SDK Development

Trivia Quiz Challenge

Student Name: Gianmarco Ortolani
Student Number: 16652
Web Module: 601
Submission Code: SDKD
Class Code:WDHE1012

Table of Contents

1 Introduction

A trivial quiz application for android devices will be developed, according with the characteristics written on the precedent submitted proposal, the intension is to be as faithful as possible to the original idea, but for technical and time reasons, some aspects of the application will be modified, without limiting the good final result. A technical weekly report will be written in this document with the purpose to illustrate all the steps went throughon the development of the quiz. There will be included all the problems found, and how they were resolved, also a final chapter with the conclusion will be written at the end, to give my per sonal ideas of the final result of the application, and some aspects that can be improved the next time that another project of this type will be approached .

2 Project Development

2.1 Week One

Starting with the concept that the application is built using a single Fragment for each layout, the first step was to create a SingleFragmentActivity class (see appendix 5.1), which extends FragmentActivity, whit the intent to manage (add, remove, replace), using the Frag-mentManager API, all the fragments inside the application. An activity_fragnemt-.xml will be created, which will be used as container for all the other XML files (Android Programming: The Big Nerd Ranch Guide, 2013). The next step was to create all the necessary classes to create the required number of layouts for the application (see appendix 5.2). Following the MVC structure, for example QuizQuestions.java has been used as Model, QuizQuestionsActivity.java and QuizQuestionsFragment.java as Controllers and fragment_ quiz_questions.xml as a View. Each activity class, will be extends to the SingleFragment-

Activity, and then, through a protected Fragment createFragment() method, connected to his related Fragment (see appendix 5.3). The Fragments classes, instead, will be extends with the Fragment API, and connected with the own XML file through the LayoutInflater (see appendix 5.4). Then, all the XML files has been created (see appendix 5.5) and drawn, adding the various required styles. It has been used RelativeLayout, using as a reference, a small screen device (480 x 800 hdpi), with the purpose to increase the various dimensions of the objects, adapting them for different resolutions, this will be done later, based on the time left. The screen orientations for the layout as been managed from the onCreate() method, adding the setRequestedOrientation(Activ-ityInfo.SCREEN_ORIENTATION_PORTRAIT), and every pages is called using an onClick methods called from the XML file to the Activity class (see appendix 5.6).

The first week has been used to create the base of the application, and more in particular, assimilating the concept of Fragments and how they need to be used correctly. This has been challenging because of the impossibility to the Fragments to communicate between each others directly, but always just with the own Activity , so just the Activities can parse informations between them and give the informations taken to its own Fragment. That is why the QuizQuestions.java is essential to retrieve data between the different classes.

2.2 Week Two

After have given the style and the classes organization, the next step has been to create a remote database using MySQL. Because the application only need to read data from the database, without the needs to write any informations, the first impression was for an easy task to achieve, but this is not what it was supposed to be. After created the database and all the related tables and rows, the first problem encountered was to connect the database with

the application, db_config.php (see appendix 5.7) with the connections variables, a db_connect.php (see appendix 5.8) for connecting and closing the database and get_questions_answers.php (see appendix 5.9) for read all inquiries from the tables. After that a JSONParser.java (see appendix 5.10) class has been created with the purpose to get Jason from the URL by making HTTP POST and GET methods. The last passage was to create the QuizGetEnquiries.java (see appendix 5.11) where the IP address of the wireless connection was set trough a static variable and create the Arrays where to store and read the information when requested using JSONArrays and JSONObjects. Unfortunately even after managed to read the information from the database to the application and being able to read the informations even from the mobile device, for lack of time it has not been possible to finish the project using this approach. An alternative solution has been found using SQlite database management classes, which give you the possibility to store data in a private database.

