CIS 257 Programming Lab 6 – **Wellness app (part 1)**

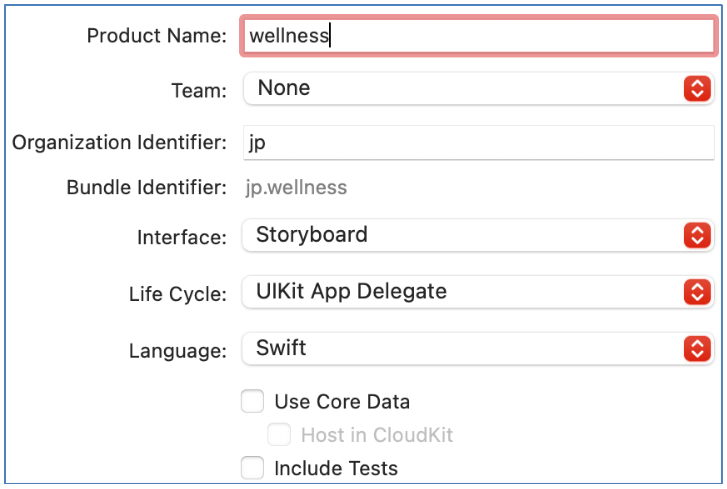
This lab will be part one of a two part app that will get you acquainted with storyboarding an app. Having a storyboard will allow for *multiviews* to help the user to navigate easily through your wellness app. The actions to go from view to view will be connected via a **Tab Bar Controller**. For this lab part you will create a Home view screen and a BMI screen view which will calculate the BMI of an indvidual.

See page 5 of this lab for an early view of how the app will initially look like.

Step 1. Setting up storyboard area

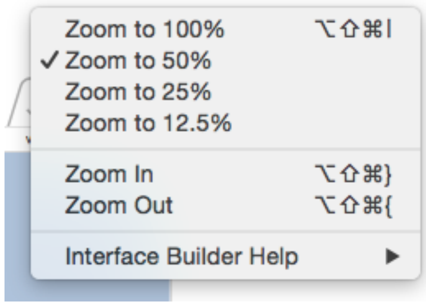
Open Xcode and create a Single View project. Call the app project **wellness**. Add any Organization Identifier you like if required.   Keep defaults and for **Interface** choose **Storyboard** as the option. You can choose to uncheck the option Include Tests.

Ex. snapshot of Xcode 12.4



Product Name: wellness
Team: None
Organization Identifier: jp
Bundle Identifier: jp.wellness
Interface: Storyboard
Life Cycle: UIKit App Delegate
Language: Swift
Use Core Data [ ]
Host in CloudKit [ ]
Include Tests [ ]

To make some room for storyboarding, open up your Main.storyboard file and right click onto any white space area and choose to set Zoom at 50%.

