ITECH2000 Mobile Development Fundamentals
Assessment Task 3 – Trivial Pursuit App

Overview

You will implement in AppInventor a multi-screen app, based on a given project specification. This app will use a range of components taught up to and including Week 11 of the course. You will also submit a brief report containing pseudocode and a description of how your solution utilises various concepts learned in class.

Timelines and Expectations

Percentage value of task: 30% (of final course mark)

Due date: 11:59pm, Sunday 2nd October 2022 (Week 11) – after this date, you will receive a penalty

Cut-off date: 11:59pm, Sunday 9tt October 2022 – after this date work will not be accepted

Minimum time expectation: 20 hours

Learning Outcomes Assessed

The following course learning outcomes are assessed by completing this assessment.

K1. Understand constructs typical of many programming languages such as: variables, expressions, assignment, sequence, selection, iteration, procedures, parameters, return values.

A1. Design, develop, test and debug mobile apps from a given textual program specification.

S1. Analyse the input, processing and output needs of small programming problems.

S2. Design code sequences to realise algorithms in a programming language.

S3. Design basic user interfaces and develop storyboards to convey designed interaction sequences.

S5. Develop test cases to ensure correct behaviour.

Assessment Details

This assignment contains two parts: an app and a brief report.

1. App Details

Your assignment is to develop an Android application, using MIT AppInventor, which is a general knowledge trivia app. The app should utilise programming concepts covered in class from Weeks 2 – 10, including persistence of data using files and databases, the dictionary data structure, web and networking functionality, and sorting algorithms.

The application should contain several screens to fulfill the requirements described below. You are free to design the interfaces of your app as you like, but your solution should also meet the requirements in a logical way.

Specifically, the logic requirements that must be supported in the app are as follows:

- When the application is first opened, there should be a screen that provides users with a way to select the game difficulty (“easy”, “medium”, or “hard”) and start a new game and view high scores.

- There should be a separate screen in which the trivia questions are displayed, opened from the first screen. When this trivia screen opens, it should load questions of the selected difficulty, following the following logic:
    - If the device is currently connected to the network, questions should be loaded from the Open Trivia Database API, as described in the API section of this document.
    - If the device is not connected to the network, questions should be loaded from the provided text files, which contain static question data retrieved from the Open Trivia Database API.
- A single question should be displayed at a time, including its category, question text, and four possible answers (including the correct answer and three incorrect answers in a random order). The user should be able to select one of these answers, at which point the following should occur:
    - If the selected answer is the correct answer, the user's score is incremented by 1 point (the current score should also be visible on the trivia screen). The user should then be prompted as to whether they wish to continue the game or end it. If they choose to continue, a new question is displayed and the process repeats.
    - If the selected answer is incorrect, the user should be made aware that they selected the wrong answer, and the correct answer should be shown. The app should then navigate to another screen where the user can save their score to a database, passing the achieved score with it.
- The save score screen should have a mechanism for the user to enter their name, which along with their score and a current date and timestamp, should be saved to a database. Once the value is saved, the app should navigate back to the main screen.
- There should be a separate screen on which users can view the high scores by loading them from the database and displaying in an appropriate UI component. The following additional functionality should exist on this screen:
    - There should be a way for the user to sort the scores, based on your implementation of a sorting algorithm. The scores can be sorted by the score value, alphabetical order of the user's name, or the date timestamp – you only need to support one of these.
    - The user should also be able to delete a score, which will remove it from the display and the database entirely.

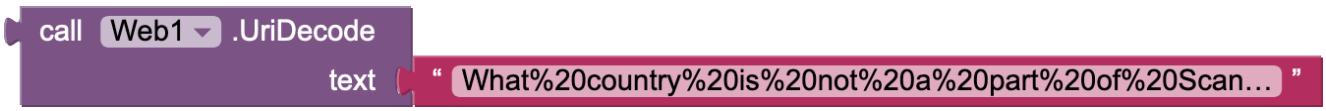
Downloading trivia questions from the Open Trivia Database using the API

The Open Trivia Database API should be used to download the trivia questions used in this app (when a network connection is available). You can get a random list of questions of a particular difficulty in JSON format using the following URL – all you need to do is replace **<#>** with a positive integer, and ***<difficulty>*** with the preferred difficulty ("easy", "medium", or "hard") as required:

```
https://opentdb.com/api.php?amount=<#>&difficulty=<difficulty>&type=multiple&encode=url3986
```

For example, https://opentdb.com/api.php?amount=20&difficulty=hard&type=multiple&encode=url3986 will give 20 random questions of hard difficulty. Note that the *type* parameter is set to "multiple", so that multiple-choice questions are returned. The *encode* parameter specifies how special characters are rendered in text, in this case it is set to URL Encoding (RFC 3986).

In order to correctly display any of the text in your app, you'll need to use the [.UriDecode] block (as shown below), which can be accessed from any Web type block you add to your app. For example, the RFC 3986 text that is displayed in the example image below would be decoded to "*What country is not a part of Scandinavia?*"



It is recommended that you visit the Open Trivia DB API link in a browser and view the JSON in a web browser to get a feel for its structure. You will see that the JSON data contains an array (*results*) of questions, with each question containing a range of information that describes the *question*, its *category*, *correct_answer*, and *incorrect_answers*. Your app should extract this data and load into an appropriate data structure to achieve the required functionality for this assignment.

