Course: BSc Software Engineering
Module: Mobile Development – Assignment1 - An Interactive Game - Report
Student: Navin Bajgai (@00433761 / stc392)
Lecturer: Dr Ian Drumm
University: The University of Salford

1 INTRODUCTION

This is a report of evaluation of my finished interactive game application named "**Science Quizzes**" for Mobile Development module. The report will look at different topics discussing the contents and layouts of the interface along with the coding skills being used to enhance and implement the interface itself of a desired user and system requirements. The report will mainly focus on the analysation and the breakdown of the design solution that has been developed, with the use of some familiar visual paradigm and driver design tools and materials such as, use-cases, classes, activity and sequence diagrams. The use of Android Studio's tools and features for the development of an application will also be looked into detail along with the design patterns implemented for creating the classes in a structural order. The key features and functionalities of the game application will then be identified and tested systematically which will be presented in the screenshots with annotations, in the latter part of the report. Additionally, the report will reflect on my self-assessment of the work done in the task discussing the various criteria of the task and how I am satisfied with each aspect of work assignment. The report will eventually be concluded with the presentation of valid reference and permission of any copyrighted materials used in the building of interface design solution. The report will cover following topics;

- Application Content and Design
- Use of Android Studio
- Discussion on use of Android Studio's Tools and Features
- System and Driven Designs
- Functional Testing of Design Solution
- Comparison and Evaluation
- Self-Reflection on the Work Done
- References

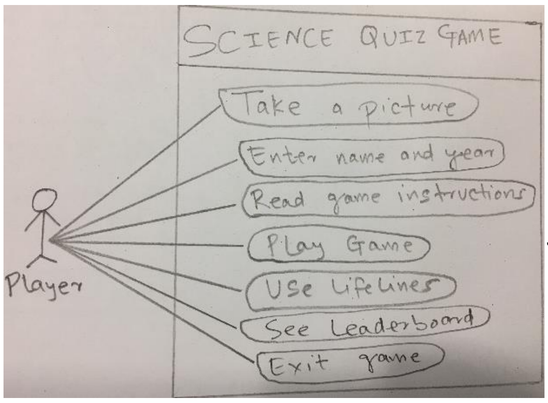
2 APPLICATION CONTENT AND DESIGN

With the game application being titled "**Science Quizzes** – can you use some maths in science?", the game has primarily been created for teaching secondary school students to learn basic physics and chemistry's formulas to calculate a combination of physics and chemistry's equations on the topics including force, speed, velocity, atoms & molecule, chemical reactions, energy, power and so on. The game consists of 10 multiple choice questions with 3 answers options where a player must answer all questions correctly to successfully win a game. As the game had to be developed for secondary school level students, everything was needed to be kept simple and intermediate in terms of questions quality and game playing time to avoid complexity in the game. There were a lot of things to be taken into a consideration before actually starting to develop a game. Most particularly, the requirements elicitation was taken into a high consideration to ensure that the finished game application is fit-for-purpose and that it has been developed for a desired user-type and requirements.

The preliminary requirements were initially recorded in the form of wireframes and use-cases diagrams. The wireframes and use-cases diagrams were drawn to describe the user interactions that the application must provide.

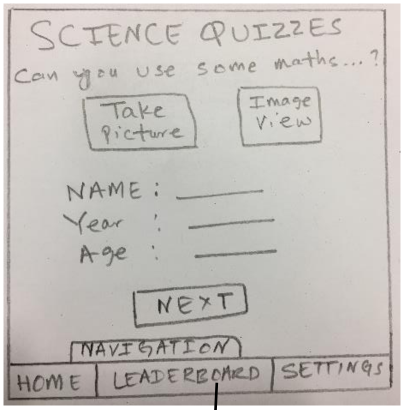
2.1 Use-case Diagram

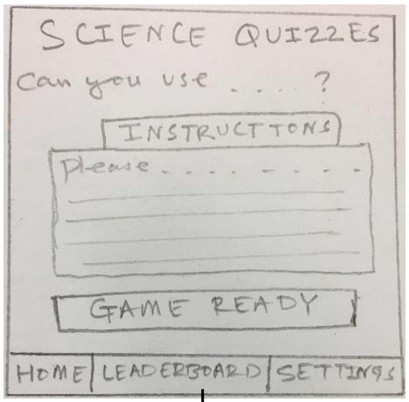
The diagram below shows the use-cases for the complete game application.

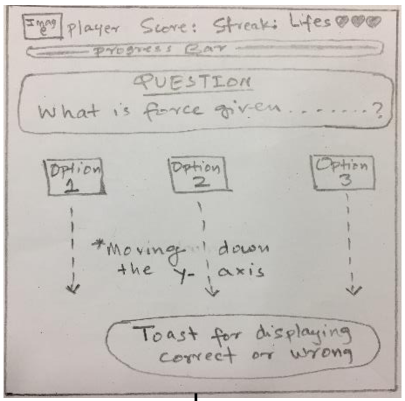


As you can see in the screenshot, player is the only actor of a system and that he/she can do a lot of different things to an application.

