your overall score and has no time limit
- Offers a series of levels (9 total) which are unlocked on completion of the previous level
- Has multiplayer mode
- The false answers are randomly generated so the same questions will have different options each time (except correct answer)
- Links to Google play account

Cons:

- Questions are generally easy and most questions are often repeated albeit with different false answers
- Lots of ads
- False answers tend to be other countries capitals making it even easier

6. Trivia Crack(Etermax):

Trivia crack is one of the largest trivia apps on the Google play store. The user plays against other players in a series of questions, each round is a different topic which is chosen at random by a spinner.

Pros:

- Can rate questions
- nice animations
- head to head modes
- The topics are chosen at random using an animated spinner
- You can suggest a question to add, you fill out a question with correct answers and false choices and send it off for review

Cons:

- No single player options
- The settings doesn't work correctly on first use: when choosing your country while registering I chose the UK, but when playing for the first time the settings for language were in Spanish not English.

7. QuizOid (Habanen Quiz apps):

QuizOid is an app where the user keeps going until they get a question wrong, the further you get the points start increasing exponentially.

Pros:

- Allows user to suggest a question
- keeps stats of how many answers you have answered and correct percentage, what modes you have played etc
- Can change theme and turn off sound
- linked to Google account and has achievements and leader boards

Cons:

- Annoying adds
- Only one game mode

2.6 What my app will offer:

My app is aimed at allowing users to create self-studying quizzes which will enable them to use these quizzes to revise for exams and improve their knowledge in another way as oppossed to reading lecture slides etc. The app that I will be creating will allow users to create their own quizzes on the app and provide questions for the quiz, this is similar to the **QuizUp app,** however on my app the quizzes can be created in the app itself. The app will also allow users to create groups of up to 4 players in which the group can create a question and submit it to an existing quiz, this will allow better questions and more helpful questions to be created as users can agree on question that they find will help them revise in the future and false answers which can trip up people in exams. None of the apps provide a way of creating a question as a group on multiple devices. The username of the players who create parts of the questions will be shown once a player answers the question e.g.

"matty100 created false answer 1", this will allow users to see who creates good questions/answers so they can search other quizzes created by them.

Players can then rate questions on a scale of 1-10, so in future attempts the quiz will generate questions from the highest rated questions first. The app will have quizzes which are specific to the user and other users who are studying similar subjects compared to the apps that I have reviewed above which are contain specific types of questions **(Capitals Quiz)** or general knowledge questions. The user can create a title of the quiz so users can browse quizzes to choose from and see if that quiz maybe helpful to them e.g. "Java Syntax Quiz" or "ECS634U - Algorithms and Complexity PvNP".

3. Design of my App:

The user interface for the app will be based on a whiteboard background to mimic using them in lessons. Text will be displayed in easy to read fonts and colours to make the app simple and pleasant to use. I will stick to only a couple of colours to keep the design consistent throughout which is one of the design principles which helps reduce user errors. Buttons will be used which will allow users to tap on the screen to use them and they will be sufficiently large enough so that users don't accidently tap on the wrong part of the screen. I will use tables to display information such as leaderboard and group views which will allow the user to swipe up and down on them to go through the rows. I will not include back buttons as users will be able to use the android built in buttons as it will help keep the design uncluttered and simple.

The app will have a welcome screen which has two buttons-one for login and one for registration. Allowing users to create an account will make it easier to store progress such as high scores and edit questions that they created. The registration section will ask users to enter a username, password and an email address. The email address is required so that when a user forgets their login details they can recover them through an email. There will be checks to make sure that none of the fields are empty, then a connection will be made to the registration script which will store the details into a new row in the user table of the database. However if a username or email already exists then a message will be returned to the app which will be displayed to the user explaining that the username is in use or the email address already has a username attached.

For logging in a user can enter a username and password which will then be checked against the database user table to check if that user exists, they can also request an email to be sent to them to recover their details. I have chosen to send an email instead of displaying the details on screen because another user on a different device could find out someone else's email address and enter it which would then display that users details, whereas my solution will keep the details private as the user can only find them out by logging into their email server.

A home page will appear after a user successfully logs in and it will display the logged in user with their current highscore. Here the user can then choose one of three sections (create, play and leaderboard). The create and play sections are the main part of the app these are where the user can create quizzes and play them. I have added a leaderboard section as an extra feature to give some competition between users to give them another reason to use the app and motivate them more to revise using this app. The leaderboard will display the username and highscore of the top 50 players, logged in users will have their position highlighted if they are on the leaderboard to make it easier for them to see.

The create section will give the users the option to create a quiz or create a question or view current groups that they are in. I have split up the create a quiz and create a question to make it easier for groups/users to create new questions for existing quizzes by having less steps - as they won't have to go through the create quiz section each time. When creating a question users will be able to choose whether to create them on their own or as a group, one of the requirements of the project is for users to be able to create questions in a group as a collaborative effort. My solution will allow up to 3 other players (4 including question creator) to take part in a single question.

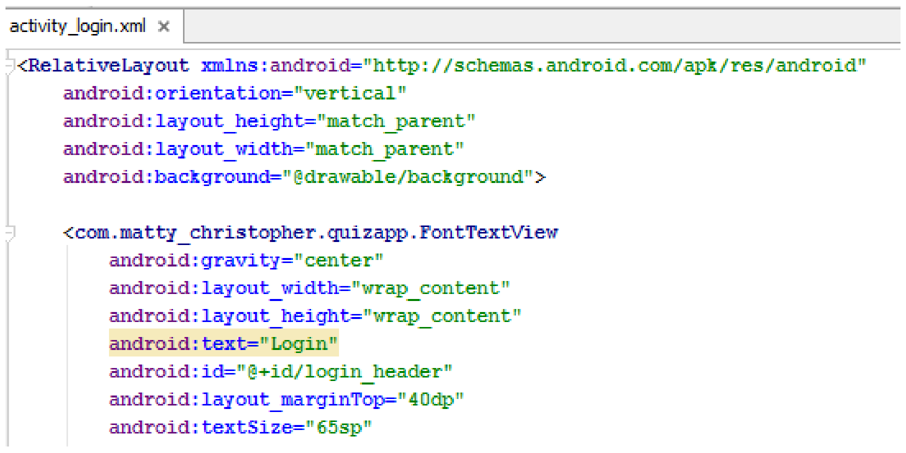
The question creator will be able to enter 3 other names they want to join to the group which will then be displayed in the view groups section. Users will be able to view the groups they are in then select one of the groups to edit the question where they will be able to edit the answer that they have been assigned to e.g. "false answer 1". Once all of a question are filled in then the creator will be able to send the question off as complete so it can be used for the play section.

An extra feature that I will include in the app is to allow users to play a highscore mode which chooses random questions from different topics and will keep going until user gets one wrong, this will help improve replay ability even after users don’t need to study anymore. Players will be able to rate questions after they have answered them to help ensure that better rated questions are more likely to appear, they will also be able to view stats for the questions which will display the percentage for each answer what users tended to select. This will help users see possible patterns for questions so that they won't make these mistakes in exams.

3.1 Implementation:

Creating the user interfaces:

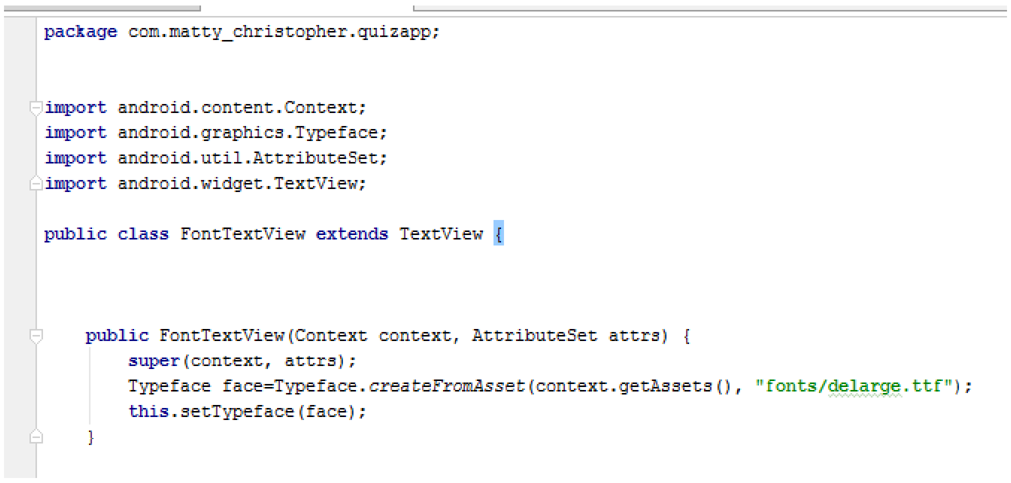
To create the interfaces I added components such as buttons and text views to the .xml files in the layout folder for each created activity. Here I used the Relative layout which allows me to display components on different parts of the screen easily without having to use multiple linear layouts.



```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_height="match_parent"
    android:layout_width="match_parent"
    android:background="@drawable/background">

    <com.matty_christopher.quizapp.FontTextView
        android:gravity="center"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Login"
        android:id="@+id/login_header"
        android:layout_marginTop="40dp"
        android:textSize="65sp"
```

Figure 2: showing partial xml of login

To create the headings of each panel such as Login I decided to use the delarge.ttf font which mimics a marker pen font. To do this I used this class-which I have named FontTextView, below which I found online[http://www.101apps.co.za/index.php/articles/using-custom-fonts-in-your-android-apps.html]:

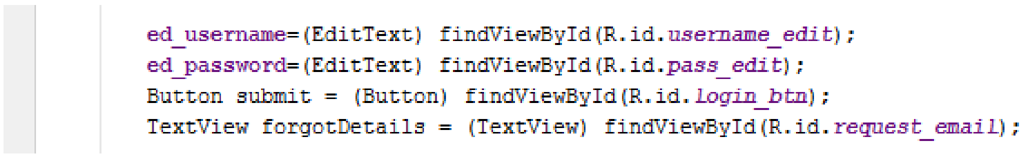


```
package com.matty_christopher.quizapp;


import android.content.Context;
import android.graphics.Typeface;
import android.util.AttributeSet;
import android.widget.TextView;

public class FontTextView extends TextView {



    public FontTextView(Context context, AttributeSet attrs) {
        super(context, attrs);
        Typeface face=Typeface.createFromAsset(context.getAssets(), "fonts/delarge.ttf");
        this.setTypeface(face);
    }
```

Figure 3: showing FontTextView class for custom font

This class allows me to create a custom text view in the xml layout files so I don't have to reuse the createFromAssets() method in the java files for each panel.

Then to add functionality to components such as Edit Text and buttons, I created private variables inside the java file for the corresponding panel where I would cast the component on the xml file to the new variable so I can add an onclick listener or retrieve text entered into the textboxes:



```
ed_username=(EditText) findViewById(R.id.username_edit);
ed_password=(EditText) findViewById(R.id.pass_edit);
Button submit = (Button) findViewById(R.id.login_btn);
TextView forgotDetails = (TextView) findViewById(R.id.request_email);
```

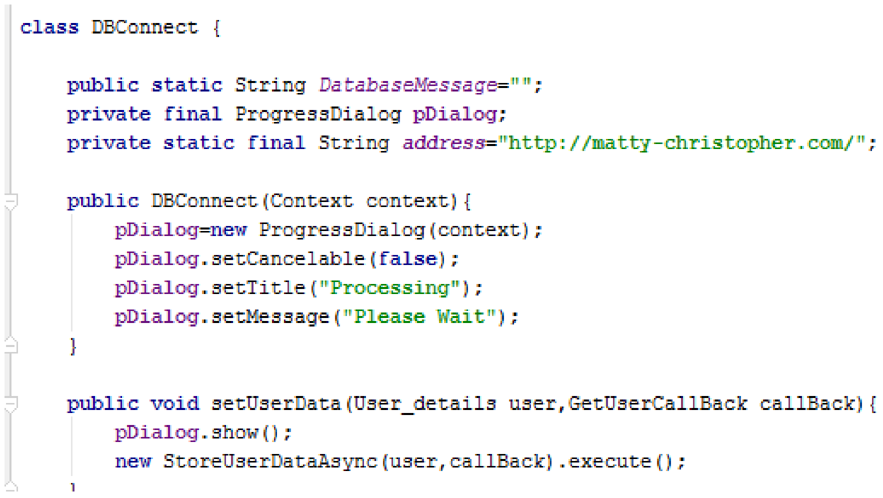
Figure 4: showing casting of xml components

Creating the database classes:

As my app requires a database I needed to create a series of php scripts which are stored on my server at "InMotionHosting.com". I set up a "DBConnect" class which handles all of the scripts for logging in and registration of users. First I created a "User_details "class which stores all the data belonging to a user such as username, password, email, etc... I then setup a shared Preferences class called "User_details_store" which allows data from the "User_details" class to be shared between classes.

The next step was to create a static variable to hold my domain address of where my scripts are held and a ProgressDialog variable to allow use of a progress bar to show the user that the data was being retrieved. The reason for storing the domain address in a global variable is so that in each asynctask which calls the required script I don't have to write the address each time, and if I decide to change the hosting site later I only have to change the address held in the variable once.

To start the progress dialog I created a public constructor for "DBConnect" in which I initialised the dialog bar in the current panel using Context context. Then a series of methods are implemented which are used to display the dialog bar and execute the correct AsyncTask class:



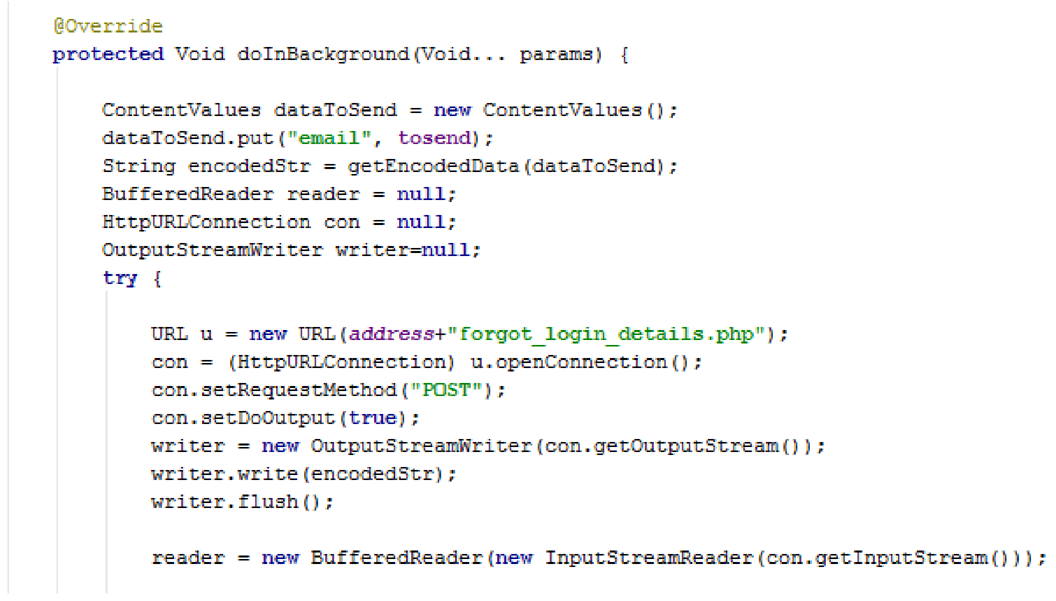
```
class DBConnect {

    public static String DatabaseMessage="";
    private final ProgressDialog pDialog;
    private static final String address="http://matty-christopher.com/";

    public DBConnect(Context context){
        pDialog=new ProgressDialog(context);
        pDialog.setCancelable(false);
        pDialog.setTitle("Processing");
        pDialog.setMessage("Please Wait");
    }

    public void setUserData(User_details user,GetUserCallBack callBack){
        pDialog.show();
        new StoreUserDataAsync(user,callBack).execute();
```

Figure 5: showing DBconnect class with constructor

The reason for displaying progress bars to the user when loading data is to show them that the app is doing something as otherwise they may think that when they load a panel which calls the database they may think while getting the data that the app has frozen or that nothing happened.

The next step was to create the AsyncTask classes such as "StoreUserDataAsync" which allows tasks to be executed on a separate thread so that they can be executed in the background therefore not blocking other threads. This allows use of progress dialogs to be active to the user while the backend code is being executed.

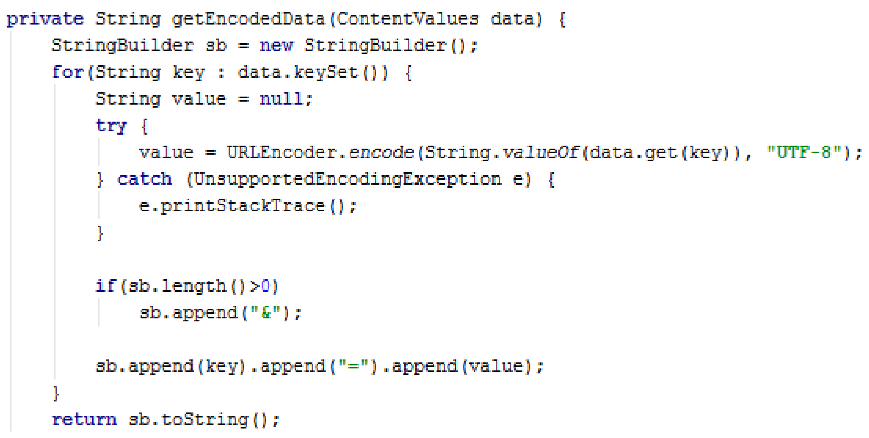
I used 2 of the 4 methods for each AsyncTask in the "DBConnect" class which are: "onPostExecute()" and "doInBackground()", the "onPostExecute()" method is called after the "doInBackground()" method has finished executing. A call-back method will be called which will allow data retrieved from the scripts to be used on the section classes:



```
@Override
protected Void doInBackground(Void... params) {

    ContentValues dataToSend = new ContentValues();
    dataToSend.put("email", tosend);
    String encodedStr = getEncodedData(dataToSend);
    BufferedReader reader = null;
    HttpURLConnection con = null;
    OutputStreamWriter writer=null;
    try {

        URL u = new URL(address+"forgot_login_details.php");
        con = (HttpURLConnection) u.openConnection();
        con.setRequestMethod("POST");
        con.setDoOutput(true);
        writer = new OutputStreamWriter(con.getOutputStream());
        writer.write(encodedStr);
        writer.flush();

        reader = new BufferedReader(new InputStreamReader(con.getInputStream()));
```

Figure 6: showing doInBackground structure of AsyncTask

To make a connection I used the "HTTPURLConnection" class which allows sending data to the script and then getting data back from the response. Data to be sent is stored in ContentValues which gets data stored from the "User_details" class. To send data to the scripts when using the 'POST' request I decided to use 'ContentValues' to store the data as it uses key-value combinations so in the scripts I can call the key using 'POST[]' to retrieve the corresponding piece of data. The data is then encoded using the 'getEncodedData()' method:



```
private String getEncodedData(ContentValues data) {
    StringBuilder sb = new StringBuilder();
    for(String key : data.keySet()) {
        String value = null;
        try {
            value = URLEncoder.encode(String.valueOf(data.get(key)), "UTF-8");
        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        }

        if(sb.length()>0)
            sb.append("&");

        sb.append(key).append("=").append(value);
    }
    return sb.toString();
```

Figure 7: showing getEncodedData method

The method takes in the ContentValue object from the AsyncTask and returns and encoded string of the data to allow it to be passed to the script using the HttpUrlConnection. The method was modified to allow contentValues to be used from [http://stackoverflow.com/questions/30740359/namevaluepair-httpparams-httpconnection-params-deprecated-on-server-request-cl, by Gagandeep Singh].

