|  |  |  |
| --- | --- | --- |
|  | **Faculty of Computing, Engineering and Science** | Final mark awarded:\_\_\_\_\_ |

**Assessment Cover Sheet and Feedback Form 2017-18**

|  |  |  |  |
| --- | --- | --- | --- |
| Module Code: SE4S701 | Module Title:  Mobile Application Development | | Module Lecturer:  Abdulkareem Karasuwa, Berndt Muller, Peter Plassmann, Ian Wilson |
| Assessment Title:  A Learning App on Android | | | Assessment No.  1 |
| No. of pages submitted in total including this page:  **38** | | | Word Count of submission  (if applicable): **5089** |
| Date Set:  29-Jan-2018 09:00 | | Submission Date:  20-Apr-2018 23:59 | Return Date:  18-May-2018 23:59 |

|  |  |
| --- | --- |
| ***Part A: Record of Submission (to be completed by Student)*** | |
| **Extenuating Circumstances**  If there are any exceptional circumstances that may have affected your ability to undertake or submit this assignment, make sure you contact the Advice Centre on your campus prior to your submission deadline. | |
| **Fit to sit policy**:  The University operates a fit to sit policy whereby you, in submitting or presenting yourself for an assessment, are declaring that you are fit to sit the assessment. You cannot subsequently claim that your performance in this assessment was affected by extenuating factors. | |
| **Plagiarism and Unfair Practice Declaration:**  By submitting this assessment, you declare that it is your own work and that the sources of information and material you have used (including the internet) have been fully identified and properly acknowledged as required[[1]](#footnote-1). Additionally, the work presented has not been submitted for any other assessment. You also understand that the Faculty reserves the right to investigate allegations of plagiarism or unfair practice which, if proven, could result in a fail in this assessment and may affect your progress. | |
| **Intellectual Property and Retention of Student Work:**  You understand that the University will retain a copy of any assessments submitted electronically for evidence and quality assurance purposes; requests for the removal of assessments will only be considered if the work contains information that is either politically and/or commercially sensitive (as determined by the University) and where requests are made by the relevant module leader or dissertation supervisor. | |
| **Details of Submission:**  Note that all work handed in after the submission date and within 5 working days will be capped at 40%[[2]](#footnote-2). No marks will be awarded if the assessment is submitted after the late submission date unless extenuating circumstances are applied for and accepted (Advice Centre to be consulted). | |
| You are required to acknowledge that you have read the above statements by writing your student number(s) in the box: | Student Number(s):  **14031329** |

**IT IS YOUR RESPONSIBILITY TO KEEP RECORDS OF ALL WORK SUBMITTED**

|  |
| --- |
| **Part B: Marking and Assessment**  **(to be completed by Module Lecturer)** |
| This assignment will be marked out of 100%  This assignment contributes to 50% of the total module marks.  This assignment is bonded |
| **Learning Outcomes to be assessed** (as specified in the validated module descriptor <https://icis.southwales.ac.uk/> ):  *1) Understand and appreciate the challenges and opportunities in developing mobile applications. 2) Develop skills to design and program web applications on mobile devices. 3) Understand the different programming skills needed for developing desktop and mobile platforms. 4) Develop a capability to create mobile applications on a specific mobile platform.* |

|  |  |  |
| --- | --- | --- |
| **Feedback/feed-forward** (linked to assessment criteria):   * Areas where you have done well: * Feedback from this assessment to help you to improve future assessments: * Other comments | | |
| **Mark:** | **Marker’s Signature:** | **Date:** |
| * **Work on this module has been marked, double marked/moderated in line with USW procedures.** | | |
| *Provisional mark only: subject to change and/or confirmation by the Assessment Board* | | |

