Interim Report

Matty Christopher - 130080943

supervisor: Dr Anne Hsu

Project Title: Quiz Maker App

Table of Contents

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 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 [1]. 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[2], 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.[3]

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.

2.2 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.[4]"

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 clients 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[4].

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 users 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[5]. 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[6].

2.3 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[7].

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[8]. 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.

2.4 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

2.5 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 eg "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.

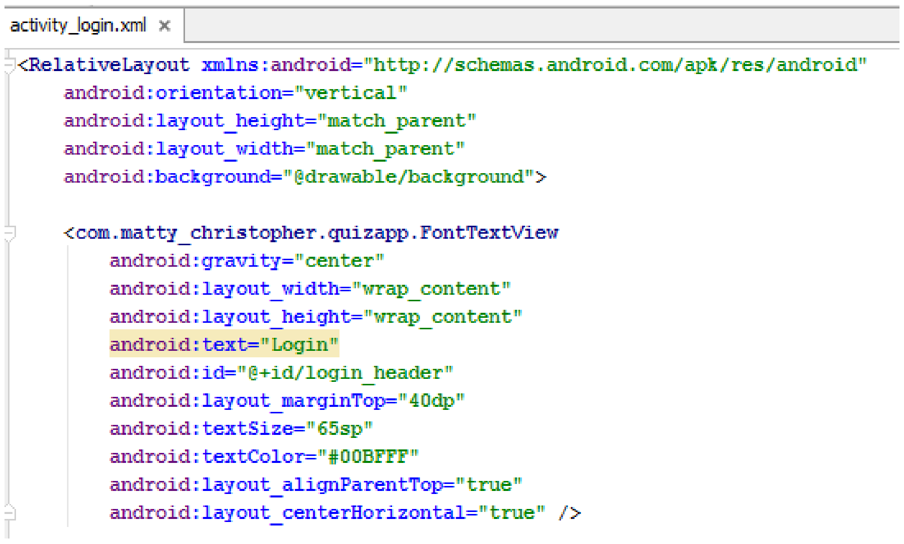
The question creator will be able to enter 3 other names they want to join to the group which will then be displayed in the view groups section. Users will be able to view the groups they are in then select one of the groups to edit the question where they will be able to edit the answer that they have 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:

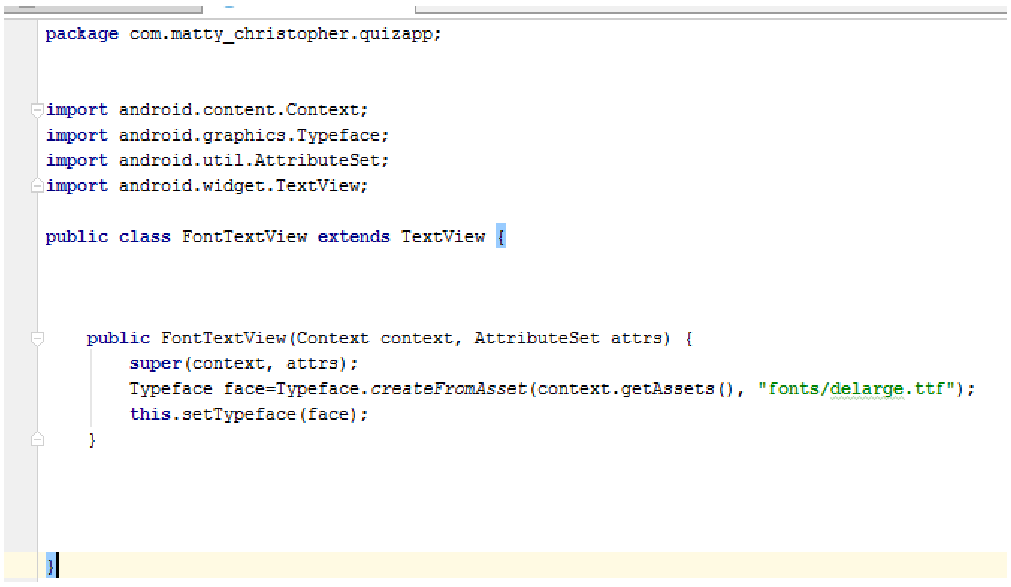
To create the interfaces I added components such as buttons and text views to the .xml files in the layout folder for each created activity. Here I used the Relative layout which allows me to display components on different parts of the screen easily without having to use multiple linear layouts.



activity_login.xml ×

```
<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"
        android:textColor="#00BFFF"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true" />
```

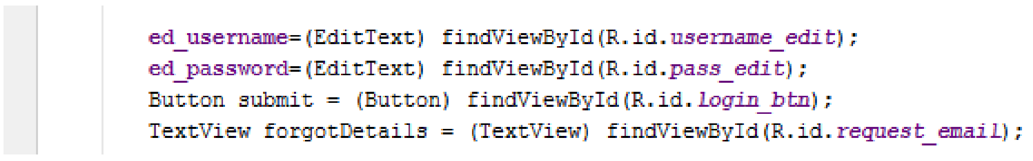
To create the headings of each panel such as Login I decided to use the delarge.ttf font which mimics a marker pen font. To do this I used this class-which I have named FontTextView, below which I found online[http://www.101apps.co.za/index.php/articles/using-custom-fonts-in-your-android-apps.html]:



```
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);
    }




}
```

