



# Don Miller

Presenting:

# Getting Off the Ground with iOS Development

Don Miller  
Founder of GroundSpeed™  
Co-founder of Seed Coworking  
@donmiller



Don Miller | GroundSpeed™  
rapid web + mobile software

# EXPECTATIONS

- Very casual and safe learning space
- Ask questions, be playful. Be the mad scientist you always wanted to be
- I will discuss and demo and then challenge you to create what we just discussed or something similar to what we just covered.
- Somethings will be repetitive by design. If you are annoyed you will probably never forget it. 😊
- I have taught a similar format to this many times; however, this is the first time not in person. Let's be patient with one another knowing that our goal is to get you a foundation for iOS development



# WHAT WE ARE NOT GOING TO COVER

- AutoLayout

Although we will chat about sizing and some layout control.

- SwiftUI

But you would probably be one of the two other workshops if you were looking for SwiftUI. Wednesday I have a talk about XIB, Storyboard, and SwiftUI.

- Everything

There is so much to discuss the we cannot cover everything in one workshop. This will give you the tools to get some momentum

- MVVM

Here is a great resource for learning MVVM by Bart Jacobs:

<https://cocoacasts.com/swift-and-model-view-viewmodel-in-practice>



# SCHEDULE

Introduction

Discuss Terminology, Xcode Overview

Create Your First App: Hello World!