|  |  |
| --- | --- |
| **Part C: Reflections on Assessment**  **(to be completed by student – optional)** | |
| **Use of previous feedback:**  In this assessment, I have taken/took note of the following points in feedback on previous work: | |
| **Please indicate which of the following you feel/felt applies/applied to your submitted work**   * A reasonable attempt. I could have developed some of the   sections further.   * A good attempt, displaying my understanding and learning, with   analysis in some parts.   * A very good attempt. The work demonstrates my clear   understanding of the learning supported by relevant literature and  scholarly work with good analysis and evaluation.   * An excellent attempt, with clear application of literature and   scholarly work, demonstrating significant analysis and evaluation. | |
| **What I found most difficult about this assessment:** |  |
| **The areas where I would value/would have valued feedback:** |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Fail | Narrow Fail | 3rd Class / Pass | Lower 2nd Class / Pass | Upper 2nd Class / Merit | 1st Class / Distinction |
| Design - User Friendliness 15% | * The app is impossible or almost impossible to use * No Layout or default layout used. Components poorly aligned | * The app is very difficult to use and requires a lot of experimentation to access the functionalities implemented * An attempt was made to use layouts but on rotation the arrangement of the App's components is seriously flawed | * The app can be used after some experimentation but may have some design inconsistencies and unusual ways of using it * Layouts used that rearrange the App on device rotation with some obvious flaws. Component alignment flawed but just about usable | * The app is easy to use but leaves the user in doubt about functionality at times. Could have some design inconsistencies * Appropriate layouts used that rearrange the App on device rotation with some unobtrusive flaws. Reasonably accurate component alignment | * The app is straightforward to use, and implements most Android design principles without design inconsistencies * Appropriate layouts used that rearrange the App on device rotation with no or only hard to spot minor flaws. Very good component alignment | * The app is intuitively designed, straightforward to use, adheres to all Android design principles and has a professional 'look and feel' * Appropriate nested layouts used that perfectly rearrange the App on device rotation. Perfect component alignment |
| Report 10% | * No description of the programming problems to solve * No discussion of possible solutions (advantages / disadvantages * No decision on design choices apparent | * A very rudimentary description of the programming problems to solve * Very basic discussion of possible solutions (advantages / disadvantages), no or very little background research apparent (references) * Few and unsubstantiated design decisions made | * A good description of the programming problems to solve with minimal insight into the problems to be solved * A reasonable discussion of possible solutions (advantages / disadvantages) with some fundamental background research (references). Generic and not about THIS project at times * Most design decisions are based on previous discussion | * A very good description of the programming problems to solve providing good insight into the problems to be solved * A well-argued discussion of possible solutions (advantages / disadvantages) a good amount of background research (references). Mainly non-generic and about THIS project * Design decisions very clear and supported by an engaged discussion | * An excellent description of the programming problems to solve providing deep insights into the problems to be solved * A very well argued and deep discussion of possible solutions (advantages / disadvantages) a large amount of background research (including non www difficult references). Almost entirely non-generic and about THIS project * Well-argued and fact supported informed design decisions supported by good engagement with source reference material | * A very professional description of the programming problems to solve providing deep insight into the problems to be solved in a succinct professional manner * An excellently argued and deep discussion of possible solutions (advantages / disadvantages) with a large amount of relevant background research (including several non www difficult references). Entirely non-generic and about THIS project * Professionally argued and fact supported informed design decisions supported by excellent and deep engagement with source reference material |
| Functionality 9 functions splash screen, 5 topics present, quiz with 5 question on each topic, questions include graphics, questions asked randomly, question bank of 10 questions present, scorecard created, scorecard saved, scores supported by graphics/ date/time 35% | * Does not launch or less than 3 functions are working to some extend | * Less than half of the functions working correctly | * Approximately half of the functions are working correctly (e.g. 3 fully and another 3 with half functionality) | * Approximately 2/3 of the functions are working correctly (e.g. 4 fully and another 4 with half functionality) | * Almost all of the 9 functions are working correctly (e.g. 7 fully and the remaining 2 with half functionality) | * All of the 9 functions are working correctly. Some additional user guidance and/or features add a 'professional touch' |
| Completeness all 9 functions (see above) attempted to a more than trivial degree (even if they may not work 100%) 20% | * Only one or 2 functions attempted | * Only 3 or 4 functions attempted | * 5 or 6 functions attempted | * 7 or 8 functions attempted | * All functions attempted | * All functions attempted. Some additional user guidance and/or features/functions add a 'professional touch' |
| Testing 10% | * Testing lacks any valid meaning or was not done at all * No test plan | * Testing is poor and lacks proper meaning | * About half of the functions were tested * Tests were superficial * A rudimentary test plan was present | * Approximately 2/3 of the functions were tested * Tests covered approximately half of the possibilities to a good degree * A good test plan and some structuring of tests attempted | * Almost all of the functions were tested * Most possibilities were tested to a very good degree * A well thought out test plan that is well structured | * All functions were tested * All possibilities tested exhaustively in several test categories * A professional test plan with excellent structuring and layout |
| Quality of Code 10% | * No commenting * Code layout all over the place / random * Haphazard code that only just about works but relies on many assumptions that may not be true * No error checking (e.g. user input errors, file I/O errors) | * Poor commenting * Very little and incomplete/incorrect error checking * Code layout reasonable in places but largely inconsistent | * Some meaningful commenting * Some error checking (e.g. user input errors, file I/O errors) * Mostly reasonable code layout | * Good commenting * Code layout (i.e. indentation, spacing) correct in most places * Good error checking (e.g. user input errors, file I/O errors) | * Very good commenting * Almost complete error checking (e.g. user input errors, file I/O errors) | * Excellent commenting, only where really required and code not self-explanatory * Perfect code layout perfect and without flaws * Professional error checking (e.g. user input errors, file I/O errors, boundary errors in conditionals, array/capacitor errors, etc) |
| Self-assessment is a requirement of the coursework, you will not receive a mark until this process is complete. | | | | | | |

