

Mobile Application Assignment 2

Instructors: Legand Burge, Asad Ullah Naweed

Due Date: 09/27/2018

Be sure to click on the links in this doc to view more information about certain topics. These may or may not help you in your lab, but can always provide some extra context and information.

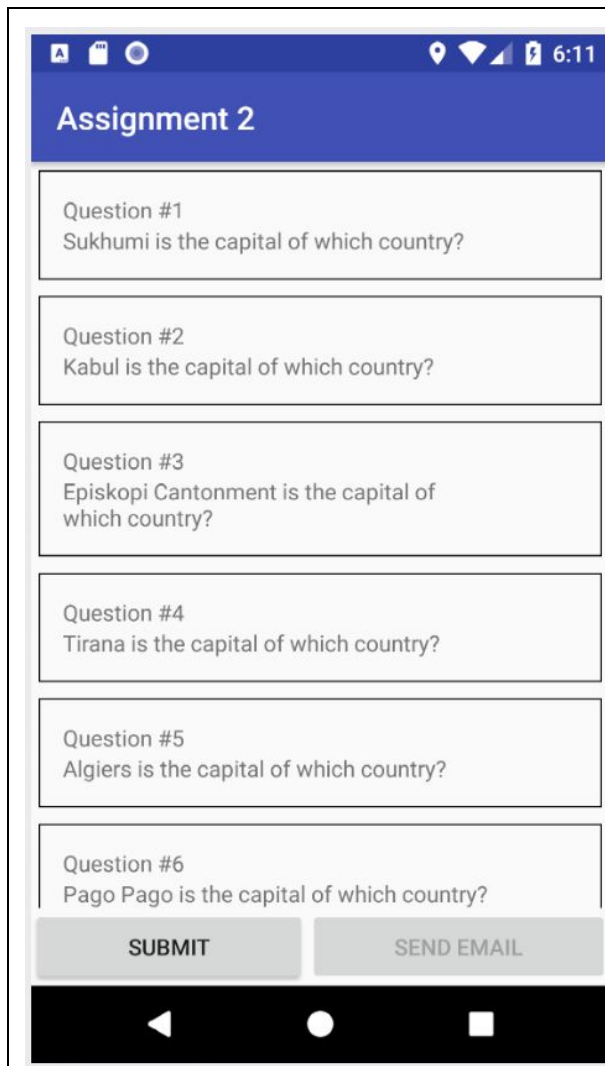
This assignment is far more structured and has more concrete deliverables than the last one, so pay attention to the individual parts.

For this assignment, you'll be making yet another version of the Geo Quiz app.

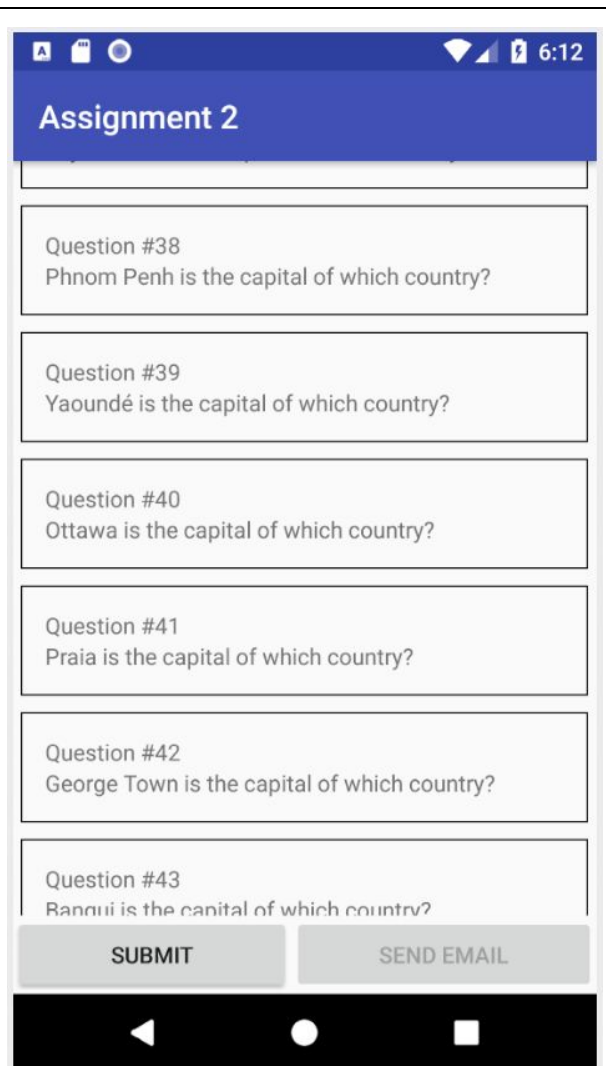
Basic Product Requirements

As with every good software design, it mostly begins with a product manager giving a set of requirements and desired app behavior. For this assignment, we're going to go through a mini-version of that.