2.3 Week Three

The QuizQuestions.java (see appendix 5.12) is the only class used as Model, and contain all the Getter and Setter which connect the variables mId, mQuestion,mAnswer1, mAnswer2, mAnswer3 and mRightAnswer with the other classes. The QuizSQliteDb.java (see appendix 5.13) class has been created, as the name suggest, to manage the SQlite database between the classes. Three static variable manage the version, name and table of the private database, the other static variables call the id, question, the three answers and the right answer, the last variable “dbase” is for the SQLiteDatabase class where all the data will be stored. In onCreate method a table is created inside a String, added to the SQLiteDatabase and then the addQuestions() method is called. The addQuestions() method get the variables from the QuizQuestions.java class for the questions and store the actuals questions and an-

swer. The method getAllQuestions() create an ArrayList looping through all rows and adding them to the database. The last method rowcount() return the numbers of row in the Cursor, which provides random read-write access to the result set returned by a database query.

2.4 Week Forth

The next step has been to create the class QuizQuestionsFragment.java, (see appendix 5.13) it start creating a List which maintains an ordering for the elements inside the QuizQuestion.java class and returned in the variable quesList, to create the score, a static variable with the starting value of 0 as been set. Another variable qid has been set to 0 and the QuizQuestion.java class has been set through the variable curentQ. The other variables left set the widgets for the TextView, RadioButton and RadioGroup. In onCreateView() method, the QuizSqliteDB.java is instantiated in the db variable and inserted in the QuizQuestion.java List where it call the method getAllQuestions() from the QuizSQliteDb.java class and then, parse in the currentQ variable which trough the questList get the qid variable with a 0 value. The setQuestionView() method is called to populate the questions and answers to the view. As We can see in the onChecekChange() method once one of the radio buttons is clicked if is the answer is right, the score is incremented by one and for the following questions all the radio buttons are unchecked to let the user able to respond to another question. There are five questions in total inside the quiz so the if statement said that if the questions are less than five the application have to display again the questions and answers uncheck the radio buttons, else store the score given in the Bundle class trough the b variable and sent to the QuizGame overActivity.java class, this is just an example of how the fragments can not parse informations between them, but the activity have to be the bridge between them.

Inside the QuizGameOverFragment.java (see appendix 5.14) the Bundle class trough

the b variable get the score stored trough the intent from the QuizQuestionsFragment.java and store it in the score variable. A RatingBar has been set with five stars, and based on the number of correct answers, the score will be visualized on it, giving a end result text, depending always on the numbers of answer that has been responded correctly, trough an if statement.

The QuizGameoverActivity.java (see appendix 5.15) have the role, through the re turnMenu() method, to reset the score to 0 and go back to the menu so the user is able to start a new game.

Inside the QuizMenuActivity (see appendix 5.16) there are the method startGame() to start the game and rulesMenu() for display the page with the rules, in this case there will be just a welcome message.

6 Conclusions

A big issue encounter in the development of this project was dealing with a remote database, that is done in part in the development of this project, which it has been possible send data to the application from the database, but that later encountered difficulties to com bine the logics of dealing with fragments with remote database. For this small application use such a database it would have been quite heavy and the approach of using a private database as SQLite probably has been a good choice for the needs and the dimension of this project., but it is something that it will be to improve for the future. A very important achievement, has been working with fragments, which is a fundamental part to understand today, because of the much better options when we consider the potential scalability and maintainability that application can offer. Even if for the actual small dimension of this project, and the applica tion could be programmed using only the activity, it has been a great source of knowledge to

start programming in a better way aiming for a more large and consistent applications in the future.

4 References

- Android Programming: The Big Nerd RAnch Guide, Phillips B., Hardy B., (2013), "*chapter 9*" [Book], Big Nerd Ranch, Inc [2013].