Creating the registration section:

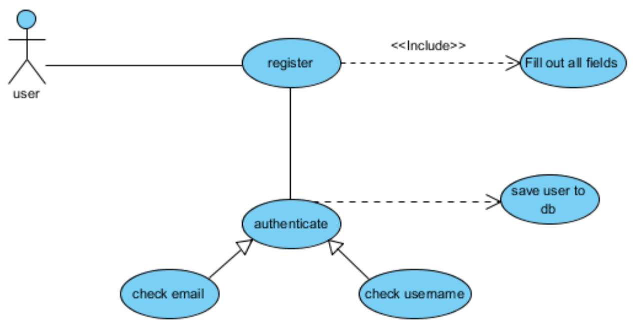
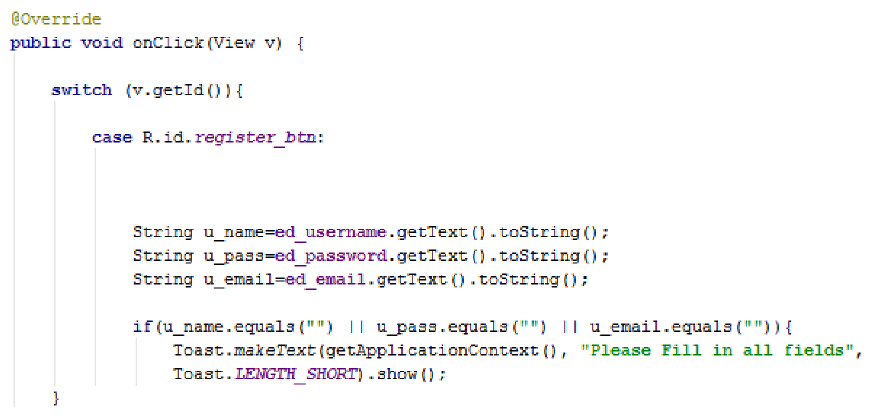


Figure 8: shows use case for registering a new user

The use case shows that to register the user must fill out all the fields provided by the registration panel which upon completing will be authenticated through checking the username and email to see if they are unique, if they are then the user will be added to the database and they can then login.

To create a new user the user must register a username and password, the entered data will then need to be authenticated by checking the relevant database table to check if the entered username is unique.

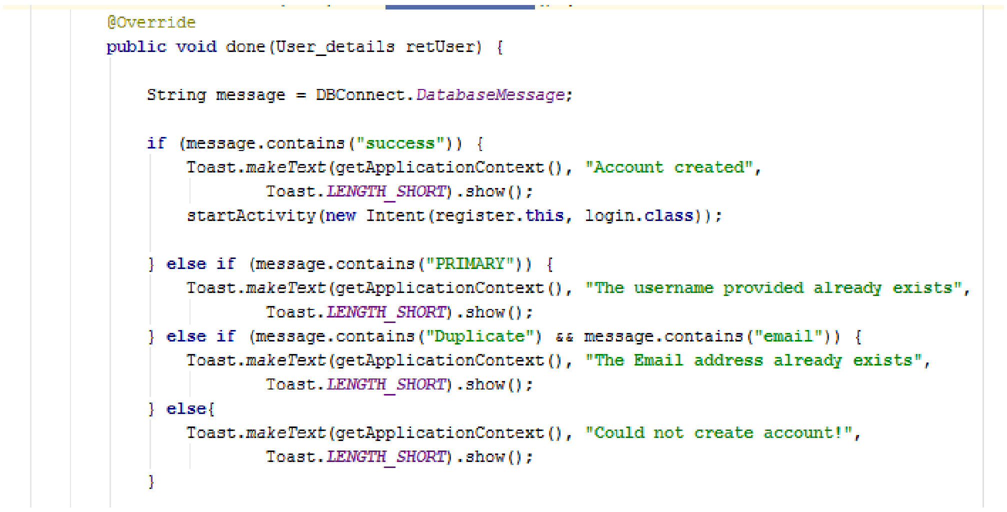
To start with I created the interface through the xml layout file called:"activity_register.xml" and then giving the components functionality in "register.java". The next step was to create the onclick listener for the register button which I made the register class implement "View.OnClickListener" which let me implement the "onclick()" method. In this method I used the "getText()" methods to retrieve the text from the edit text fields which I stored in new local variables where I then was able to use an if else statement to check if any of these fields were empty, here a Toast message would be displayed if any field was empty:



```
@Override
public void onClick(View v) {

    switch (v.getId()){

        case R.id.register_btn:


            String u_name=ed_username.getText().toString();
            String u_pass=ed_password.getText().toString();
            String u_email=ed_email.getText().toString();

            if(u_name.equals("") || u_pass.equals("") || u_email.equals("")){
                Toast.makeText(getApplicationContext(), "Please Fill in all fields",
                Toast.LENGTH_SHORT).show();
    }
```

Figure 9: showing onClick validation

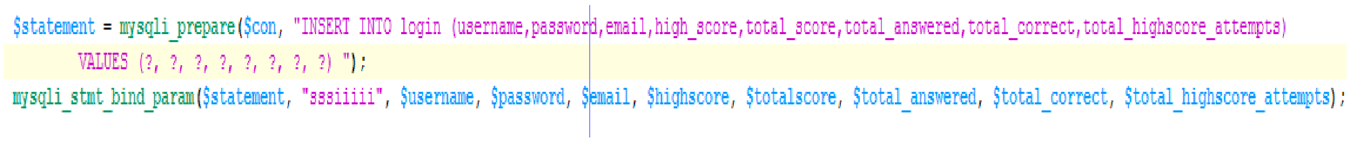
Otherwise the database class would be called. I created a new method "registerUser(user)" which handles calling the "DBConnect" class. A new object of the class is created which then allows me to call the "setUserData()" method which in turn executes the "StoreUserDataAsync" class which calls the "register.php" script on the server. I implemented a global variable to keep track of the return message of the script which I then call in the "done()" method from the passed through "getusercallback" class, here I implemented an if -else if- else block where I display a series of Toast messages depending on the response by implementing the statements to use contains() which look for specific words. If the response is "success" i call the "startActivity()" method to change the panel to login:



```java
@Override
public void done(User_details retUser) {

    String message = DBConnect.DatabaseMessage;

    if (message.contains("success")) {
        Toast.makeText(getApplicationContext(), "Account created",
                Toast.LENGTH_SHORT).show();
        startActivity(new Intent(register.this, login.class));

    } else if (message.contains("PRIMARY")) {
        Toast.makeText(getApplicationContext(), "The username provided already exists",
                Toast.LENGTH_SHORT).show();
    } else if (message.contains("Duplicate") && message.contains("email")) {
        Toast.makeText(getApplicationContext(), "The Email address already exists",
                Toast.LENGTH_SHORT).show();
    } else{
        Toast.makeText(getApplicationContext(), "Could not create account!",
                Toast.LENGTH_SHORT).show();
    }
```

Figure 10: showing validation of database retrieved value

For database I created a user table called login which has a primary key called 'username' and an 'email' field which is unique. I decided to make the email unique so to only allow one user per account to make managing forgotten passwords easier and to try and keep one account per person so help make it more social. The data from the contentValues are placed in variables in the register.php script and then called in the 'mysqli_stmt_bind_params' method as I decided to use a prepared statement as they are more efficient as the overhead of compiling is done once, and they help prevent against sql injections in which a user could manipulate the data. The script will execute the statement and return either 'success' or 'failure' along with the reason why it failed. The message is then filtered in the registerUser method to check whether the insert failed or passed.



```php
$statement = mysqli_prepare($con, "INSERT INTO login (username,password,email,high_score,total_score,total_answered,total_correct,total_highscore_attempts)
        VALUES (?, ?, ?, ?, ?, ?, ?, ?) ");
mysqli_stmt_bind_param($statement, "sssiiiii", $username, $password, $email, $highscore, $totalscore, $total_answered, $total_correct, $total_highscore_attempts);
```

Figure 11: shows the prepared statement for registering a user

Creating the login section:

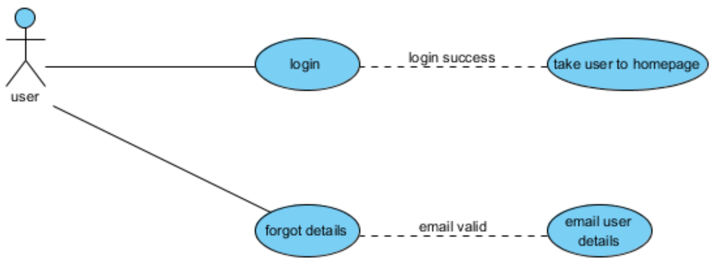
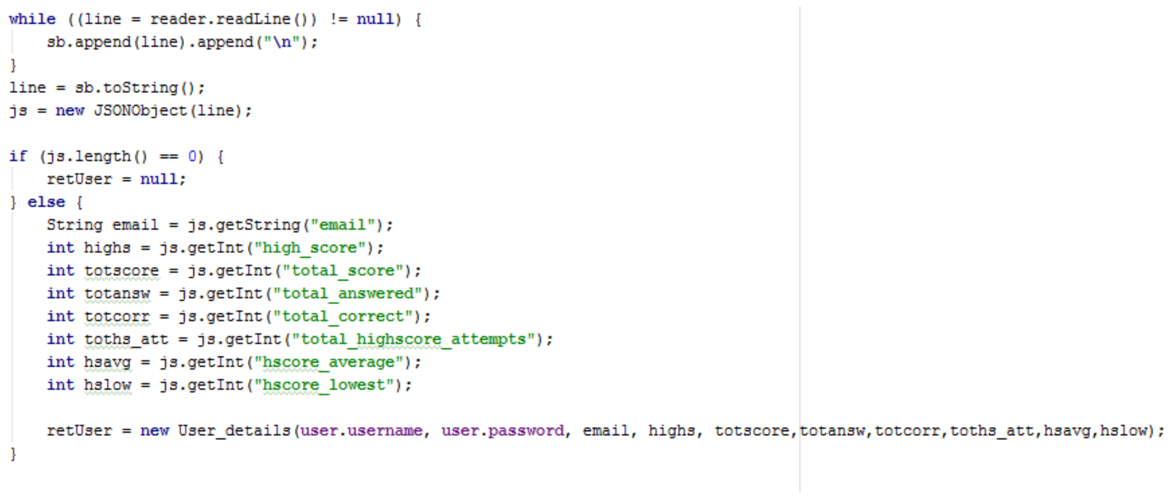


figure 12: shows use case for login screen

The use case diagram shows that the user has to ways of interacting with the system to login, they can enter login details and if the details are successful they will be transported to the homepage. Another interaction that they can do is if they have forgotten their login details they can enter their email address in which the system will then email the user with the corresponding username and password if the email address is valid.

For the login class my implementation is similar to the register class for setting up the interface and calling the database. Here I call the "getUserData()" method in the "DBConnect" class which calls the "GetUserDataAsync" class which I implemented to call the "login.php" script. Here if the response is successful I implemented it so that the returned value Is the new logged in user which is then stored in the "User_details" class as retUser object.

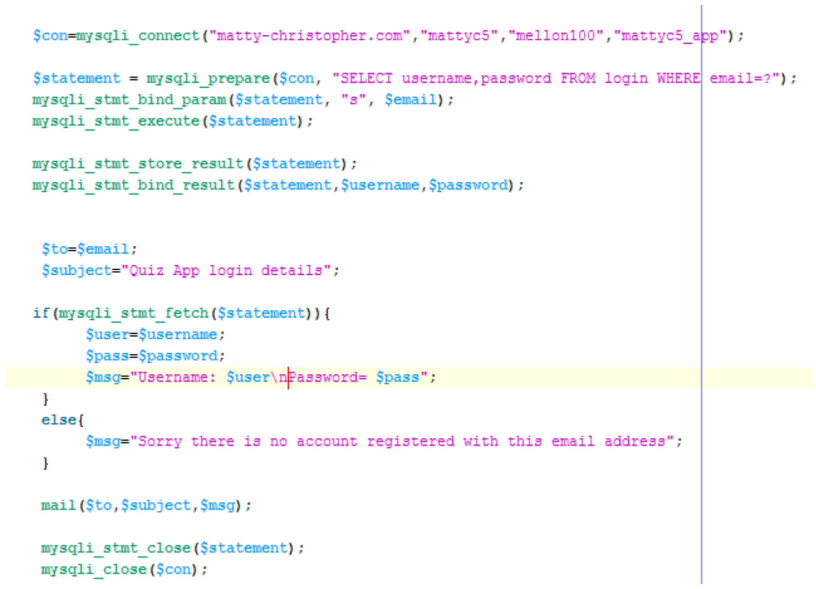


```
while ((line = reader.readLine()) != null) {
    sb.append(line).append("\n");
}
line = sb.toString();
js = new JSONObject(line);

if (js.length() == 0) {
    retUser = null;
} else {
    String email = js.getString("email");
    int highs = js.getInt("high_score");
    int totscore = js.getInt("total_score");
    int totansw = js.getInt("total_answered");
    int totcorr = js.getInt("total_correct");
    int toths_att = js.getInt("total_highscore_attempts");
    int hsavg = js.getInt("hscore_average");
    int hslow = js.getInt("hscore_lowest");

    retUser = new User_details(user.username, user.password, email, highs, totscore,totansw,totcorr,toths_att,hsavg,hslow);
}
```

Figure 13: showing getting json data from script

If the returned user is not null then the login class will call the "startActivity()" method and store the user details into the "User_details_store" shared preference class. A shared preference is used to store user data as they keep a single instance of a class so that the data placed in the preference can be accessed by any activity which calls the preference, this reduces the need of passing data through to activities and helps easily store and retrieve data into the user class.

I also implemented a textbox to be a button in which allows users to send an email to request their details. To do this I created another "DBConnect" object which then calls the "getUserEmail()" method which will execute the AsyncTask class which sends the email address provided by the user to the email.php script which will send an email with the username and password if it exists or a message saying that this email address has no assigned user.



```
$con=mysqli_connect("matty-christopher.com","mattyc5","mellon100","mattyc5_app");

$statement = mysqli_prepare($con, "SELECT username,password FROM login WHERE email=?");
mysqli_stmt_bind_param($statement, "s", $email);
mysqli_stmt_execute($statement);

mysqli_stmt_store_result($statement);
mysqli_stmt_bind_result($statement,$username,$password);


 $to=$email;
 $subject="Quiz App login details";

if(mysqli_stmt_fetch($statement)){
      $user=$username;
      $pass=$password;
      $msg="Username: $user\nPassword= $pass";
 }
 else{
      $msg="Sorry there is no account registered with this email address";
 }

 mail($to,$subject,$msg);

 mysqli_stmt_close($statement);
 mysqli_close($con);
```

Figure 14: showing script for emailing user details

Creating the leaderboard section:

As one of the features that I intend to add is a highscore mode, I thought it would be best to also have a leaderboard displaying the best users. This gives motivation for users to keep using the app as they will be able to see where they stack up against their friends and other players, it should hopefully motivate them to get better and help them revise for tests in a fun and enjoyable way.

To create the leaderboard section I initially created the table component with a single row which displays rank, username and highscore text fields, I then added a scrollview ontop of the table to allow users to scroll up and down, this is what the screen initially looks like:

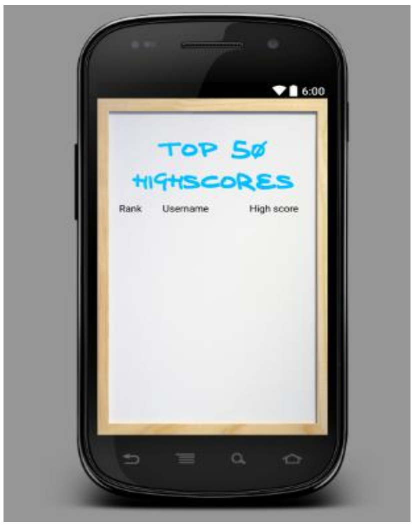
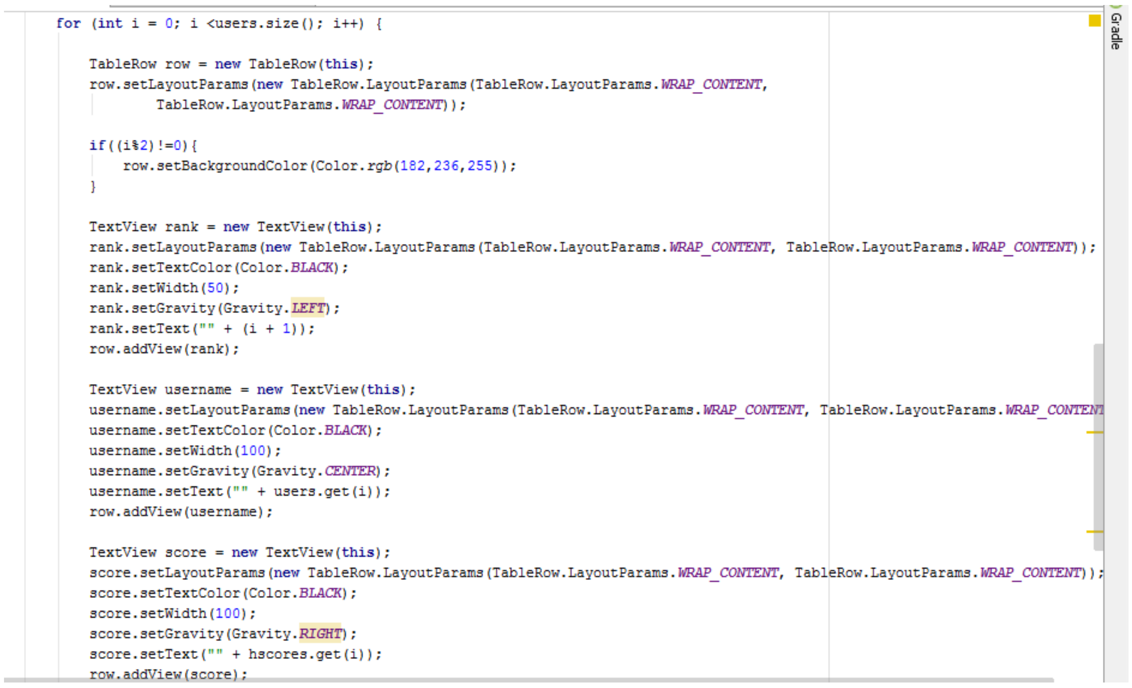