2.2 Wireframes for Game Application



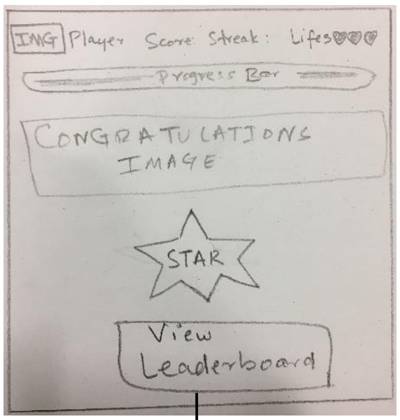


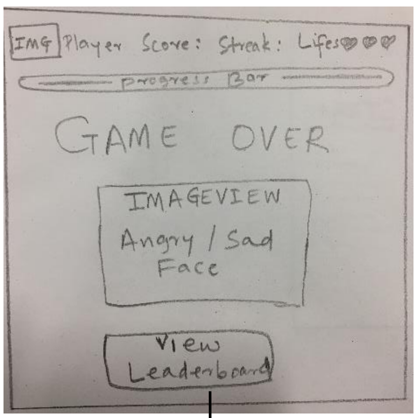


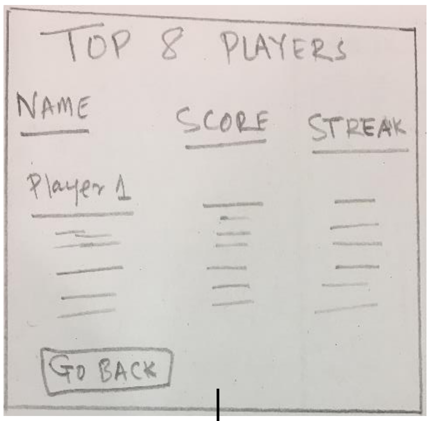
This is a start-view of a game. A player is required to enter his name and year in school to start a game. The player can also take a picture using a camera which will be used later in the actual game view. There is a navigation menu at the bottom of the screen. The leaderboard will take the player to a different activity which will display the top 8 scores from the database.

On the click of a next button in the start-up screen of the game, the player is redirected to another activity which will, as shown in the image above, displays the instructions for playing the game. The button 'Game Ready' will take the player to the game view where the first question will come up with the answer choices.

This is the actual game view. The 3 buttons containing the answer choices will start moving down (y-axis) and the player needs to click on any of the answer buttons before it goes past the screen. One button contains the right answer and other two contains the fake. If the player clicks on correct answer, the next question will show up with the new answers on the screen.







Once the player has answered all questions correctly, the above screen will appear to congratulate the player. The player can also click on 'View Leaderboard button which would then take a player to a game leaderboard view.

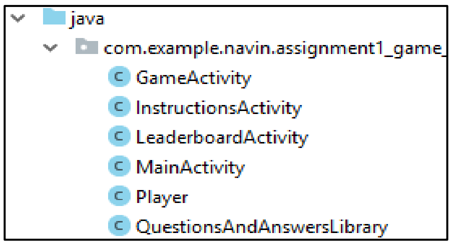
The above screen will appear once the player has used all three lifelines because this means a game is over. The player can also click on 'View Leaderboard button which would then take a player to a game leaderboard view.

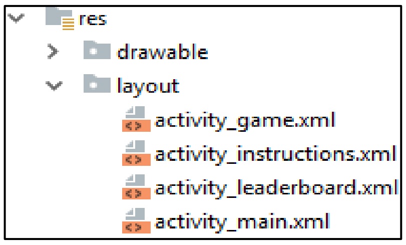
The is a leader-board for displaying top 8 players with the highest scores. Once the game is over, either by answering all question or by using all three lifelines, the score board is updated with the top 8 scores.

3 USE OF ANDROID STUDIO

Android Studio, an official Integrated Development Environment (IDE) for Android app development, was solely used for the development of a game application using JAVA Object-Oriented Programming language, which was in fact the only programming language used in the entire development process of a game application. According to ("What is Android Studio and Android SDK tools? - Android edX Community", 2017), Android Studio offers a great feature, an emulator that lets developer develop and test the apps without the use of any physical device which would consequently improve the performance of the development process with an instant visualisation and faster coding. The emulator will additionally enable the developers to debug and test the applications in a run-time environment which will make the development process easier, faster, smarter and more enjoyable.

