Homework 1

The purpose of this assignment is to demonstrate your understanding of views, view controllers, and the relationship between them; view controller containment; Interface Builder, actions, and outlets; and the view loading process.

In addition, this assignment is designed to increase your familiarity with the process of reading and using Apple's documentation, as well as other external resources like Stack Overflow. Some portions of the assignment leave implementation choices to you.

In order to successfully complete this assignment, you must successfully implement all of the Functional Requirements below. As with all apps that you submit for this class, you should also:

- Ensure that your app does not crash or behave in an unstable manner
- Make use of good design patterns (e.g. MVC) and structure your app accordingly
- Follow the Human Interface Guidelines when designing your app's UI
- Test your app on an actual device running the latest version of iOS

Functional Requirements

The app should consist of two view controllers managed by a tab bar controller. The two tabs should be named "Colors" and "Data." This assignment includes images `colors.png` and `data.png`, each with corresponding `@2x` and `@3x` variants; the app should display these images in the appropriate tabs.

Colors tab

On the first tab, you should include five buttons and three text fields:

- Three buttons labeled "Red," "Green," and "Blue" in a row
- Three text fields in a row immediately above a "Custom" button
  - The text fields should have placeholder text of "red," "green," and "blue"
- A "Random" button on its own row

Each row should be fully visible on screen, leaving at least a small margin on the left. You may choose any amount of space to separate the groups.

When any button is pressed, you should display a modal view controller with the appropriate background color:

- For the “Red,” “Green,” and “Blue” buttons, use a solid background of that color
- For the “Custom” button, interpret the text fields’ text as floating-point values and create a new UIColor from those values
- For the “Random” button, use a random color[1]

The modal controller that you present should display two things:

- A label that indicates how many times the original button was pressed
- A “Dismiss” button that, when pressed, dismisses the modal controller

Data tab

On the second tab, you should include five rows of two labels each. Every row should have:

- One label that names a modal controller color (“Red,” “Green,” “Blue,” “Custom,” or “Random”) from the first tab
- One label that gives how many times that color has been presented

These counts should update as often as necessary so that they never appear inaccurate.

In addition, the second tab should have a button labeled “Reset” under the other labels. When pressed, the Reset button should clear all the counts on the data tab, resetting them to zero. This reset should also be reflected on future presented modal controllers.

Bonus Opportunities

You may receive up to 10 bonus points for adding a confirmation sheet to your Reset button. This confirmation sheet should use the UIAlertController class configured using the UIAlertControllerStyleActionSheet style.

Present the sheet when the Reset button is tapped, and provide two options: “Reset” and “Cancel.” Only when “Reset” is pressed should you proceed to set the data counts back to zero. If “Cancel” is pressed, dismiss the sheet and do nothing else.

To learn more about UIAlertController and its use, see the Apple class documentation.

---

[1] This assignment includes a category `UIColor(UWExtensions)` to help you create a random color.

Submitting Your Work

To submit your work, upload a .zip file to the appropriate drop box that contains your entire Xcode project directory, including:

- Your .xcodeproj bundle and all its contents
- All your source files, including code, .xib files, and any image resources
- Any additional files that your app requires to run

Name this zip file "UW_HW1_<*UW NetID*>.zip", where <*UW NetID*> is the username assigned to you by UW. (For example, the instructor's submission would be named UW_HW1_tekl.zip.)

Your submission should compile cleanly on the first try, throwing absolutely no errors, warnings, or static analyzer problems. You may lose points if your solution does not compile cleanly.

Your submission should, once compiled, run well on an actual device running the latest version of iOS. You may choose to support past versions of iOS, but all testing will be done on the newest version available on the due date.