Assignment 1: *Dynamic GUIs and Multiple Activities*

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, including GUI design, adding events to widgets, and adding multiple activities and passing data among them. This assignment is to be completed individually.

Description

Three different project ideas are presented. You are very welcome to choose any of those or create your own (this is an opportunity to use your creativity and preferences!). You must only submit one application.

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 not re-start the activity when rotating from portrait to **landscape orientation.** Hint: see note at the end of document.
- Your app should use several (more than 2) widgets and several widget events.

Choices

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

B. Break 10/10

The purpose of this App is to define a 10 min / 10 exercises workout to complete during our course breaks.

- The idea is that every time the user starts a workout, a sequence of 10 different exercises should be randomly generated, by selecting 10 exercises from a predefined list of exercises. You can use a *ListView* to display the list of exercises.
- This pre-defined list should have 15 different exercises (or more if you want).
- When the user selects any of the exercises from the list in the first activity, the app should display a Dialog or *TextView* with the instructions for the selected exercise, and/or a picture of a demonstration (you can use an I*mageView*). The picture should be added to the res/drawable folder.

An example app would be the following. You are welcomed to adapt it to your preferences.
The main activity contains a predefined list of exercises/stretches (for example, burpees, knee-chest, back stretch, etc). When the App starts, there should be a default number of repetitions for each exercise. Then the user can modify the number of repetitions for each exercise that thinks feasible for 1 minute. Then, the user will press a start button. The activity will **randomly** select 10 of those exercises from the list and pass the data (10 selected exercises and their repetitions) to the second activity. Start the second activity using an Intent and put into the intent the 10 selected exercises and the corresponding repetitions selected by the user. In the second activity, the user will see a list of the 10 exercises generated and the number of repetitions. When the user is done, the user will press the end/back button to return to the main activity.

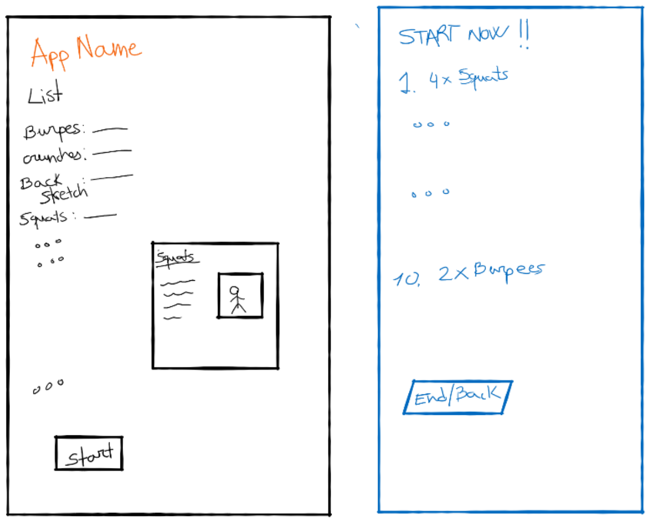
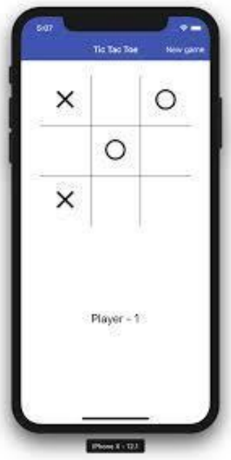


Figure 1: Example structure of App

C. 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 if you are an experienced programmer).
- The app should have a main (initial) activity, which displays the game instructions, and lets the user choose the icon to use in the game (for example, cross, circle, custom, etc). You may include any other settings that you consider relevant.
- The main activity should have a start game button, and when the user clicks on it, a second activity should start displaying the game. Start the second activity with an Intent and put into the Intent the icon type that the user selected in the main activity.
- The game activity should notify the user when the game is over, and who won the game (you can use a TextView, Toast, snackBar, Dialog, for example)