***[*This page was intentionally left blank*]***

Table of contents

[2. Overview 3](#_Toc512031627)

[3. Quiz app structure 4](#_Toc512031628)

[3.1. Project file structure 4](#_Toc512031629)

[3.1. High level structure and execution flow (flowchart) 8](#_Toc512031630)

[4. Quiz content choice 11](#_Toc512031631)

[5. Fundamental programming problems and solutions 12](#_Toc512031632)

[6. Testing methodologies 14](#_Toc512031633)

[Appendices 1](#_Toc512031634)

[Appendix A: 1](#_Toc512031635)

[Appendix B: 1](#_Toc512031636)

List of tables

[Table 3‑1 – Detailed project file structure 6](#_Toc512031624)

[Table 3‑2 – User score object structure – quiz\_userdata.json 10](#_Toc512031625)

[Table 4‑1 – List of questions chosen 11](#_Toc512031626)

List of figures

[Figure 3‑1 - Project file structure: top level 4](#_Toc512031616)

[Figure 3‑2 - Project file structure: inside directory 'app' 4](#_Toc512031617)

[Figure 3‑3 - Project file structure: inside directory 'app/src' 4](#_Toc512031618)

[Figure 3‑4 - Project file structure: inside directory 'app/src/main' 4](#_Toc512031619)

[Figure 3‑5 - Android Studio IDE: top level structure 5](#_Toc512031620)

[Figure 3‑6 - Android Studio IDE: complete file structure 5](#_Toc512031621)

[Figure 3‑7 - Flowchart: Program execution flow 8](file:///E:\Users\Miguel\Dropbox2\Dropbox\My-Uni-Assignments\Year%204\SE4S701%20-%20Mobile%20Application%20Development\SE4S701_CW1_14031329_Miguel_Santos.docx#_Toc512031622)

[Figure 3‑8 - Quiz databank structure – quiz\_config.json 9](#_Toc512031623)

# Overview

In the ambit of the module SE4S701 – Mobile Application Development, this report will cover the only assignment regarding an Android application required to be developed by the students. This application consists of a teaching quiz which contains questions specifically about the Android platform, which may or may not encompass the Java programming language.

In this quiz we may find 5 topics/categories, each of which has 5 questions with images including. These questions are to be picked randomly from a databank of 10 questions. The results (plus date and time) must be permanently stored in order to consequently display them in a score card list and in a graph.

In regards to the structure of the report, the first section will cover the structure of the Android project in three different levels of detail, from the highest level (summarised) to the lowest, most detailed level using a flowchart.

Following this is a section that discusses the motives for the choice of the content of the application. Specifically, the reason why these quiz questions in particular were picked.

Next, the fundamental issues and/or topics in programming are to be discussed, with perspective to the contents being taught in the lectures.

Finally, this report will conclude on the assignment by performing a set of unit tests in order to determine the robustness of the application being developed.

# Quiz app structure

## Project file structure

The application project follows a very common structure found in any project generated by the Android Studio IDE.

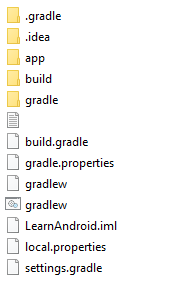


Figure 3‑1 - Project file structure: top level

The Java code, built objects and Gradle scripts are found on the ‘app’ directory. Inside, we may find the following contents.

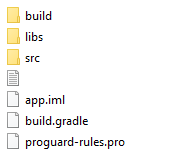


Figure 3‑2 - Project file structure: inside directory 'app'

Again, the desired content is to be found inside the directory ‘src’.

Figure 3‑3 - Project file structure: inside directory 'app/src'

This contains:

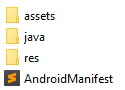


Figure 3‑4 - Project file structure: inside directory 'app/src/main'

In fact, this is the main project structure found on Android Studio on the sidebar.

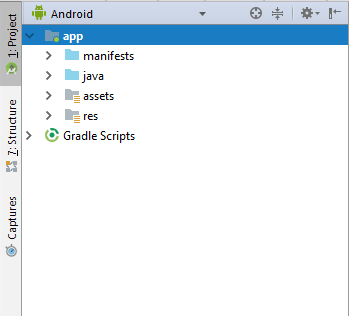


Figure 3‑5 - Android Studio IDE: top level structure

Finally, the code and the layouts for the Quiz application may be found inside the folders ‘java’ and ‘res’.