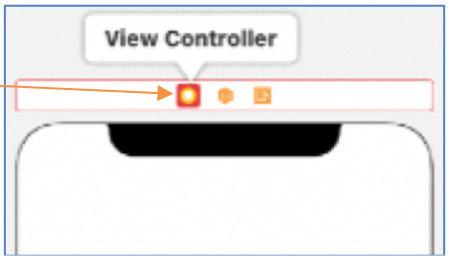


Step 2. Making a Tab Bar Controller

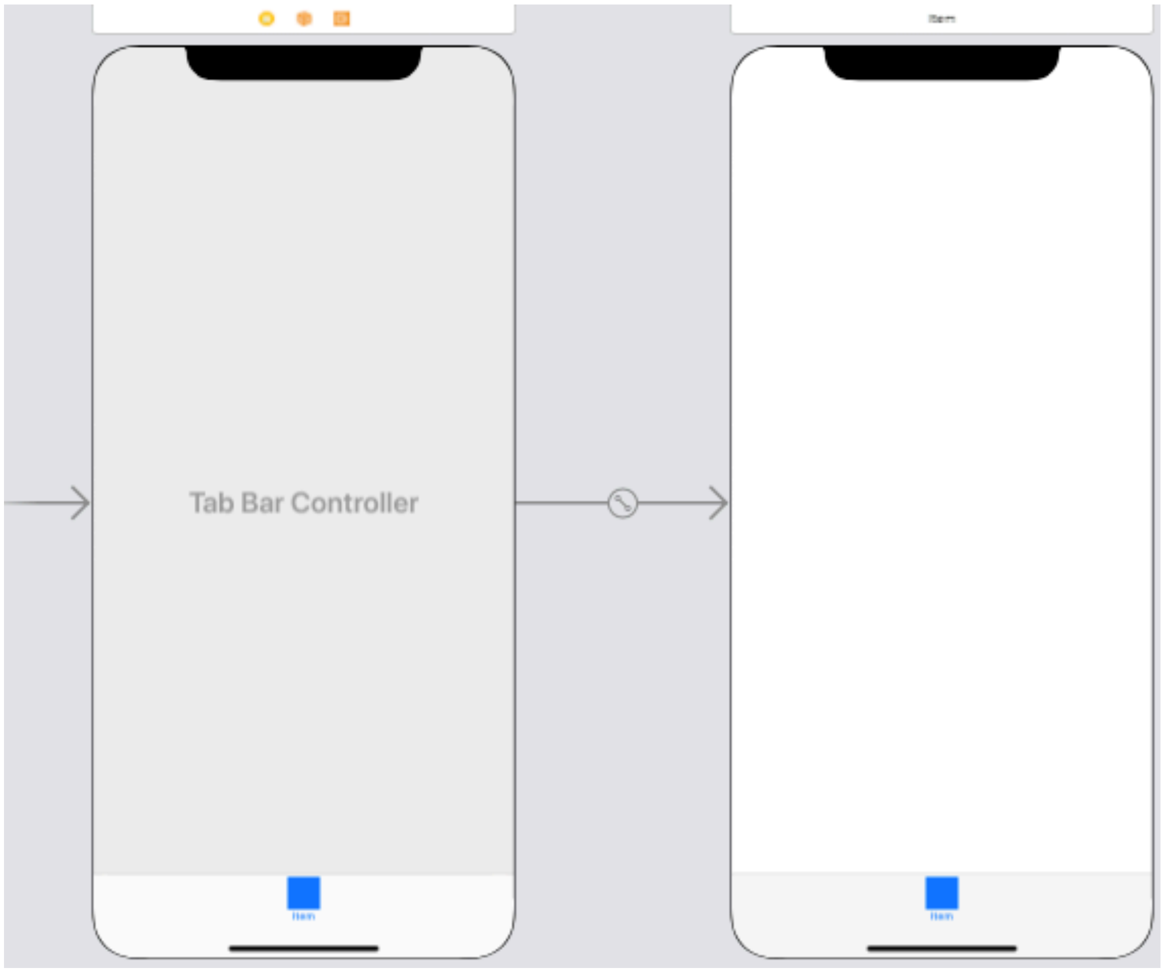
Click on one of your borders of the default view controller so all the borders focus on the entire Controller, then from your menu go to **Editor>Embed In>Tab Bar Controller**. Voila – you now have a

Tab Bar Controller on your stage! Notice what happened, your ViewController has now been attached to a Tab Bar Controller (the parent) and has become a subview or child, in with a single tab item!