D. *MemoryMatching* Game:

This is a one-player game. You can learn the game logistics by playing yourself https://dkmgames.com/Memory/pairsrun.php

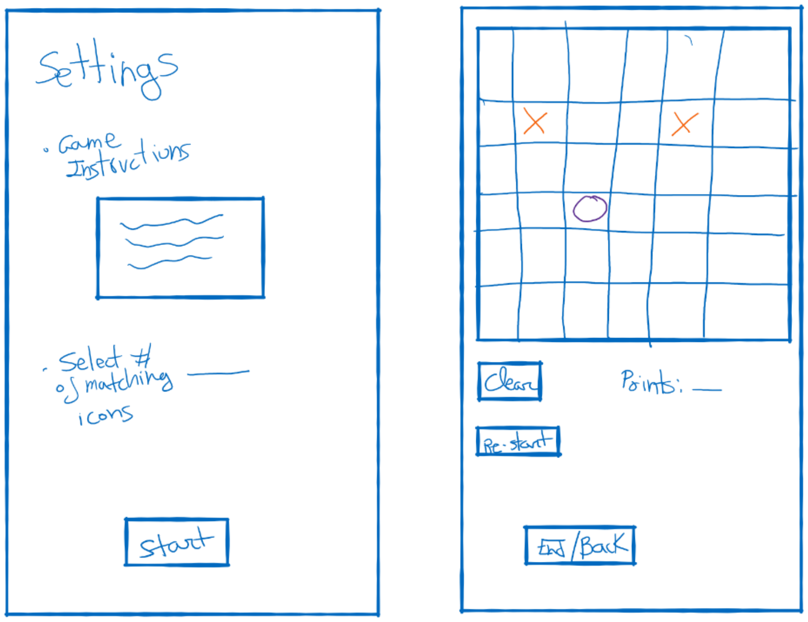


Figure 2: Example structure of App

- You can implement the game as you prefer but make it simple. For example, you can use buttons as the tiles.
- The app should have a main (initial) activity, which displays the game instructions, and the user can select the number of required matching icons (2 is the default). You may include any other settings that you consider relevant.
- The main activity should have a start game button, and when the user clicks on it, a second activity should start displaying the game. Start the second activity with an Intent and put into the intent the number of matching icons required (the value that the user entered in the main activity).
- If the user clicks on 2 (or more) buttons whose icon does not match, then the user should press the button "clear" to try again.
- Set the tiles/buttons icons randomly at every new game.
- You can implement a points-system if you want. Give points to the user for every successful attempt and subtract points for every non-matching attempt.
- The game activity should notify the user when the game is over, and who won the game (you can use a TextView, Toast, snackBar, Dialog, for example)
- The game activity should have a Back button. When the user clicks on it, it should return to the main activity
- Note that the game logic can be as simple as you like, as long as you explain the game rules.

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
- Short video demoing the App
- README.txt file (see instructions below)

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 (0-55 points)

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.

- 0-25 points: the app meets all the minimum requirements.
- 0-20 points: the app builds and runs without errors and the functionality is correct.
- 0-10 points: The User Interface is well organized.

Getting help and resources

- Installing Android Studio and using the smartphone or virtual device: refer to the document *Local_platform_setup.pdf* on Canvas.
- For the board games, the most simple but efficient structure is to use a grid of buttons.
- To create list, a very simple way (one option, but there are more) is to use a *ListView*. Refer to the example App provided in the Assignment folder in canvas named *ListViewExample*.
- Getting help: Use Canvas discussion section, Piazza, and office hours.
- Review Slides from Lecture 1 and 2, and class examples.

Note: Simple strategy to retain your activity's GUI state on rotation
A quick way is to set the `configChanges` attribute of the activity in AndroidManifest.xml. (note that this doesn't solve the other cases like loading other apps/activities)

In your Manifest file:

```
<activity android:name=".MainActivity"
android:configChanges="orientation|screenSize"
```