Figure 15: showing leaderboard design

I then implemented the database connection which retrieves the data from the script and stores the data into 2 arraylists: user and highscore. Here I then created a loop which goes around user.size()-1 times which inside the loop I create new rows for each user.



```
for (int i = 0; i <users.size(); i++) {

    TableRow row = new TableRow(this);
    row.setLayoutParams(new TableRow.LayoutParams(TableRow.LayoutParams.WRAP_CONTENT,
            TableRow.LayoutParams.WRAP_CONTENT));

    if((i%2)!=0){
        row.setBackgroundColor(Color.rgb(182,236,255));
    }

    TextView rank = new TextView(this);
    rank.setLayoutParams(new TableRow.LayoutParams(TableRow.LayoutParams.WRAP_CONTENT, TableRow.LayoutParams.WRAP_CONTENT));
    rank.setTextColor(Color.BLACK);
    rank.setWidth(50);
    rank.setGravity(Gravity.LEFT);
    rank.setText("" + (i + 1));
    row.addView(rank);

    TextView username = new TextView(this);
    username.setLayoutParams(new TableRow.LayoutParams(TableRow.LayoutParams.WRAP_CONTENT, TableRow.LayoutParams.WRAP_CONTENT
    username.setTextColor(Color.BLACK);
    username.setWidth(100);
    username.setGravity(Gravity.CENTER);
    username.setText("" + users.get(i));
    row.addView(username);

    TextView score = new TextView(this);
    score.setLayoutParams(new TableRow.LayoutParams(TableRow.LayoutParams.WRAP_CONTENT, TableRow.LayoutParams.WRAP_CONTENT));
    score.setTextColor(Color.BLACK);
    score.setWidth(100);
    score.setGravity(Gravity.RIGHT);
    score.setText("" + hscores.get(i));
    row.addView(score);
```

Figure 16: shows dynamic table layout adding rows

The final step was to implement an if else statement at the end of the loop which checks if the current user matches the user in the shared preference class, if the user matches then I made that row's background green so that the user can clearly see their rank.

```
if(users.get(i).equals(currentUser.username)){
    row.setBackgroundColor(Color.rgb(0,255,51));
}
```

Figure 17: changing background colour of row for logged in user

To retrieve users from the database, the script used returns all usernames and corresponding highscore fields which are then ordered by the highscore field and limited to the first 50 users.

```
mysqli_query($con, "SELECT username,high_score FROM login ORDER BY high_score DESC LIMIT 50");
```

Figure 18: select statement for leaderboard from script

The reason for limiting to the top 50 as I felt that there needed to be a cut off for when the app is used by lots of people, as if all users were displayed then it would make browsing harder and would include lots of lower scores. Users not on the leaderboard will be able to gage roughly how good their score is compared to the best users as their score would be displayed on the homepage.

The create section:

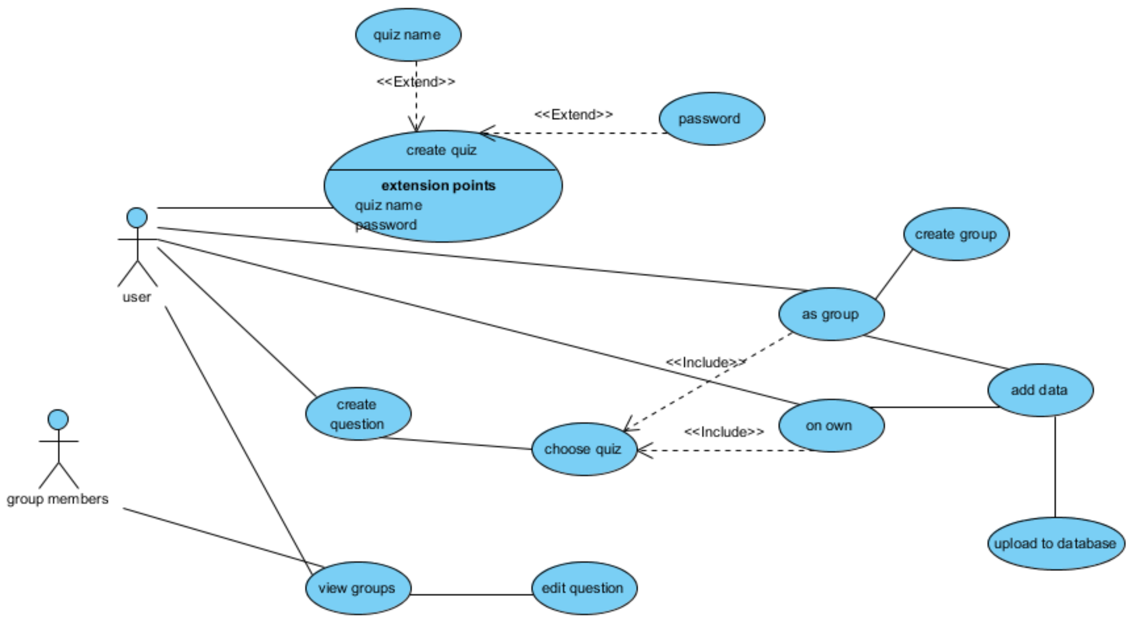
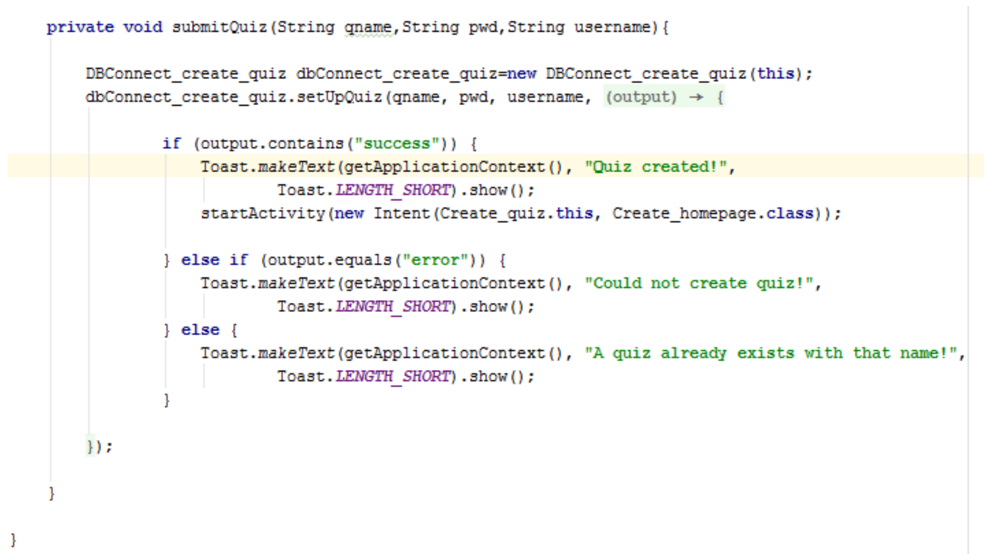


Figure 19: showing use case interactions for create section

For creating a question the user will need to first choose a quiz to add it to, the diagram above shows the interactions that can be taken by the user to add a question. The user will need to first create a quiz in which they must enter a unique name and a password to access the quiz. The user can also add a question to another users quiz if they are given the password to access it. The reason for requiring a password is that the app is intended for use as a self-study aid for a user/ groups of users so a user won't want anyone who they don't know to add a question to a specific quiz as it may not be relevant to the quiz topic. This way will allow a user to choose who should have access to certain quizzes. Once a quiz is chosen the user can decide to create it as part of a group or on their own. If they choose to create it as part of a group they can choose up to other 3 users to have access to editing the question. Once the user has added a question and answer the group members will have access to the question which will be shown in the view groups section where they can choose a question that they are part of then add an answer to it in which they have been assigned to.

Creating the create quiz section:

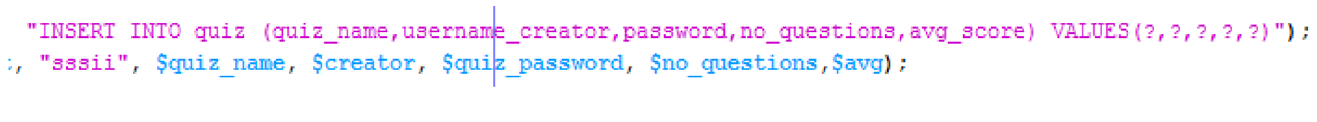
To create this section is very similar to login as there are two edit text boxes to enter a quiz name and password, here the "DBConnect_create_quiz" class will be called which calls the php script to check whether a quiz with that name exists or not, if it exists then a new entry is added with all the stats colums set to 0. Otherwise a toast message is called and displays an exists already message:



```
    private void submitQuiz(String qname,String pwd,String username){

        DBConnect_create_quiz dbConnect_create_quiz=new DBConnect_create_quiz(this);
        dbConnect_create_quiz.setUpQuiz(qname, pwd, username, (output) → {

                if (output.contains("success")) {
                    Toast.makeText(getApplicationContext(), "Quiz created!",
                            Toast.LENGTH_SHORT).show();
                    startActivity(new Intent(Create_quiz.this, Create_homepage.class));

                } else if (output.equals("error")) {
                    Toast.makeText(getApplicationContext(), "Could not create quiz!",
                            Toast.LENGTH_SHORT).show();
                } else {
                    Toast.makeText(getApplicationContext(), "A quiz already exists with that name!",
                            Toast.LENGTH_SHORT).show();
                }

        });

    }

}
```

Figure 20: showing validation for quiz depending on output result

The quiz name and password will be sent to the corresponding script which contains an insert sql prepared statement to insert a new row into the quiz table:



```
 "INSERT INTO quiz (quiz_name,username_creator,password,no_questions,avg_score) VALUES(?,?,?,?,?)");
:, "sssii", $quiz_name, $creator, $quiz_password, $no_questions,$avg);
```

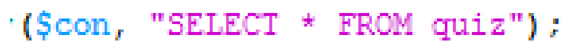
Figure 21: insert statement using prepared statement

The number of questions and average fields will be initially set to 0 indicating that they contain no questions.

Creating the create question section:

The first step in adding a new question to a quiz is for the user to enter the quiz that they wish to add it to and then add the corresponding password. The user can then authenticate by pressing a submit button which will check the database quiz table to see if there exists that quiz name and whether the password is correct.

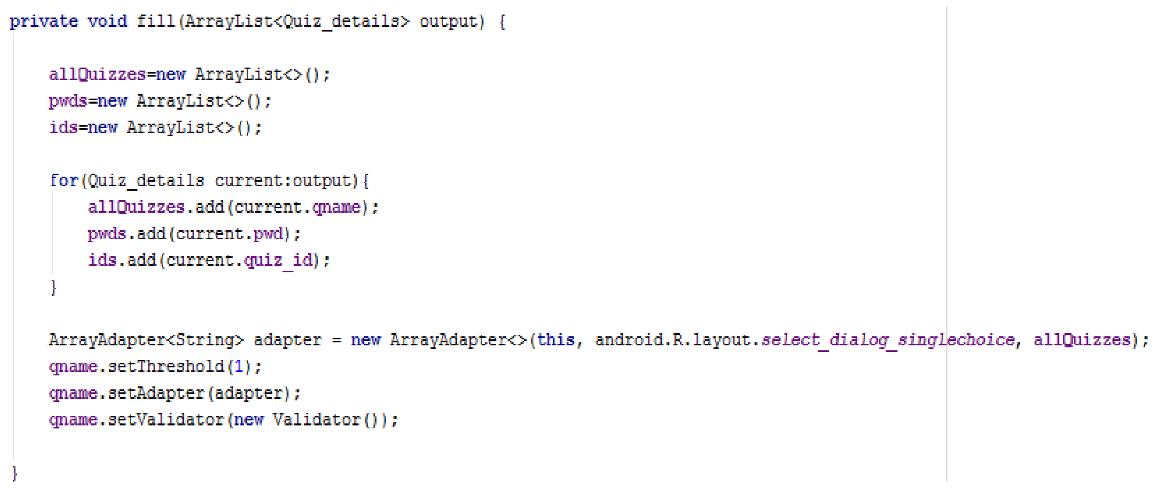
To do this I first had to create the panel which contains one EditText box, an AutoCompleteTextView and a Button component. In the java file 'create_question_start' the 'onStart' method calls a method 'getQuizzes' which creates a new AsyncTask to make a database connection to retrieve all quizzes currently in the database.



```
($con, "SELECT * FROM quiz");
```

Figure 22: shows the simple select query to retrieve all quizzes.

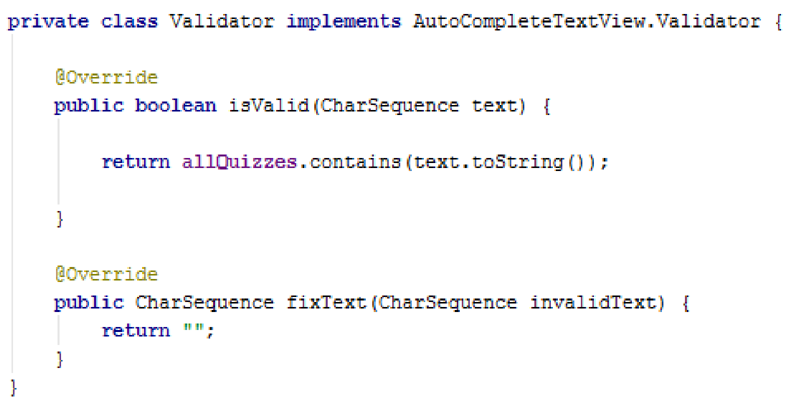
The figure above shows that it is a simple select call to retrieve all of the quizzes which are then stored in an array in the script where each row is a position in the array. The array is then encoded in json format using ‘json_encode ' and returned to the Activity using the AsyncResponse interface using the implemented method 'processFinish' where the json data is stored into Quiz_details objects which are added to an array list of Quiz_details. The returned array list is passed onto the 'fill' method which initializes 3 array lists to store the quiz names, passwords and id for each quiz. The data from the array list of Quiz_details is stored in these array lists by going through each Quiz_details object using a for each loop:



```
private void fill(ArrayList<Quiz_details> output) {

    allQuizzes=new ArrayList<>();
    pwds=new ArrayList<>();
    ids=new ArrayList<>();

    for(Quiz_details current:output){
        allQuizzes.add(current.qname);
        pwds.add(current.pwd);
        ids.add(current.quiz_id);
    }

    ArrayAdapter<String> adapter = new ArrayAdapter<>(this, android.R.layout.select_dialog_singlechoice, allQuizzes);
    qname.setThreshold(1);
    qname.setAdapter(adapter);
    qname.setValidator(new Validator());

}
```

Figure 23: fill method where array adapter created

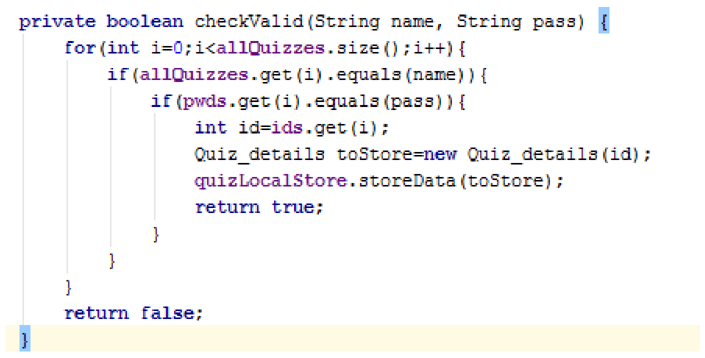
The allQuizzes, pwds and ids array lists are global allowing use throughout the activity. Here I can then implement the AutoCompleteTextView with the allQuizzes data, which stores all the quiz names. To do this I first created an ArrayAdapter-which can provide a view for each object in the array. I set the threshold to one so as soon as the user enters a single character names in the array with that character will be returned. A new Validator object is assigned to the adapter which is used to check if data input in the text box is contained in the array or not. If the input is invalid then it is removed from the textbox displaying no text as soon as the user removes focus from the text box.



```
private class Validator implements AutoCompleteTextView.Validator {

    @Override
    public boolean isValid(CharSequence text) {

        return allQuizzes.contains(text.toString());

    }

    @Override
    public CharSequence fixText(CharSequence invalidText) {
        return "";
    }
}
```

Figure 24: showing the Validator class dealing with invalid text

Once a valid quiz name is entered then the user can enter the corresponding password which will then be checked as the user clicks on the submit button.



```
private boolean checkValid(String name, String pass) {
    for(int i=0;i<allQuizzes.size();i++){
        if(allQuizzes.get(i).equals(name)){
            if(pwds.get(i).equals(pass)){
                int id=ids.get(i);
                Quiz_details toStore=new Quiz_details(id);
                quizLocalStore.storeData(toStore);
                return true;
            }
        }
    }
    return false;
}
```

Figure 25: showing submit button validation for correct password

As the quiz name is already valid when the submit button is pressed only the password needs to be checked, first the quizname value and password value from the text boxes are passed to the 'checkValid' method. This method returns true or false depending on whether the password matches or not, the array lists of allQuizzes and pwds are searched through using a loop until the name variable matches the value in allQuizzes, then the password value is checked against the pwds list at the stopped position and returns true if they match and creates a Quiz_details object and stores it in the shared preference belonging to quizzes. The user is then taken to the choose mode activity if true is returned otherwise a toast message is output explaining to the user that the password is incorrect.

The reason for choosing an AutoCompleteTextView is that initially when testing it I found that with an edit text box it was hard to remember the quiz names, so I decided to start again and have the AutoCompleteTextView as it allows the user to only partially enter a quiz name before seeing the quiz name that they are after. I also decided to store all passwords in an array list as it reduces the need for an extra database call to pass the users password to a script to check if it is valid, as the validation can be done straight away requiring only the initial database call to fill the auto complete box.

Create group section:

Once the user has successfully chosen a quiz they will be presented with the option of creating the question in a group or on their own:

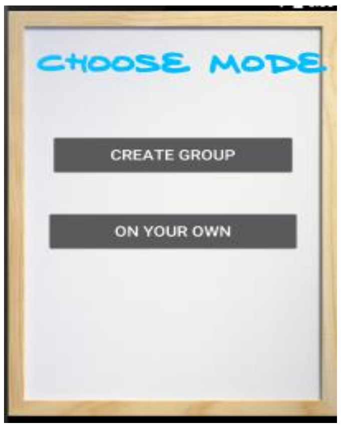
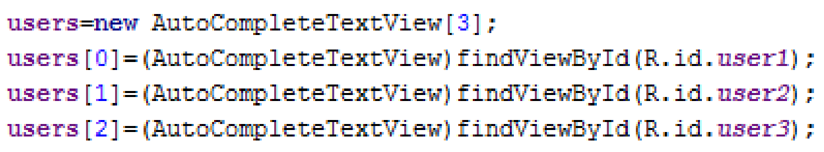


Figure 26: showing user option for creating question

The 'on your own' button will take the user straight to the 'add question' panel where the user can fill out the question on their own. However if the 'create group' button is pressed then they will be taken to the 'create group' panel where they can assign up to 3 members to have access to the quiz.