Click on View Controller symbol to grab focus of object



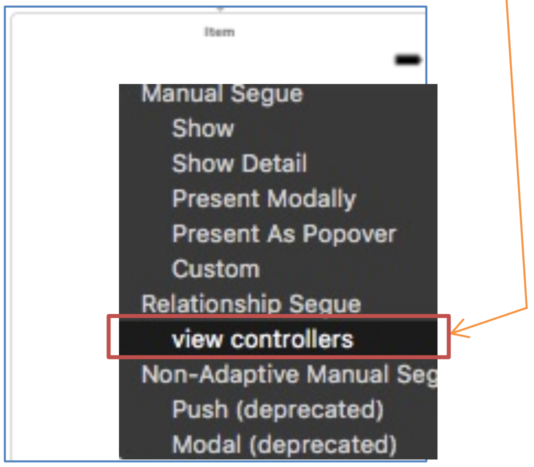
Snapshot of Tab Bar Controller add on



Step 3. Adding in another subview to your Tab Bar Controller

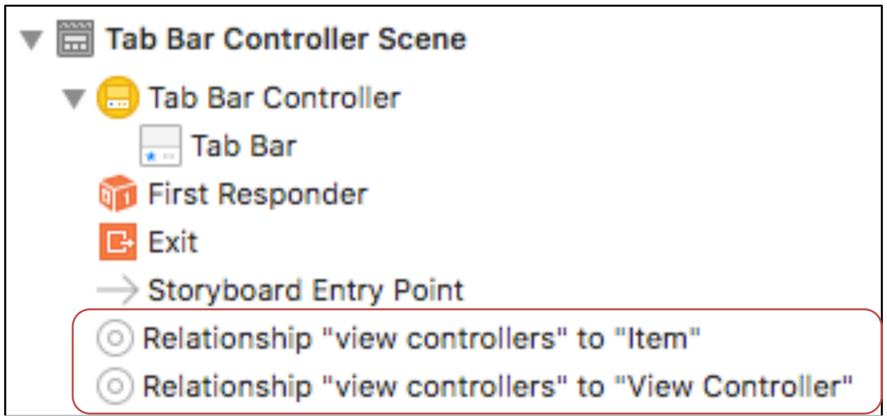
Let's now add in another subview to complete your storyboard. Go to your object libray and drag in a new **View Controller**. This will serve as another child to the parent controller *stack*.

Next, to add or connect the new View Controller to the Tab Bar Controller, click on your Tab Bar Controller and press and hold your Ctrl key and then right click on your mouse and drag over a connection to your *new* View Controller, then release your mouse click and choose from the pop up menu under Relationship Seque, the **view controllers** option.



You now have two 'subviews' attached to your parent or master controller. Your new View Controller you'll notice has an automatic Tab Item added in because it's part of the Tab Bar controller stack and therefore inherits a Tab Item!

Notice now the Relationships now under your Tab Bar Controller Scene. Relationships will obviously be built onto the stack each time you create and attach a new subview onto your master stack, as just performed.


Tab Bar Controller Scene
Tab Bar Controller
Tab Bar
First Responder
Exit
Storyboard Entry Point
Relationship "view controllers" to "Item"
Relationship "view controllers" to "View Controller"

You'll hopefully notice that when you add in any new View Controllers onto your stage that you do not however have a corresponding **ViewControllerSomeNumber/Name.swift** file automatically added to your project folder. That's alright as you can add that logic in anytime and have an added **ViewController** be associated with an added **swift** file, so you can do some programming for the new view and not just have a view with objects placed on it.

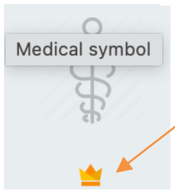
Step 4. Adding icons and title to your subviews.

So you know which *subview* controller does what, let's at least add in item titles and icons so you and the user can easily identfy what view does what.

Grab a few icons that are medically or 'wellness' associated from the web. A good site maybe here:

http://www.freepik.com/free-icons/medical

From that site, click on a free icon of your choice, appropriate let's say for the intial view to sort of an icon depicted for a Home/Welcoming view). Avoid any icon with the Premium icon under the icon itself as shown below.



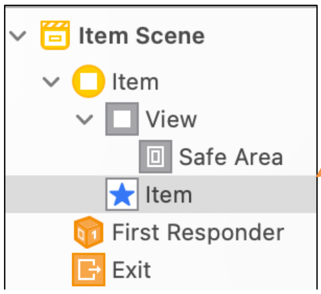
Then at the next page, click at the bottom on the right hand side just click the button (size 32px is good from the dropdown choices) and then at the Download options pop up display, click on the button and your file will download.

Do the same steps as above for another icon you'll need for your second view, this time the icon should associate with a BMI calculator session which will help determine a user's overall health status given the user's weight and height. Note- you can rename your icons to something easy like, BMI, heart, home, etc.