This class allows me to create a custom text view in the xml layout files so I don't have to keep calling 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:

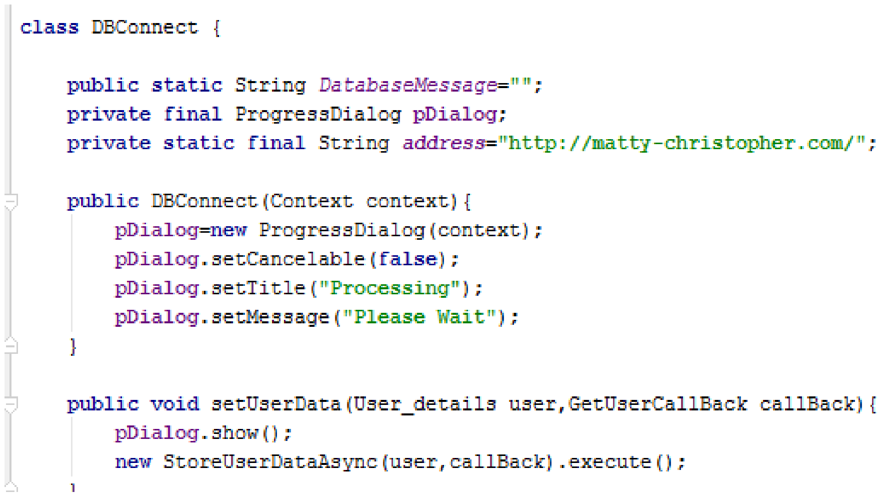


```
ed_username=(EditText) findViewById(R.id.username_edit);
ed_password=(EditText) findViewById(R.id.pass_edit);
Button submit = (Button) findViewById(R.id.login_btn);
TextView forgotDetails = (TextView) findViewById(R.id.request_email);
```

Creating the database classes:

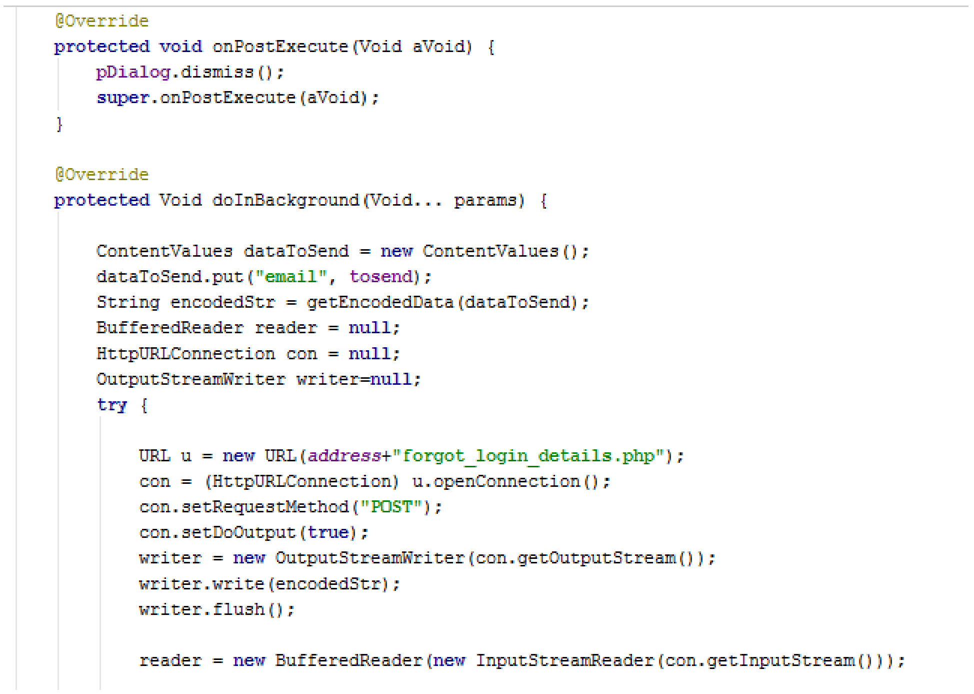
As my app requires a database I needed to create a series of php scripts which are stored on my server at "InMotionHosting.com". I set up a "DBConnect " class which handles all of the scripts for logging in and registration of users. First I created a "User_details "class which stores all the data belonging 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. 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();
```

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. 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:

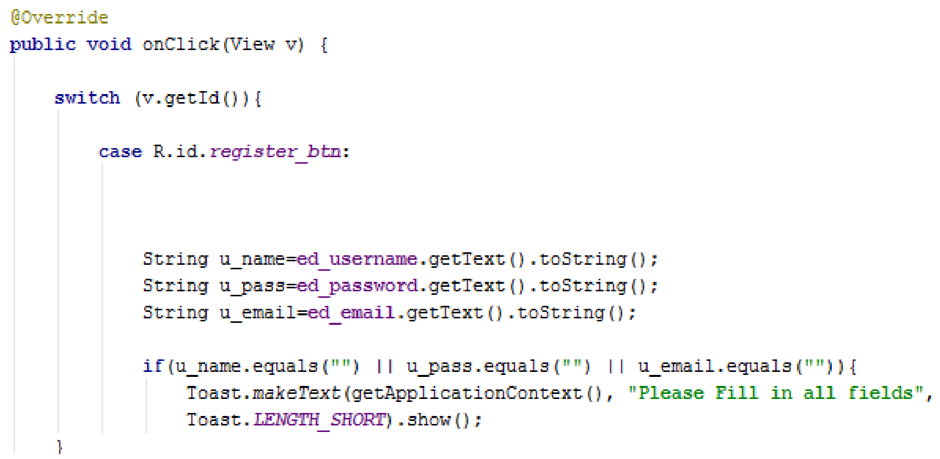