5 Appendix

5.1 SingleFragmentActivity

```
package com.gianmarcoortolani.trivialquizchallenge;

import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentActivity;
import android.support.v4.app.FragmentManager;

public abstract class SingleFragmentActivity extends FragmentActivity {

	protected abstract Fragment createFragment();

	@Override
	public void onCreate( Bundle savedInstanceState ) {

		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_fragment);

		FragmentManager manager = getSupportFragmentManager();
		Fragment fragment = manager.findFragmentById(R.id.fragmentContainer);

		 if (fragment == null) {
		fragment = createFragment();
		manager.beginTransaction()
		   .add(R.id.fragmentContainer, fragment)
		   .commit();
	    }

	}
}
```

5.2 First Classes Created

QuizGameoverActivity.java

QuizGameoverFragment.java

QuizMenuActivity.java

QuizMenuFragment.java

QuizQuestions.java

QuizQuestionsActivity.java

QuizQuestionsFragment.java

QuizRulesActivity.java

QuizRulesFragment.java

5.3 Protected Fragment CreateFragment()

```
public class QuizGameoverActivity extends SingleFragmentActivity {

        protected Fragment createFragment() {

        return new QuizGameoverFragment();
}
```

5.4 Fragments Connection with the XML Files

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
Bundle saveInstanceState) {

        // Inflate the layout for this fragment

        View v = inflater.inflate(R.layout.fragment_quiz_gameover, container, false);

        return v;
}
```

5.5 XML Files

activity_fragment.xml

fragment_quiz_gameover.xml

fragment_quiz_menu.xml

fragment_quiz_questions.xml

fragment_quiz_rules.xml

5.6 Calling pages

```
public void startGame (View view) {

        Intent intent = new Intent (view.getContext(), QuizStartActivity.class);

        startActivity(intent);

}
```

5.7 Db_config.php

```
<?php

/*
 * All database connection variables
 */

define('DB_USER', "root"); // db user
define('DB_PASSWORD', ""); // db password
define('DB_DATABASE', "quiz_connect"); // database name
define('DB_SERVER', "localhost"); // db server
?>
```

5.8 Db_connect.php

```
<?php
/**
 * A class file to connect to database
 */
class DB_CONNECT {

    // constructor
    function __construct() {
        // connecting to database
        $this->connect();
    }

    // destructor
    function __destruct() {
        // closing db connection
        $this->close();
    }

    /**
     * Function to connect with database
     */
    function connect() {
        // import database connection variables
        require_once __DIR__ . '/db_config.php';

        // Connecting to mysql database
        $con = mysql_connect(DB_SERVER, DB_USER, DB_PASSWORD) or die(mysql_er-
ror());

        // Selecing database
        $db = mysql_select_db(DB_DATABASE) or die(mysql_error()) or die(mysql_error());

        // returing connection cursor
        return $con;
    }
    /**
     * Function to close db connection
     */
    function close() {
        // closing db connection
        mysql_close();
    }
}

?>
```

5.9 Get_questions_answers.php

```
<?php
// array for JSON response
$response = array();
// include db connect class
require_once __DIR__ . '/db_connect.php';
// connecting to db
$db = new DB_CONNECT();
// get all inquiries from products table
$result = mysql_query("SELECT *FROM table_quiz") or die(mysql_error());
// check for empty result
if (mysql_num_rows($result) > 0) {
    // looping through all results
    // table node
    $response["table_quiz"] = array();

    while ($row = mysql_fetch_array($result)) {
        // temp user array
        $table = array();
        $table["id"] = $row["id"];
        $table["question"] = $row["question"];
        $table["answer1"] = $row["answer1"];
        $table["answer2"] = $row["answer2"];
        $table["answer3"] = $row["answer3"];
        $table["answer4"] = $row["answer4"];
        $table["right_answer"] = $row["right_answer"];
        // push single enquiry into final response array
        array_push($response["table_quiz"], $table);
    }
    // success
    $response["success"] = 1;


     // echoing JSON response
    echo json_encode($response);
} else {
    // no enquiry found
    $response["success"] = 0;
    $response["message"] = "No products found";


     // echo no users JSON
    echo json_encode($response);
}
?>
```

5.10 JSONParser.java

```
public class JSONParser {

    static InputStream is = null;
    static JSONObject jObj = null;
    static String json = “”;