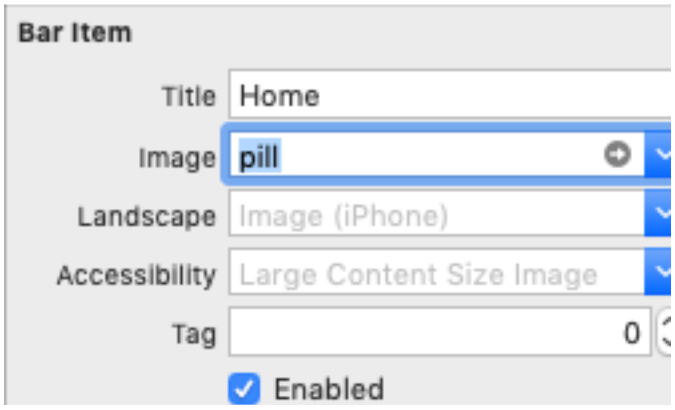
To continue, drag and drop your images into your projects folder in any designated area you see fit. If you like, you can drag the icons into your **Assets.xcassets** folder area (double click the folder first to open it up) which helps set sizes based on your phone/tablet dimension possibilities (ex. 1x/2x/3x sizes).

Note the names of your icons (mine are called life3 and pill) so you'll know which icon should be placed as an image to a desired Tab Item.

From your Document Outline view or within your Controller, click on the **Tab Item** of your very first (i.e. initial) controller you had and go to your **Attributes Inspector** and add in the following...



Under **Bar Item,** add in the following **Title** and your choice image (by name only) as depicted below...


Bar Item

Title: Home

Image: pill

Landscape: Image (iPhone)

Accessibility: Large Content Size Image

Tag: 0

✓ Enabled

Do the same as above for your newly added controller and this time add the Title **BMI** and an alternative image than your first view.

Cool! At this point, if you like, run your app. You should see each item appear with titles and varying icons which become highlighted when pressed. If something is amiss with your app, retrace steps above over again till you get things right.

Okay now onto the action. Let's get our subviews populated and rocking with some nice images and interactive features. Try to make your app, "theme" based, matching the look and feel of things across all subviews with varying but complimentary colors, shadings, font colors and sizes, imagery, etc... plus think on some cohesive concepts your app can use to benefit the users along the way that will make for even a greater *more* robust app (you'll add in more concepts for the next lab)!

For your first or initial view, add in a nice image from your favorite search engine that will serve as a background to the entire view (see next page for snapshot). You can place your image into the **Assets.xcassets** folder area again.

Next, add in an **ImageView** from your Object Library to the View of the home controller. From your **Attribute's** Inspector under the **Image View** section, select your image for the control and from the **View** section, choose **Aspect Fill** for your image sizing and select a desired image file to fill up your view.