Swift Basics and the Playground

MVC Overview

Color Me Demo

Storyboard Navigation

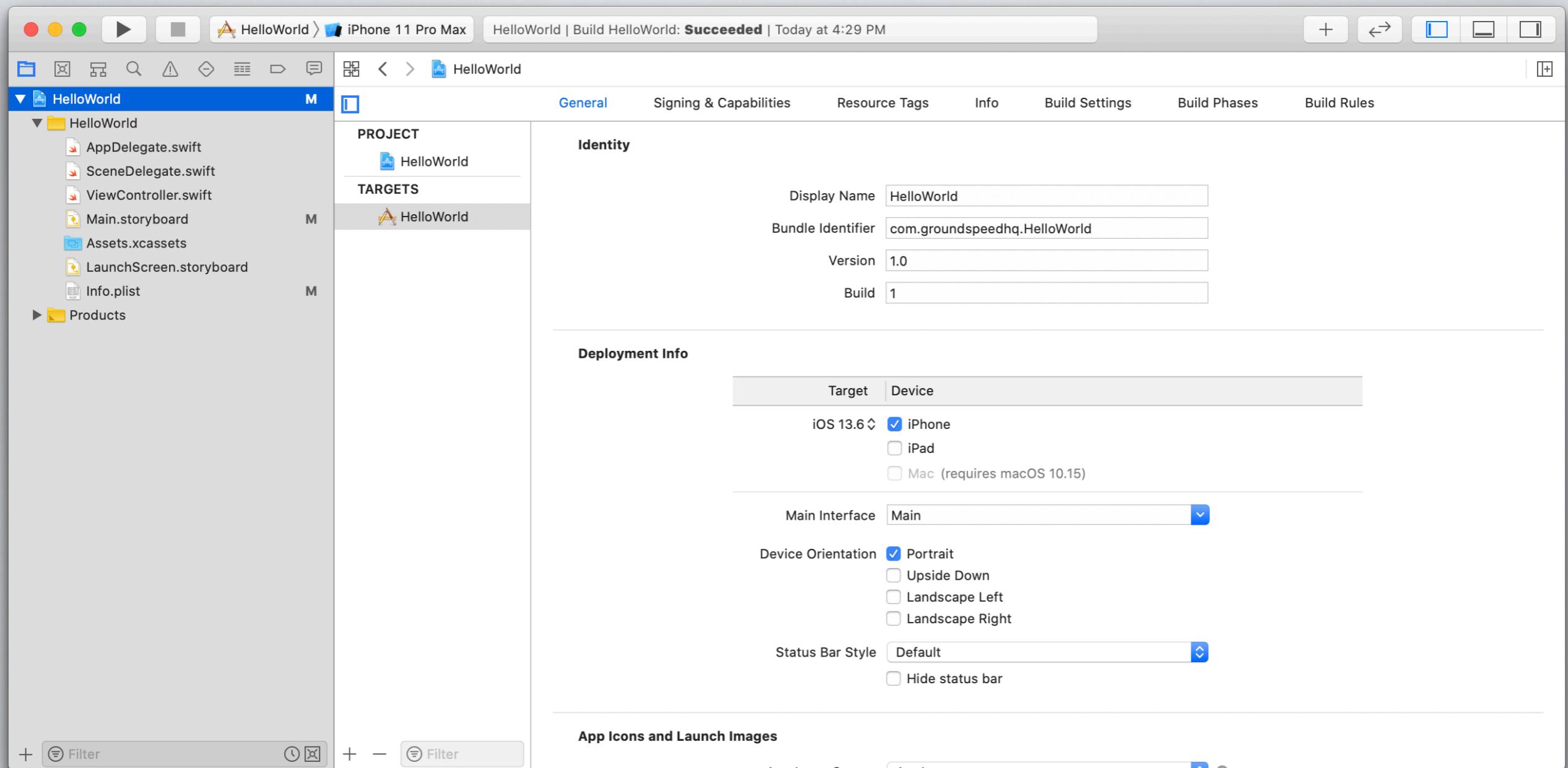


# TERMINOLOGY

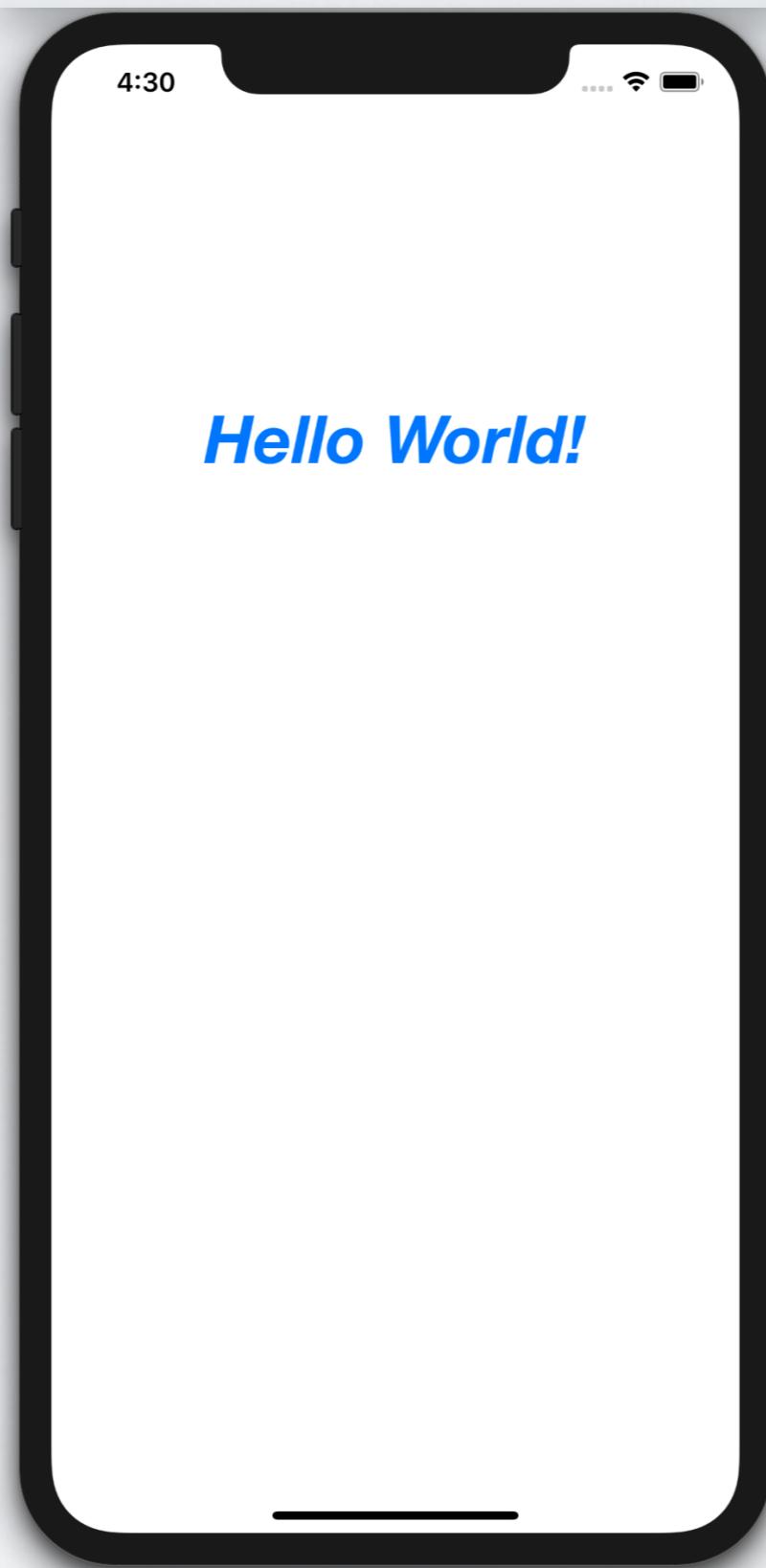
- ▶ iOS
- ▶ iPhone
- ▶ iPad
- ▶ MacBook
- ▶ iMac
- ▶ SDK
- ▶ Xcode
- ▶ OSX
- ▶ Catalina (10.15)
- ▶ Mojave (10.14)
- ▶ High Sierra (10.13)
- ▶ Sierra (10.12)
- ▶ El Capitan,  
Yosemite,  
Mavericks,  
Mountain Lion, etc.