    // constructor
    public JSONParser() {

    }

    // function get json from url
    // by making HTTP POST or GET mehtod
    public JSONObject makeHttpRequest(String url, String method,
            List<NameValuePair> params) {

        // Making HTTP request
        try {

            // check for request method
            if(method == “POST”){
                // request method is POST
                // defaultHttpClient
                DefaultHttpClient httpClient = new DefaultHttpClient();
                HttpPost httpPost = new HttpPost(url);
                httpPost.setEntity(new UrlEncodedFormEntity(params));

                HttpResponse httpResponse = httpClient.execute(httpPost);
                HttpEntity httpEntity = httpResponse.getEntity();
                is = httpEntity.getContent();

            }else if(method == “GET”){
                // request method is GET
                DefaultHttpClient httpClient = new DefaultHttpClient();
                String paramString = URLEncodedUtils.format(params, “utf-8”);
                url += “?” + paramString;
                HttpGet httpGet = new HttpGet(url);

                HttpResponse httpResponse = httpClient.execute(httpGet);
                HttpEntity httpEntity = httpResponse.getEntity();
                is = httpEntity.getContent();
            }

        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
```

```
} catch (ClientProtocolException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }

    try {
        BufferedReader reader = new BufferedReader(new InputStreamReader(
                is, "iso-8859-1"), 8);
        StringBuilder sb = new StringBuilder();
        String line = null;
        while ((line = reader.readLine()) != null) {
            sb.append(line + "\n");
        }
        is.close();
        json = sb.toString();
    } catch (Exception e) {
        Log.e("Buffer Error", "Error converting result " + e.toString());
    }

    // try parse the string to a JSON object
    try {
        jObj = new JSONObject(json);
    } catch (JSONException e) {
        Log.e("JSON Parser", "Error parsing data " + e.toString());
    }

    // return JSON String
    return jObj;

  }
}
```

5.11 QuizGetInquiries.java

```
public class QuizGetInquiries extends Activity {

   // Progress Dialog
   private ProgressDialog pDialog;

   // Creating JSON Parser object
   JSONParser jParser = new JSONParser();

   ArrayList<HashMap<String, String>> inquriesList;

   // url to get all products list
   private static String url_all_inquiries = “http://192.168.1.7/quiz_connect/get_questions_an-
swers.php”;

   // JSON Node names
   private static final String TAG_SUCCESS = “success”;
   private static final String TAG_TABLE = “table_quiz”;
   private static final String TAG_QUESTIONS = “question”;
   private static final String TAG_ANSWERS = “radioGroup1”;

   // products JSONArray
   JSONArray inquiries = null;

   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.fragment_quiz_questions);

      // Hashmap for ListView
      inquriesList = new ArrayList<HashMap<String, String>>();

      // Loading products in Background Thread
      new LoadAllInquiries().execute();

   }

   class LoadAllInquiries extends AsyncTask<String, String, String> {

      @Override
      protected void onPreExecute() {
         super.onPreExecute();
         pDialog = new ProgressDialog(QuizGetInquiries.this);
         pDialog.setMessage(“Loading quiz. Please wait...”);
         pDialog.setIndeterminate(false);
         pDialog.setCancelable(false);                }
```

```
pDialog.show();
        }

        protected String doInBackground(String... args) {
            // Building Parameters
            List<NameValuePair> params = new ArrayList<NameValuePair>();
            // getting JSON string from URL
            JSONObject json = jParser.makeHttpRequest(url_all_inquiries, “GET”, params);

            // Check your log cat for JSON reponse
            Log.d(“All Inquiries: “, json.toString());

            try {
                // Checking for SUCCESS TAG
                int success = json.getInt(TAG_SUCCESS);

                if (success == 1) {
                    // Getting Array of inquiries
                        inquiries = json.getJSONArray(TAG_TABLE);

                    // looping through All inquiries
                    for (int i = 0; i < inquiries.length(); i++) {
                        JSONObject c = inquiries.getJSONObject(i);