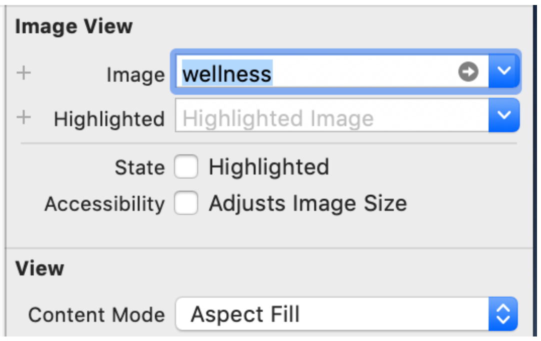


Image View

Image: wellness

Highlighted: Highlighted Image

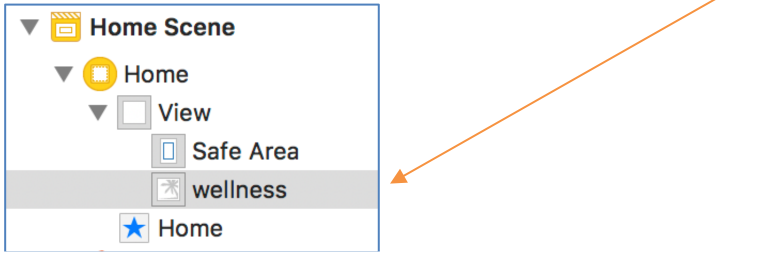
State: ☐ Highlighted

Accessibility: ☐ Adjusts Image Size

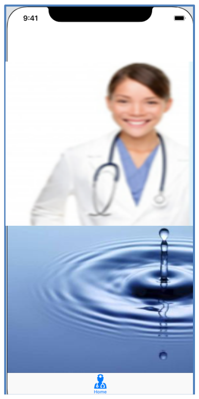
View

Content Mode: Aspect Fill

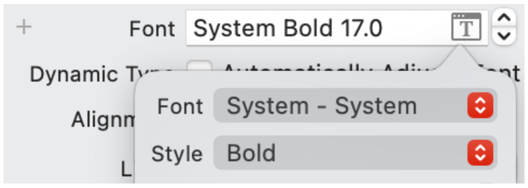
Make sure to stretch out your ImageView amply to fit enough of your view area (grab focus of your image to allow for stretching). Leave some headroom at the top for a label to be added next.



Shot of image resized with some head room at the top



Next place a label towards the top of your view and label it accordingly with your app's desired title, for ex., "The Wellness Clinic" perhaps. From the **Attributes** Inspector make the title text bold as well.



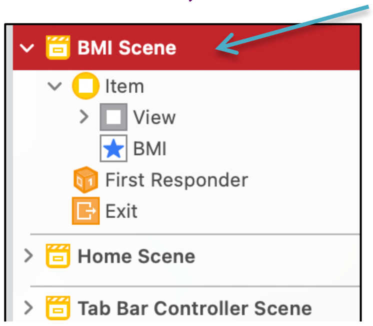
Font System Bold 17.0

Font System - System

Style Bold

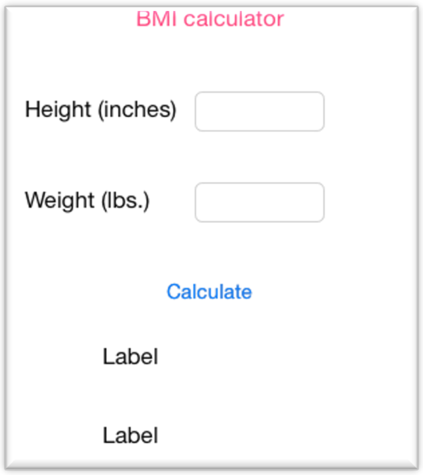
Note- if you would like to preview how your image(s) and label(s) or other controls for that matter looks "live" at anytime without running your emulator, go to **Editor > Preview** on your menu. You now should have a nice splitview to work with setting up designs for any particular view "focused" on, from the left side of your splitview and seeing a preview of the view on the right, which acts like a live view!

Step 5. Adding the UI design to your second view, now titled BMI.



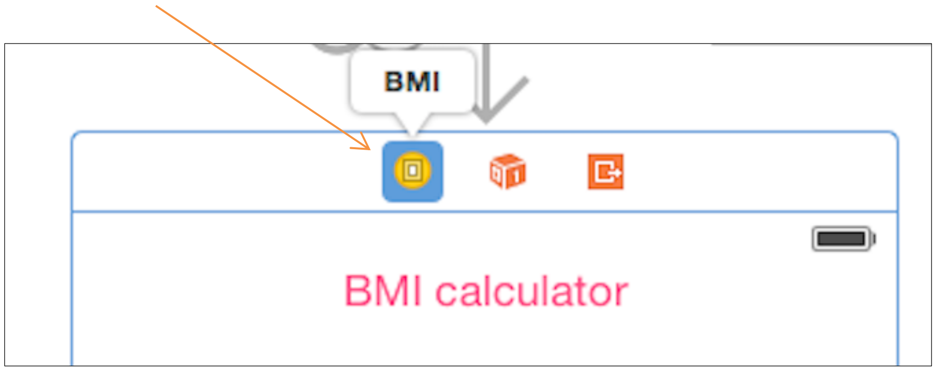
For your **BMI controller** lets create a simple UI to grab the users height and weight and with a click of a button, you'll calculate the user's BMI and display the resulting BMI value to the user along with a response of their weight status as detemined by the WHO or World Health Organization.