```
@Override
protected void onPostExecute(Void aVoid) {
    pDialog.dismiss();
    super.onPostExecute(aVoid);
}

@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()));
```

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.

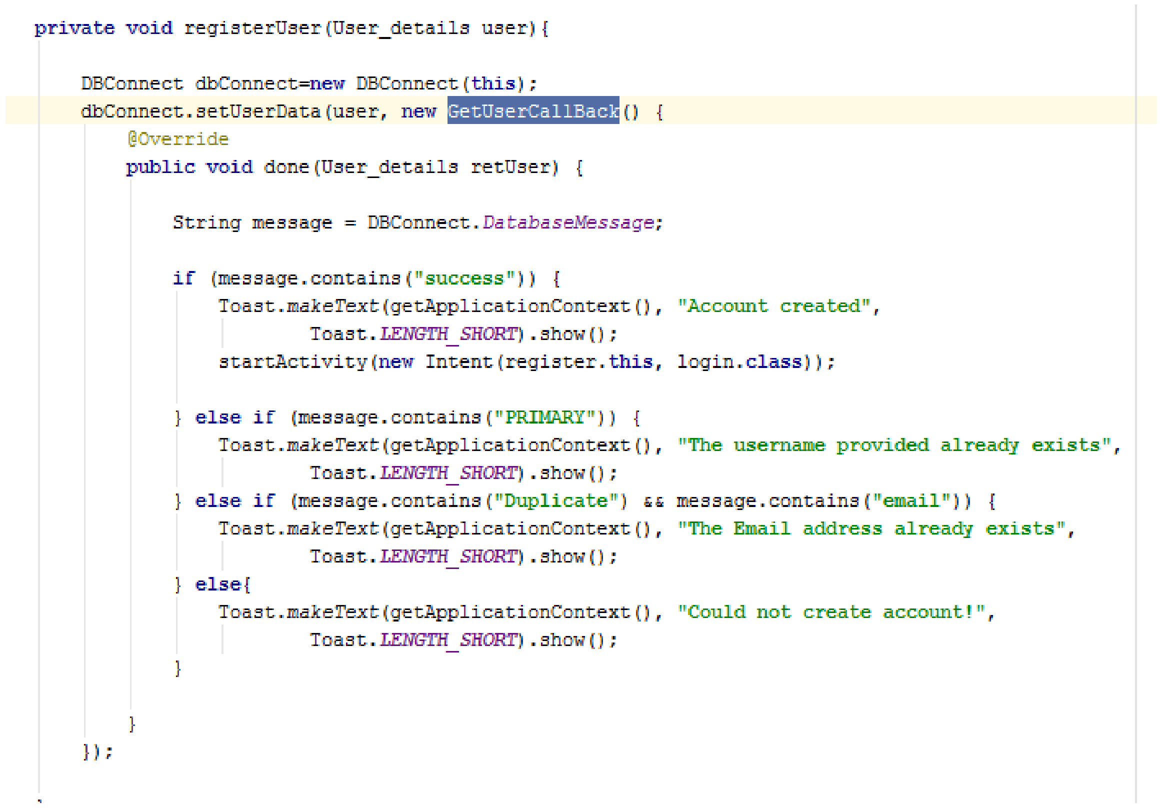
Creating the registration section:

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();
    }
```

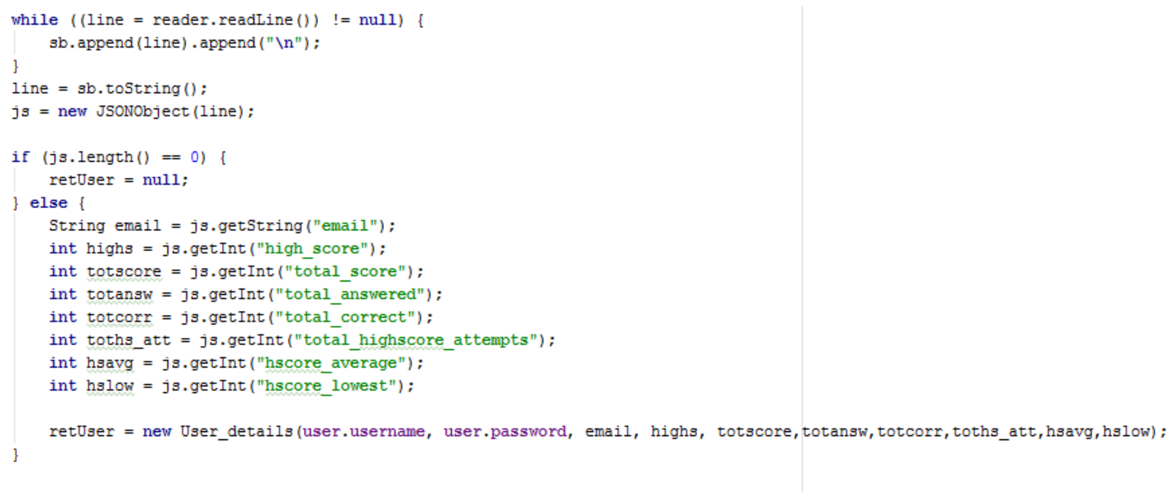
Otherwise the database class would be called. I created a new method "registerUser(user)" which handles calling the "DBConnect" class. A new object of the class is created which then allows me to call the "setUserData()" method which in turn executes the "StoreUserDataAsync" class which calls the "register.php" 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.



```
private void registerUser(User_details user){

    DBConnect dbConnect=new DBConnect(this);
    dbConnect.setUserData(user, new GetUserCallBack() {
        @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();
            }

        }
    });
```