Android Studio has been used in my project to create an interactive game application for teaching secondary school level students the basic science formulas and calculations in a fun and user-friendly environment. The JAVA activity classes were created which acts as the models and controllers of an application very similar to MVC design pattern. The JAVA activity classes contain the source code files including the Junit test code. The views are located on the '**res**' directory which contains all non-code resources including XML layouts, multimedia files, navigations, etc.


```
java
  com.example.navin.assignment1_game_
    GameActivity
    InstructionsActivity
    LeaderboardActivity
    MainActivity
    Player
    QuestionsAndAnswersLibrary
```


```
res
  drawable
  layout
    activity_game.xml
    activity_instructions.xml
    activity_leaderboard.xml
    activity_main.xml
```

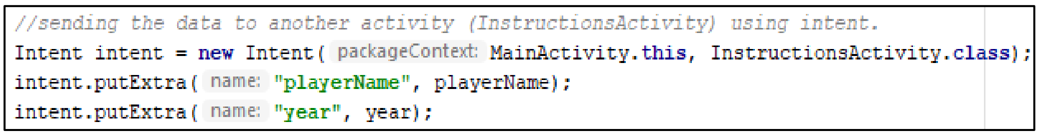
4 DISCUSSION ON USE OF ANDROID STUDIO'S TOOLS AND FEATURES

As it is said that Android Studio is a set of graphics and media packages that enables developers to design, create, test and deploy rich mobile applications, a wide range of different types of User Interface (UI) elements have been used across the application in order to provide secondary school students with user-friendly interactions and environment with the use of colourful interfaces elements, audio clips and animations.

4.1 INTENT

It is obvious that almost all of the web or mobile applications have a set of multiple screens or pages and that the user is redirected to another screen on the click on some button. Similarly, Android Studio uses **Intent** to redirect the user from one screen to another. Intent has been used various times in my dynamic game application for navigating user from one game activity to another, such as, from **MainActivity** to **InstructionsActivity**, from **InstructionsActivity** to **GameActivity**, from **GameActivity** to **LeaderboaadActivity** and so on.

The screenshot below shows that I am redirecting the user to a **InstructionsActivity** from the **MainActivity** along with passing some values in transition.

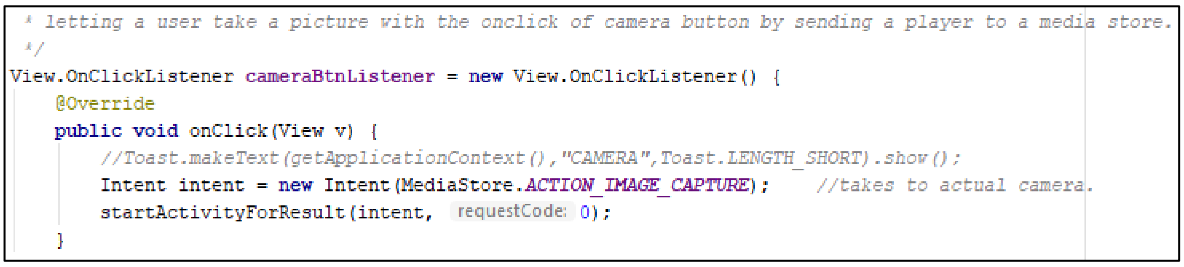


```
//sending the data to another activity (InstructionsActivity) using intent.
Intent intent = new Intent( packageContext: MainActivity.this, InstructionsActivity.class);
intent.putExtra( name: "playerName", playerName);
intent.putExtra( name: "year", year);
```

The values passed from the **MainActivity** were then received using the Bundles in the **InstructionsActivity**. The **getExtras()** method of type **Bundle** was used to retrieve the extended data from the intent that started the current activity, **InstructionsActivity**.

4.2 CAMERA INTENT

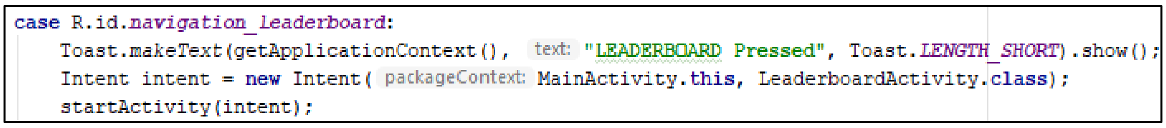
The game application also lets a player capture a photo to use in the game. This feature has been implemented using a camera intent. The standard intent action, ***ACTION_IMAGE_CAPTURE***, available from the media store which provides all available media was used to implement the camera application by letting it capture an image and return it. An ImageView was also created to temporarily store the player's picture which would later get sent for the use in the **GameActivity** using an Intent.



```
 * letting a user take a picture with the onclick of camera button by sending a player to a media store.
 */
View.OnClickListener cameraBtnListener = new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        //Toast.makeText(getApplicationContext(),"CAMERA",Toast.LENGTH_SHORT).show();
        Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);    //takes to actual camera.
        startActivityForResult(intent,  requestCode: 0);
    }
```