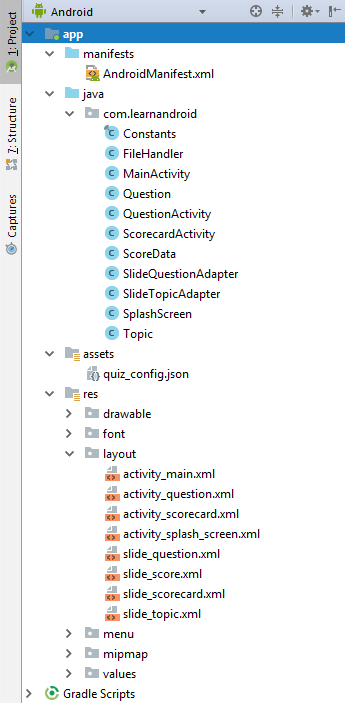


Figure 3‑6 - Android Studio IDE: complete file structure

This brings attention to the following files:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Java file | Role | Layout File(s) | |
| 1 | Constants.java | Global constant values | - | |
| 2 | FileHandler.java | Util library | - | |
| 3 | MainActivity.java | Activity | activity\_main.xml | |
| 4 | Question.java | Data class | - | |
| 5 | QuestionActivity.java | Activity | activity\_question.xml | |
| 6 | ScorecardActivity.java | Activity | activity\_scorecard.xml | slide\_scorecard.xml |
| 7 | ScoreData.java | Data class | - | |
| 8 | SlideTopicAdapter.java | View controller | slide\_topic.xml | |
| 9 | SlideQuestionAdapter.java | View controller | slide\_question.xml | slide\_score.xml |
| 10 | SplashScreen.java | Activity | activity\_splash\_screen.xml | |
| 11 | Topic.java | Data class | - | |
| 12 | quiz\_config.json | Configuration file | - | |

Table 3‑1 – Detailed project file structure

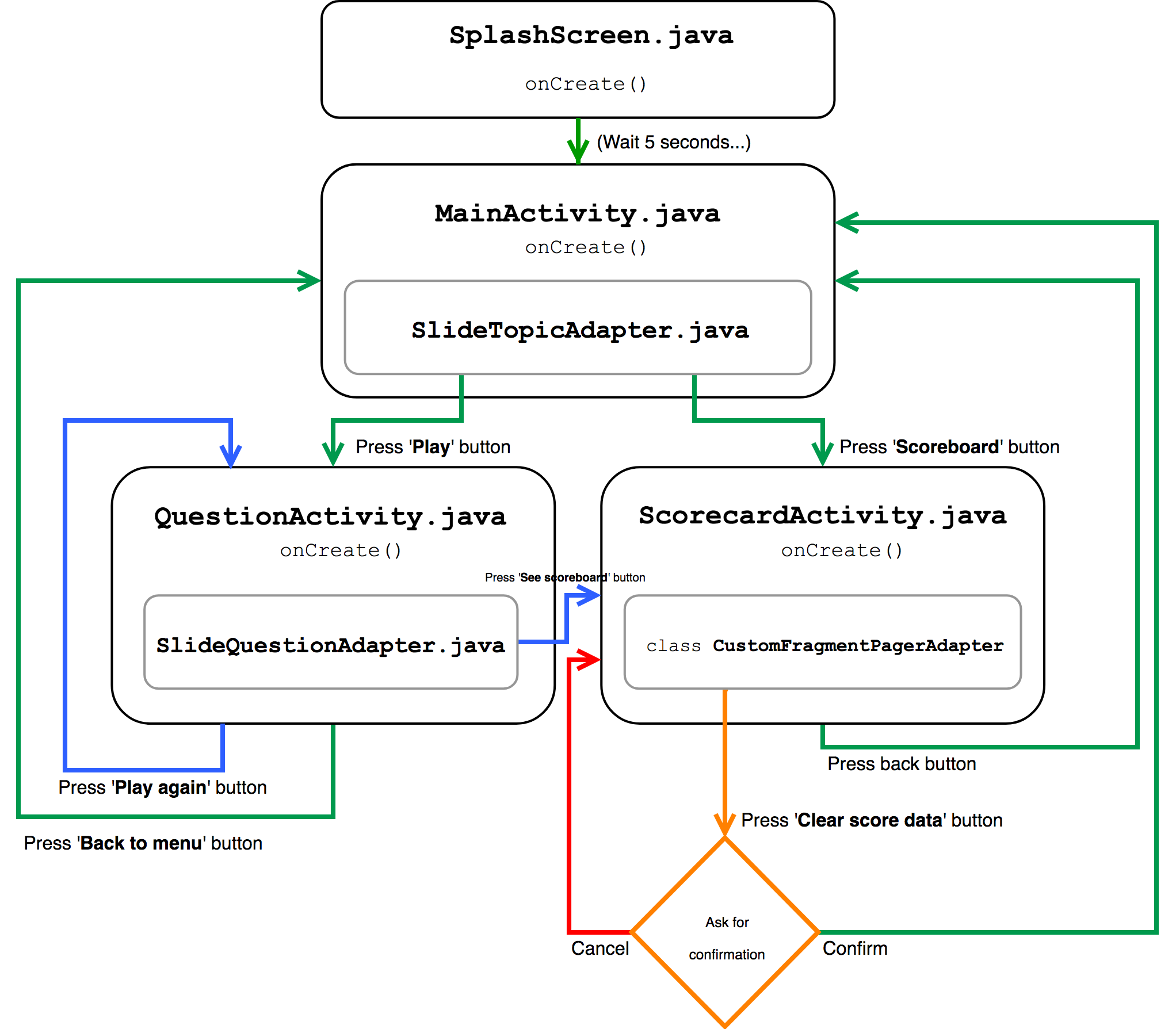
This table should be clear enough to describe the actual structure of the project in terms of relationships between Code ⬄ Layout.