Creating the login section:

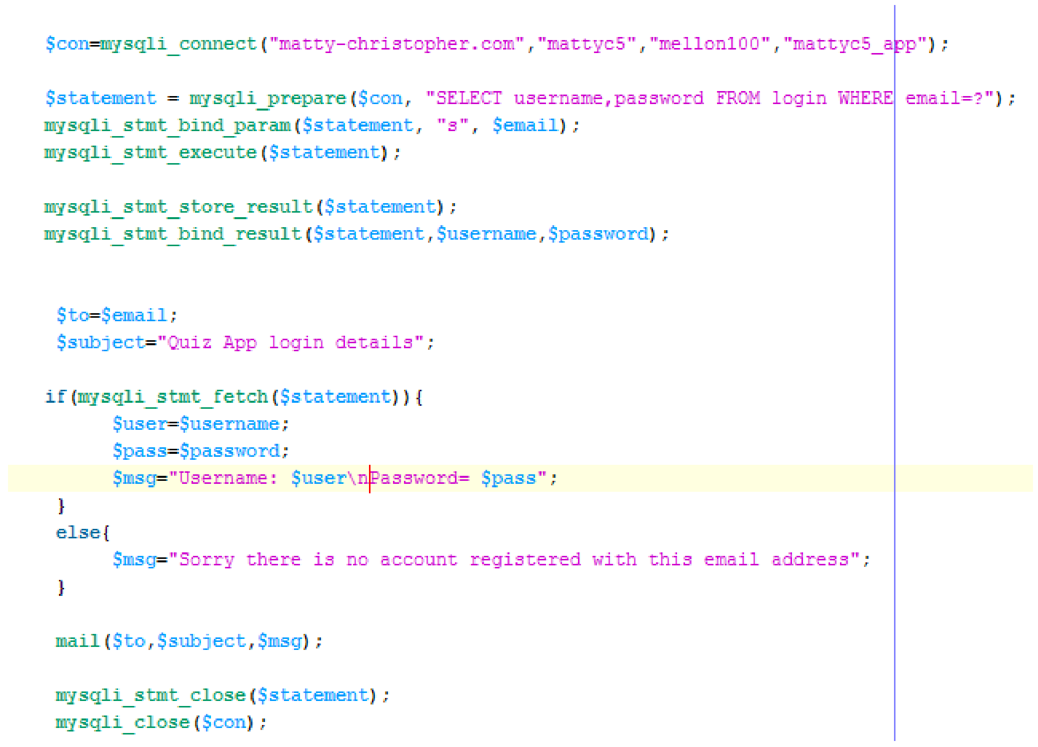
For the login class my implementation is similar to the register class for setting up the interface and calling the database. Here I call the "getUserData()" method in the "DBConnect" class which calls the "GetUserDataAsync" class which I implemented to call the "login.php" script. Here if the response 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.



```java
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);
}
```

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.

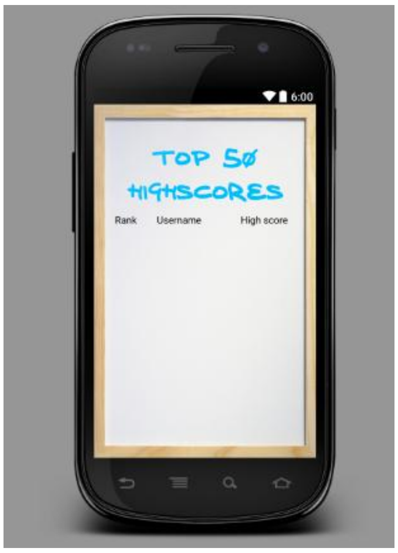
I also implemented a textbox to be a button in which allows users to send an email to request their details. To do this I created another "DBConnect" object which then calls the "getUserEmail()" method which will execute the AsyncTask class which sends the email address provided by the userto the email.php 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.