General Requirements for Coding

While your app should meet the functionality described above, you need to ensure that you demonstrate the concepts we have covered in ITECH2000 so far. To achieve full marks, you will need to ensure that you have correctly made use of each of the following components, constructs or concepts somewhere in your app:

- Dictionary
- WebViewer and/or Web component
- ListView or ListPicker
- TinyDB and File
- Local and global variables
- Procedures

You should follow best practices for coding that we have described this semester, including the use of procedures to promote code reuse. Make sure you thoroughly test your application to ensure that it is robust.

Please read through all of the requirements *before* you commence work on the app, so you get a full sense of what is required to be done. It is recommended that you first model any events using pseudocode before commencing programming.

2. Brief Report

As well as completing the program described above in AppInventor, you are also required to submit a brief report that includes the following:

- A title page that includes your name and student ID number.
- Pseudocode describing the behaviour of two (2) events that your app will respond to. Ensure that you clearly label your pseudocode so that it is clear what aspect you are modelling.
- For each of the design components/blocks listed in the previous section (General Requirements for Coding), you should describe in 2-3 sentences how you used the component(s) in your solution and justify why. If you have used a component multiple times, please describe one example. **Note:** There are 6 in total.
- A description of how you tested your application to ensure that it functioned correctly, with respect to user inputs, outputs, and networking.

Getting Assistance and Clarification

If any part of the task is unclear to you, or you are not quite sure how to do some aspect of the task, you should either contact your lecturer directly (via email, or in person while you are in class), or else post a question to the Discussion Forum on Moodle. However, any questions posted to the forum on Moodle should not include anything that you plan to submit (such as screenshots of code you might want to submit).

Plagiarism

Plagiarism is the presentation of the expressed thought or work of another person as though it is one's own without properly acknowledging that person. You must not allow other students to copy your work and must take care to safeguard against this happening. More information about the plagiarism policy and procedure for the university can be found at https://federation.edu.au/current-students/learning-and-study/online-help-with/plagiarism

Submission

You must export your AppInventor project for submission. To do this, go to the "Projects" menu, and select "Export selected project (.aia) to my computer". Rename the .aia file to include both your name and student ID number. **App files submitted in any other format than .aia (e.g. apk) will not be accepted and you will receive zero marks.**

You should also save your brief report as a PDF including both your name and student ID in the file name.

Upload these files to Moodle through the assignment link labelled "Submit Assignment 3". This link will only become available after you have completed the "Declaration of Originality" form for the assignment, which requires you to accept the Student's Statement. It is a legal declaration that the work was done by you, without any part of it being done by someone else.

Feedback

You can expect to receive your final mark and feedback comments within 3 weeks of the due date or the date which you submitted your work – whichever is later. You may be required to attend an interview with the marker to answer questions about your work; if this is the case, your mark will be withheld until you have attended.

Continue to the next page for the Marking Criteria/Rubric

Marking Criteria/Rubric

Student Name: _____ Student ID: _____

App Logic	Marks	Awarded
Difficulty and Menu Screen		
Screen exists and has appropriate input/output elements for requirements	1	
Functionality provided for the user to select difficulty and open trivia screen	2	
Functionality to navigate to high scores screen implemented correctly	1	
Trivia Game Screen		
Screen exists and has appropriate input/output elements for requirements	1	
Questions of selected difficulty loaded from web API if network connectivity available	4	
Questions of selected difficulty loaded from appropriate File if no network available	4	
Question category, text, and correct/incorrect answers displayed on screen correctly	2	
Selecting a correct answer behaves correctly (loading next question or exiting to menu)	2	
Selecting an incorrect answer behaves correctly, including sending score to next screen	2	
User score is displayed on screen and correctly implemented	1	
Save Score Screen		
Screen exists and has appropriate input/output elements for requirements	1	
Date timestamp is generated using appropriate methods	2	
User's name, score and date timestamp correctly saved to a TinyDB with appropriate tag	2	
App navigates back to menu screen upon saving data	1	
View Scores Screen		
Screen exists and has appropriate input/output elements for requirements	1	
Scores are loaded from database and displayed on screen (with name and timestamp)	2	
Sorting functionality works using an appropriate sorting algorithm	4	
Score can be deleted from the database using appropriate methods	2	
Method to navigate back to the menu screen is implemented	1	
General Programming Requirements [For each of the following criteria – full marks if included and no issues; half marks if included by some issues; no marks if not demonstrated in solution]		
Has used TextBoxes, CheckBoxes, Labels, and Buttons appropriately (including using their action blocks and event blocks)	1	
Has used the screen-changing mechanisms correctly and appropriately	1	
Has used repetition construct(s) to repeat code sequences appropriately and in appropriate places	1	
Has used decision constructs appropriately / in appropriate places	1	

Has formed appropriate boolean and relational expressions	1	
Has used the list construct for appropriate purposes and used its action blocks in an appropriate manner to manipulate/use lists	1	
Has used the dictionary construct for appropriate purposes and used its action blocks in an appropriate manner to manipulate/use dictionaries	1	
Has formed procedures appropriately and in appropriate circumstances	1	
Has used the TinyDB component appropriately	2	
Has used the File component appropriately	2	
Has used the Web and/or WebViewer components appropriately	2	
The names used for variables, components, procedures and their argument slots are appropriate and clearly communicate their purpose in the code	1	
The app is designed to be robust and does not crash unexpectedly	2	
Brief Report		
Two (2) events are adequately modelled using pseudocode	2	
Examples and explanations are provided for each of the 6 components/concepts listed in the *General Requirements for Coding* section (0.5 marks each)	3	
Description of testing plan conducted to ensure robustness of app	2	
Total:	60 marks	
Total Course Marks:	30%	
Feedback:		