For instance, the first column describes the Java file which has the ‘business’ logic of the application, whereas the two last columns contain the layout specifically associated with that Java file in that same row, e.g. ‘MainActivity.java’ has a layout resource associated with it, named ‘activity\_main.xml’. Similarly, ‘SlideAdapter.java’, which is an adapter for controlling the main activity’s ‘ViewPager’ component also has a layout associated with it – the ‘slide\_topic.xml’ XML file.

## High level structure and execution flow (flowchart)

Each of these files has a role in the entire application, as well as a relationship between the next file. In fact, there is a program flow from a specific Java file to another, and back.

On startup, the ‘onCreate()’ method of the file ‘SplashScreen.java’ is executed, as is shown on the following flowchart:



This is the underlying structure of the app. Every possibility is covered in this diagram; however, it does not describe the behaviour of the activities in themselves. This matter will be discussed on section 6 - ‘Testing methodologies’.

Figure 3‑7 - Flowchart: Program execution flow

In addition to these Java classes, we may find other classes such as ‘Constants.java' and ‘FileHandler.java’, which are just miscellaneous static classes that deal with global values and loading and saving of files. The public quiz data which contains the actual questions is stored in JSON format, as follows:



Figure 3‑8 - Quiz databank structure – quiz\_config.json

This file (‘quiz\_config.json’) stores the quiz questions inside topic objects. This object also contains the name of the topics and their descriptions. As can be seen, each question inside a topic object is in the form of an array of objects, which contains a field “query” with the actual question string, followed by an “answers” field, with all of the possible answers in it. The correct answer is always the first answer in this array, so, for instance, in this given example, “Answer 1” is the correct answer. In addition, an extra field “img” (not present in the screenshot) is used to assign an image to the question. This field contains the filename of the image (no file extension) which must be stored inside the directory ‘res/drawable’.

From this, class ‘Question.java' stores the inner object (fields: “query” and “answers”), whereas class ‘Topic.java’ stores the outer object comprised of the fields “name”, “desc” and “questions”. The last class ‘ScoreData.java’ stores private user score data also in JSON format, however, this data is only stored inside the permanent storage and as such it is not available for access as opposed to the ‘quiz\_config.json’ file.

As was just described, the user scores are stored in the JSON format with the keys:

|  |  |  |
| --- | --- | --- |
|  | Field / Key | Meaning / Role |
| 1 | “time” | Keeps the time and date of the question attempt |
| 2 | “topic\_index” | The topic index from which the question belongs to |
| 3 | “correct\_answers” | How many correct answers the user got on that attempt |
| 4 | “total\_questions” | How many questions did the user answer - useful to determine the score percentage (= correct\_answers / total\_questions) |
| 5 | “questions” | An array for each answered question that indicates which questions were answered correctly and which ones weren’t. This field also contains the index of the question that was randomly picked from the databank in this format: **{“q2” : “0”}** OR **{“q8” : “1”}** |

Table 3‑2 – User score object structure – quiz\_userdata.json

All values in this private JSON file (‘quiz\_userdata.json’) are stored in string format.