4.3 Navigation

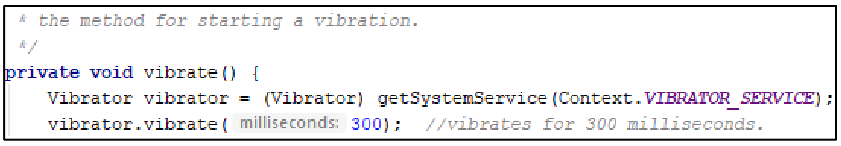
The **BottomNavigationActivity** was selected every time the new activity was created because the whole purpose of the application was to implement a game with at least 2 tabs which where one would let the player play a game and another would be to display the game leaderboard with top scores. As shown in the screenshot below, the leaderboard navigation button on the **MainActivity** will redirect the user to **LeaderboardActivity** which displays the players with top 8 scores.



```
case R.id.navigation_leaderboard:
    Toast.makeText(getApplicationContext(),  text: "LEADERBOARD Pressed", Toast.LENGTH_SHORT).show();
    Intent intent = new Intent( packageContext: MainActivity.this, LeaderboardActivity.class);
    startActivity(intent);
```

4.4 Vibrator

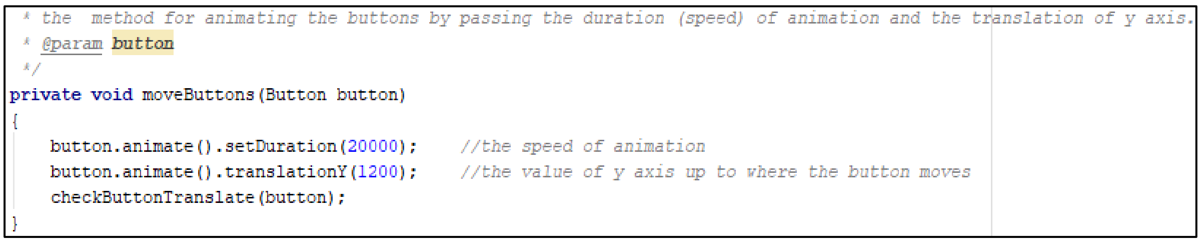
Vibrator was another Android Studio feature that was used in the game application for making the phone vibrate every time a player clicks on a button that contains the wrong answer. Vibrator, the class that operates the vibration on the device was used for the implementation. The method **vibrate()** takes a millisecond of type **long** in the arguments which would indicate how the phone will vibrate for.



```
 * the method for starting a vibration.
 */
private void vibrate() {
    Vibrator vibrator = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE);
    vibrator.vibrate( milliseconds: 300);   //vibrates for 300 milliseconds.
```

4.5 Animation

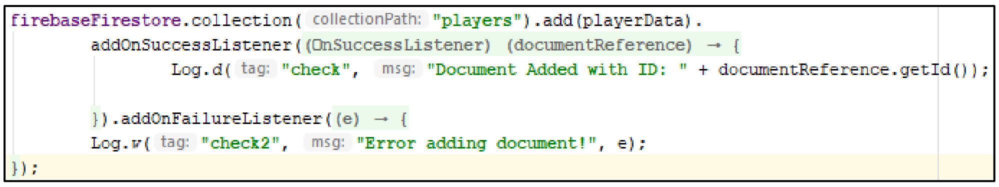
On the game view, the 3 answer choice buttons have been animated to move down the y-axis as soon as the question gets updated. The main theme of a game is that the answers will start moving down and the player has to click on the correct answer before the answers get past the screen and disappear in order to remain in the game. The method **animate()** is used to animate the specific properties on the view, which in my case, is to animate the answer buttons by passing the duration/ speed of an animation and as well as the value of y-axis up to which the button should be animated to.



```
 * the  method for animating the buttons by passing the duration (speed) of animation and the translation of y axis.
 * @param button
 */
private void moveButtons(Button button)
{
    button.animate().setDuration(20000);    //the speed of animation
    button.animate().translationY(1200);    //the value of y axis up to where the button moves
    checkButtonTranslate(button);
}
```

4.6 Firebase Firestore

Firebase, usually known as cloud-service technology for real-time data synchronisation has been implemented in the game application for storing player's information and game statistics. From the different types of firebase databases, Cloud Firestore has been chosen which will store the players' data in document which are then stored in the Maps collections using key and value for every single player's data. The firebase has primarily been used for the game application leaderboard. It has been implemented in such a way where as soon as the game is over, the player information including name, points earned, and the best streak achieved from the game are populated into a database. Additionally, after populating the player's information into a database, the information is also queried from the database into the leader board. The screenshot below shows the backend implementation of populating players data into the database using Firebase Firestore.