To create an appropriate UI for your BMI controller, add labels (5), text fields (2) and a button for a completed UI. An example follows...



Now to code things you need to attach the **View controller** to some swift file. Well we have the **ViewController.swift** file on hand when we created a Single View project. Let's use that.

To connect with the swift file as a reference to the BMI controller, click on the entire controller (again a good idea is to click the BMI object icon at the top of the controller so the whole controller is focused on,



and go to your **Identity Inspector** and enter in **ViewController** as the **Class** name under the **Custom Class** section, if its not defaulting to the class name. Press enter to commit your entry. ViewController is actually the swift file name, so it will map out to the BMI controller correctly for you no problem! 😎

Step 6. Adding functionality to your ViewController.swift file.

From your menu go to **Editor>Assistant** to easily have muliptle views between your Swift file and your storyboard.

Click on your ViewController.swift file and add in IBOutlets for each of your text fields from your storyboard, with the following names as follows:

```
    @IBOutlet weak var txtHeight: UITextField!
    @IBOutlet weak var txtWeight: UITextField!
```

Add in IBOutlets for each of your labels (**lblBMI** will be for the first label under your button and **lblStatus** will serve as the second label name under your button) with the outlet names as follows:

```
    @IBOutlet weak var lblBMI: UILabel!
    @IBOutlet weak var lblStatus: UILabel!
```

Create next an IBAction for your button (keep other settings as defaults) with the following name

```
    @IBAction func calc(_ sender: UIButton) {   }
```

Next code your `calc` function for your button action as follows...

```
    @@IBAction func calc(_sender: UIButton) {

        let wt = Double(txtWeight.text!)
        let ht = Double(txtHeight.text!)

        let BMI  = ((wt! * 703.0) / (ht!*ht!));  //BMI formula

        lblBMI.text = (BMI as NSNumber).stringValue
   }
```

Notice the use of **!** in the syntax to allow for the unwrapping of data types (ex. String->Double)

Step 7. Run and test your app.

Run your app and choose your BMI tab item (should be rightmost icon). Enter into your text fields both your weight (be honest now) and height (in inches) values, then press your Calculate button to see if a BMI result is shown. Always make room for your labels so a full display is shown.

To enter in text fields directly with the help of the built in IOS keyboard you can always use the command shown below, to toggle the keyboard Hardware to on /off.

Command-K -Toggle (Show/Hide) Keyboard

Step 8. Modify your app to show the result of the BMI status

Modify your function such that the BMI result (aka your BMI variable) yielded can be used against the following data chart to determine the health status of the individual. Display the result of the status appropriately into your **lblStatus** label.

BMI yielded result:	**Health status label response:**
BMI less than 18.5	- Underweight
BMI between 18.5 and 25	- Healthy weight
BMI between 25 and 30	- Overweight
BMI between 30 and 40	- Obese
BMI over 40	- Very obese, morbidly obese

Also give a pop up warning message if the person reaches an Obese type status, something like
Lose Weight now or Dr. Feelgood will be calling on you verrrry soon!

An alert message or pop up can be coded simply as the following example shows:

```
let alertController = UIAlertController(title: "My App", message: "Hey",
    preferredStyle: UIAlertController.Style.alert)
alertController.addAction(UIAlertAction(title: "Ok", style:
    UIAlertAction.Style.default, handler: nil))
self.present(alertController, animated: true, completion: nil)
```

Step 9. Submissions for credit

Snapshot results into a Word doc results for someone who is healthy AND someone who is obese. Show your pop up message as well, flagging Feelgood's message of dire warnings! Submit also your source code for your ViewController.swift file for full possible credit.

Next lab will be a continuation of this using more subviews with mapkit and also a view depicting a health video! Oh yesss! Think of added concepts you would like to add to your project, error trappings & also what controls would be needed to make your app fully functional, intuitive and allow for the ease of user interfacing! Perhaps a tableview! Check your text for that as a guidance.

To be continued...

Good luck as always.