The reason JSON was picked was to simplify the way questions were added to the app. Besides, JSON is extremely simple to parse (the file ‘Topic.java’ has a public static method implemented to parse this and instantiate Topic and Question objects), since it is just a key-value data structure. The last reason was to easily allow the addition of RESTful features. This way, new quiz questions may be added through a simple http request.

# Quiz content choice

Several topics and questions were picked for this quiz. The five topics are:

1. Android Activity Lifecycle
2. MVC (Model-View-Controller)
3. Layouts
4. Widgets
5. Interface Controls

These topics are at the core of this course module. They are the most fundamental topics in Android development and as such they were picked for this quiz. Plus, the course module covers these topics in this order, which strengthens the motive for the choice of such topics. Ideally, these topics should strengthen the knowledge of a student learning Android development, which is the main goal of the case-study provided.

Despite the choice of these topics, the questions themselves only cover the first topic. The case-study requirements indicate that students must pick 5 topics with 10 questions each (images included), which equals in a total of 50 unique questions and images. As per the lecturer’s instruction, only the first topic was populated with unique questions. These questions were consequently copied and pasted across the other four topics. Lastly, the source of the questions may be found on the section Appendix A.

The ten questions are:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Question | Possible answers (1st is correct) | | |
| 1 | 2 | 3 |
| 1 | **How to pass the data from activity to services in android?** | Using putExtra() method in intent, we can pass the data using setResult() | We can store the data in a common database and access the data on services as well as in Activity | We can store the data in a common database and access the data on services as well as in Activity |
| 2 | **What is the life cycle of foreground activity in android?** | onCreate() −> onStart() −> onResume() −> onStop() −> onRestart() | onCreate() −> onStart() −> onResume() −> onStop() | onCreate() −> onStart() −> onResume() |
| 3 | **How to kill an activity in Android?** | finish() | finishActivity(int requestCode) | None of the Above |
| 4 | **What is the application class in android?** | Base class for all classes | A class that can create only an object | Anonymous class |
| 5 | **What is a fragment in android?** | Layout | JSON | A portion of an Activity |
| 6 | **Which specific characteristic does not differentiate the Dalvik Virtual Machine (DVM) from other standard VMs?** | It uses Java Virtual Machine compatible .class without conversion | It uses a register-based architecture | It is just a Just-In-Time (JIT) compiler |
| 7 | **Which of the following is not a method of the Activity class that user can override while implementing his/her own Activity?** | onKill() | onPause() | onCreate() |
| 8 | **Which layer is not included in the architecture diagram of an Android Platform?** | Data interchange | Application Framework | Linux Kernel |
| 9 | **During the initialization of your Activity, \_\_\_\_\_\_ might be used to set the layout resources to define the UI of your Activity.** | setContentView(int layoutResID) | getInflater() | startActivity(Intent intent) |
| 10 | **Components of the underlying OS are written in \_\_\_, while user and built-in applications are built for Android in \_\_\_\_?** | C/C++, Java | Java, Java | Java, C/C++ |

Table 4‑1 – List of questions chosen

# Fundamental programming problems and solutions

The development of this application was a challenge. Most of the problems were directly related to Android development, however, programming was at the core of it. A successful development of such project involves and obviously demands a rich and broad skillset on programming, specifically Java programming, and as such, this section will identify those exact programming challenges and skills required to solve these challenges.

The first and most important challenge is in the basics of the Java language: data types, statements (if, switch, for, while, do-while), I/O, Arrays (statically or dynamically allocated,) Collections, methods, classes and OOP in general.

These are challenges that the student must be able to solve already by default. The student is expected to be familiar with such concepts (excluding OOP). Plus, this course module should allow the student to either regain the knowledge required or to learn from the beginning.

After the basics, we may find challenges in interfacing this knowledge with the Android platform. Executing Activities, passing data arguments to those Activities, managing the state of the application, instantiating Views programmatically and dynamically changing the background colour of a View are examples of challenges that require strong knowledge of Java. This is because the ‘act’ of passing arguments through Activities may be unsafe, due to the fact that the two Activities must mutually agree on what data is being passed. Besides, if ‘Activity A’ passes arguments to ‘Activity B’ but ‘Activity C’ does not pass arguments to ‘Activity B’, it may be possible that ‘Activity B’ will receive a null reference from ‘Activity C’. A mistake of this kind would never be committed by a student with strong programming knowledge, let alone ‘null pointer’ and ‘index out of bounds’ exceptions.

These challenges are general problems found in any Android project. This project in particular covered other ‘not-so-common’ challenges. The task of parsing JSON data into Java objects was not so trivial. It required the use of a JSON parsing library (provided by the Android SDK) followed by iteration over the fetched objects. For every key found in this JSON object, a new object was pushed into a ‘HashMap<String, String>’ collection. This essentially composed the underlying structure of the ‘Topic’ and ‘Question’ classes.