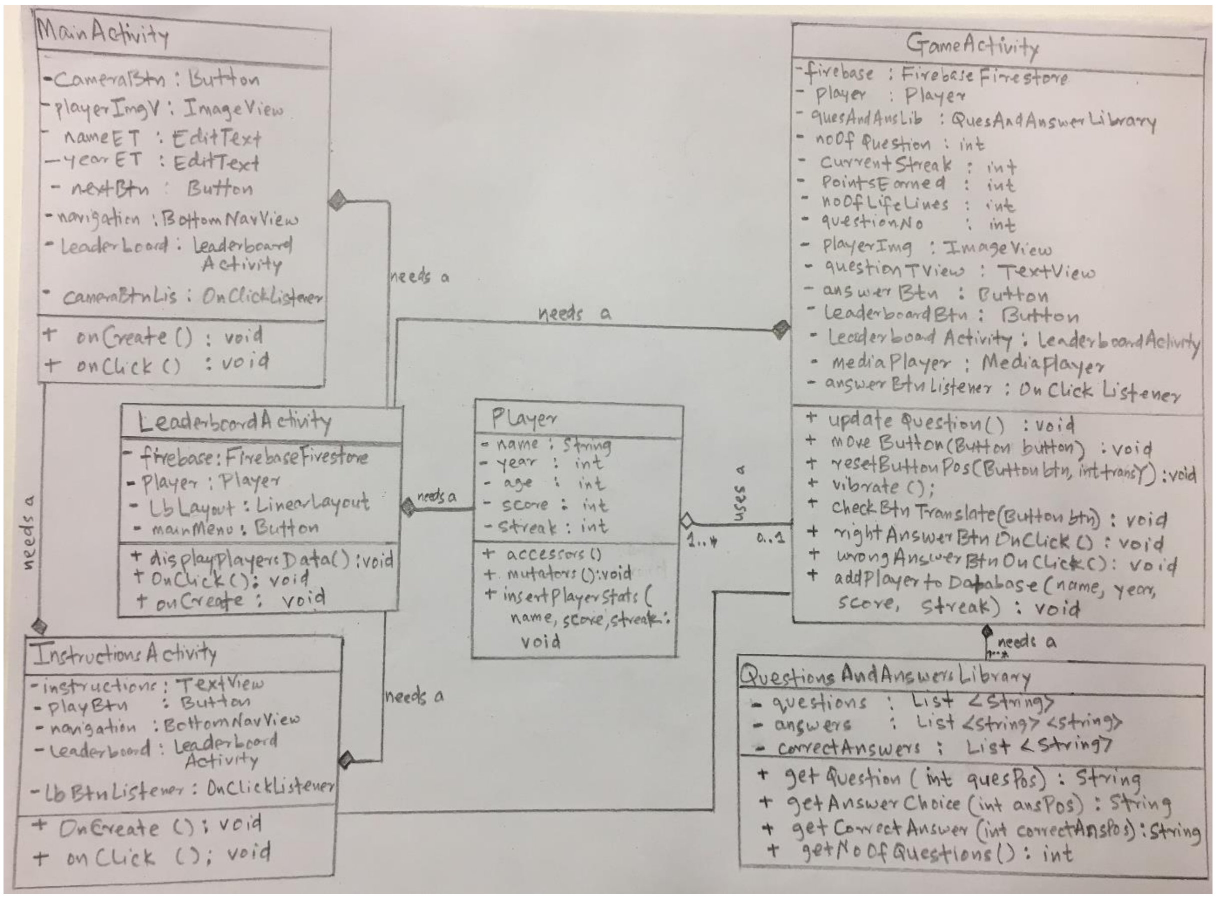
```
firebaseFirestore.collection( collectionPath: "players").add(playerData).
        addOnSuccessListener((OnSuccessListener) (documentReference) → {
                Log.d( tag: "check",  msg: "Document Added with ID: " + documentReference.getId());

        }).addOnFailureListener((e) → {
        Log.w( tag: "check2",  msg: "Error adding document!", e);
});
```

5 SYSTEM AND DRIVEN DESIGNS

The Unified Modelling Languages (UML) diagrams have also been used for modelling the structures, behaviours and interactions of the application.

5.1 CLASS DIAGRAM

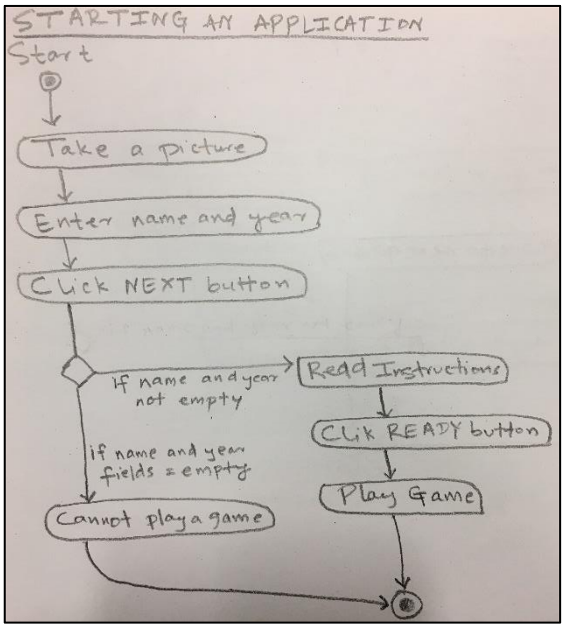
The diagram below is a class diagram which models a whole game application. The classes are linked to each other using various kinds of special relationships including inheritance, composition, aggregation and cardinality.

