Project 1B - MultiQuiz Screenshot

10/10 Points

2/8/2022

Attempt 1

REVIEW FEEDBACK
2/6/2022

Attempt 1 Score:
10/10

View Feedback

Unlimited Attempts Allowed

Details

Due: February 8
Points: 10
Resources:

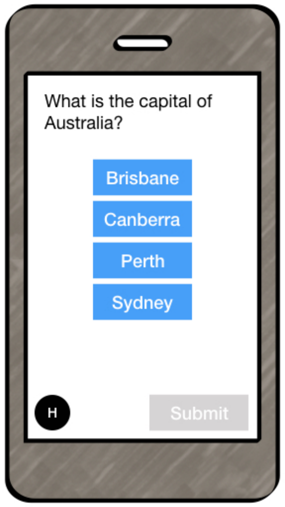
- Leverage files from AnswerButton

Deliverables:

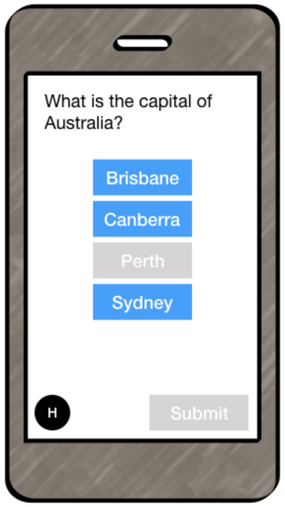
- Screenshot (jpg)

Overview

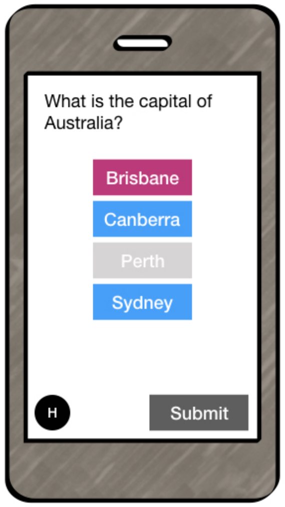
Write a quiz application named MultiQuiz. It will be similar to the quiz application in BNR. Create four multiple choice questions. The questions should not be longer than two or three lines. The answers should all be one word answers that fit inside of a button. When the application starts up, the first question appears at the top of the screen. Below the question is a column of four answer choices in the form of buttons that can be pressed. In the bottom left of the screen is a hint button and in the bottom right of the screen is a submit button. Initially, the submit button is disabled. The figure below illustrates this state.



When the hint button is pressed, one of the wrong answers becomes disabled (choose a random wrong answer). If there are no wrong answers left (in other words, all three wrong answers have been disabled, the hint button itself becomes disabled). The figure below indicates the state of the screen when one hint has been used.

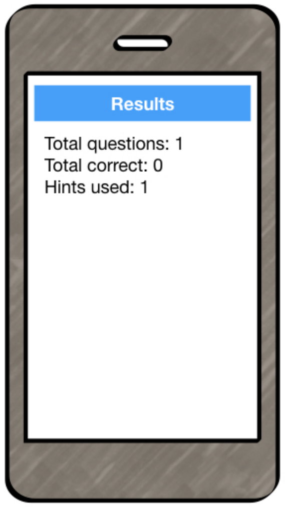


When a button is selected as an answer, the selected button changes color and the submit button becomes enabled. The user can still change their answer at this point, and the user can even unselect a selected answer by clicking on the button again. The figure below shows the state of the screen after one hint has been used and after one of the remaining answers has been selected.



Note that once an answer has been selected, the hint button can still be used, and if the wrong answer chosen to be disabled is the one the user had selected, the button is disabled (leaving no buttons selected) and the submit button is also disabled. Note that when a button is disabled, it should no longer be selected in the model.

When the submit button is pressed, the answer is recorded and the next question is displayed. If there are no more questions, the client is taken to a results screen that displays: (1) the total number of questions, (2) the number of questions answered correctly, and (3) the total number of hints used. The figure below shows the results screen for a quiz in which only one question was asked and the client used a hint but still got the question wrong.



Implementing MultiQuiz

Implement MultiQuiz by modifying the code from the AnswerButton app.As you evolve AnswerButton into MultiQuiz, make sure you can meet the following milestones:

1. Modify the Kotlin code in AnswerButton so that it has a functional style (see Project 1A)
2. **Create a MultiQuiz app that works with a single question**. Copy the code from AnswerButton into a new MultiQuiz project. Change the disable button to a hint button (that works as described here, and change the reset button to a submit button. When the submit button is clicked (only if an answer is selected), display a toast indicating whether the selected answer is correct or not (similar to what you did with GeoQuiz in chapter 1). Note that you will *not* have toast in the final project. I am suggesting it here solely as a sanity check.
3. **Make MultiQuiz work with multiple (four) questions**. Choose your questions and answers so that the first answer is correct for the first question, the second answer is correct for the second question, the third answer is correct for the third question, and the fourth answer is correct for the fourth question. This will make it easier for us to write test cases for the application.
4. **Make MultiQuiz rotate without losing the model state**. Create a view-model (see chapter 4 of BNR) that saves the state when you rotate the phone. You will move most of the functions that are called from the listeners into the view-model.
5. **Create a result screen for MultiQuiz**. Finally, create your second activity and use an intent to call it a pass data to it. Note that unlike in BNR chapter 6, you will only be passing data to the second activity, you will not be getting data back from the second activity.

After completing each of these steps, run MultiQuiz and make sure they work. Remove the toasts from your final project. If you are having trouble with MultiQuiz and have questions about the implementation, let us know which of these milestones you were able to achieve in your post.

Screenshot

After you complete step 3, submit a screenshot of Android Studio (and your application running in the foreground) to this assignment. Make sure of the following.

- The project window is displayed on the left
- The editor window contains QuizActivity.kt
- The emulator displays a quiz question in which one of the answers is disabled, one of the answers is selected, and at least one of the answers is enabled but unselected. Your button colors should **not** the same as those in the screenshots above. The layout of the app should look professional - all buttons should have space between them and elements should be aligned nicely.
- Your name is displayed prominently just above the emulator.

Naming and Layout Requirements

In the next part of this project (Project 1C), we will test your application. In order for our tests to work correctly, you *must* have the following names.

- Your application package name must be "edu.vt.cs.cs5254.multiquiz"
- Your application name must be "MultiQuiz"
- The IDs for your buttons must be:
  - *answer_0_button*, *answer_1_button*, *answer_2_button*, and *answer_3_button*
  - *hint_button* and *submit_button*
- On the result screen, the IDs for the text-views that hold your values must be:
  - *total_questions_value*, *total_answers_correct_value*, and *total_hints_used_value*

We will give you a subset of the tests that you must pass in Project 1C.

You are unable to submit to this assignment as your enrollment in this course has been concluded.