# ECE558 Project 2 Design Report

# Android Quiz App

Ryan Bornhorst

5-8-2019

## Introduction

The goal of this project was to gain experience design an Android app that contains more than a single activity. For this project we were assigned with the task of creating a way for a user to take multiple quizzes. The design requirements specified that we had to implement at least two different quizzes of at least eight questions. The quiz questions, choices, and answers had to be created using a .json file format. The first screen upon opening the app greets the user and gives them a list of possible quizzes that they could take. After the user selects a quiz, they can hit the button for starting the quiz which will open a new activity that contains the first quiz question. The app must keep track of the amount of questions a user got correct while they are progressing through the questions. The app also features a 'Cheat' button that allows the user to view the answer if they want to, but application will also keep track of the amount of times a user decides to cheat. The final screen after finishing the quiz will show the user the amount of questions they got correct out of the total amount of questions as well as the number of times that they cheated. From this screen, the user can go back to the main screen and choose to take the quiz again or a different quiz from the list.

## Design Flow

Upon starting the process of coding out the app, I decided to create the main activity screen completely before implementing any of the other activities. I assumed that I was likely to model the following activities after the main screen, but the main activity layout ended up being a lot simpler than the actual quiz activity screens. The splash screen consisted of two TextViews, one that welcomed the user to the app, and another that asked the user to pick from a Spinner list the quiz that they wanted to take. To meet the requirement of having an item in the app that took advantage of SharedPreferences I added the ability for the user to type in their name in an EditText field that would persist through closing and reopening the app or taking more quizzes. Once the user selects a quiz, they can interact with a Button that launches the selected quiz. The MainActivity class will then launch the QuizActivity1 intent and use a call to intent.putExtras() to send the activity information about which quiz the user selected.

QuizActivity1's job is to provide the user with the first question of the quiz. The application gets the quiz information from a .json file located in the res/assets folder within the project. I created a class called JSONAssetLoader, which upon creating an object, will read in the .json file and extract the important pieces of quiz information from the file. These pieces are the quiz questions, choices, and answers for each question. The user can select one of the possible choices using a RadioButton and then hitting the 'Next' Button on the bottom of the app screen. The user may also cheat to get the correct answer by selecting the 'Cheat' button, which launches a new CheatActivity that reveals the correct answer to the user. The QuizActivity that launched the CheatActivity must inform the CheatActivity what the answer was by passing it a string containing the answer to the question when it launches the intent to cheat. The CheatActivity contains an instance variable that it also must pass back to the activity that launched it, so that the activity is now aware that the user has cheated. In my application I also disabled the 'Cheat' Button once a user cheated so that they could not reset the cheat variable. Each of the eight QuizActivities have instance variables for keeping track of if the user answered the question correctly and if they cheated. Each activity passes all the previous activities results along with its own results to the next activity.

Once all the questions have been answered, the FinalActivity gets launched. The role of the FinalActivity is to inform the user how many questions they answered correctly as well as how many times they decided to cheat. The FinalActivity can gather all the data from the previous eight activities by using a Bundle to hold all of their passed in values. Each value that gets passed in contains a unique 'key' identifier that the FinalActivity uses to retrieve the results from each QuizActivity. Since these are Boolean variables, I was just able to check if the values were True to determine how many questions a user got correct, or how many times they chose to hit the 'Cheat' button.

Another thing unique to this project compared to the last one was that because we need to keep track of data that's being passed between multiple activities, we need to save the state of our instance variables when the state changes. There are a few things that can make the application change state, such as rotating the screen and launching new activities. To keep track of instance variable values before a state change, there needs to be an override to the

function onSaveInstanceState(Bundle). This will allow the activity a chance to save its state before it gets destroyed. Also, since the CheatActivity is not launching a new activity, but instead returning to the previous activity, it must use the ability to save its result. This can be done by using the override to the onActivityResult() method, and grabbing the Extra that was saved by the CheatActivity.

**Conclusion**

Although, creating multiple activities that performed the same function was a little tedious, I really enjoyed working on this project. This project gave me a good introduction on what it takes to make a basic Android application. Most applications have many different activities and keep track of a lot of different types of data, so this project was a good introduction into the kinds of things someone designing their first app should consider.