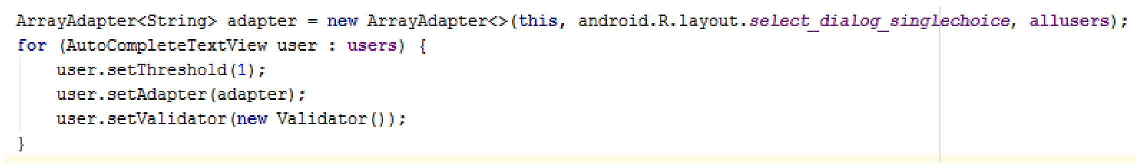
To create a group I decided to use AutoCompleteTextView boxes for entering usernames as it is allows the user to only partially complete entering a username, and it allows validation of a user as soon as focus is removed from the box. To do this instead of 1 text view I needed 3 so I created in the java file an array of AutoCompleteTextView:



```
users=new AutoCompleteTextView[3];
users[0]=(AutoCompleteTextView)findViewById(R.id.user1);
users[1]=(AutoCompleteTextView)findViewById(R.id.user2);
users[2]=(AutoCompleteTextView)findViewById(R.id.user3);
```

Figure 27: shows array of AutoCompleteTextView being initialized

This then allowed me to use loops to add array adapters and other data to the text boxes. An array list containing all users is initialized in the 'onStart' method were a single database call is made retrieving all users via a simple *select all from login query*.

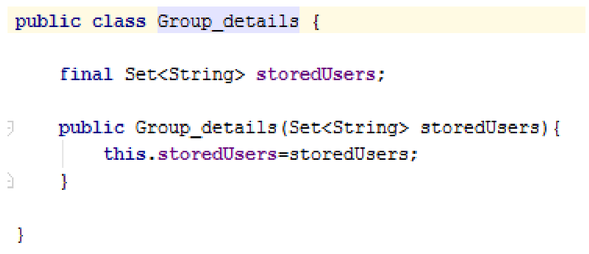


```
ArrayAdapter<String> adapter = new ArrayAdapter<>(this, android.R.layout.select_dialog_singlechoice, allusers);
for (AutoCompleteTextView user : users) {
    user.setThreshold(1);
    user.setAdapter(adapter);
    user.setValidator(new Validator());
}
```

Figure 28: showing use of loop to create array adapters for text view array positions

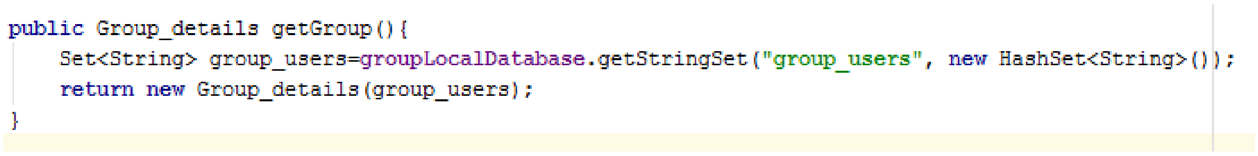
The logged in user's username is removed from the array list of usernames by using the array list method 'remove' to remove the username retrieved from the shared preference 'user_details_store'.

Once at least one text box is valid (not empty) then all the values from the text boxes are added into a Hash Set- the reason for this is that Hash sets don’t allow duplicate values so if the user enters the same user 3 times then only one copy of the name will be saved in the set. The set is then stored in a shared preference 'group_details_store' as the 'group_details' class takes in a set:



```
public class Group_details {

    final Set<String> storedUsers;

    public Group_details(Set<String> storedUsers){
        this.storedUsers=storedUsers;
    }

}
```

Figure 29: Group details class



```
public Group_details getGroup(){
    Set<String> group_users=groupLocalDatabase.getStringSet("group_users", new HashSet<String>());
    return new Group_details(group_users);
}
```

Figure 30: getGroup method from group_details_store to retrieve group_details

When going to the 'add question' activity a string value is passed to the activity using putExtra() method in intent to send a value 'group' to indicate that the 'add question' activity should access the data specifically for when part of a group.

If the user presses the submit button without entering a valid user (all text boxes being empty) a toast message will appear explaining to the user that at least one user is required to be added to the group. The reason for requiring at least one user is that otherwise the user would be creating it on their own- if they want to do that then can press the back button on their device and choose 'on your own'. It wouldn’t make a difference if they didn’t add any members but when submitting as part of a group initially the complete variable in the database is set to 0 so the user will have to go to view groups to submit a complete version, so it makes more sense for a user to not be allowed to create a group on their own.

Create Add Question section:

The next stage is to add the question, this activity uses 5 edit text boxes, a radio group with 4 radio buttons, 4 spinner components, and a button. Here the user can fill in all the text boxes with data and then choose the correct answer using the radio buttons located next to each answer box. The spinner components are drop down boxes containing all the users assigned to the question with the creator as first option. The user can then tap on a spinner located next to each answer to assign the user they wish to fill in this answer-unless the user choose to do it on their own the spinner will only have their name.

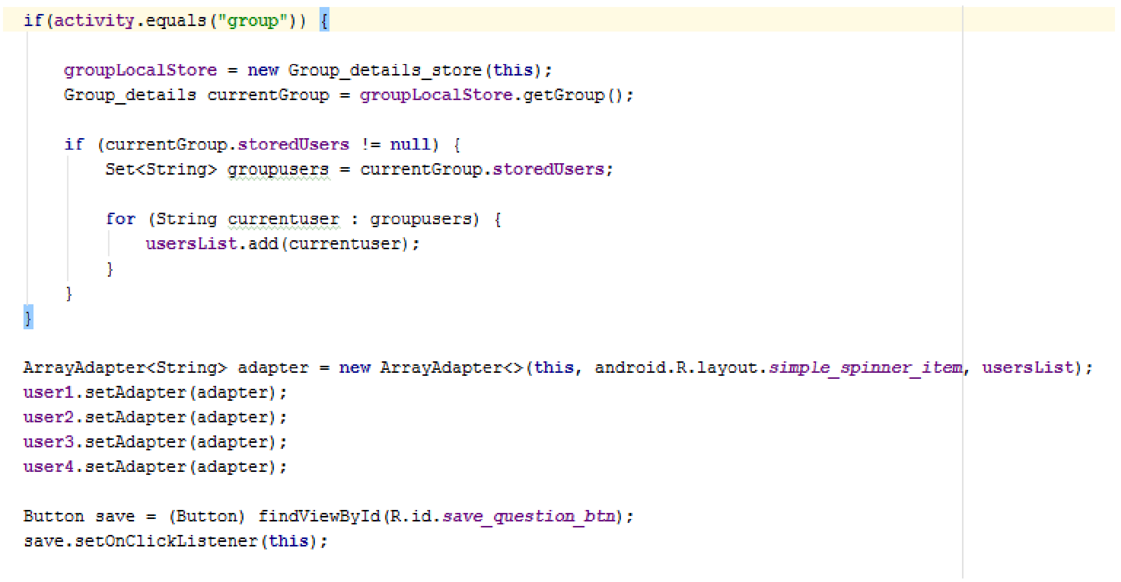
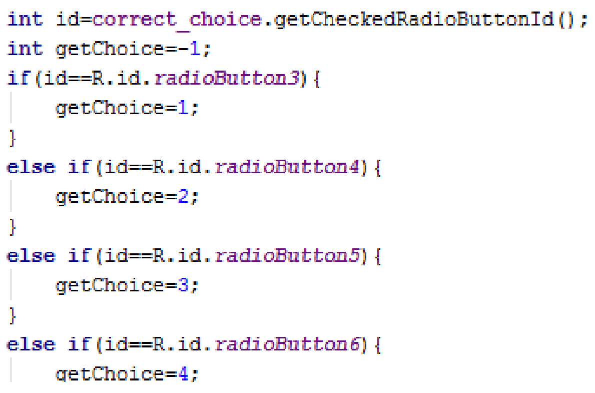

```
if(activity.equals("group")) {

    groupLocalStore = new Group_details_store(this);
    Group_details currentGroup = groupLocalStore.getGroup();

    if (currentGroup.storedUsers != null) {
        Set<String> groupusers = currentGroup.storedUsers;

        for (String currentuser : groupusers) {
            usersList.add(currentuser);
        }
    }
}

ArrayAdapter<String> adapter = new ArrayAdapter<>(this, android.R.layout.simple_spinner_item, usersList);
user1.setAdapter(adapter);
user2.setAdapter(adapter);
user3.setAdapter(adapter);
user4.setAdapter(adapter);

Button save = (Button) findViewById(R.id.save_question_btn);
save.setOnClickListener(this);
```

Figure 31: showing adding usernames to spinners

The figure above shows that if the passed through value is equal to 'group' from the putExtra() method then the activity will access the shared preference holding the 'Group_details' object and fill the array list of users with the retrieved users. I choose to use spinners as it allows the user to just tap on a spinner to choose users instead of typing them in and relates it to 'add groups' activity where the user already choose the group-if group selected.

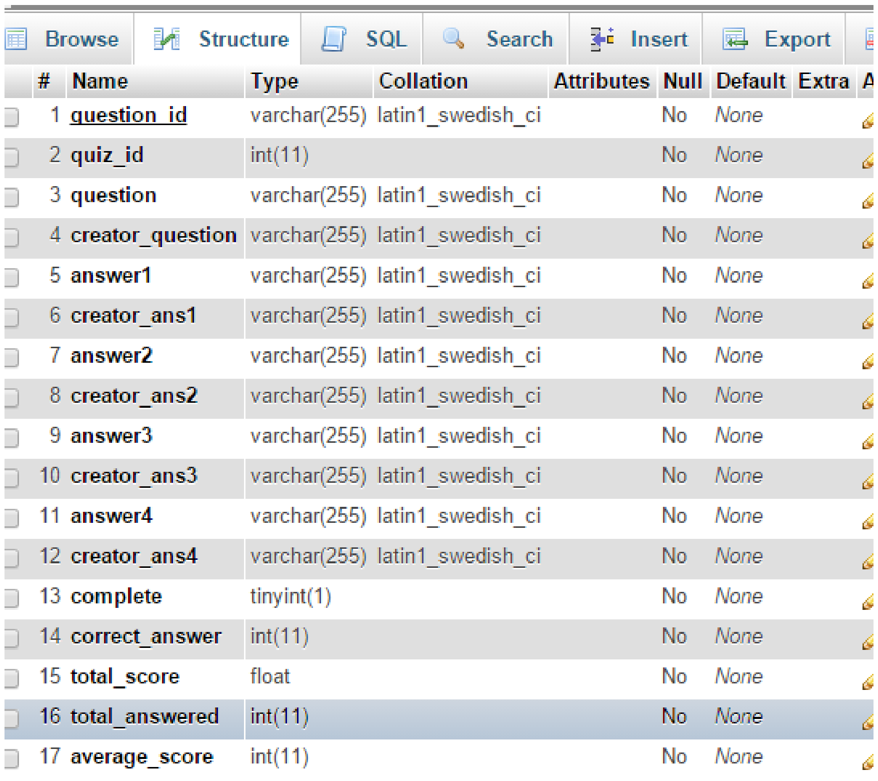
When the user clicks on the submit button, the 'onClick' method will call the 'authenticate' method which retrieves all the text from the text boxes, users from the spinners and the radio button number (1-4).



```
int id=correct_choice.getCheckedRadioButtonId();
int getChoice=-1;
if(id==R.id.radioButton3){
    getChoice=1;
}
else if(id==R.id.radioButton4){
    getChoice=2;
}
else if(id==R.id.radioButton5){
    getChoice=3;
}
else if(id==R.id.radioButton6){
    getChoice=4;
```

Figure 32: shows getting the radio button id for selected answer

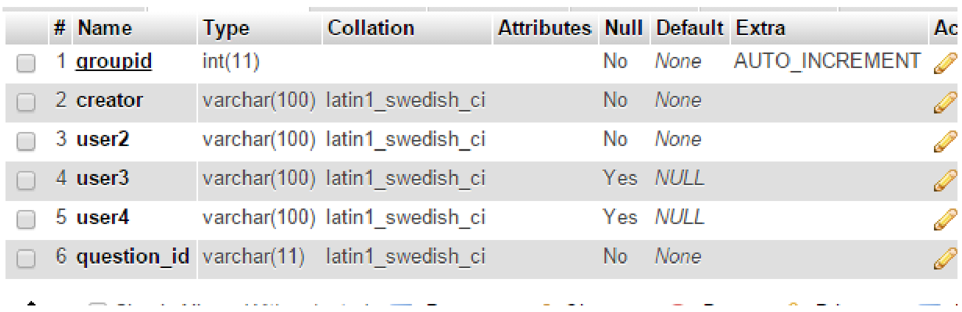
As the figure above shows if a user doesn't select a radio button then the getChoice value will stay as -1 indicating not selected. A series of if else statements will then check whether the user is in a group or not. If the user is on their own then it will check to see if all data is filled in- all answer text boxes filled in, the question text box added and a radio button selected to indicate the correct answer. If the validation passes then the data will be sent to the script 'setupQuestion' with the answers, question, answer, users and complete which will be inserted into the database, upon adding the question a new row in the question stats table will be added which corresponds to the question id and will set the stats fields to 0 and then update the quiz table row of the quiz number of questions field to be incremented by 1. If the user is part of a group then the complete variable sent will be set to 0 indicating that it should not be used in the play section and can be edited.



#	Name	Type	Collation	Attributes	Null	Default	Extra	A
1	question_id	varchar(255)	latin1_swedish_ci		No	None		
2	quiz_id	int(11)			No	None		
3	question	varchar(255)	latin1_swedish_ci		No	None		
4	creator_question	varchar(255)	latin1_swedish_ci		No	None		
5	answer1	varchar(255)	latin1_swedish_ci		No	None		
6	creator_ans1	varchar(255)	latin1_swedish_ci		No	None		
7	answer2	varchar(255)	latin1_swedish_ci		No	None		
8	creator_ans2	varchar(255)	latin1_swedish_ci		No	None		
9	answer3	varchar(255)	latin1_swedish_ci		No	None		
10	creator_ans3	varchar(255)	latin1_swedish_ci		No	None		
11	answer4	varchar(255)	latin1_swedish_ci		No	None		
12	creator_ans4	varchar(255)	latin1_swedish_ci		No	None		
13	complete	tinyint(1)			No	None		
14	correct_answer	int(11)			No	None		
15	total_score	float			No	None		
16	total_answered	int(11)			No	None		
17	average_score	int(11)			No	None		

Figure 33: shows the table for questions where the added question goes

If in a group once the question has been added the group will then be saved to the database allowing group members to access and edit the question on another device.



#	Name	Type	Collation	Attributes	Null	Default	Extra	Ac
1	groupid	int(11)			No	None	AUTO_INCREMENT	
2	creator	varchar(100)	latin1_swedish_ci		No	None		
3	user2	varchar(100)	latin1_swedish_ci		No	None		
4	user3	varchar(100)	latin1_swedish_ci		Yes	NULL		
5	user4	varchar(100)	latin1_swedish_ci		Yes	NULL		
6	question_id	varchar(11)	latin1_swedish_ci		No	None		

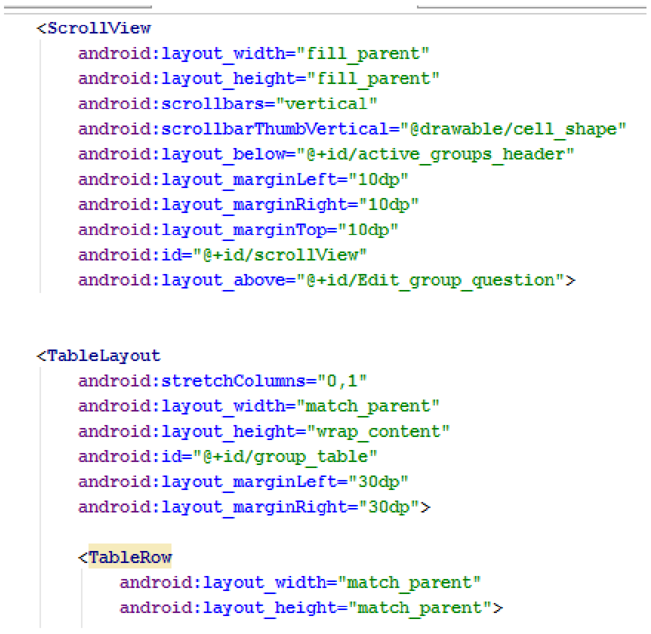
Figure 34: This shows the groups table which corresponds to a question

The figure above shows the table for created groups which allows user3 and 4 to be null so any combination of 2, 3 or 4 members can be part of a group. This is a design choice to allow for more flexibility as it can often be hard to get more than a couple of members in a group so it allows for more flexibility.

Creating view groups:

To allow a user to edit a question in which they have access to as part of a group, they can view all the incomplete questions here. To do this I decided to display the retrieved questions in a simple table which a user can click on a row which will take them to the corresponding question to edit it.

Firstly I created the table layout in the xml file 'activity_view_groups' where I defined a table layout and nested it inside a scrollview to allow the user to scroll through the table when it gets too big.

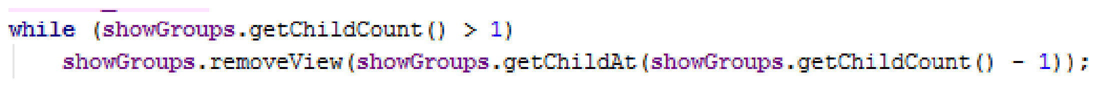


```
<ScrollView
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:scrollbars="vertical"
    android:scrollbarThumbVertical="@drawable/cell_shape"
    android:layout_below="@+id/active_groups_header"
    android:layout_marginLeft="10dp"
    android:layout_marginRight="10dp"
    android:layout_marginTop="10dp"
    android:id="@+id/scrollView"
    android:layout_above="@+id/Edit_group_question">


<TableLayout
    android:stretchColumns="0,1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/group_table"
    android:layout_marginLeft="30dp"
    android:layout_marginRight="30dp">

    <TableRow
        android:layout_width="match_parent"
        android:layout_height="match_parent">
```

Figure 35: xml file showing part of table

As the image above shows, I created an initial table row which serves as the column headers, as these won't change it made sense to create them in the xml. Meaning that I don't have to keep adding the same headers over and over again when reloading the page/refreshing. The next step was to define a global variable called 'current_groupid' which is initally set to '-1' in the 'onStart' method. This variable is used to keep track of the current row selected with '-1' indicating no selected row. This is then checked when the submit button is pressed to see which question to load.

In the 'onStart' method I decided to delete all rows of the table except the first- this is to prevent when the 'onStart' is called which will increment the rows to the bottom of the table layout, so this solution below is used to always keep the headers row from being deleted and preventing multiple rows which are the same.