                        // Storing each json item in variable
                        String questions = c.getString(TAG_QUESTIONS);
                        String answers = c.getString(TAG_ANSWERS);

                        // creating new HashMap
                        HashMap<String, String> map = new HashMap<String, String>();

                        // adding each child node to HashMap key => value
                        map.put(TAG_QUESTIONS, questions);
                        map.put(TAG_ANSWERS, answers);

                        // adding HashList to ArrayList
                        inquriesList.add(map);
                    }
                }
            } catch (JSONException e) {
                e.printStackTrace();
            }

            return null;
        }
    }
}
```

5.12 QuizQuestions.java

```
public class QuizQuestions {

	private int mId;
	private String mQuestion;
	private String mAnswer1;
	private String mAnswer2;
	private String mAnswer3;
	private String mRightAnswer;

	public QuizQuestions(String question, String answer1, String answer2, String an
		swer3,String rightAnswer) {

		mQuestion = question;
		mAnswer1 = answer1;
		mAnswer2 = answer2;
		mAnswer3 = answer3;
		mRightAnswer = rightAnswer;
	}

	public QuizQuestions() {

		mId=0;
		mQuestion = “”;
		mAnswer1 = “”;
		mAnswer2 = “”;
		mAnswer3 = “”;
		mRightAnswer = “”;
	}

	public int getID() {
		return mId;
	}

	public void setID(int id) {
		mId = id;
	}

	public String getQuestion() {
		return mQuestion;
	}
```

```
public void setQuestion(String question) {
		mQuestion = question;
	}

	public String getAnswer1() {
		return mAnswer1;
	}

	public void setAnswer1(String answer1) {
		mAnswer1 = answer1;
	}

	public String getAnswer2() {
		return mAnswer2;
	}

	public void setAnswer2(String answer2) {
		mAnswer2 = answer2;
	}

	public String getAnswer3() {
		return mAnswer3;
	}

	public void setAnswer3(String answer3) {
		mAnswer3 = answer3;
	}

	public String getRightAnswer() {
		return mRightAnswer;
	}

	public void setRightAnswer(String rightAnswer)  {
		mRightAnswer = rightAnswer;
	}
}
```

5.13 QuizSQliteDb.java

```
public class QuizSQliteDB extends SQLiteOpenHelper {

        private static final int DATABASE_VERSION = 2;
        // Database Name
        private static final String DATABASE_NAME = “mTriviaQuiz”;
        // tasks table name
        private static final String TABLE_QUEST = “mQuest”;
        // tasks Table Columns names
        private static final String KEY_ID = “mId”;
        private static final String KEY_QUES = “mQuestion”;
        //correct option
        private static final String KEY_RIGHT_ANSWER = “mRightAnswer”;
        //option a
        private static final String KEY_ANSWER1= “mAswer1”;
        //option b
        private static final String KEY_ANSWER2= “mAswer2”;
        //option c
        private static final String KEY_ANSWER3= “mAswer3”;
        private SQLiteDatabase dbase;

        public QuizSQliteDB(Context context) {

                super(context, DATABASE_NAME, null, DATABASE_VERSION);
        }

        @Override
        public void onCreate(SQLiteDatabase db) {

                dbase=db;

                String sql = “CREATE TABLE IF NOT EXISTS “ + TABLE_QUEST + “ ( “
                                + KEY_ID + “ INTEGER PRIMARY KEY AUTOINCRE-
MENT,                           “ + KEY_QUES + “ TEXT, “ + KEY_RIGHT_ANSWER+ “
TEXT, “+ KEY_                           ANSWER1 +” TEXT, “+ KEY_ANSWER2 +” TEXT,
“+ KEY_ANSWER3 +”                               TEXT)”;

                db.execSQL(sql);