Also, regex was used to delete score data from a specific topic without deleting the entire user score data. The reason for this is, since each time the player finishes a quiz, a JSON object in string format is appended to the ‘quiz\_userdata.json’. Deletion becomes extremely difficult especially if the player attempts different topics every time.

If this was the case, the user data structure would be in the state:

[{“topic\_index”: “0”, …}, {“topic\_index” : “1”, …}, {“topic\_index”: “0”, …}, …]

In order to delete the score of only the topic 1, we’d have to get the index of the start of that substring and delete the string contents from that index up to the start of the next score entry. Regex was the perfect solution for this.

Finally, the case-study requirements include a graph to display the score data. Unfortunately, there is no default graph view provided by the Android SDK, which meant researching for a decent and simple graphing library. This demands knowledge about libraries, open-source and how to effectively use the code that someone else developed. The solution for this was to use the (very popular) open-source library ‘GraphView’ developed by Jonas Gehring (reference on Appendix-B). Thanks to Google’s build tools (and Gradle), importing the library was as easy as adding the line

implementation 'com.jjoe64:graphview:4.2.1'

to the Gradle script of the app module (not the Project Gradle script).

# Testing methodologies

|  |  |
| --- | --- |
| Activity: SplashScreen.java | |
| Test 1: Splash screen delay | |
| Prompt: | **Expected output:** |
| Real output: same as expected output. The image scales up slowly for 5 seconds. | |
| Steps: wait 5 seconds for the splash screen to execute MainActivity.java. | |
| Possible user inputs: none (except screen rotation). | |

|  |  |
| --- | --- |
| Activity: SplashScreen.java | |
| Test 2: Splash Activity screen rotation | |
| Prompt: | **Expected output:** |
|  |
|  | |
| Steps: rotate the device horizontally in order to readjust the layout. | |
| Possible user inputs: none. | |

|  |  |
| --- | --- |
| Activity: MainActivity.java | |
| Test 3: Swipe test | |
| Prompt: | **Expected output (first swipe):** |
| Prompt: | **Expected output (second swipe):** |
| Prompt: | **Expected output (third swipe):** |
| Prompt: | **Expected output (fourth swipe):** |
| Real output: same as expected output. | |
| Steps: swipe right/press the left/right arrows right before they fade out 4 times. | |
| Possible user inputs: press the buttons ‘Play’ and ‘Scoreboard’ and rotate. | |
| Notes:   1. The navigation arrows are clickable, up until the point they fade out (1.5 seconds). 2. The first View Pager slide has the left arrow hidden whereas the last slide has the right arrow hidden instead. | |

|  |  |
| --- | --- |
| Activity: MainActivity.java | |
| Test 4: Main Activity screen rotation | |
| Prompt: | **Expected output:** |
|  |
| Real output: same as expected output. | |
| Steps: rotate the device horizontally in order to readjust the layout. | |
| Possible user inputs: press the buttons ‘Play’ and ‘Scoreboard’. | |

|  |  |
| --- | --- |
| Activity: MainActivity.java | |
| Test 5: Press ‘Play’ Button | |
| Prompt: | **Expected output:** |
|  |
| Real output: same as expected output. | |
| Steps: press ‘Play’ button. | |
| Possible user inputs: rotate the device. | |
| Notes:   1. The quiz question on the expected output will vary since questions are randomly picked. | |

|  |  |
| --- | --- |
| Activity: MainActivity.java | |
| Test 6: Press ‘Scoreboard’ Button | |
| Prompt: | **Expected output:** |
|  |
| Real output: same as expected output. | |
| Steps: press ‘Scoreboard’ button. | |
| Possible user inputs: rotate the device. | |
| Notes:   1. The expected output will vary if the user has previously played the quiz. | |

|  |  |
| --- | --- |
| Activity: QuestionActivity.java | |
| Test 7: Enable ‘Next question’ Button | |
| Prompt: | **Expected output:** |
|  |
| Real output: same as expected output. | |
| Steps: select any answer from the Radio Group. | |
| Possible user inputs: press ‘Back to menu’ Button and rotate device. | |
| Notes:   1. The View Pager is not scrollable until an answer has been given. | |

|  |  |
| --- | --- |
| Activity: QuestionActivity.java | |
| Test 8: Question Activity screen rotation | |
| Prompt: | **Expected output:** |
|  |
| Real output: same as expected output. | |
| Steps: rotate the device horizontally in order to readjust the layout. | |
| Possible user inputs: press ‘Back to menu’ and ‘Next question’ Buttons. | |
| Notes:   1. Rotation has been disabled for this activity since the question string, image and answers would interfere with the navigation buttons. Dynamically adjusting the height of the View Pager or setting its height to ‘wrap\_content’ is not a solution since this interference is caused by the length of the question string and the size of the image rather than the View Pager itself. | |

