Assignment 2: *Dynamic GUIs, Multiple Activities, Camera, Files*

The purpose of this assignment is to get help you become experienced with Android application programming and using the Android Studio editor, as well as using widgets/views and events to produce an interactive graphical application. In this assignment, you are going to practice what we saw in the first two weeks of class: apps with multiple activities, apps that data, interacting with the device's camera, read data from files.

Instructions

4 different project ideas are presented. You are very welcome to choose any of those or create your own. You must only submit one application.

It is a good idea that you start thinking about your final project (instructions will be posted within one week), and you could start building the skeleton or basic functionalities.

Your app must meet the following minimum requirements (whether you choose one of the suggested apps or you define your own):

- Set up your app as an Android Studio project
- At least 2 different activities
- Your app should pass at least one piece of **"extra"** data between the activities.
  (you need to override the function *onActivityResult*)
- Your app should gracefully handle rotation from portrait to **landscape orientation** (at least for the main activity).
- Your app should read data (text for example) **from a text file** save in your project Resources directory **OR** read data from the strings.xml file in the res/values folder of your project (later in the course we will use a database to store the data)
- Your app should use the camera (take picture and do something with it).
- Your app should use a motion event (swipe for example)

Suggestions

Tic-Tac-Toe Game:

Write a basic game of tic-tac-toe, where two players take turns pressing buttons in a 3x3 grid. If any player can place three of their letters/icon in a row horizontally, vertically, or diagonally, that player wins the game.

You have two options: (simple) write your code as though two human players were playing it on the same screen, (a bit more complex): have the computer play as the second player (randomly, or with certain intelligence).

The app should have a main (initial) activity, where the user can configure her/his profile, the icon to use in the game (cross, circle, custom icon, etc), and any other option that you consider relevant. The user profile should contain at least the user name, and the number of correct/incorrect guesses (you can have only one user, this is enough for this assignment). The user should be able to take a picture with the camera and display it as the user profile. Then main activity should display the instructions of the game. To implement a swipe event, you could for example, change the user profile when the user swipes right. The app could have a rating bar, where the user enters rates the game.

Hang-Man Game:

Make a basic Hangman game that displays an image, along with a word that the player is trying to guess. An image should appear by default. However, the user should have the option to pick an image from the gallery and/or take an image with the camera. To implement a swipe event, you could for example, display a different word to guess when the user swipes right.

The word is chosen randomly from a provided dictionary. You can save the words in a text file inside your project resource directory res/raw. Another option is to use the strings.xml file in the res/values folder of your project. Another option is to create a new Kotlin Class and write here the words to guess.

At all times the game displays a preview of the letters the player has guessed correctly. For example, if the word is "android" and the player has guessed n,i the app should display: "_ n _ _ _ i _". The user should only type single-letter guessed. A robust game would handle multiple-letter attempts gracefully, as well as other errors like trying to guess the same letter twice, etc. The app should tell the user the total number of attempts, and also the remaining attempts left.
The app should have a main (initial) activity, where the user can configure her/his profile, and the maximum length of the word to guess. The user profile should contain the user name, and the number of correct/incorrect guesses (you can have only one user, this is enough for this assignment). Then main activity should display the instructions of the game. The app could have a rating bar, where the user enters rates the game.

Keep My Plants Alive

This app consists on a record of the user plants at home. The idea is that the user keeps track of when was the last time that the plant was watered, and to know when the plant should be watered again.
The main activity should show the plants registered by the user. The plants record could be displayed as a list of plant names or a list of pictures, for example. The user should be able to add new plants form the main activity. A new activity is used to handle the entry of new plants.
When a new plant is added, the user should add a picture taken with the camera or select one from the gallery. Every plant could have more than one picture associated.
When the user selects a plant, a new activity is started. This activity should display some basic information about the plant selected, such as how often the plant should be watered, depending on the season/month. In this activity, the user will type in the last time that the plant was watered, and the app will calculate when it should be watered again.
A more intelligent version (for future work) could add some intelligence to the algorithm, such as the current weather, the humidity of the room, the user feedback, etc.

Give Me A recipe!

The main activity of the app displays a list of food ingredients, with a picture of each one. The user selects some of them, and the app displays a dish/recipe containing those ingredients in a different activity. Optionally, the user could enter a new recipe. Each recipe should have a list of ingredients associated. It is a good idea to create a new Class named Recipe, where the attributes/properties of each Recipe object would be the list of the ingredients, and a description of the recipe.

****Make your own:** they only restriction is that it should meet the minimum requirements presented before.

Submission

Submit your assignment using Canvas, under the Assignment section. The due date is shown in Canvas. You must submit a **.zip** file, containing the following items:

- Android Studio Project folder
- README.txt file (see instructions below)
- video file with a demo (optional for this assignment)

The file named README.txt should contain your name and email address along with the name of your app and a very brief description of it, along with any special instructions that the user might need to know in order to use it properly (if there are any). For example:

```
Laura Arjona <arjonal@uw.edu>
Sorting Hat - This app displays the sorting hat and a picture of the user. It returns the house where the user belongs.
```

Grading

Your submission will be graded by building and running it and evaluating its functionality. Your code will not be graded on style, but we still encourage you to follow good overall coding style for your own good. See the appendix about Model.

Appendix: Model-View-Controller in Android

This website presents a good example of an App using the MVC model
https://openclassrooms.com/en/courses/4661936-develop-your-first-android-application/4679186-learn-the-model-view-controller-pattern

Android applications are designed around an architecture called Model-View-Controller, or MVC for short. MVC states that any object in your application must be a *model object*, a *view object*, or a *controller object*.

• A *model object* holds the application's data and "business logic." Model classes are typically designed to *model* the things your app is concerned with, such as a user, a product in a store, a photo on a server, or a television show. Or a true-false question. Model objects have no knowledge of the user interface; their sole purpose is holding and managing data.
In Android applications, model classes are generally custom classes you create. All of the model objects in your application compose its *model layer*.

• *View objects* know how to draw themselves on the screen and how to respond to user input, like touches. A simple rule of thumb is that if you can see it on screen, then it is a view.
Android provides a wealth of configurable view classes. You can also create custom view classes.
An application's view objects make up its *view layer*.

• *Controller objects* tie the view and model objects together. They contain "application logic." Controllers are designed to respond to various events triggered by view objects and to manage the flow of data to and from model objects and the view layer. In Android, a controller is typically a subclass of **Activity**, **Fragment**, or **Service**.

This figure shows the flow of control between objects in response to a user event, like a press of a button. Notice that model and view objects do not talk to each other directly; controllers sit squarely in the middle of everything, receiving messages from some objects and dispatching instructions to others.