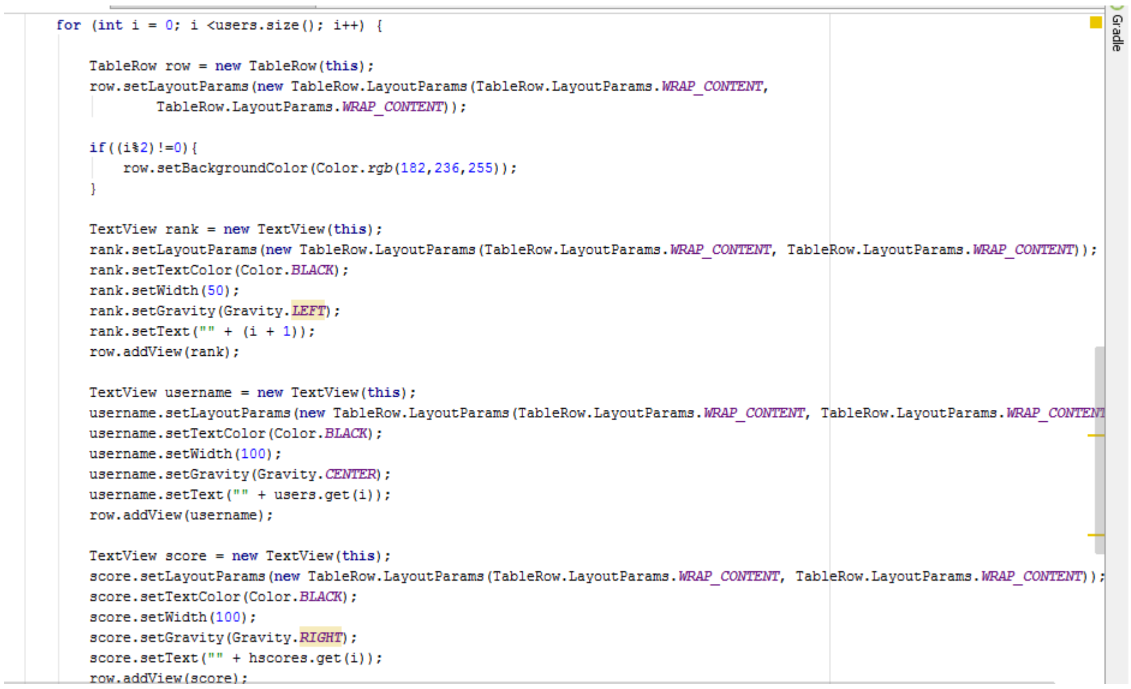
```php
$con=mysqli_connect("matty-christopher.com","mattyc5","mellon100","mattyc5_app");

$statement = mysqli_prepare($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);
```

Creating the leaderboard section:

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:

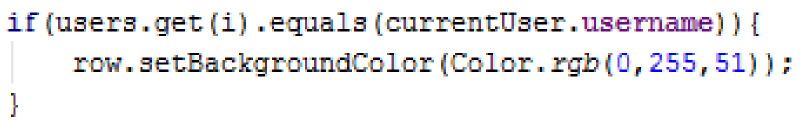


I then implemented the database connection which retrieves the data from the script and stores the data into 2 arraylists: user and highscore. Here I then created a loop which goes around user.size()-1 times which inside the loop I create new rows for each user.



```
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_CONTEN
    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);
```

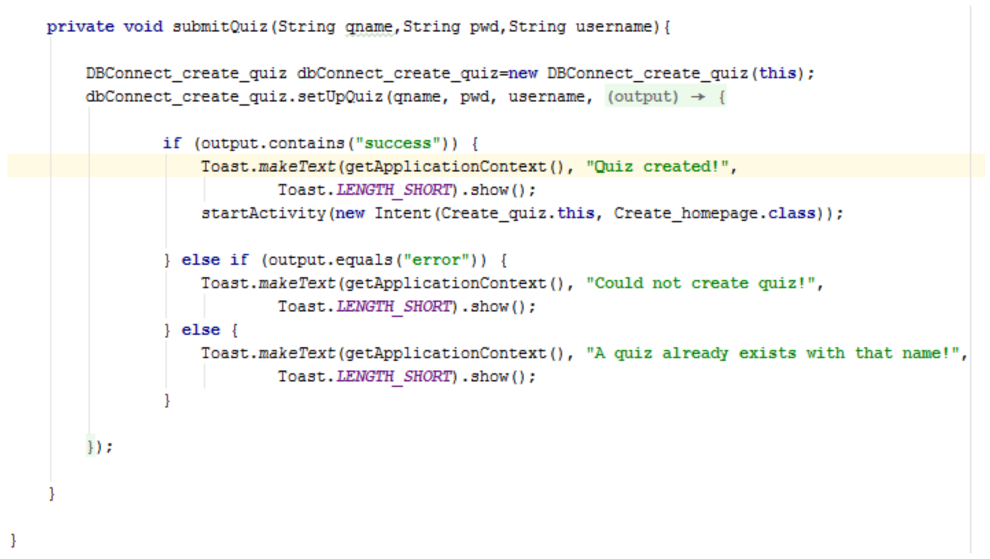
The final step was to implement an if else statement at the end of the loop which checks if the current user matches the user in the shared preference class, if the user matches then I made that row's background green so that the user can clearly see their rank.