                addQuestions();
        }
```

```
private void addQuestions() {

        QuizQuestions question1 = new QuizQuestions("What type of creature" +
        " is a flickertail ?", "Dragonfly", "Monkey", "Squirrel", "Squirrel");
        addQuestion(question1);
        QuizQuestions question2 = new QuizQuestions("Who was south Africa's" +
        " first black president ?", "Nelson Mandela", "Jhon Gary", "Wiliam Burton",
                "Nelson Mandela");
        addQuestion(question2);
        QuizQuestions question3 = new QuizQuestions("Which colours" +
        " make purple ?","Red and Green", "Green and Orange", "Red and Blue",
"Red                and Blue");
        addQuestion(question3);
        QuizQuestions question4 = new QuizQuestions("What name is given to the
        Buddhist practice" +
        " of mental concentration?", "Moderation", "Modification", "Meditation",
"Medi               tation");
        addQuestion(question4);
        QuizQuestions question5 = new QuizQuestions("What score must a Ten-Pin
        bowler make" +
        " to achieve a perfect game?", "200", "300", "150", "300");
        addQuestion(question5);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldV, int newV) {

        // Drop older table if existed
        db.execSQL("DROP TABLE IF EXISTS " + TABLE_QUEST);

        // Create tables again
        onCreate(db);
    }

    // Adding new question
    public void addQuestion(QuizQuestions quest) {

        ContentValues values = new ContentValues();
        values.put(KEY_QUES, quest.getQuestion());
        values.put(KEY_RIGHT_ANSWER, quest.getRightAnswer());
        values.put(KEY_ANSWER1, quest.getAnswer1());
        values.put(KEY_ANSWER2, quest.getAnswer2());
        values.put(KEY_ANSWER3, quest.getAnswer3());
        // Inserting Row
        dbase.insert(TABLE_QUEST, null, values);
    }
```

```
public List<QuizQuestions> getAllQuestions() {

        List<QuizQuestions> quesList = new ArrayList<QuizQuestions>();

        // Select All Query
        String selectQuery = "SELECT  * FROM " + TABLE_QUEST;
        dbase = getReadableDatabase();
        Cursor cursor = dbase.rawQuery(selectQuery, null);

        // looping through all rows and adding to list
        if (cursor.moveToFirst()) {
            do {
                QuizQuestions quest = new QuizQuestions();

                quest.setID(cursor.getInt(0));

                quest.setQuestion(cursor.getString(1));

                quest.setRightAnswer(cursor.getString(2));

                quest.setAnswer1(cursor.getString(3));

                quest.setAnswer2(cursor.getString(4));

                quest.setAnswer3(cursor.getString(5));

                quesList.add(quest);

            } while (cursor.moveToNext());
        }

        // return quest list
        return quesList;
    }

    public int rowcount() {

        int row=0;

        String selectQuery = "SELECT  * FROM " + TABLE_QUEST;

        SQLiteDatabase db = getWritableDatabase();
        Cursor cursor = db.rawQuery(selectQuery, null);
        row=cursor.getCount();

        return row;
    }
}
```

5.13 QuizQuestionsFragment.java

```
public class QuizQuestionsFragment extends Fragment {

        List<QuizQuestions> quesList;
        public static int score=0;
        int qid=0;
        QuizQuestions currentQ;
        TextView txtQuestion;
        RadioButton mAnswer1, mAnswer2, mAnswer3;
        RadioGroup grp;

        @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        }

        @Override
        public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
                savedInstanceState) {
                // Inflate the layout for this fragment
                final View v = inflater.inflate(R.layout.fragment_quiz_questions, container,
                false);
                QuizSQliteDB db = new QuizSQliteDB(getActivity());
                quesList=db.getAllQuestions();
                currentQ=quesList.get(qid);
                txtQuestion=(TextView)v.findViewById(R.id.questions_text);
                mAnswer1 = (RadioButton)v.findViewById(R.id.answer1);
                mAnswer2 = (RadioButton)v.findViewById(R.id.answer2);
                mAnswer3 = (RadioButton)v.findViewById(R.id.answer3);
                setQuestionView();
                grp = (RadioGroup)v.findViewById(R.id.radioGroup1);
                grp.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeLis
                tener() {