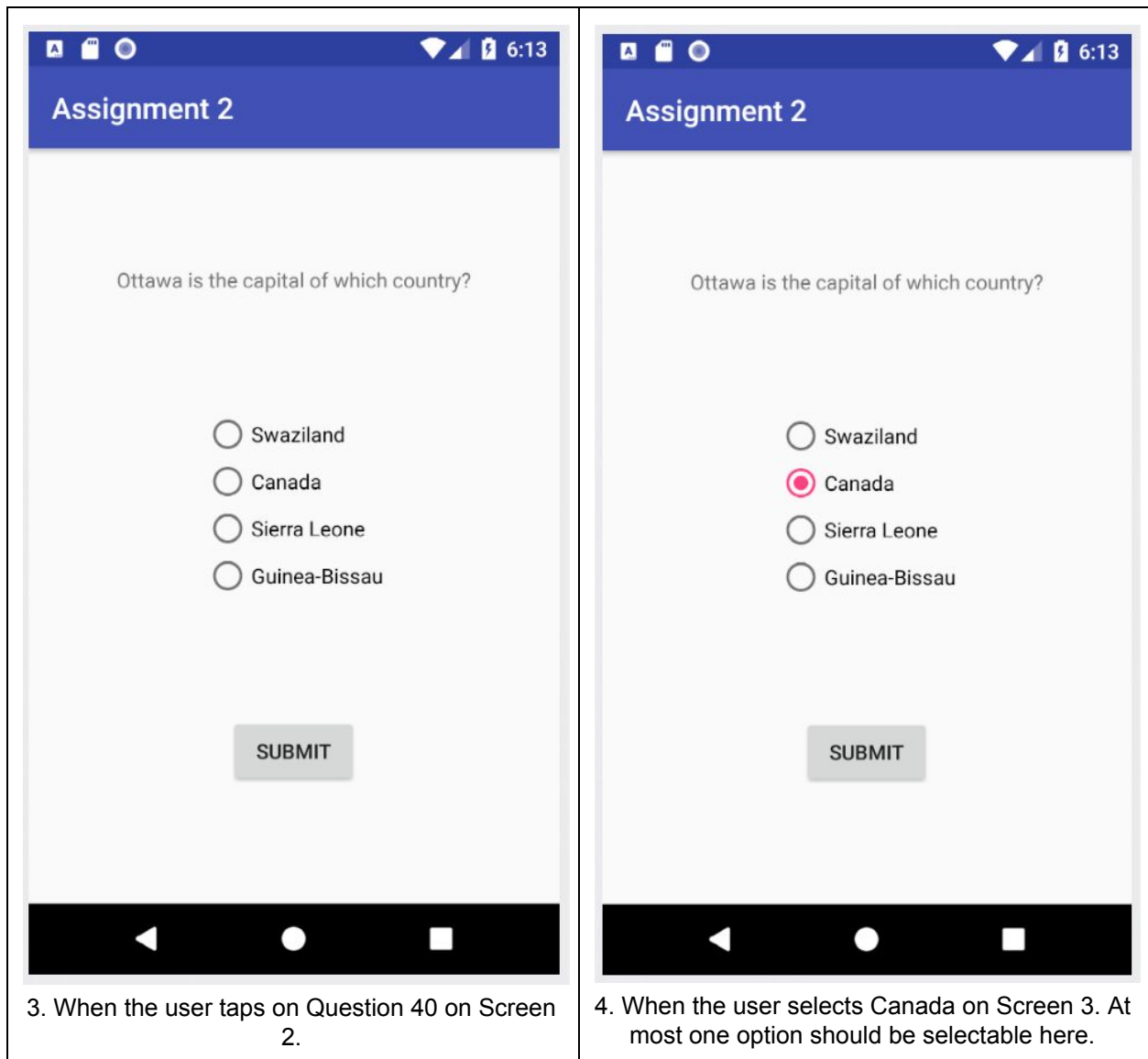
Here are some user flows you will need to implement. You can also think of these requirements as your grading criteria, because that's often what happens in the real world.