# XCODE 11.X & IOS 13.X



# HELLO WORLD!



# CREATE A NEW SINGLE VIEW APPLICATION

Choose a template for your new project:

iOS watchOS tvOS macOS Cross-platform Filter

**Application**

 Single View App	 Game	 Augmented Reality App	 Document Based App	 Master-Detail App
 Tabbed App	 Sticker Pack App	 iMessage App		

**Framework & Library**

 Framework	 Static Library	 Metal Library
--	--	--

Cancel Previous Next



# ENTER THE PRODUCT NAME

Choose options for your new project:

Product Name:  HelloWorld

Team:  None ▼

Organization Name:  GroundSpeed

Organization Identifier:  com.groundspeedhq

Bundle Identifier: com.groundspeedhq.HelloWorld

Language:  Swift ▼

User Interface:  Storyboard ▼

Use Core Data

Use CloudKit

Include Unit Tests

Include UI Tests

Cancel Previous Next



# UPDATE THE GENERAL PROJECT ATTRIBUTES

The screenshot shows the Xcode interface with the "General" tab selected in the top navigation bar. The left sidebar displays the project structure for "HelloWorld". The main area shows the "Identity" section for the "HelloWorld" target, which includes fields for Display Name (HelloWorld), Bundle Identifier (com.groundspeedhq.HelloWorld), Version (1.0), and Build (1). Below this is the "Deployment Info" section, which specifies the target as "Device" and "iOS 13.6". It also includes options for Main Interface (Main), Device Orientation (Portrait checked, Upside Down, Landscape Left, Landscape Right), and Status Bar Style (Default, Hide status bar). Further down are sections for App Icons and Launch Images (App Icons Source: AppIcon, Launch Screen File: LaunchScreen) and Frameworks, Libraries, and Embedded Content (Name: Embed). The right side of the screen contains tabs for Identity and Type, Project Document, and Text Settings.

>HelloWorld > iPhone 11 Pro Max HelloWorld | Build HelloWorld: **Succeeded** | Today at 4:29 PM

General Signing & Capabilities Resource Tags Info Build Settings Build Phases Build Rules

**PROJECT**  
HelloWorld

**TARGETS**  
HelloWorld

Display Name: HelloWorld  
Bundle Identifier: com.groundspeedhq.HelloWorld  
Version: 1.0  
Build: 1

**Deployment Info**

Target: Device  
iOS 13.6:  iPhone  
 iPad  
 Mac (requires macOS 10.15)

Main Interface: Main

Device Orientation:  Portrait  
 Upside Down  
 Landscape Left  
 Landscape Right