              @Override
              public void onCheckedChanged(RadioGroup arg0, int arg1) {
                 // TODO Auto-generated method stub

                 RadioButton answer=(RadioButton)v.findViewById(grp.getCheckedRa
                 dioButtonId());

                        if(currentQ.getRightAnswer().equals(answer.getText())) {
                                //add score
                                        score++;
```

```
// uncheck radial button
                                        mAnswer1.setChecked(false);
                                        mAnswer2.setChecked(false);
                                        mAnswer3.setChecked(false);
                                    }
                // check if the number of question is lower then 5
                                    if(qid<5){
                                        currentQ=quesList.get(qid);
                                        // set question
                                        setQuestionView();

                                        mAnswer1.setChecked(false);
                                        mAnswer2.setChecked(false);
                                        mAnswer3.setChecked(false);

                                    }else{

                                        // sent intent to QuizGameoverActivity for the
                                        score
                                        Intent intent = new Intent();
                                        intent.setClass(getActivity(), QuizGameoverAc
                                        tivity.class);
                                        Bundle b = new Bundle();

                                        //Your score
                                        b.putInt("score", score);

                                        //Put your score to your next Intent
                                        intent.putExtras(b);
                                        startActivity(intent);
                                    }
                            }

                        });

                return v;
                    }

                    private void setQuestionView() {

                            // set the answers on the view
                            txtQuestion.setText(currentQ.getQuestion());
                            mAnswer1.setText(currentQ.getAnswer1());
                            mAnswer2.setText(currentQ.getAnswer2());
                            mAnswer3.setText(currentQ.getAnswer3());
                            qid++;
                    }
}
```

5.14 QuizGameoverFragment.java

```
public class QuizGameoverFragment extends Fragment {

	@Override
	public void onCreate(Bundle savedInstanceState) {

		super.onCreate(savedInstanceState);
	}

	@Override
	public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
saveInstanceState) {
		// Inflate the layout for this fragment
		View v = inflater.inflate(R.layout.fragment_quiz_gameover, container, false);

		RatingBar bar=(RatingBar)v.findViewById(R.id.ratingBar1);
		bar.setNumStars(5);
		bar.setStepSize(0.5f);

		//get text view
		TextView t = (TextView)v.findViewById(R.id.textResult);

		//get score
		Bundle b = getActivity().getIntent().getExtras();
		int score= b.getInt("score");

		//display score
		bar.setRating(score);

		switch (score) {
			case 1:
			case 2: t.setText("Is it a jocke???");
			break;

			case 3:
			case 4:t.setText("come on....you can do it...");
			break;

			case 5:t.setText("Wow you are a genius.....");
			break;

		}
		return v;
	}
}
```

5.15 QuizGameoverActivity.java

```
public class QuizGameoverActivity extends SingleFragmentActivity {

        @Override
    protected Fragment createFragment() {
        return new QuizGameoverFragment();
        }

    @Override
    public void onCreate( Bundle savedInstanceState ) {

        super.onCreate(savedInstanceState);

        setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
    }

    public void returnMenu (View view) {

        QuizQuestionsFragment.score = 0;
        Intent intent = new Intent (view.getContext(), QuizMenuActivity.class);
        startActivity(intent);

        }
}
```

5.15 QuizMenuActivity.java

```
public class QuizMenuActivity extends SingleFragmentActivity {

        @Override
   protected Fragment createFragment() {
      return new QuizMenuFragment();
        }


        @Override
   public void onCreate( Bundle savedInstanceState ) {

                super.onCreate(savedInstanceState);

                setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_POR-
TRAIT);
   }

        public void startGame (View view) {

        Intent intent = new Intent (view.getContext(), QuizQuestionsActivity.class);
        startActivity(intent);
        }

        public void rulesMenu (View view) {

                Intent intent = new Intent (view.getContext(), QuizRulesActivity.class);
                startActivity(intent);
        }
}
```