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

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();
                }

        });

    }

}
```

How I will implement create question:

To implement the create question section I have made the login to the quiz step where the user selects the quiz they want to add the app to. The next step is to add the panel which deals with choosing whether to create the question as a group or on their own. To create a question on their own the panel will display the logged in username for each answer and question, whereas with creating a group the user will enter 3 other usernames which will then be checked against the user table in the database to see whether they exist. Then on the question panel the user will have the option to assign each member to part of the question through drop down boxes next to each answer edit text box. When users edit the question the panel will check the shared preferences logged in user against the assigned text box, wherever they are matched they will be able to edit that text box.

How I will implement view groups section:

To implement this section I will create a table with a scrollbar attached to it similar to the leaderboard section. On each row returned will have an onclick listener which will allow the user to select a row and then be taken to an edit question panel where they can edit the assigned part of the question. To do the script I will have to send the current users username to the script which will have a query which checks the username against the groups table and will store each row where that user is part of into an array which will be returned. This array will then be used to fill each row in the table.

How I will implement play section:

I will need to create a new question class which stores the question along with the four answers and correct answer number. I will then need to create scripts which retrieve questions from the database and store them into the class. I will also need to keep track of stats when a user is playing by storing them into a local arraylist/s and upload them to the database through a script at the end of the play through. This will be better than continuously calling the database at each question as there will be only one call making it quicker.

4. Testing and Evaluation proposal:

To make sure that the app I have created works as it was designed to I will do extensive testing. To do this I will find a number of users to test my prototype for a set number of time. Here I will listen to their feedback to check whether they found major bugs or unexpected behaviour. I will take into account all feedback to see what changes are needed such as changing font colours, making the layout easier etc. I will do this iteratively so that I can see whether changes that I make are improved or not and see if the latest prototype is better overall for the user.

My proposed testing schedule is:

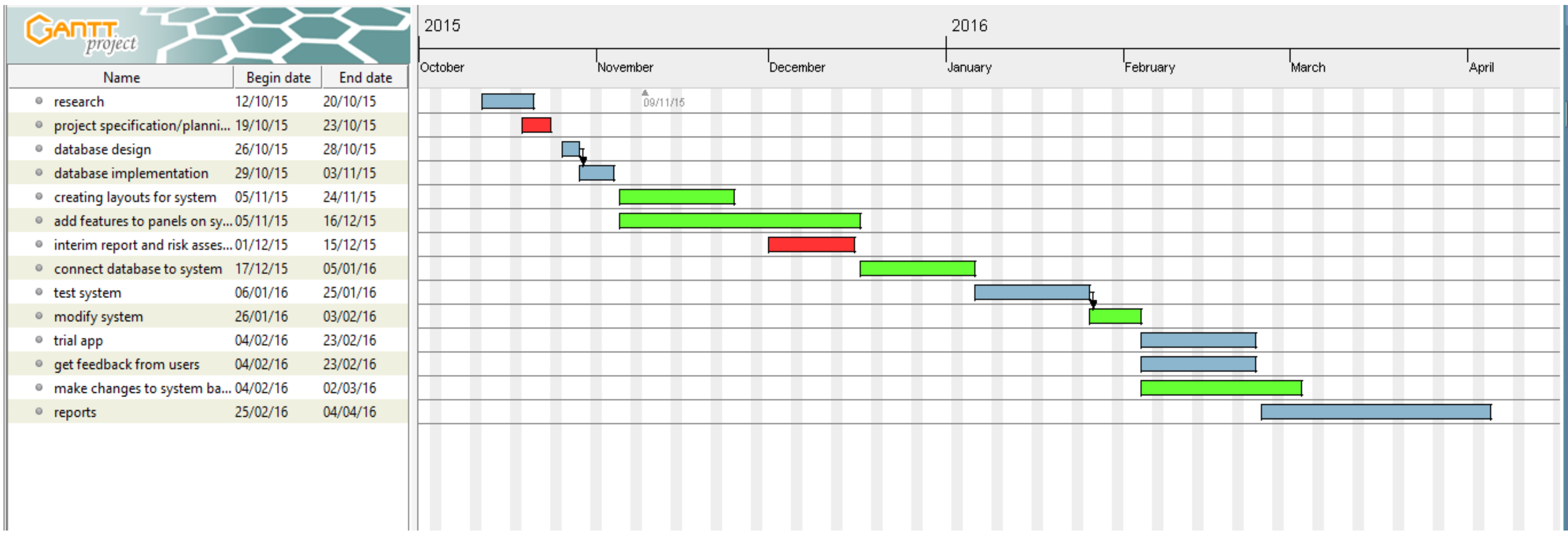
1. Design initial prototype
2. Do initial bug testing to find as many bugs as possible
3. Get a number of users to play with the app
4. Listen to feedback and make a list of issues
5. While users test the app I will start to work on fixing the most common/major issues/errors
6. After the initial user testing phase I will work on improving the app using feedback to make general improvements
7. Get users to play with the app again

I aim to have at least 2 major user testing and feedback stages before the final app is completed.

Once I have made the final version of the app I will evaluate it against the requirements that I created at the start of the project to check whether my app satisfies all of the set out requirements.