Status Bar Style: Default  
 Hide status bar

**App Icons and Launch Images**

App Icons Source: AppIcon  
Launch Screen File: LaunchScreen

**Frameworks, Libraries, and Embedded Content**

Name: Embed

Add frameworks, libraries, and embedded content here

**Identity and Type**

Name: HelloWorld  
Location: Absolute  
Full Path: /Users/donmiller/Projects/Speaking/360iDev-2020/HelloWorld/HelloWorld.xcodeproj

**Project Document**

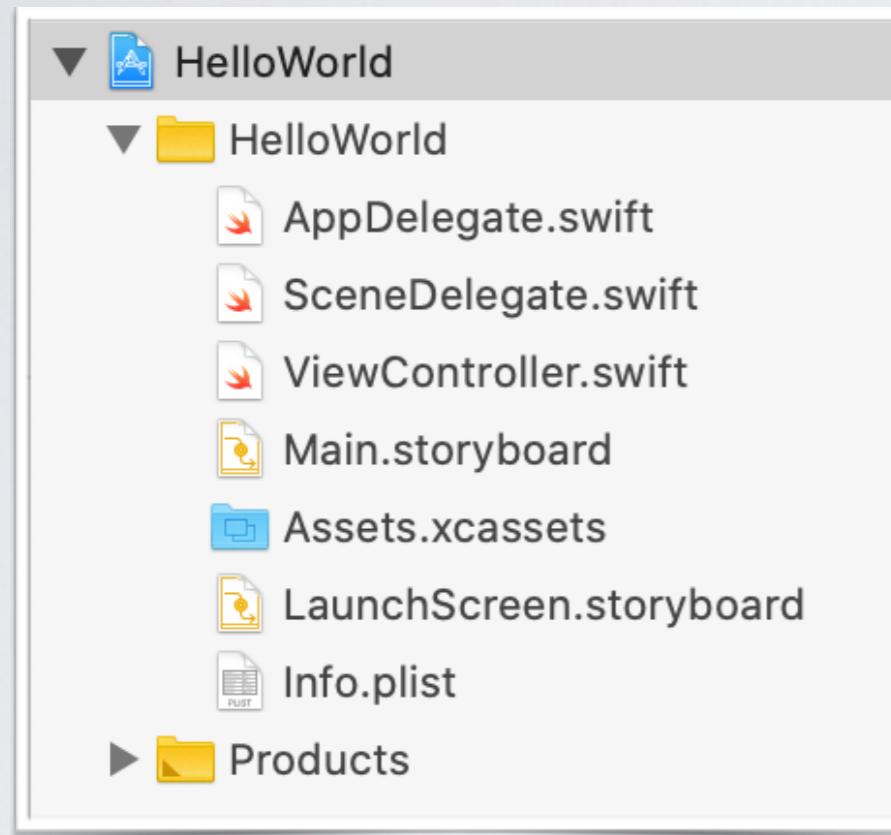
Project Format: Xcode 9.3-compatible  
Organization: Don Miller  
Class Prefix:

**Text Settings**

Indent Using: Spaces  
Widths: 4 Tab, 4 Indent  
 Wrap lines

+ Filter + - Filter

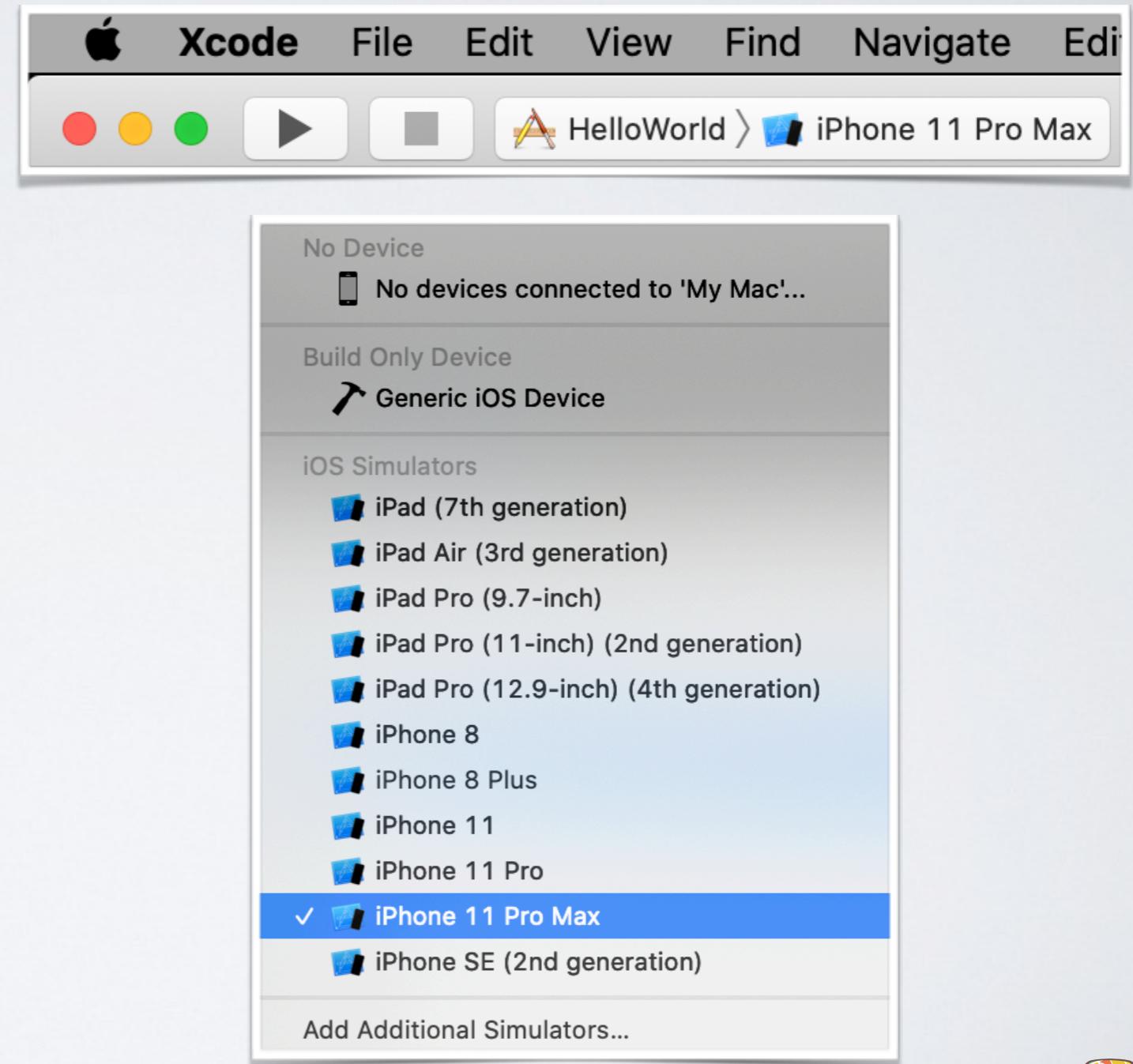
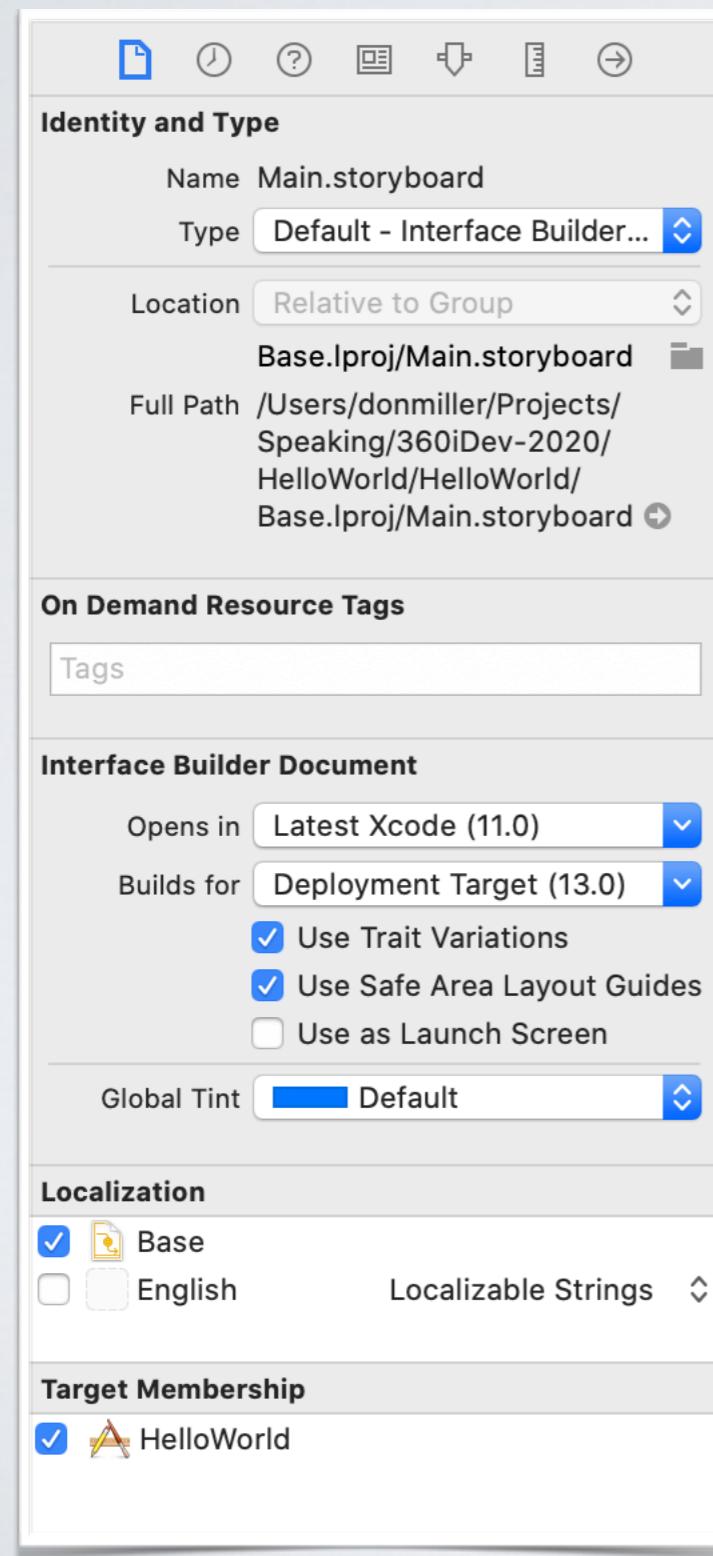
# PROJECT STRUCTURE