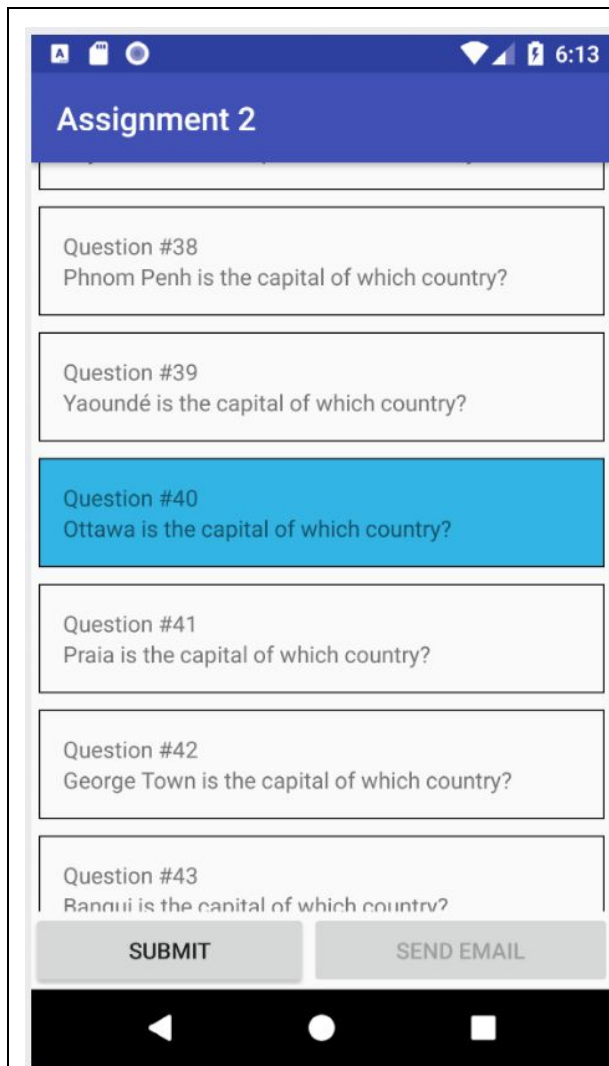


1. Initial state upon launch. There should be some amount of spacing between the questions borders and the question text, and also the space between the borders of adjacent questions.