|  |  |
| --- | --- |
| Activity: QuestionActivity.java | |
| Test 9: Press ‘Next question’ Button | |
| Prompt: | **Expected output:** |
|  |
| Real output: same as expected output. | |
| Steps: press ‘Next question’ button. | |
| Possible user inputs: press ‘Back to menu’ Button and rotate device. | |
| Notes:   1. The View Pager is not scrollable until an answer has been given. 2. Selecting an answer re-enables ‘Next question’ Button. | |

|  |  |
| --- | --- |
| Activity: QuestionActivity.java | |
| Test 10: Enable ‘Finish’ Button | |
| Prompt: | **Expected output:** |
|  |
| Real output: same as expected output. | |
| Steps: solve quiz and select last answer. | |
| Possible user inputs: press ‘Back to menu’ and ‘Previous question’ Buttons and rotate device. | |
| Notes:   1. It is not possible to scroll past this point even after selecting the answer. The user must either:    1. Press ‘Finish’ Button    2. Press ‘Previous question’ or scroll left in order to fix already answered questions    3. Press ‘Back to menu’ Button to return to the Main Activity.    4. Press the system’s back button on the bottom of the screen. | |

|  |  |
| --- | --- |
| Activity: QuestionActivity.java | |
| Test 11: Press ‘Finish’ Button | |
| Prompt: | **Expected output:** |
|  |
| Real output: same as expected output. | |
| Steps: press ‘Finish’ Button. | |
| Possible user inputs: press ‘Back to menu’ and ‘Previous question’ Buttons and rotate device. | |
| Notes:   1. The user may swipe (only) left to see his answers. 2. Now, each question has its answer highlighted with a colour. Green background on the correct answer and red background on the wrong answer. If the user’s answer matches the correct answer, no radio button is highlighted red. 3. Right swipe is disabled on this slide. 4. Pressing ‘See answers’ Button takes the user back to the first question 5. Pressing ‘See Scoreboard’ Button opens the Scoreboard Activity and closes this activity. 6. Pressing ‘Play again’ Button re-initializes this Activity with new random questions. | |

|  |  |
| --- | --- |
| Activity: ScorecardActivity.java | |
| Test 12: Scorecard Activity screen rotation | |
| Prompt: | **Expected output:** |
|  |
| Real output: same as expected output. | |
| Steps: rotate the device horizontally in order to readjust the layout. | |
| Possible user inputs: press ‘Clear score data’ Button. | |
| Notes:   1. The user may scroll down in case the list overflows outside the screen. | |

|  |  |
| --- | --- |
| Activity: ScorecardActivity.java | |
| Test 13: Press ‘Clear score data’ Button | |
| Prompt: | **Expected output:** |
|  |
| Real output: same as expected output. | |
| Steps: press ‘Clear score data’ button. | |
| Possible user inputs: swipe right, press ‘Score Graph’ tab and rotate screen. | |
| Notes:   1. Pressing confirm deletes the score entries from the private JSON file used to store scores. 2. Only the scores for a given topic are deleted. The scores for the other topics are preserved (until all data is deleted). | |

|  |  |
| --- | --- |
| Activity: ScorecardActivity.java | |
| Test 14: Press ‘Score Graph’ tab or swipe right | |
| Prompt: | **Expected output:** |
|  |
| Rotated: | |
|  | |
| Real output: same as expected output. | |
| Steps: press ‘Score Graph’ tab or swipe right. | |
| Possible user inputs: press ‘Clear score data’ Button, press back button or rotate device. | |

|  |  |
| --- | --- |
| Activity: ScorecardActivity.java | |
| Test 15: Press Graph’s ‘Bar Graph’ | |
| Prompt: | **Expected output:** |
|  |
| Real output: same as expected output. | |
| Steps: press Graph’s ‘Bar Graph’. | |
| Possible user inputs: swipe left, press ‘Score List’ tab and rotate screen. | |
| Notes:   1. The date and time are displayed on the Toast at the bottom of the screen. | |

Appendices

Appendix A:

Quiz questions: <https://www.tutorialspoint.com/android/android_online_quiz.htm>

Appendix B:

GraphView plotting library: <http://www.android-graphview.org/>

1. University Academic Misconduct Regulations [↑](#footnote-ref-1)
2. Information on exclusions to this rule is available from the Advice Centre at each Campus [↑](#footnote-ref-2)