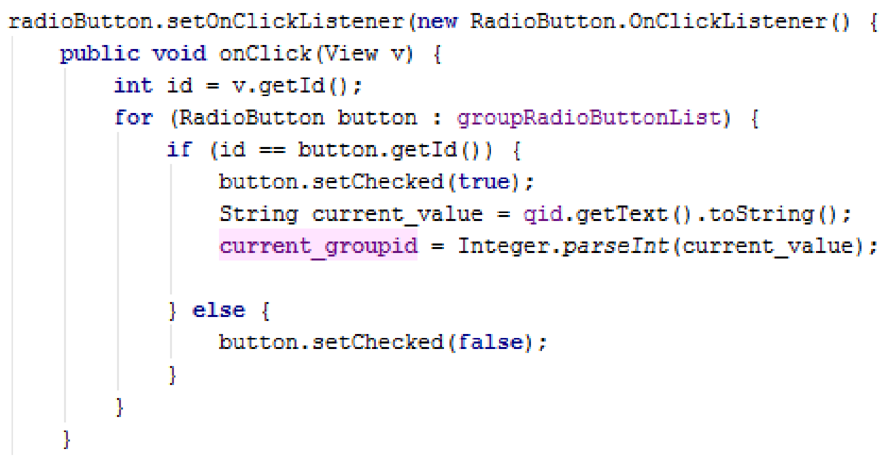
```
while (showGroups.getChildCount() > 1)
    showGroups.removeView(showGroups.getChildAt(showGroups.getChildCount() - 1));
```

Figure 36: solution to preventing duplicate rows at table bottom when page reloaded

id	**Question id**
1	32_0
2	39_7
1	32_0

Figure 37: table highlighting problem that Fig 36 above fixes

To allow users to select a row, I decided to add radio buttons to the side of the created rows. A user can tick the radio button next to the row which will set the global variable 'current_groupid' to the rows id value, eg if row 1 is selected from **figure 37** the variable will be set to 1.



```
radioButton.setOnClickListener(new RadioButton.OnClickListener() {
    public void onClick(View v) {
        int id = v.getId();
        for (RadioButton button : groupRadioButtonList) {
            if (id == button.getId()) {
                button.setChecked(true);
                String current_value = qid.getText().toString();
                current_groupid = Integer.parseInt(current_value);

            } else {
                button.setChecked(false);
            }
        }
    }
```

Figure 38: This is solution to creating dynamic radio buttons

The figure above which I modified from [http://stackoverflow.com/questions/22208468/radio-buttons-all-selectable-in-radio-group-after-adding-them-dynamically-android, Kristy Welsh] checks the current row radio button against the array list containing all the row radio buttons to see if it has been checked. If it has then it retrieves the id value of the group and parses it to get the integer value to set the 'current_groupid' value with. The reason for having to do this is because android's Radio button groups don’t allow for adding radio buttons dynamically as they have to be set beforehand, as the table can have any range of rows I cannot define how many rows I think there will be beforehand.

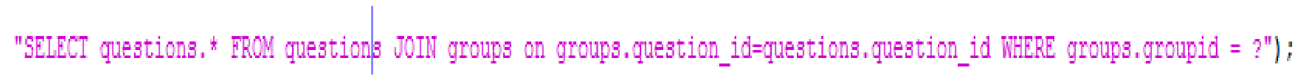
When the user clicks on the submit button it will check if 'current_groupid' is not equal to -1 and then pass the value to the edit question activity using putExtra method of intent.

Creating Edit Question:

The edit question activity is the final activity in the create class, here the user can edit the question selected from view groups. I decided that they should only be allowed to edit the assigned answer/question that the creator had assigned them in the add questions class. The reason for this is to avoid conflict where one user may be happy with their answer/ question and doesn’t want it changed, so another group member won't be able to access it unless they login as that user.

The first step in doing this was to define a similar xml file to the add questions activity as I wanted to keep the same structure and layout. The reason for separating the edit question and add question classes is that I wanted to keep both as simple as possible and separate the separate logic from each other.

The first step was retrieving the correct question selected from the previous activity, I defined a script called 'retrieve_question.php' which uses a single select statement using a join to combine the tables groups and questions, which selects all the fields from questions where the question id matches the question id from the correct groups row:

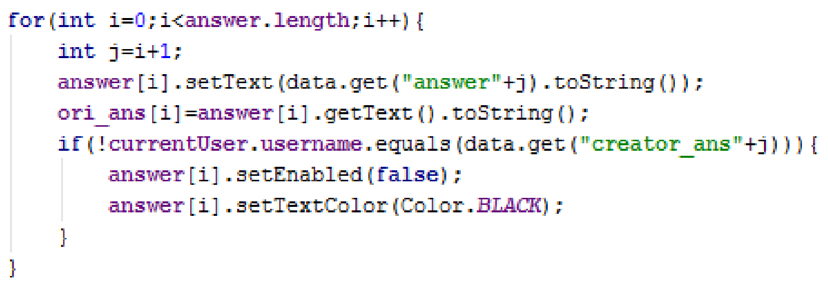


```
"SELECT questions.* FROM questions JOIN groups on groups.question_id=questions.question_id WHERE groups.groupid = ?");
```

Figure 39: shows select statement from script to retrieve correct question to edit

The reason for using a join statement is that it combines the results of both tables if there is a match so it allows me to use a single statement instead of multiple select statements.

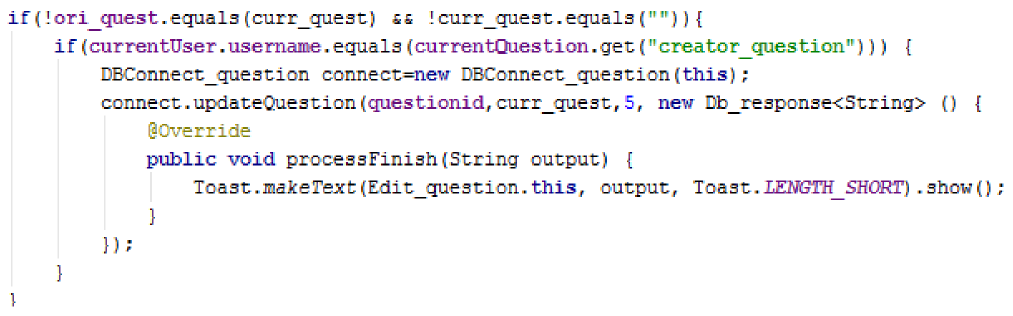
Once the data is retrieved from the database, the data is used to fill out the components(edit text boxes, text boxes for usernames and correct radio button to be shown). The usernames for each edit text box are compared against the current user and set to disabled if they don’t match so the user cannot edit other users answers.



```
for(int i=0;i<answer.length;i++){
    int j=i+1;
    answer[i].setText(data.get("answer"+j).toString());
    ori_ans[i]=answer[i].getText().toString();
    if(!currentUser.username.equals(data.get("creator_ans"+j))){
        answer[i].setEnabled(false);
        answer[i].setTextColor(Color.BLACK);
    }
}
```

Figure 40: loop going through the answer edit text boxes and disabling them if users don't match

When the user clicks submit a method called 'update' is called which checks to see if the user entered any new data before saving it to the database- for example if the user leaves the fields as they were originally then pressing submit wont upload the data to the database. The reason for this is that it would be unnecessary to keep updating the database row with the same data over and over again.



```
if(!ori_quest.equals(curr_quest) && !curr_quest.equals("")){
    if(currentUser.username.equals(currentQuestion.get("creator_question"))) {
        DBConnect_question connect=new DBConnect_question(this);
        connect.updateQuestion(questionid,curr_quest,5, new Db_response<String> () {
            @Override
            public void processFinish(String output) {
                Toast.makeText(Edit_question.this, output, Toast.LENGTH_SHORT).show();
            }
        });
    }
}
```

Figure 41: example piece of code which checks to see if question has been edited before uploading

The figure above shows that the 'ori_quest' variable which stores the initial value of the question which was retrieved from the database is compared to the current value to see if the are still equal, if it has changed and is not empty then the question will be updated in the database. The code for checking the answers is similar to the above code but is in a loop to go through the answer array.

If the question is fully complete and the creator presses submit it will bring up a pop up dialog box asking the user whether they wish to complete the question, if they press yes then the complete field for the corresponding question will be set to 1 and the question can no longer be edited. However if the user selects no then a message will tell the user that they can still edit the question. When a question becomes complete then the question will be removed from the view groups table.

The Play section:

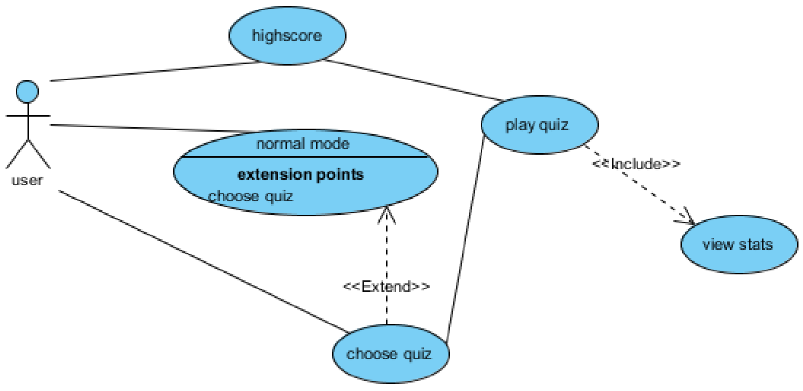


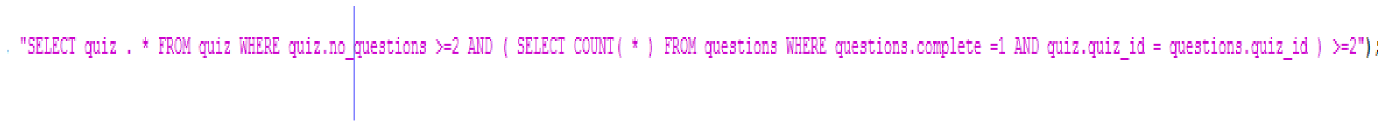
Figure 42: use case for play section highlighting interactions within section

The use case diagram above shows that the user will have the option of choosing to play a quiz through two possible interactions, they can either play the highscore mode or normal mode. The highscore mode is intended to pick random questions from all available complete questions and the user then attempts to get as many right in a row before getting one wrong. For the normal mode a user is then given a list of all quizzes which have complete questions attached to them and they can pick a quiz to play. The system will retrieve questions from the corresponding quiz in random order and the user can then play through all the available questions. While playing the user can view stats of each question once they have answered the question so they can see how tricky that question is and see possible patterns of answers.

As the use case shows I split this section into 2 parts-highscore and normal mode sections, both sections will then join up at the question activity where the relevant questions will be loaded for the user to play. I decided to have a highscore mode as I felt that it would offer another fun way of learning for the user as they can play other questions from other users to see if they can beat their friends. It will help keep users replaying the app when they don't need to use if for studying and help build the users knowledge.

Creating the normal mode:

The normal mode is the main part of this section as this is where the user can play their own quizzes and others. The first step was to create a 'TableLayout' component - which serves as the component in which all of the returned quizzes are stored. I created a script which is called on the activities 'onStart' method to fill the table.



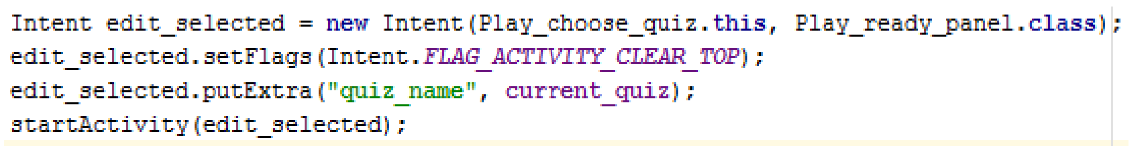
```
"SELECT quiz . * FROM quiz WHERE quiz.no_questions >=2 AND ( SELECT COUNT( * ) FROM questions WHERE questions.complete =1 AND quiz.quiz_id = questions.quiz_id ) >=2");
```

Figure 43: The sql statement used to retrieve quizzes

I choose to only return quizzes which had at least two completed questions attached, this was chosen as I felt that it would be frustrating for the user to choose a quiz which has a couple of questions but none which are complete and then therefore ending as soon as it is loaded. I believe that a quiz should contain more than one question as otherwise, but on the other hand I didn't want to force users to create a large amount of questions before being available to play as not all quizzes may need many questions. For instance a user may struggle on remembering only a couple of questions so they may only want to play those specific questions to help them study.

The quizzes are stored in the table which is created similarly to the table in the 'view_groups' activity-contains a series of radio buttons for each row which is created in the same way. The table rows include the quiz creators name and the quiz name as I felt these would be most important to the user-they may not remember all quiz id's that they are after, so including the quiz name and creator will help them navigate through the table in a more efficient way.

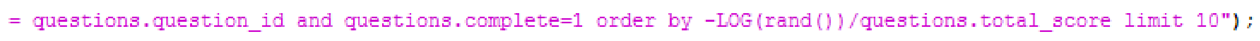
Once a user has selected a row and presses submit, the question name variable for that row is passed onto the 'play_ready_panel' activity to indicate the quiz to load. I chose to pass the quiz name as the quiz name is only used in that one activity so creating a shared preference would be unnecessary.



```
Intent edit_selected = new Intent(Play_choose_quiz.this, Play_ready_panel.class);
edit_selected.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
edit_selected.putExtra("quiz_name", current_quiz);
startActivity(edit_selected);
```

Figure 44: piece of code showing quiz name being passed to the next panel

The next activity is the 'play_ready_panel' which is access from the 'play_choose_quiz' panel above and from the button for high score mode in the 'choose_mode' activity. This panel contains two buttons 'play' and 'cancel', the play button will check the mode selected-if highscore mode or to retrieve a specific quiz. A script will be called from an AsyncTask class which will either select random questions or up to 10 questions from the selected quiz. The data returned is then stored in an array list of type 'Question_details' and passed through the intent to the question panel.



```
= questions.question_id and questions.complete=1 order by -LOG(rand())/questions.total_score limit 10");
```

Figure 45: part of the query used if a specific quiz is required

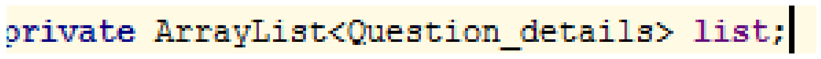
The query above uses -LOG() function to get a weighted random, so new questions with 0 total score are more likely to appear first followed by questions which are more highly rated. This way then allows new questions to be played by the user first and then to play more often than not the more helpful questions rated as good by the user and others.

The cancel button is used to cancel the selected mode/quiz and take the user back to the previous panel. I decided to create this panel to give the user the option of backing out as they may have accidently selected the wrong mode or question, and instead of having to play the quiz they can simply cancel.

For the highscore option when pressing on the button for high score mode in the 'choose_mode' activity it will simply tell the ready panel that the user wants to play the highscore mode.

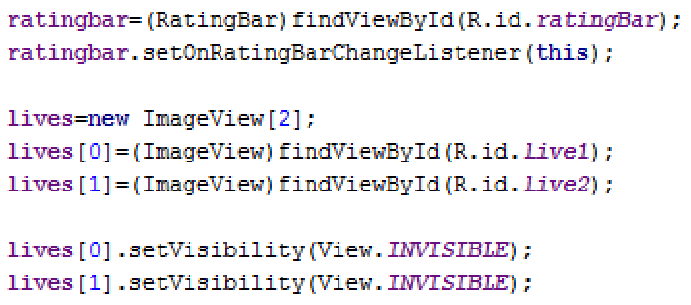
Question activity:

The main activity in the play section is 'question_template' activity, here the user plays the loaded questions and can rate each question and view stats. It is a single panel using an arraylist of 'Question_details' objects which is incremented when the user presses to go to the next question.



```
private ArrayList<Question_details> list;
```

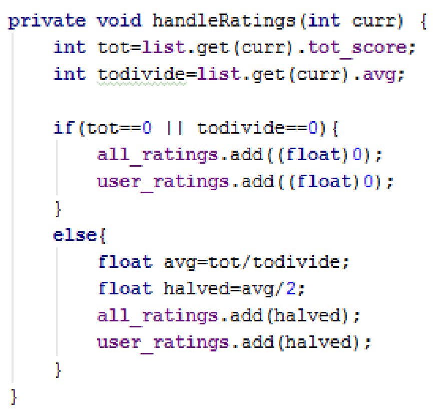
Figure 46: The global variable which stores each question retrieved



```
ratingbar=(RatingBar)findViewById(R.id.ratingBar);
ratingbar.setOnRatingBarChangeListener(this);

lives=new ImageView[2];
lives[0]=(ImageView)findViewById(R.id.live1);
lives[1]=(ImageView)findViewById(R.id.live2);

lives[0].setVisibility(View.INVISIBLE);
lives[1].setVisibility(View.INVISIBLE);
```

Figure 47: initialization of lives images and rating bar

For the high score mode the user has 2 lives indicated in the top right hand corner by two images in the form of hearts. They are set invisible initially along with the rating bar. They are set visible in the 'onCreate' method if in high score mode, the rating bar is set visible after the user selects an answer and locks it in.



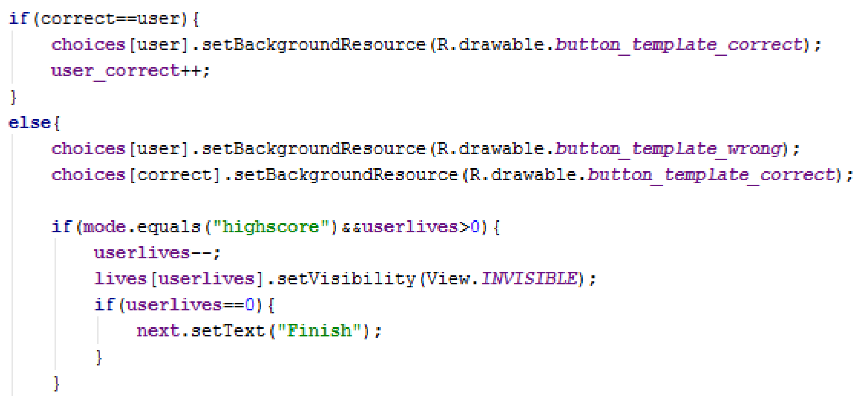
```java
private void handleRatings(int curr) {
    int tot=list.get(curr).tot_score;
    int todivide=list.get(curr).avg;

    if(tot==0 || todivide==0){
        all_ratings.add((float)0);
        user_ratings.add((float)0);
    }
    else{
        float avg=tot/todivide;
        float halved=avg/2;
        all_ratings.add(halved);
        user_ratings.add(halved);
    }
}
```

Figure 48: This method is used to load the array list with ratings of each question

The 'handleRatings' method is called in a loop which passes the current position 'curr' on each round of the loop, this checks the current question rating retrieved from the database and then stores it in an array list which is used to display that rating on the rating bar when the user is on it.