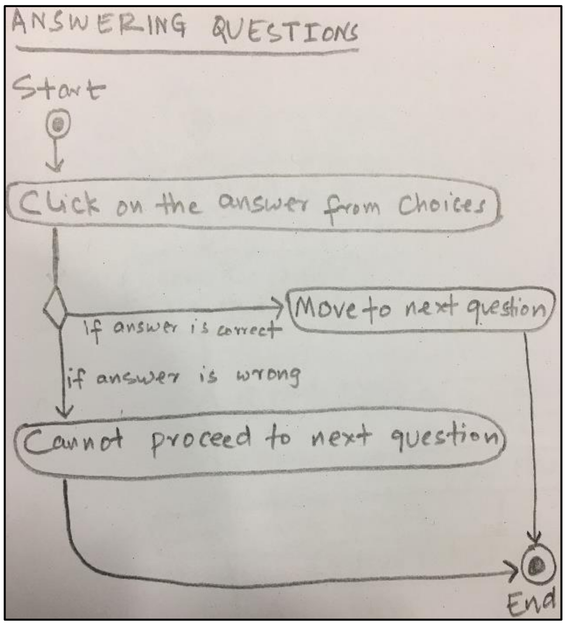


MainActivity
-CameraBtn : Button
-playerImgV : ImageView
- nameET : EditText
-yearET : EditText
- nextBtn : Button
-navigation : BottomNavView
-LeaderBoard : Leaderboard Activity
- cameraBtnLis : OnClickListener
+ onCreate () : void
+ onClick () : void
GameActivity
-firebase : FirebaseFirestore
- player : Player
- quesAndAnsLib : QuesAndAnswerLibrary
- noOf Question : int
- CurrentStreak : int
- PointsEarned : int
- noOfLifeLines : int
- questionNo : int
- playerImg : ImageView
- questionTView : TextView
- answerBtn : Button
- LeaderboardBtn : Button
- Leaderboard Activity : LeaderboardActivity
- mediaPlayer : MediaPlayer
- answerBtnListener : On Click Listener
+ update Question() : void
+ move Button(Button button) : void
+ resetButtonPos(Button btn, int transY) :void
+ vibrate ();
+ checkBtnTranslate(Button btn) : void
+ rightAnswerBtnOnClick () : void
+ wrongAnswerBtnOnClick(): void
+ addPlayer to Database (name, year, score, streak) : void
needs a
needs a
needs a
needs a
needs a
needs a
uses a
1..*
0..1
1..*
LeaderboardActivity
- firebase:FirebaseFirestore
- Player : Player
- LbLayout : LinearLayout
- mainMenu : Button
+ displayPlayersData() :void
+ OnClick(): void
+ onCreate : void
Player
- name : String
- year : int
- age : int
- score : int
- streak : int
+ accessors ()
+ mutators ():void
+ insertPlayerStats (name, score, streak : void
InstructionsActivity
-instructions : TextView
- playBtn : Button
- navigation : BottomNavView
- Leaderboard : Leaderboard Activity
- LbBtnListener : OnClickListener
+ OnCreate (); void
+ onClick (); void
QuestionsAndAnswersLibrary
- questions : List <String>
- answers : List <string> <string>
- correctAnswers : List <String>
+ get Question (int quesPos) : String
+ getAnswerChoice (int ansPos) : String
+ get Correct Answer (int correctAnsPos) : String
+ getNoOfQuestions () : int

5.2 Activity Diagrams

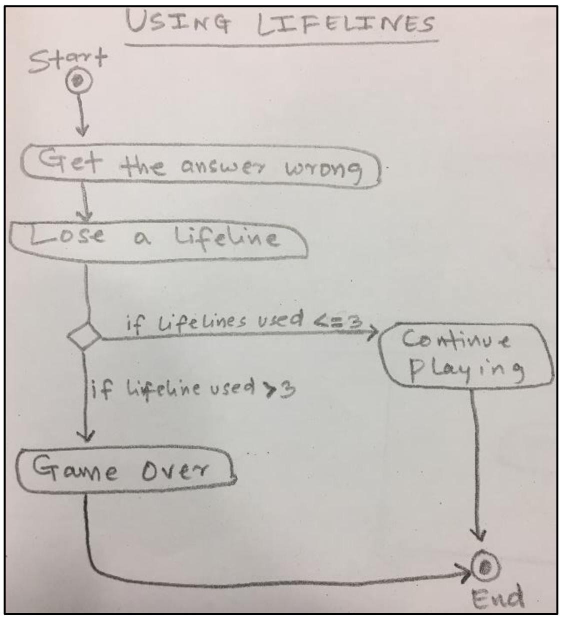
Activity diagrams have also been produced for modelling some of the use cases behaviours of the application. The diagrams below are basically the flow charts that represent the flow from one activity to another for certain use-cases.