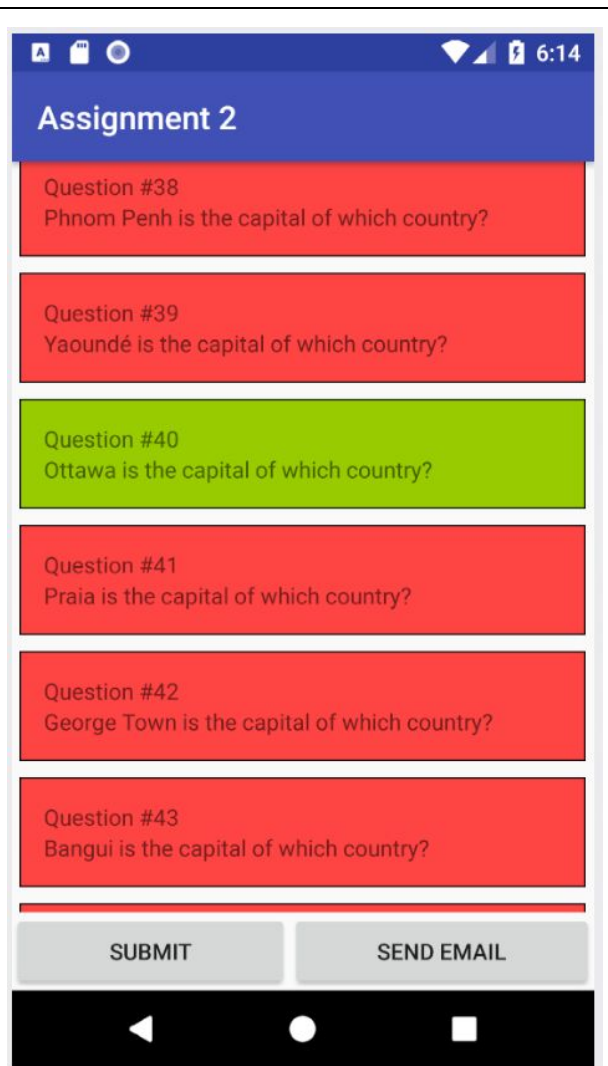


2. After some vertical scrolling on Screen 1.

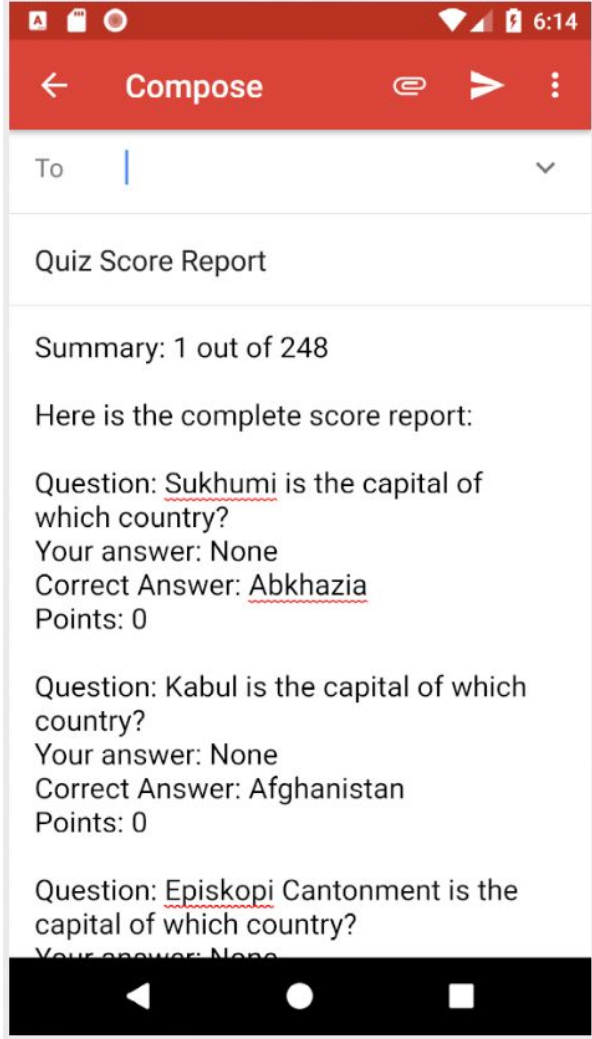




5. When the user clicks the "Submit" button on Screen 4. The answered question has now been turned blue. The user should still be able to click on blue questions to change their answers. Clicking on a blue question should display the same question, the same answer choices, and their previous answer pre-selected.



6. When the user clicks the "Submit" button on Screen 5. The correctly answered questions are turned green, and the incorrectly answered or unattempted questions are turned red. The "Send Email" button is now enabled, and clicking on the questions does nothing.



7. When the user clicks the "Send Email" button on Screen 6. The email text is populated with both a summary score as well as the user's individual final answers to all the questions.

Some other important requirements:

- Android Back Button**
 - When pressed on Screens 1, 2, 5 or 6, the button should bring the user back to the Android launcher screen.
 - When pressed on Screens 3 or 4, the button should bring the user back to the main question list.
 - At no point should this button make your app crash or stop working.
- Repeated attempts at a single question should display the options in a randomized order every single time.
- All four answer choices to each question should be unique, and valid country names.

Model from a file

You will use a [CSV file](#) containing a list of all capital cities. Place this file in the `res/raw` directory of your project. This file does not have any specific questions in it, however. You will need to write code to generate the questions based on the ground-truth data you're given in this file. Each question should be of the form: "[CITY_NAME] is the capital of which country?" with four options, of which only one is correct. You should use random incorrect country names from the file to populate the wrong choices.

The first thing to implement is the model layer. You should create a `Question` class which represents a single question, and any other data associated with that question. You should also

have a `QuestionListFactory` class, which has static methods for generating a list of questions. One of these static methods will need to read from the raw resource file and turn it into a list of questions.

Here are some links you might find useful:

- [How to read a raw resource file](#)
- [java.io.BufferedReader](#)

The View Structure

You will need to create two fragments for this assignment. The first fragment will hold the `RecyclerView` and the two buttons, and the other fragment will represent the view for a single question.

You will need to show the answer choices via [radio buttons](#). Also take a look at the [RadioGroup](#) and [RadioButton](#) class documentation pages.

When writing the layout for a single element in the `RecyclerView`, be sure to take a look at how [padding and margins](#) work for layouts.

To see how you can create borders for views and layouts, see this [StackOverflow question](#). Also take a look at the documentation for [Shape drawables](#).

The Controller Structure

Make sure to put all fragment-swapping logic inside the Activity, and not inside any fragments. You will also need to figure out how the `FragmentManager.backstack` works, or the Android back button will do weird things. The `FragmentManager.popBackStack()` method should be useful.