A global variable 'pos' is used to keep track of the current question to be shown on screen and is incremented when the user presses next. The next button listener checks to see if the 'pos' value is still less than the 'list' size to avoid any out of bounds exceptions.

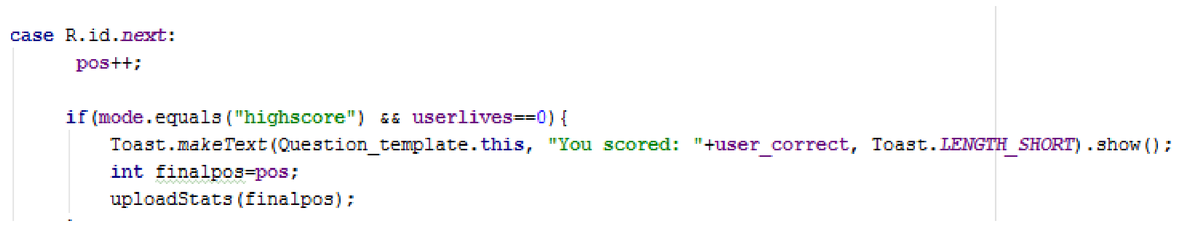
When the user clicks on an answer the button will change its graphic to indicate that it has been pressed and stays locked until the user either chooses another answer in which case it becomes unlocked and the new answer locked, or if the user submits in which case the system will see whether the selected question is right and highlight it green or whether it is wrong and highlight it red and then set the correct answer to green.



```java
if(correct==user){
    choices[user].setBackgroundResource(R.drawable.button_template_correct);
    user_correct++;
}
else{
    choices[user].setBackgroundResource(R.drawable.button_template_wrong);
    choices[correct].setBackgroundResource(R.drawable.button_template_correct);

    if(mode.equals("highscore")&&userlives>0){
        userlives--;
        lives[userlives].setVisibility(View.INVISIBLE);
        if(userlives==0){
            next.setText("Finish");
        }
    }
```

Figure 49: part of handleAnswer method to set correct answer background

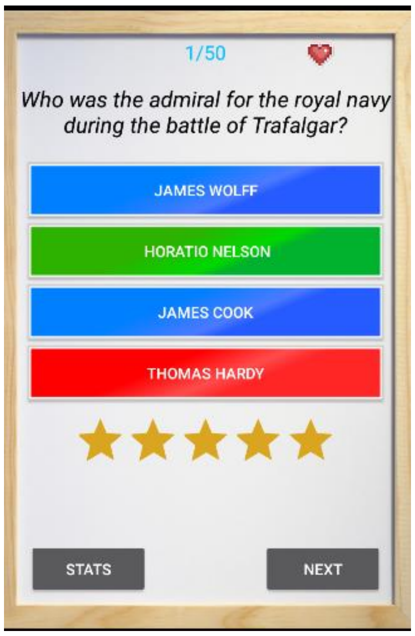
The piece of code above also handles loss of live when the user is in high score mode, a variable 'userlives' which is global and initially set to 2, is decremented when the user gets a wrong answer. The lives 'imageview' array has the outer most life become invisible to let the user know that they have one less life.



```
case R.id.next:
     pos++;

    if(mode.equals("highscore") && userlives==0){
        Toast.makeText(Question_template.this, "You scored: "+user_correct, Toast.LENGTH_SHORT).show();
        int finalpos=pos;
        uploadStats(finalpos);
```

Figure 50: partial part of next button listener

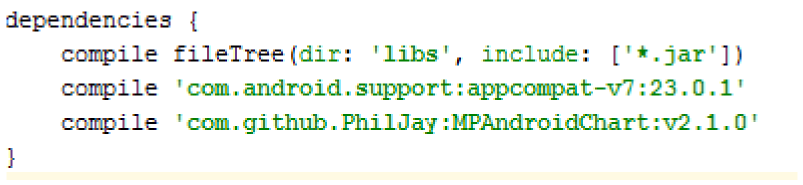
The next button listener handles whether the user has no lives remaining or if the will be an out of bounds exception as the last question is reached, in either case the quiz will end and the method 'uploadStats' is called to upload the users stats and ratings. The user is then taken back to the choose mode activity and told their score as the stats are uploaded. The ratings and question stats are uploaded in the background even after the question template activity is left. The reason for this is that these aren't important to the user so they shouldn't have to wait for a progress bar to finish before exiting the quiz.



Who was the admiral for the royal navy during the battle of Trafalgar?

Figure 51: shows an in progress high score mode with wrong answer selected

The final part is the stats button shown in the figure above, here the four answers are placed into a pie chart indicating how many times each answer was selected out of the total number. The pie chart is used from the API MPAndroidChart accessed from github [https://github.com/PhilJay/MPAndroidChart, PhilJay]. To import this I had to add it to the list of dependencies in the gradle build:



```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile 'com.android.support:appcompat-v7:23.0.1'
    compile 'com.github.PhilJay:MPAndroidChart:v2.1.0'
}
```

Figure 52: inclusion of MPAndroidChart API

The answer stats are stored in a multidimensional array list and each one is updated for the user's choice:

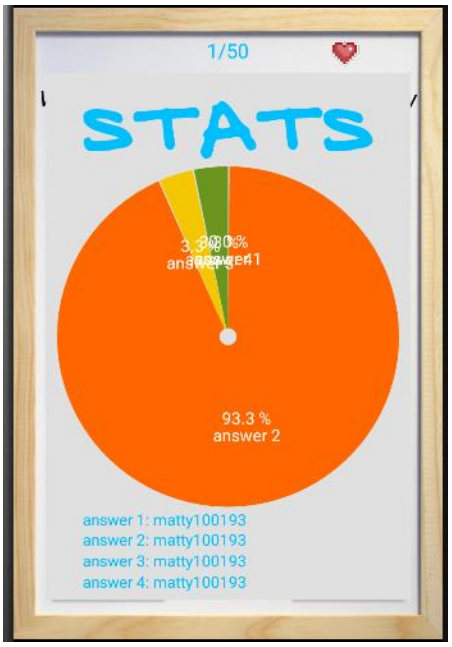


Figure 53: stats page which is a custom popup window and displays the creator of each answer

4. Testing and Evaluation of phase 1:

The aim of this step is to test the usability and functionality of my prototype created in phase 1. The Testing will be separated into 3 separate phases:

- Internal Testing of core functionality-this step is where I will focus on fixing as many errors as possible such as crashes and design problems.
- User testing and feedback-this step comes after the internal testing phase where I send out the prototype to users who will trial the app and give feedback on how to improve it and errors that may have been encountered
- Device testing-here I will use a variety of devices with different screen sizes and resolutions to make sure that the app appears correctly on as many devices as possible.

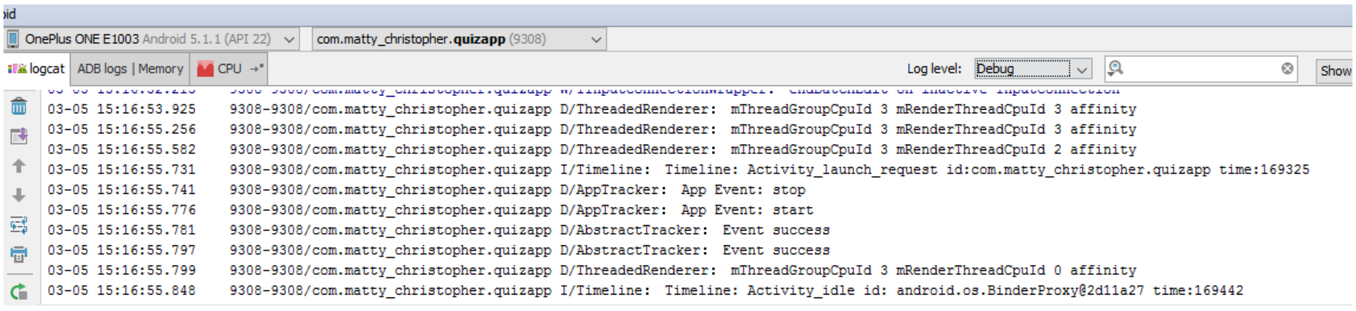
Once these 3 steps are completed I will use the results to see what features I should improve and features that I should add for phase 2 of development.

The main aims of testing and evaluation are:

1. See whether the prototype I created works correctly by being able to complete the expected tasks set out in the requirements phase
2. Make sure that the app is usable on a wide range of devices
3. Find out if the app works in a variety of conditions and does not cause unexpected errors
4. Get impartial feedback on the good and bad
5. Check that the server can cope with multiple users at the same time
6. Use feedback and results of testing to improve prototype ready for final release-what features cause problems for users and should be focused on in phase 2 for fixing them as well as features to be added suggested from feedback

4.1 Internal Testing:

This step is perhaps the most important in terms of spotting and fixing errors as errors relating to the database may be harder for external users to spot. A lot of minor errors were caught in the implementation phase when testing the completed sections. For internal testing I used my phone along with the emulator provided by android studio, using the android logcat I was able to switch between viewing debug logs, errors relating to the device and errors relating to the system while the app was running.



OnePlus ONE E1003 Android 5.1.1 (API 22)
com.matty_christopher.quizapp (9308)
logcat
ADB logs | Memory
CPU
Log level: Debug
Show

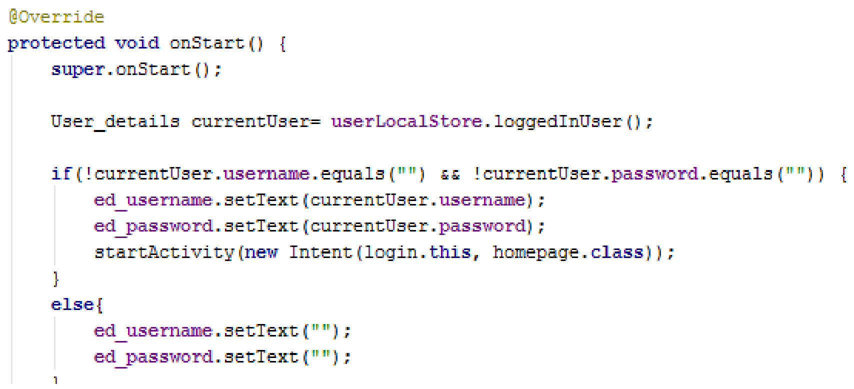
```
03-05 15:16:53.925    9308-9308/com.matty_christopher.quizapp D/ThreadedRenderer:  mThreadGroupCpuId 3 mRenderThreadCpuId 3 affinity
03-05 15:16:55.256    9308-9308/com.matty_christopher.quizapp D/ThreadedRenderer:  mThreadGroupCpuId 3 mRenderThreadCpuId 3 affinity
03-05 15:16:55.582    9308-9308/com.matty_christopher.quizapp D/ThreadedRenderer:  mThreadGroupCpuId 3 mRenderThreadCpuId 2 affinity
03-05 15:16:55.731    9308-9308/com.matty_christopher.quizapp I/Timeline:  Timeline: Activity_launch_request id:com.matty_christopher.quizapp time:169325
03-05 15:16:55.741    9308-9308/com.matty_christopher.quizapp D/AppTracker:  App Event: stop
03-05 15:16:55.776    9308-9308/com.matty_christopher.quizapp D/AppTracker:  App Event: start
03-05 15:16:55.781    9308-9308/com.matty_christopher.quizapp D/AbstractTracker:  Event success
03-05 15:16:55.797    9308-9308/com.matty_christopher.quizapp D/AbstractTracker:  Event success
03-05 15:16:55.799    9308-9308/com.matty_christopher.quizapp D/ThreadedRenderer:  mThreadGroupCpuId 3 mRenderThreadCpuId 0 affinity
03-05 15:16:55.848    9308-9308/com.matty_christopher.quizapp I/Timeline:  Timeline: Activity_idle id: android.os.BinderProxy@2d11a27 time:169442
```

Figure 54: shows logcat while app is being run

The logcat helped me pinpoint crashes to see which panel and exactly where in the code they occurred. Using this I found crashes in the create question and edit question panels, these crashes were all related to the process dialog leaking in the AsyncTask being called when connecting to the database. The reason for these crashes were due to the process dialogs being active while the activity switched to another causing the dialog to remain open and cause the crash. To solve this I made sure to utilize callbacks which returned to the current activity signifying when the 'doInBackground' methods were finished, using the callbacks I kept he user on the activity on the activity until the dialog finishes which will then finish the activity. Another way of solving the crashes was to remove the dialogs from certain AsyncTask's when it wasn't required for the user to wait for the result such as in the play section when uploading question stats-as these can take a couple of seconds it makes sense to not prevent the user using the app.

Another problem that I found was when creating a question as a group-when the question was open on 2 separate devices and the creator completes the question it would close the question for the creator but the other members panel would remain unchanged so they could still edit the question. This was mentioned in the implementation phase- I solved this by modifying the script for updating the question to only update when the complete field was not equal to 1. As I was not using push notifications this seems to be the best way of handling this without keeping a database connection open to keep checking - which would cause the device battery to run out quickly.

One improvement that I made while testing was to keep track of the last user logged in as I found when testing having to keep retyping the username and password to be frustrating.



```
@Override
protected void onStart() {
    super.onStart();

    User_details currentUser= userLocalStore.loggedInUser();

    if(!currentUser.username.equals("") && !currentUser.password.equals("")) {
        ed_username.setText(currentUser.username);
        ed_password.setText(currentUser.password);
        startActivity(new Intent(login.this, homepage.class));
    }
    else{
        ed_username.setText("");
        ed_password.setText("");
```

Figure *55*: implementing improvement to solve remembering last user login

The solution was very simple which used the shared preference for the user details checking whether the username and password stored were empty or not, if not then it would auto log the user in and take them to the homepage. I also added a button on the homepage to logout which would wipe the shared preference data for a new user to login.

4.2 Heuristic evaluation:

A Heuristic evaluation is an evaluation technique used to identify usability problems in the User interface of a system. The evaluator/s judge the interface using a set of predefined usability principles an assign a severity rating from 0-4 with 4 being major problems. There are 10 heuristics which need to be measured:

1. Visibility of system status
2. Match between system and the real world
3. User control and freedom
4. Consistency and standards
5. Error prevention
6. Recognition rather than recall
7. Flexibility and efficiency of use
8. Aesthetic and minimalist design
9. Help users recognize, diagnose, and recover from errors
10. Help and documentation

4.3 Device Testing:

The aim of device testing is to see how my current design looks and performs on a variety of different devices. It is important that it performs correctly on lots of different screen sizes and android operating system versions as the main demographic of this app are likely to use many different devices, from smaller phones to tablets. As I only have a single android mobile it is hard to get hold of multiple devices, therefore I decided to use "*Testdroid.com*" and "*Testobject.com*". These sites are mobile testing sites which have emulators of different devices in which you can test apps. I chose these two sites as they both offer free testing on a couple of devices and are simple to use.

Using the sites I was able to test the app on 6 other devices: *Acer Iconia Tab 8*, *Asus Fonepad ME371MG*, *Galaxy Nexus I9520*, *Nvida Shield Tablet*, *LG Nexus 4 E960* and the *Nexus 10*.

On the Acer Tab 8 I found that most of the app displayed correctly except for on the question template and add question activities, I noticed that the components clipped the border of the background and the lives images were very small.

On the Asus Fonepad there was clipping seen above as well as the leaderboard and other tables where the row heights were very tall taking up a lot of the screen compared to on smaller devices.

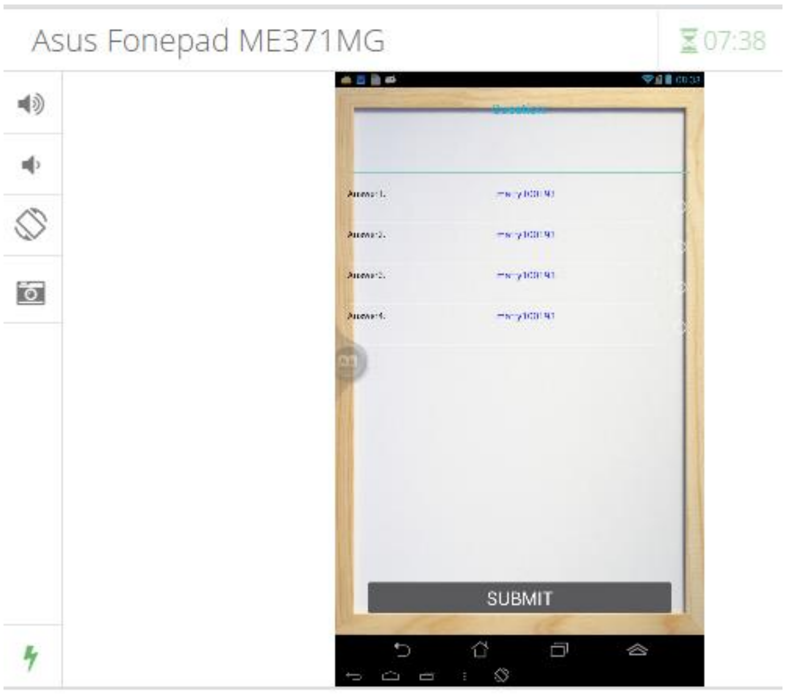


Figure 56: image shows clipping on Fonepad and wasted space

The main cause of problems testing the devices was on the Galaxy Nexus I9520 where the background images didn’t load up resulting in a black background. The result was that it was impossible to use the app as the text was blacked out and the resulting loading error caused the app to become unresponsive. The error occurred because the bitmap was too large to load resulting in the app to repeatedly recall the background.