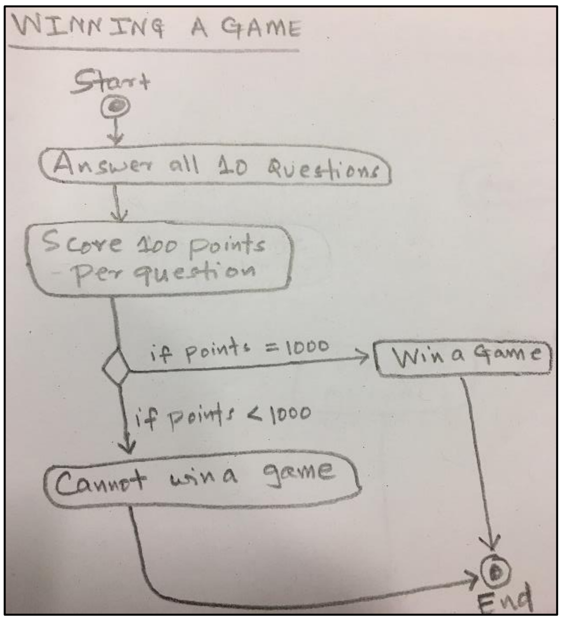




Activity Diagram: Starting an application

Activity Diagram: Answering the questions



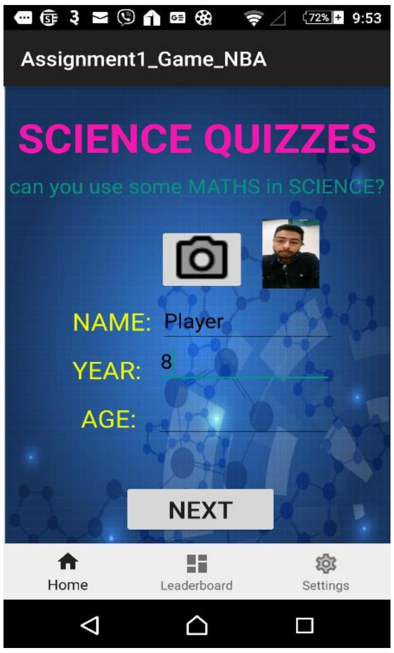


Activity Diagram: Using Lifelines

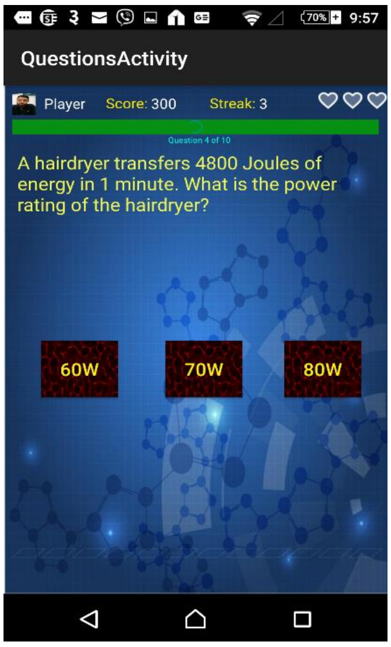
Activity Diagram: Winning a game

6 Functional Testing of Design Solution

The functionalities implemented, most particularly in the controller, based on the action performed by the users all works fine, and the game can be played proficiently (start to finish) with no lagging in between.

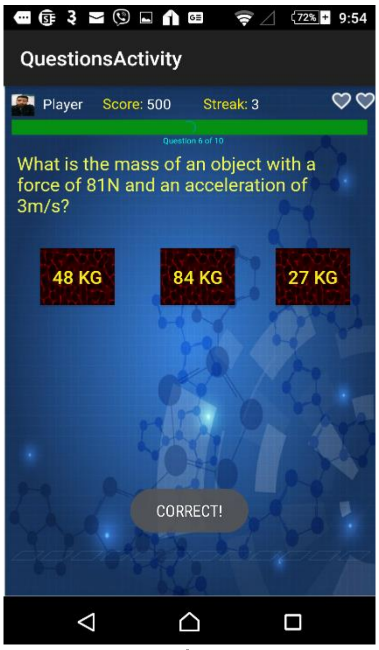


The start-up screen of a game where you can see that the player has successfully taken a picture using a camera.



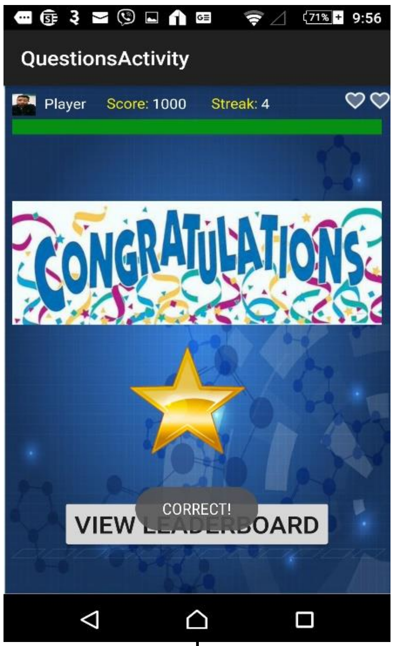
A hairdryer transfers 4800 Joules of energy in 1 minute. What is the power rating of the hairdryer?

The view is updated with the new question and new set of answers and the buttons are now moving down the y-axis.