5. Review of Progress as of 29th Nov 2015

This section talks about what I have done as of the 29th Nov 2015 (Beginning of week 10 of semester 1) and I review whether I am on track or not. I will use the work plan below that I created at the start of the project as reference.



Name	Begin date	End date
research	12/10/15	20/10/15
project specification/planni...	19/10/15	23/10/15
database design	26/10/15	28/10/15
database implementation	29/10/15	03/11/15
creating layouts for system	05/11/15	24/11/15
add features to panels on sy...	05/11/15	16/12/15
interim report and risk asses...	01/12/15	15/12/15
connect database to system	17/12/15	05/01/16
test system	06/01/16	25/01/16
modify system	26/01/16	03/02/16
trial app	04/02/16	23/02/16
get feedback from users	04/02/16	23/02/16
make changes to system ba...	04/02/16	02/03/16
reports	25/02/16	04/04/16

5.1 What I have done:

For this semester I started planning what the app would look like by creating a project specification in which I highlighted the aim of the project and what the requirements would be to make a good app. I created a set of layouts which I want the app to look like and I put these into a storyboard which I explained for each panel what interaction there would be on them allowing me when creating the app to implement it more effectively.

After the planning stages I set up the database using MySQL, I designed the database firstly by hand plotting what tables I would need so far. I have created 5 tables so far:

login: this table will be used for handling users creating accounts and retrieving info for logging in.

quiz: this handles the individual quizzes which can be created by a user, questions will then be assigned to the quiz they were created for, keeps track of how many questions there are for each quiz and handles the average rating of all the questions assigned to the quiz.

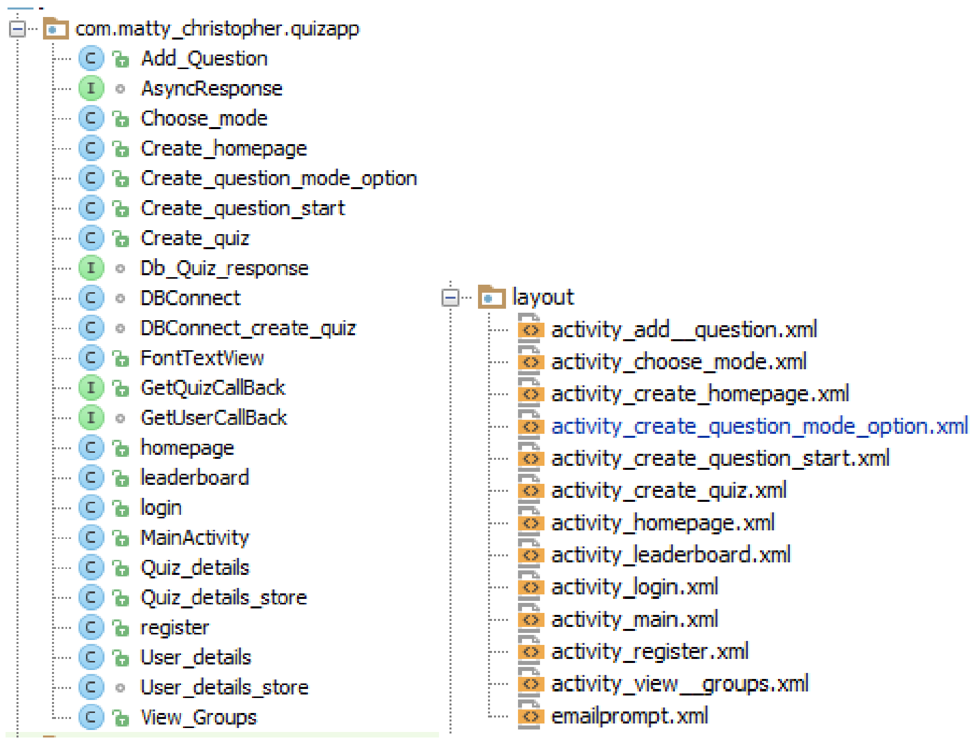
questions: this handles specific questions which contain info such as the quiz it is part of, question id, the question and correct answer, usernames of users who created the answers and question.

question_stats: assigned to each question to keep track of stats such as how many times the question has been attempted and keeps track of how many times each answer was selected.

groups: this handles user created groups by keeping track of what users are in the group (there are 4 tables for users) and what question they are assigned to. The creator of a question can choose to create the question on their own or as a group where they can then assign members.

I have created 6 php scripts which insert and retrieve data from the tables for displaying leaderboard, displaying what groups user is in, login authentication, registering and authenticating, setting up a new quiz, script to send an email to user if they forgot their login details.

For the App I have created the xml files for the welcome panel(main activity), login panel, registration panel, home panel, leaderboard, create home, create quiz panel, view groups, and create a question panel. I have implemented all of these panels so far except the view groups and I am currently working on the create question panel. I have created multiple classes to store details with shared preferences to use retrieved details on different panels, I have implemented two database classes which connect to the php scripts on my server (one is for user scripts and the other for creating quizzes and questions). These are the classes and xml files that I have implemented and working on:


```
com.matty_christopher.quizapp
    Add_Question
    AsyncResponse
    Choose_mode
    Create_homepage
    Create_question_mode_option
    Create_question_start
    Create_quiz
    Db_Quiz_response
    DBConnect
    DBConnect_create_quiz
    FontTextView
    GetQuizCallBack
    GetUserCallBack
    homepage
    leaderboard
    login
    MainActivity
    Quiz_details
    Quiz_details_store
    register
    User_details
    User_details_store
    View_Groups

layout
    activity_add__question.xml
    activity_choose_mode.xml
    activity_create_homepage.xml
    activity_create_question_mode_option.xml
    activity_create_question_start.xml
    activity_create_quiz.xml
    activity_homepage.xml
    activity_leaderboard.xml
    activity_login.xml
    activity_main.xml
    activity_register.xml
    activity_view__groups.xml
    emailprompt.xml
```

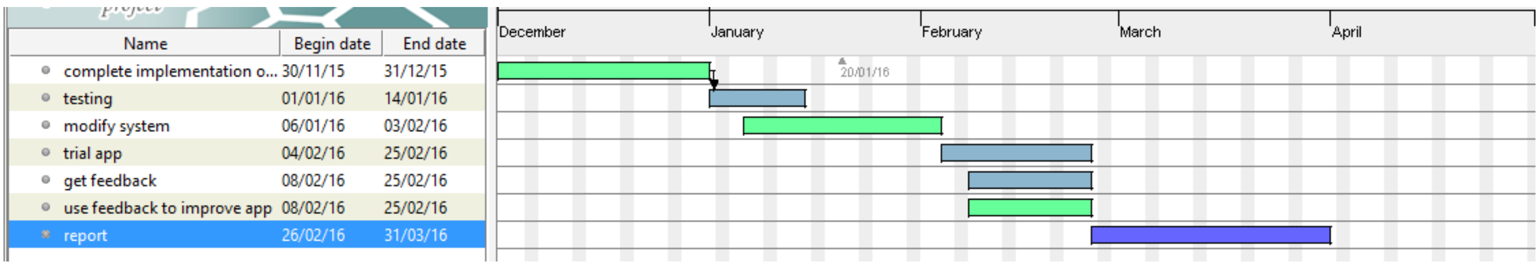
Currently with my app it allows the user to register an account which checks if provided username and email is in use already, you are able to login or request details if forgotten via an email, users can view the leaderboard which displays the top 50 (I only have 5 users currently so they all show) and the logged in user is highlighted if they are on there. Users can also setup a new quiz by entering a quiz name and password.

I have also been testing panels as I create them to make testing later easier as I should have less bugs before testing a complete version, as I have fixed every bug that I have encountered so far.

5.2 Review of progress:

I believe that I am progressing very well based on the work plan that I created previously, I am ahead of schedule in parts - I have already connected the app to the database so any extra scripts I create connection will be easy and quick to implement. However on the other hand I am behind schedule on the "creating layouts for the system" task, this is because I have been creating new panels once I have completed the previous panel in the sequence i.e. I created the login panel layout after creating the registration panel. I have also been implementing the database concurrently with the design of the panels so I could fully implement panels as I went along. Overall I am on track to complete a fully working version of the app by the start of semester 2 (11th January) and I hope to finish earlier so I can do some extra testing before semester 2. This will allow me to spend more time adding more features to the app to improve the user experience.

6. Revised work plan for Semester 2:



Name	Begin date	End date
complete implementation o...	30/11/15	31/12/15
testing	01/01/16	14/01/16
modify system	06/01/16	03/02/16
trial app	04/02/16	25/02/16
get feedback	08/02/16	25/02/16
use feedback to improve app	08/02/16	25/02/16
report	26/02/16	31/03/16

This is my revised work plan for the rest of this semester and semester 2. I have changed some of the tasks from my original for example I have removed the task "connect database to system" as I have already set that up. Also I have brought forward the testing phase by a week as I believe that I am on track to finish a working version slightly earlier than anticipated. The rest is similar to my original with the start dates brought slightly forward. This will give me roughly 3-4 weeks before the deadline in April to make any other changes that I may need, and to use if I run into any problems (illness, major bugs etc).

7. Risk Assessment:

Risk	Impact	Likelihood	Impact rating	Preventative actions
Hosting server goes down	Would not be able to connect app to database so user would not be able to access most of app.	Low	High - would make app useless to user, would not be able to do testing of parts of app that require	Could have a backup hosting server, but otherwise hard to prevent

			database connection	
Poor time management	Cause deadlines to be missed, would cause project implementation to fall behind	Medium	High	Create weekly schedules with long term ones. Regularly work on project.
Unable to find enough testers	Testers less likely to find as many bugs as having more testers.	Medium	Medium	Explore other ways to find testers e.g. online forums to ask for testers.
Failure to debug software	Would mean that users would be more likely to run into bugs when using app which would cause them to get frustrated with using it.	Low	High	Make sure to regularly debug while working on software, make sure to leave enough time after completion for debugging.
Computer/Hard drive failure	All un-backed up work would be lost-need to be recovered	Low	High	Make sure to backup work weekly on external devices and online storage (cloud/repository)
Become ill	Would cause me to fall behind as would not be able to do as much work(any work) for as long as I am ill.	Low-Medium (depending on seriousness)	Medium-High (depending on seriousness)	Make sure to do as much work as possible when I have free time and early so I can afford some days off if ill.
Misinterpret project specification	Would mean that project app would be different to what was required meaning client is unhappy	Low	High	Make sure to talk to client(supervisor) if unsure about any requirements and create a prototype so you can see if it is on track.

Bibliography

1. (2015, Aug).*Smartphone OS Market Share, 2015 Q2*. Retrieved from: http://www.idc.com/prodserv/smartphone-os-market-share.jsp

2. 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

3. 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

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

5. 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

6. 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

7. 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

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