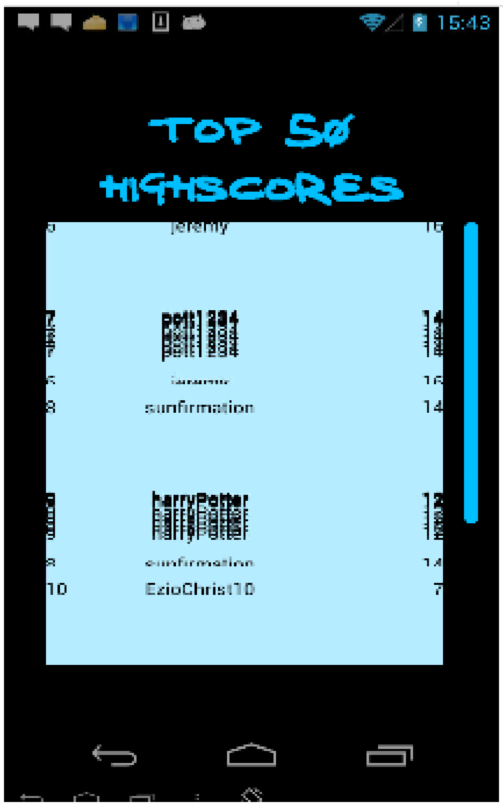
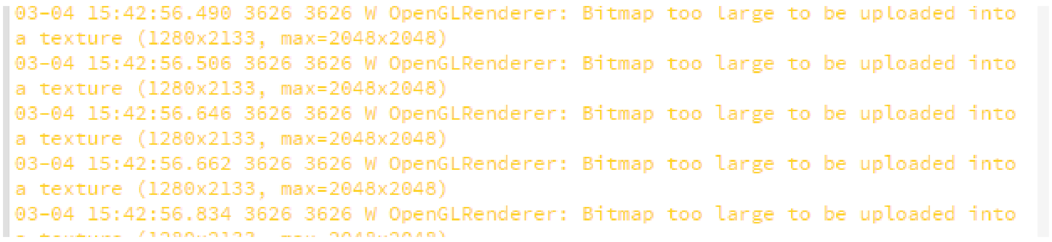


Figure 57: bitmap too large resulting in app becoming unresponsive and unexpected behaviour



```
03-04 15:42:56.490 3626 3626 W OpenGLRenderer: Bitmap too large to be uploaded into
a texture (1280x2133, max=2048x2048)
03-04 15:42:56.506 3626 3626 W OpenGLRenderer: Bitmap too large to be uploaded into
a texture (1280x2133, max=2048x2048)
03-04 15:42:56.646 3626 3626 W OpenGLRenderer: Bitmap too large to be uploaded into
a texture (1280x2133, max=2048x2048)
03-04 15:42:56.662 3626 3626 W OpenGLRenderer: Bitmap too large to be uploaded into
a texture (1280x2133, max=2048x2048)
03-04 15:42:56.834 3626 3626 W OpenGLRenderer: Bitmap too large to be uploaded into
```

Figure 58: error displayed in emulator log where bitmap too large

The nexus has a 4.65 inch screen size and 1280 by 720 resolution, and is the only device that this has occurred on-it occurred on the nexus I9520 on both sites. All other devices tested the only problems occurring were clipping and the table row heights being too large.

4.4 User testing:

The aim of the user testing phase is to get feedback from target audience users to identify the strengths and weaknesses of the prototype, with feedback acquired I will be able to improve the app before deployment.

To be able to get impartial feedback of the prototype I decided to send the app to 5 users who will first test the app going through each task and then evaluating the app where they will give their impressions on the good and bad. Users will be given the app to trial for a week in which they will be given instructions on how to install the app and the device requirements. I decided not to explain how to use the app to them as I want to find out how intuitive the current design is to see if instructions will be required in the final deployment. I also gave each user a questionnaire to fill in at the end of the week, the questionnaire's aims is to find out any problems in the app, where they were located and frequency of occurrence. The questionnaire also asked about whether the device displayed correctly and asks if any particular sections seemed to be displayed correctly to the user. It then asked the user about what current features could be improved, if they found it easy to use and what features could be added to improve their experience.

Results from the Questionnaire:

The feedback that I got indicated that none of the 5 users encountered any system problems such as crashes or incompatibilities. I got a range of interesting and helpful improvements and concerns:

Improvements:

1. Allow quick access to adding a question as a couple of users mentioned when testing together or in the provided questionnaires is that when adding multiple questions to one quiz or a select few found that it became repetitive having to retype the same data over and over again. I also got feedback telling me that people found that it was very hard remembering all the quizzes that they created especially the passwords, this led them to having to view the play now sections and search for the name of some of their quizzes and had to resort to writing down passwords on paper for future use. Therefore I propose that I will create a popup panel which lists all quizzes created by that user with the quiz name and password, the user will then be able to click on one of the quizzes which will then automatically put the data into the text fields so they don't have to type them in manually.
2. Users complained about having to search through the table to find a quiz that they created and wanted to try out. One user suggested having a search option in which you can enter a username to be shown quizzes created by that user. This is a good suggestion as my in house testing was focused on only a couple of quizzes being present so I never found the need to search through lots/ encountered this.
3. When testing the group functionality with another member we found that when one user is on the view group's panel and another user creates a group question with that user, the user has to switch activities to get the panel to update to show the new question added. We discussed that having a refresh button or push notifications would solve this.
4. One user was unsure on what the group functionality entailed as they said that it was unclear what would happen. They suggested to have an explanation/instructions on what creating a question in a group does/how to use it. I thought that this would be self-explanatory, however I didn't take into account the actual group creation process as they explained that they were in a group question but were then unable to add the question to play- they were not the group creator so they don't have access to completing a group question. Therefore to solve this I will create an information panel which can be accessed in the 'create home' panel where this will explain each section on what it does and how to use it.
5. The last improvement was to do with aesthetics in which users explained if any parts of the app which didn't display correctly etc. Users had a range of devices and those using tablets and smaller devices encountered the majority of components being displayed incorrectly. This ranged from certain components such as buttons being too wide stretching the screen to components being too small/large and to not displaying correctly (rating bar). For this I

will attempt to make my app as compatible as possible through reading the android document for supporting multiple screens which I hope will fix this. This also backs up what I found in my own testing through testing sites where I noticed some visual problems so I will be able to check using these sites after phase 2 to see if this has been resolved.

Extra features suggested:

1. The common suggested features were to do with login and account management, users asked to be able to login with their google pay account/ Facebook etc. and be able to change their email address and password.
2. Another feature suggested was having more game modes as they felt that the current modes provided was slightly lacking to keep their attention for longer periods as opposed to using other revision modes/ general use. One user would like to be able to play quizzes with friends/ group members from previous questions as they said it would improve replay ability and make learning with app more enjoyable.
3. Another user suggested having a profile page in which you can add a profile picture, view stats and have goals to follow.

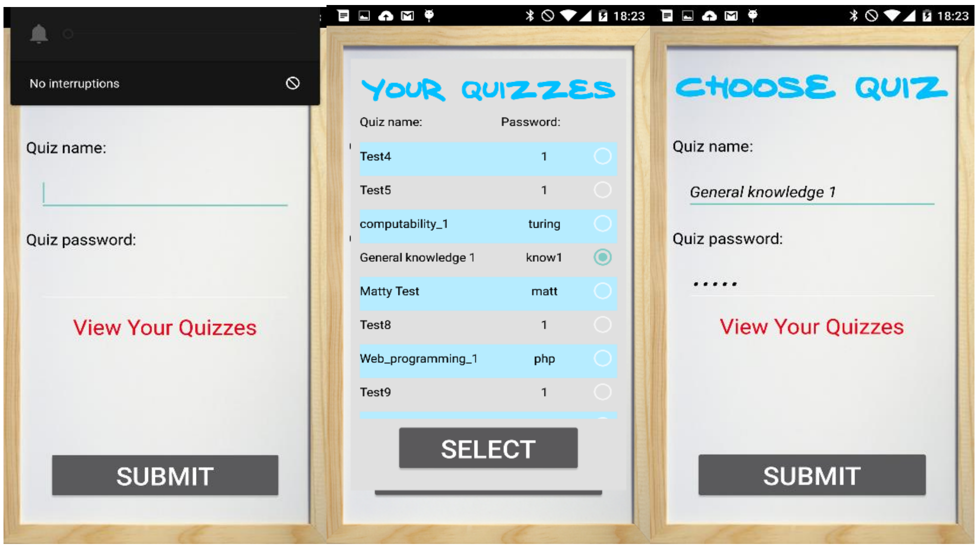
With the feedback received I will now use the data gathered to improve my design in phase 2. I will concentrate on improving existing features first before adding extra features, as I may not have enough time to do everything suggested.

The key features to be worked on in phase 2 will be:

1. **Having quiz management pop up panel to allow quick access**– as this was one of the more common suggestions which caused a lot of frustration for users.
2. **Search quiz attachment to play panel**- another common suggestion and cause of frustration, this will be beneficial for a large amount of users when the number of quizzes grows, therefore this is an important improvement that I will make sure is added.
3. **Information pop up panel in create section**- This should be easy to implement and will provide users on how to use this section when stuck which will hopefully make the app easier to use for people unfamiliar with the app.
4. **Account management allowing user to edit details** – I don't think I will have time to link login details to other accounts as researching this I found that I would need to change my app especially the database too much.
5. **Add profile and achievements/goals** – This shouldn't be too hard, I may not have enough time to be able to allow users to create their own goals so I will most likely add a generic set which I will ask the user who suggested this what achievements should be added.
6. **Head 2 Head mode** – This will be the most difficult feature to be added so I will aim to do this last so other features are completed before.
7. **Supporting multiple devices**- This will be done throughout phase 2

5. Implementation phase 2:

Having quiz management pop up panel to allow quick access:



Quiz name:	Password:
Test4	1
Test5	1
computability_1	turing
General knowledge 1	know1
Matty Test	matt
Test8	1
Web_programming_1	php
Test9	1

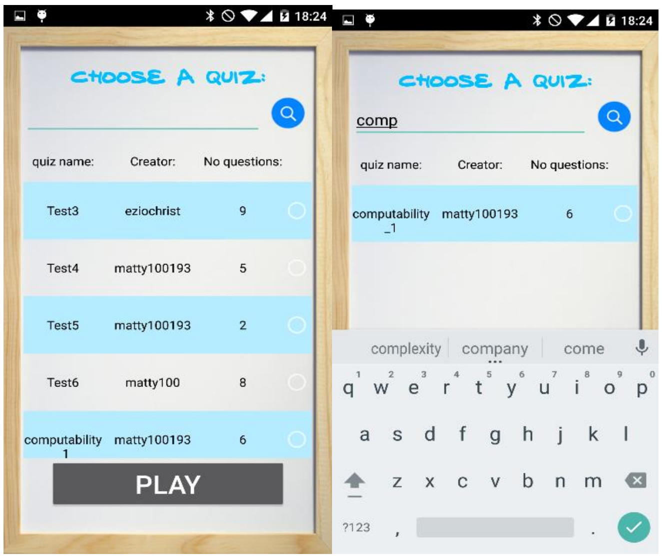
Quiz name: General knowledge 1

Quiz password: .....

Figure 59: screenshots of quiz management in action

For this feature I decided to have a popup window which has a list of user quizzes and passwords which can be selected. I decided to have a pop up window as it keeps the user on the same panel and can be easily accessed between the windows and create question start. Radio buttons are used to allow the user to select a row so that they don’t have to type in the data, when the user clicks select and has a row selected then the window will close and the data will be stored into the textboxes on the create question start as pictured above. I implemented this in the same way to previous activities which required a selectable table (view groups and choose quiz in the play section) and global variables are used to communicate between the two panels.

Search quiz attachment to play panel:



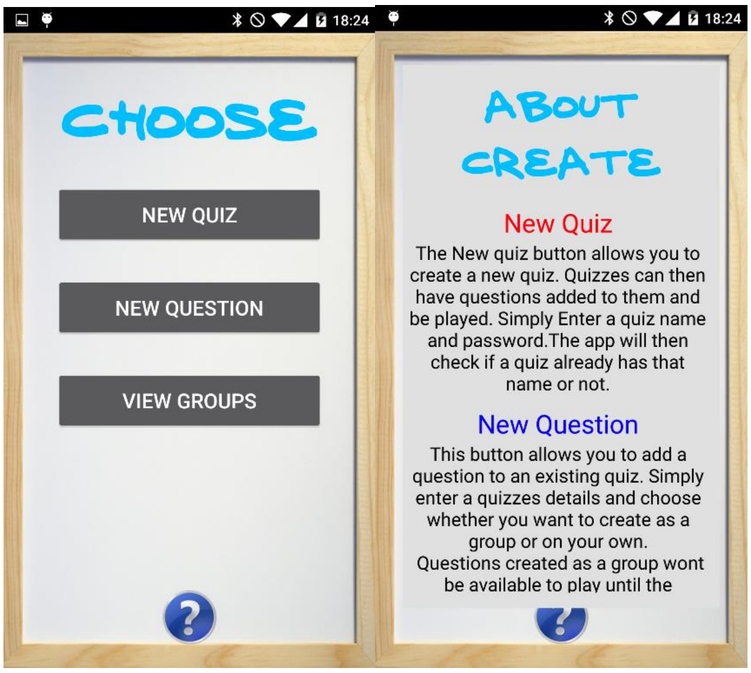
quiz name:	Creator:	No questions:
Test3	eziochrist	9
Test4	matty100193	5
Test5	matty100193	2
Test6	matty100	8
computability 1	matty100193	6

quiz name:	Creator:	No questions:
computability _1	matty100193	6

Figure 60: screenshots of search feature

For implementing the search functionality I used an edit text box and button for the search box and search button. When the onStart method is called (when user opens the panel etc.) a database call to a script returns data for the table, a copy of this data is made for use with the search functionality. When the user searches the table is filled with the arraylist containing the copied data (if the table already has data then that data is removed first), then any matches of either columns from the creator or quiz name are shown in the table which removes all rows without a match. I created a copy of the original data so that the user can do multiple searches without ever having to restart the screen to re-get data for the table. Then if the user searches with no data input then the original contents of the table will be shown. Also another reason for creating a copy is so that the user can then search another term even when the table only displays the last searched rows (such as above figure where only one row is displayed, the user can enter for example test and 5 rows will be displayed (currently))– this removes the need for the user to have to reset the table as searches are always compared against the copied arraylist.

Information pop up panel in create section:



New Quiz

The New quiz button allows you to create a new quiz. Quizzes can then have questions added to them and be played. Simply Enter a quiz name and password.The app will then check if a quiz already has that name or not.

New Question

This button allows you to add a question to an existing quiz. Simply enter a quizzes details and choose whether you want to create as a group or on your own. Questions created as a group wont be available to play until the

Figure 61: screenshots of information panel

For this feature another popup window is used which contains a scrollview and text explaining what each section does, and how to use them. At the moment I have included information that I feel will help the user but if users require more information that can be easily added to the required textbox.

Account management allowing user to edit details and goals/stats:

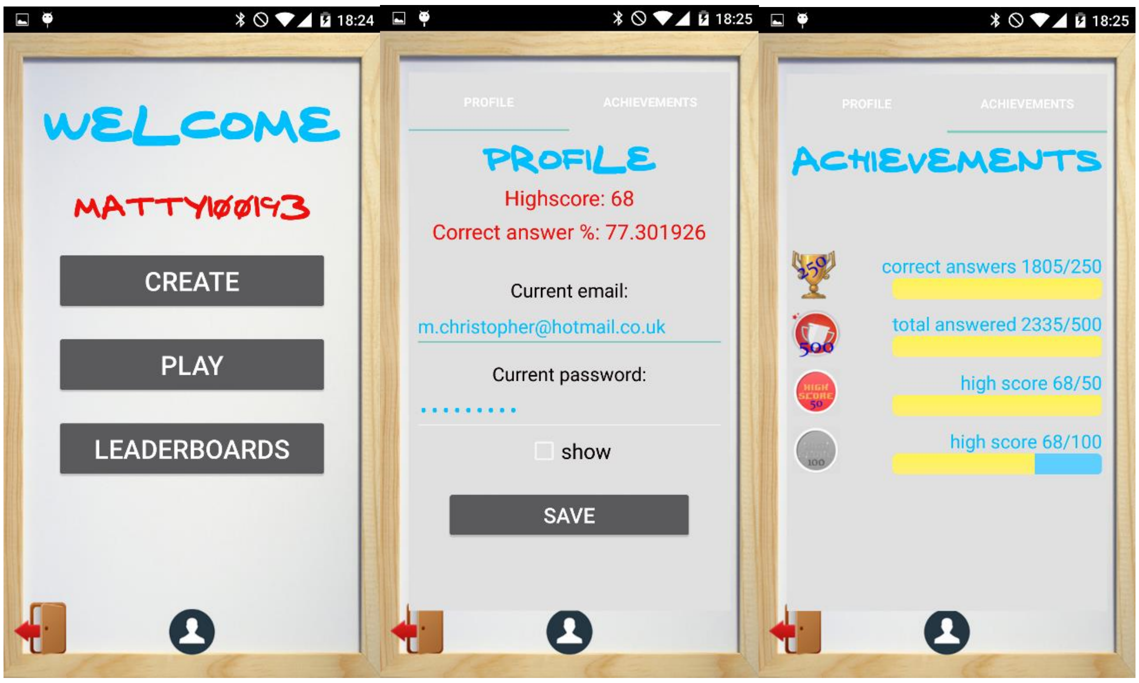
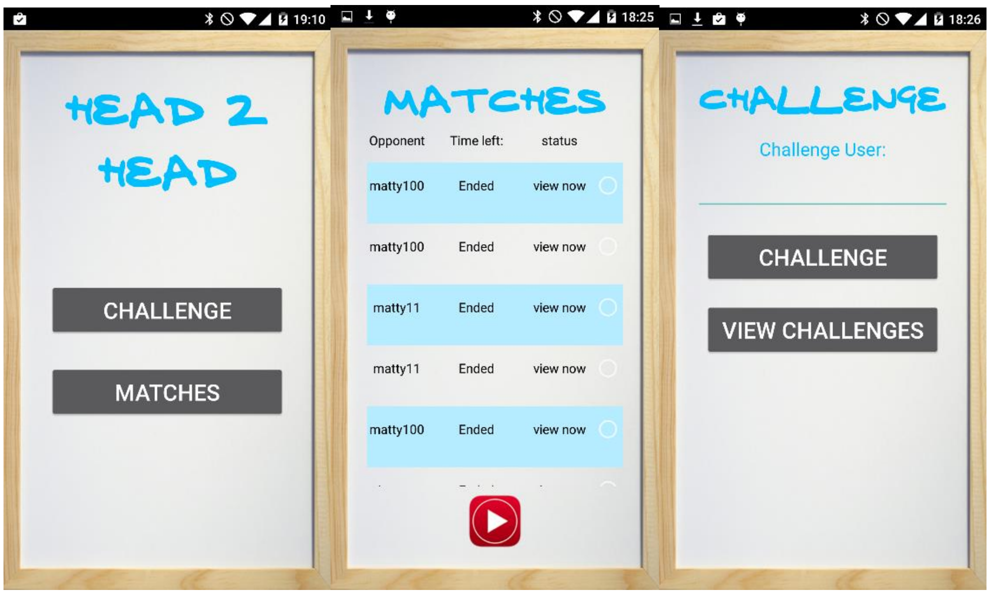


Figure 62: screenshots of profile

For these two features I decided that they can be grouped together into one section, I placed these into a profile popup window which has two tabs one for password management and stats, the other for achievements/goals. The two sections can be easily switched between via the tab buttons on the top of the window. For the profile tab the users highscore and percentage of correct answers along with the users email address and password. The password is hidden but can be switched between text and hidden via the show checkbox which has a listener checking when user presses it. The user can change their email address or password or both by simply clicking on the textboxes and inputting the new data then clicking save, if the user clicks save while not changing any data then a message will appear that no changes have been made, this is to prevent unnecessary database calls when not needed. For the achievements tab I have included 4 achievements so far which are greyed out when they are not achieved and the corresponding percentage bar is filled in to the correct percentage, when the user completes an achievement the image becomes coloured and the percentage bar fills up to fully gold.