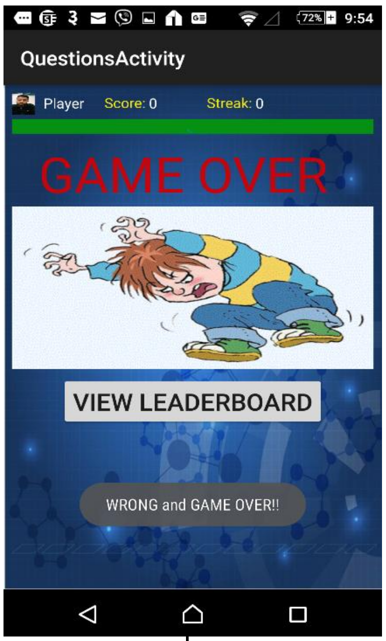


What is the mass of an object with a force of 81N and an acceleration of 3m/s?

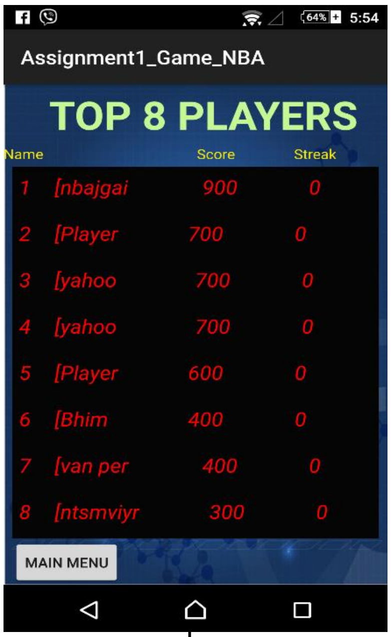
This is when the player clicks on a button that has the right answer. As you can see, the score and streak on the top of the screen also get updated.



As the player has now scored a maximum of 1000 points, the game is now over, and the player is being congratulated. The leaderboard can now be viewed to see if the score achieved is top 8.



All 3 lifelines have been used which means that the player cannot proceed further therefore the game is now over. The leaderboard can now be viewed with a click on a button.



	Name	Score	Streak
1	[nbajgai	900	0
2	[Player	700	0
3	[yahoo	700	0
4	[yahoo	700	0
5	[Player	600	0
6	[Bhim	400	0
7	[van per	400	0
8	[ntsmviyr	300	0

The player can get redirected to the leaderboard view with just one click on a button. This tells us that the firebase has been set up properly and the players data can be populated and queried with no error.

7 COMPARISON AND EVALUATION

In terms of programming language, there is a wide range of OOP language that can be used for android development that are compatible for certain IDEs and Frameworks. When it comes to Android Studio, there are two languages that can be used for the app development, i.e., **JAVA** and **Kotlin**. (SINICKI, 2017) states that JAVA is the official language of Android development supported by Android Studio and on the other hand, Kotlin is recently introduced secondary official language for android development which is very similar to JAVA in many ways and a little easier to get our head around. The difference between JAVA and Kotlin in terms of performance is that JAVA is believed to ensure faster build process than Kotlin and is considered a good choice for cross platform apps. Even though the Kotlin is interoperable with JAVA, the auto-complete and compilation tends to run slower in Kotlin in comparison to JAVA. These are the various reasons for choosing JAVA over Kotlin.

The application data can be saved using either **Firebase** database or **File IO** system. (Sulaiman, 2017) compares the two ways of storing data saying performance in File IO can be better than when you do it in a database and migration of data is also easier and simpler. On the other hand, storing the data in Firebase database is more secure than saving the data in the system. The rationale behind the choice of Firebase over File IO is the level of security and accessibility meaning that database can be accessed at any time, but the file system might not be accessible in the event of loss and accident modification.

8 SELF-REFLECTION ON THE WORK DONE

This module has been the most enjoyable one out of all the modules I have studied so far because I have always wanted to learn about dynamic mobile phones application development. This assessment task has enhanced my experience in the Android apps development in such a way where I can now understand the basic about different types of views, layouts, APIs, animations advanced tools and UI elements offered by Android Studio that can be used to create interactive Android applications. When it comes to the development of a game, I was using Android Studio for the first time, nevertheless, because of a sufficient amount knowledge of JAVA programming gained from the previous years of study, it was easier to understand the syntax and semantics of the coding standard. XML was also easier to understand because it was very similar to HTML. The official documentation of Android Studio has been God-given to me as I could use it as a guide for creating a game.

9 REFERENCES

- What is Android Studio and Android SDK tools? - Android edX Community. (2017). Retrieved from http://androiddeveloper.galileo.edu/2017/03/29/android-studio-and-android-sdk-tools/
- SINICKI, A. (2017). I want to develop Android Apps — What languages should I learn?. Retrieved from https://www.androidauthority.com/develop-android-apps-languages-learn-391008/
- Sulaiman, A. (2017). File System vs. Database - DZone Database. Retrieved from https://dzone.com/articles/which-is-better-saving-files-in-database-or-in-fil