Head 2 Head mode:



Opponent	Time left:	status
matty100	Ended	view now
matty100	Ended	view now
matty11	Ended	view now
matty11	Ended	view now
matty100	Ended	view now

Figure 63: screenshots of head 2 head mode

For the last feature I decided to make this mode a one on one turn based mode as I wasn't sure that I would have enough time to properly create and test a real time version, as based on my research into it found that it would require a lot more work and be much harder to test. For this mode users first must challenger another user through an autocomplete box which lists all registered users. Users can also view invites from another user where they can either accept or reject. Rejecting will destroy the match challenge in the database while accepting will create a match with 5 random questions and give the participants 24 hours to complete. Both users have the same questions and when complete their scores are submitted and the user can no longer play that match. The user once finished can view the status when the status is 'check' this will take the user to the score panel which will list the users score and opponents score and the current status such as waiting for opponent. When the 24 hours runs out or both users has finished, the stats panel will list the result such as you won, drew or lost. For invites and matches I implemented a countDownActivity timer for each row which checks the current time against the start time from the database, when the timer runs out the invites are automatically removed to indicate they have now expired, matches have their status changed to view now when the timer runs out. I decided to use this as my approach doesn't use push notifications so this helps make the mode more dynamic and less reliant on multiple database calls.

6. Testing and evaluating phase 2:

For phase 2 I was able to complete all of the proposed features/improvements set out in the testing phase 1 section. I was able to complete these features in only a couple of weeks with the majority of time spent on the Head 2 Head mode and fixing bugs. To test the new features most of the testing will be done by using the app in everyday use to simulate a potential user and see if any problems are found.

6.1 Performance:

The aim of performance testing is to check whether the app is usable over a longer period of time and that the app doesn't use too much memory which could affect the devices performance. To test this I spent a week monitoring the app while using it with the android studio monitor.

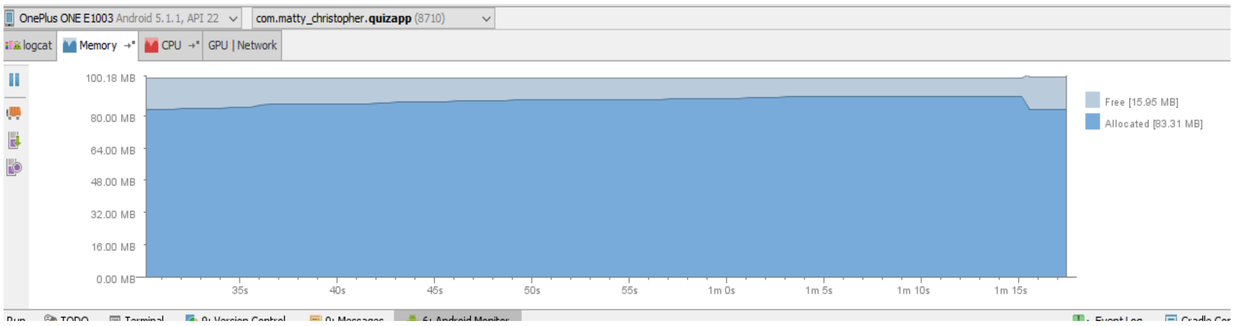
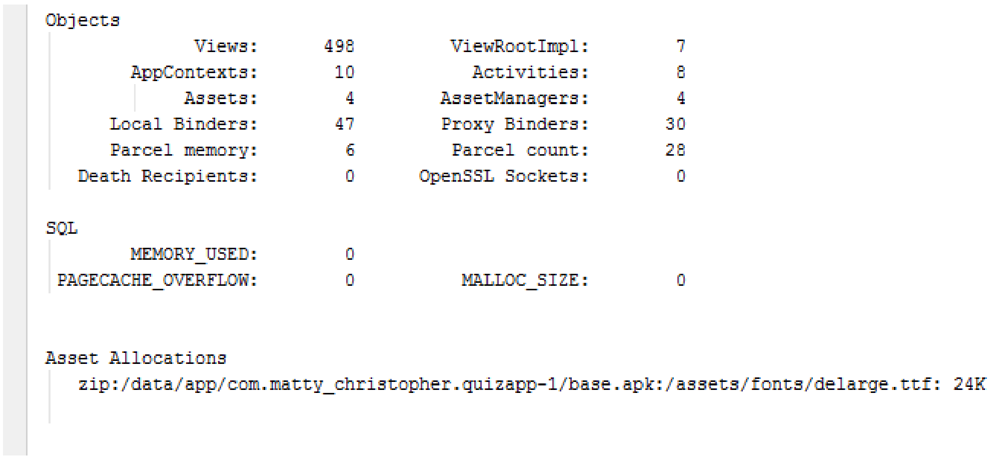


Figure 64: showing how much memory app uses at any given time

Using this I soon noticed that every time I reopened a panel multiple times such as going to the home screen and back to leaderboard and vice versa, that the memory usage grew after each panel being opened this after a couple minutes of usage the memory usage could spike up to well over 250 MB with only a few sections of the app opened. In previous testing I never noticed this as my device has 2GB of ram so it is still a small fraction of it, however most phones tend to have 512-1024 MB of ram. By looking at the status section as shown below I was able to deduce that multiple instances of the same activities were being created each time that activity is opened.



```
Objects
               Views:      498         ViewRootImpl:        7
         AppContexts:       10           Activities:        8
              Assets:        4        AssetManagers:        4
       Local Binders:       47        Proxy Binders:       30
       Parcel memory:        6         Parcel count:       28
    Death Recipients:        0      OpenSSL Sockets:        0

SQL
         MEMORY_USED:        0
  PAGECACHE_OVERFLOW:        0          MALLOC_SIZE:        0


Asset Allocations
    zip:/data/app/com.matty_christopher.quizapp-1/base.apk:/assets/fonts/delarge.ttf: 24K
```

Figure 65: shows objects etc currently in memory

To fix this I made sure that every activities back panel opens the previous panel in the sequence such as leaderboard goes back to home, with the flag set to 'REORDER_TO_FRONT'. This flag is also set for each startActivity() call- this flag checks if there is a currently open instance of the activity in memory and brings that instance to screen instead of creating a new instance (unless none in memory). With this I have been able to keep the memory usage to 100-200 MB of ram depending on whether the user uses all parts of app or not. Another problem that I found when testing performance was that the custom font I use was leaking with multiple instances of the same font being called, to do this I cached the font so that only one version is in memory at a time which I found online-referenced in code. I also made my images smaller resolution so that they take up less memory as larger images can take up lots of memory very quickly, I followed the android documentation on supporting multiple devices for the correct resolutions for background images and created a different sized one for each screen size folder.

I then tested my app on Testdroid.com again to see different screen sizes, I noticed that the app ran better generally compared to before and the nexus device which kept crashing before worked smoothly, I noticed some clipping still but generally the app seems to have less clipping on bigger devices.

6.2 User testing:

The aim of the second user testing phase is to get feedback from users who have previously tested the prototype 1. This is to see whether phase 2 of development has fixed general issues found in the first prototype and that phase 2 is a better experience to the user based on their previous feedback. After a couple of days using the app I sent the users a feedback form which asks the user what they found was better or worse than the initial prototype, what they would like to see in potential future iterations and whether they had any trouble with the new version.

The results of the user testing were that the app was much improved and found the new features helpful to improving their experience. Users didn’t encounter any crashes or potential bugs/errors with their use over the period that they tested it for. Users found that for future updates that they would like to see different topics being added in which users can create a quiz around a topic such as Computer Science or History, so that users can find quizzes by other users easier. They feel that having different topics will help separate unrelated quizzes from each other which will help users who want to use it to revise be able to find relevant quizzes quicker for the area of study that they are undertaking.

7. Conclusion:

The completion of phase 2 development and testing marks the end of the development of my project due to time constraints and other commitments. Upon evaluating my final completed prototype I would say that my initial expectations have been greatly exceeded. At the start of this project I had never created an android app which I now feel confident in creating future apps in android. While working on the project I learnt a great deal from the project which I hope to use in future work. One of the main aspects learnt was to fully plan out the system design to include easy expansion for future prototypes after the initial prototype. I found that after creating the initial prototype that some features for phase 2 required modifying existing database tables which caused me to have to modify the database calls in the code to match the new data. In the future I would try to work out potential future improvements in the initial design phase and how to incorporate them such as creating new database tables and linking them to existing tables which would reduce the need to modify existing code. Another aspect learnt was to always keep an eye on the memory usage for each new section added so that the app is as efficient as possible. I found that when testing this near the end of development I had to go back to previous sections from the start of the project and make them as efficient as possible then having to retest them.

Overall I am extremely happy with the results of this project as I was able to meet all my initial expectations and more while using knowledge from the first two years to complete this such as:

1. Algorithms and Data Structures-applying knowledge such as generics to allow reuse of methods for multiple object types and ways of keeping algorithms as efficient and elegant as possible to improve the performance of the app.
2. Database Systems- how to create and link a database to java as well as efficient ways of creating SQL statements through joins and nested selects to reduce the need of multiple calls, and also ways of planning and maintaining a database.
3. Graphical User Interfaces- ways of testing an interface such as Heuristic evaluation to evaluate ways to improve the design for the user and design principles for user interfaces such as simplicity and feedback.

7.1 Proposed further work:

If I was given more time or decided to continue working on this in the future I would add a range of features/improvements:

1. Push notifications- Adding push notifications to existing features such as group quizzes and head 2 head modes would help keep the user updated about the status of other users. Adding push notifications would help allow the user to use different parts of the app without having to check the status manually of a group question for example. They could be playing a quiz mode when a notification appears telling the user that a group question they are part of has been modified/completed.
2. Real time h2h mode and group mode-Making both modes real time/ adding the feature as an alternative mode would help reduce the need for users to refresh etc. and help keep track of users when editing a group question. Users for example would be able to see when editing a question whether another user is currently online also modifying the question (they may be able to see the user’s sections having data added to them) while they work on their section.
3. Different topics- this was suggested in testing phase 2 which I would have to modify the existing structure of the app in terms of layout and database. For this feature I would create a set of general topics such as History, Art etc. then allow users to create their own sub topics such as History topic on the Titanic/ww2 etc. Users when creating a quiz will decide what topic best describes the quiz, then when choosing a quiz to play a user can either search the quizzes as current or browse the topics to narrow down the number of relevant quizzes.

Better support for different devices- This isn’t a feature but a general improvement to the app to better support different screen types, android os versions and add extra input options such as speech based input. While working on the project I tried to best support as many devices as possible but due to time constraints and lack of different devices I had to focus on a small number of devices which are similar to my primary device. I feel better supporting multiple devices would help the amount of potential users and as the app is able to be used for education-primarily students, they will have a wide range of devices not necessarily larger premium phones.

Bibliography

1. Gualtieri,L. (2009, Sep 17). *Can Quizzes Challenge and Inspire?*. Retrieved from: http://elearnmag.acm.org/blog/?p=71

2. Aljezawi, M. and Albashtawy, M. (2015). *Quiz game teaching format versus didactic lectures*. British Journal of Nursing, 24(2), pp.86-92.Retrieved from: https://www.researchgate.net/publication/271537197_Quiz_game_teaching_format_versus_didactic_lectures

3. Gordillo, A., Barra, E. and Quemada, J. (2015). *Enhancing Web-Based Learning Resources With Existing and Custom Quizzes Through an Authoring Tool*. IEEE Revista Iberoamericana de Tecnologias del Aprendizaje, 10(4), pp.215-222. retreived from: http://ieeexplore.ieee.org.ezproxy.library.qmul.ac.uk/stamp/stamp.jsp?tp=&arnumber=7044188

4. Chen, D. (2005). *Enhancing Student Learning through Classroom Discussions in Circuits Courses*. retrieved from: http://ieeexplore.ieee.org.ezproxy.library.qmul.ac.uk/stamp/stamp.jsp?tp=&arnumber=1612201

5. (2015, Aug).*Smartphone OS Market Share, 2015 Q2*. Retrieved from: http://www.idc.com/prodserv/smartphone-os-market-share.jsp

6. Frumusanu, A.(2014, July 1). A *Closer Look at Android RunTime (ART) in Android L.* Retrieved from: http://anandtech.com/show/8231/a-closer-look-at-android-runtime-art-in-android-l

7. Brahler, S. (2010, 6 October). *Analysis of the Android Architecture*. Retrieved from: https://os.itec.kit.edu/downloads/sa_2010_braehler-stefan_android-architecture.pdf

8. Oluwatosinm, H. S. (2014, February). *Client-Server Model*. Retrieved from: http://www.iosrjournals.org/iosr-jce/papers/Vol16-issue1/Version-9/J016195771.pdf

9. Hengming, F. Jia, C. Bin, X.(2013). *The Interaction Mechanism based on JSON for Android Database Application*. Retrieved from: http://docsdrive.com/pdfs/ansinet/itj/0000/45103-45103.pdf

10. Nurseitov, N. Paulson, M. Reynolds, R. Izurieta, C.(2009). *Comparison of JSON and XML Interchange formats*. Retrieved from: http://www.cs.montana.edu/izurieta/pubs/caine2009.pdf

11. Yang, X.(2009). *Human-Computer Interaction Design in Product Design*. Retrieved from: http://ieeexplore.ieee.org.ezproxy.library.qmul.ac.uk/stamp/stamp.jsp?tp=&arnumber=4959073

12. Kim, G. J.(2015, 19 February). *Human-Computer Interaction Fundamentals and Practice*. Retrieved from: http://www.ittoday.info/Excerpts/HCI.pdf

Appendix

Heuristic evaluation of app:

Taken before start of testing phase 1 and end of phase 1 development.

Heuristic	Problem	Severity Rating	Solution to problem
Visibility of system status:	No problems as system tells users errors, when loading etc.	-	-
Match between system and the real world:	No problems as errors and messages are converted to understandable human form	-	-
User control and freedom:	On certain tasks users cannot go back as back button is disabled so user cant cancel once gone a certain way through some tasks	2	Either tell the user that they can't go back until the task is completed or allow the back button to cancel the task by giving pop up option to continue or cancel.
Consistency and	Inconsistency of	2	Change buttons with different

standards:	button/ component locations and naming		names such as save, submit which are used for the same action to be changed to a single name when appropriate such as change submit to save.
Error prevention:	When the user cant connect to the server either through no connection to internet or server problem some database calls don't check for errors causing the activity to restart or crash	4	Make sure to check when there is a timeout or server error by redirecting them back to homepage with a message saying what happened
Recognition rather than recall:	For logging in users have to constantly re-enter details when relaunching app.	3	Modify login section to allow automatic login when user closes app, add logout button on homepage to clear details for future login.
Flexibility and efficiency of use:	No problems	-	-
Aesthetic and minimalist design:	No problems	-	-
Help users recognize, diagnose, and recover from errors:	When database errors occur as highlighted in error prevention activity closes and goes to previous open activity so user cannot recover if constantly happening	2	Fix error prevention so that user doesn't have to deal with fixing or recovering from errors manually.
Help and documentation:	New users may not know how to use app/certain features	2	Provide a manual on how to use app, what features do

User feedback phase 1:

All users are placed into one version and colour coded:

User 1

User 2

User 3

User 4

device used: Sony Xperia M4 Aqua, Samsung s6 and nexus tablet, Samsung galaxy s3 mini, Samsung Galaxy S6

Android os installed: 5.1.1, 5.1.1 on s5 and 4.4.4, 4.4, 5.1

Did you create any questions? If so did you have any problems?

Yes, I didn't find any errors except for remembering quiz passwords when trying to add a new question
Yes there were no issues with the application
I didn't have any errors with creating questions, except for when creating as part of a group I noticed that when on the view groups page and another user creates a group with me in it doesn't show up until going to the previous panel then back to viewing groups.
Quiz would only show up on the quiz page if there is more than 2 questions for that quiz. Not sure if this is meant to happen or is a bug.

Did all panel components display correctly (i.e no clipping of background image border,no components on top of each other etc..)

I found that the leaderboard and the table for choosing a quiz to play didn't look right. I saw that the quiz names when long cut off so you can't see it all, the leaderboard had some clipping on the sides and the radio buttons were small.
No issues with the display on s6.
On tablet there was lots of clipping on the add question and play panels, the radio buttons and textboxes went over the border, the buttons don't look right as they are very stretched.
The button at the bottom in the page to add a question clipped with the bottom of answer 4 box, same with the page for playing.
Everything displayed correctly. Not entirely sure if this was a display error but the stars when answering a question correctly appeared as 5 and were not filled in. Unsure if its based on filling in the starts out of 5 or if it is based on how many stars appear on the bottom.

Did you find the app intuitive/easy to use? If not what would you improve

I found it nice to use as the buttons are intuitive and simple to use, however I found that some of the steps where a bit long such as being able to create a question you have to go through a lot of panels.
Yeah very easy to use
It was mostly easy to use except for adding a question I kept having to create quizzes with the same passwords as I couldn't remember passwords for multiple quizzes
App overall works very well and functions correctly. An improvement might be to have a manual/tutorial button which gives instructions on how to do certain features as at times I was slightly confused on some of the functionality.

Are there any improvements for current features that you would like to see for final version?

I would like to see all the quizzes that I have created and be able to select one for quick access to adding a new question. I would also like to see all components display correctly and have a search function for searching a quiz to play as I found that even with only a small number of quizzes that it was sometimes hard to find the quiz I wanted.
The pick a quiz looks simple but I would to see how difficult is the quiz to be used
Be able to view passwords of quizzes that I have created or added questions to in the past, have refresh buttons on the select groups page to remove need to go to previous panel then back again
Have more than 50 questions on the highscore mode.
Unsure on the group feature and how it works. Making it more clearer maybe on what it does.

Any extra features you would like to see which you feel would improve your experience?

I would like to be able to modify my details such as password or email address. Play with friends who we created questions with as a group to see who is the best, as this would add more replay ability and make learning with the app more enjoyable and improve replay ability for general use.
Make the group page option to have no password so people can go in for free
Have option to change the layout and background around the app
I would also like to see more game modes as I feel only 2 options gets a bit boring after a while
options to create quiz or question without a password
Be able to add profile pictures and view stats of my accounts, have personal goals to achieve such as how many questions I answered this week.
Possible feature of linking account with Facebook account so it is possible to login with a persons Facebook

User feedback phase 2:

User 1

User 2

Did you encounter any problems using the app?

Nope I did not have no problems with app and automatically log me in and recognised

No

How did you find the current version better to use than previous versions? If so/not why?

The currrent vision is much fluid and simple to use Yes I found that the app was much improved. I thought that the new features helped me stay interested in using the app and even though there are more features it is still intuitive and the new features are easy to use. I found that the information button was very helpful as it explained to me why some quizzes weren't being able to be played when creating them in initial version I previously used.

What improvements would you like to see for future versions?

Just add chatbox to the system I would like to be able to delete finished matches from the online mode as it seems to get cluttered quite quickly with multiple matches. I would also like to have different topics to select from so you can choose a topic such as Computer Science and have quizzes sorted into a relevant topic- I feel this would make the app be easier to navigate quizzes in the future which will help users.