- Folders do not always match physical layout on file system.
- Actually you will not find any sub folder named like Supporting Files in your physical project folder.
- The folder hierarchy in the XCode's navigator is simply a logical grouping of the resources.
- At this stage, we would be interested only on the Main.storyboard file.



# FILE INSPECTOR & SCHEMA SETTINGS



# DEVICES & TARGETS

General    Signing & Capabilities    Resource Tags    Info    Build Settings    Build Phases    Build Rules

**PROJECT**  
HelloWorld

**TARGETS**  
HelloWorld

**Identity**

Display Name: HelloWorld

Bundle Identifier: com.groundspeedhq.HelloWorld

Version: 1.0

Build: 1

**Deployment Info**

Target	Device
--------	--------

iOS 13.6 ▾  iPhone  
 iPad  
 Mac (requires macOS 10.15)

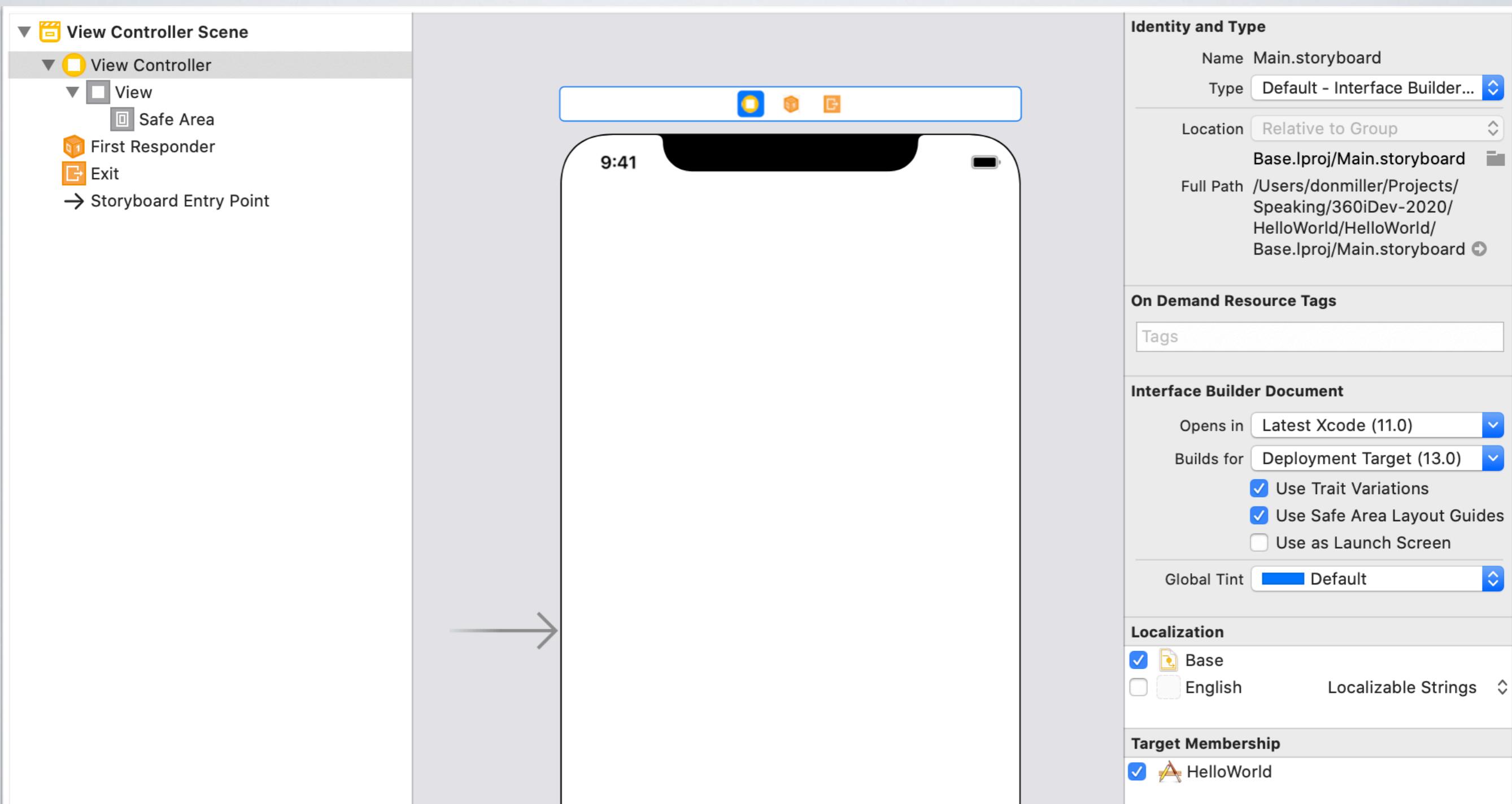
Main Interface: Main

Device Orientation:  Portrait  
 Upside Down  
 Landscape Left  
 Landscape Right

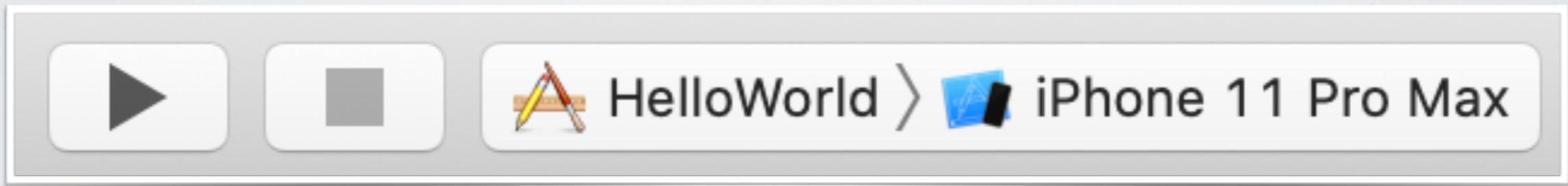
Status Bar Style: Default  
 Hide status bar



# FILE OWNER, VIEW, & FIRST RESPONDER

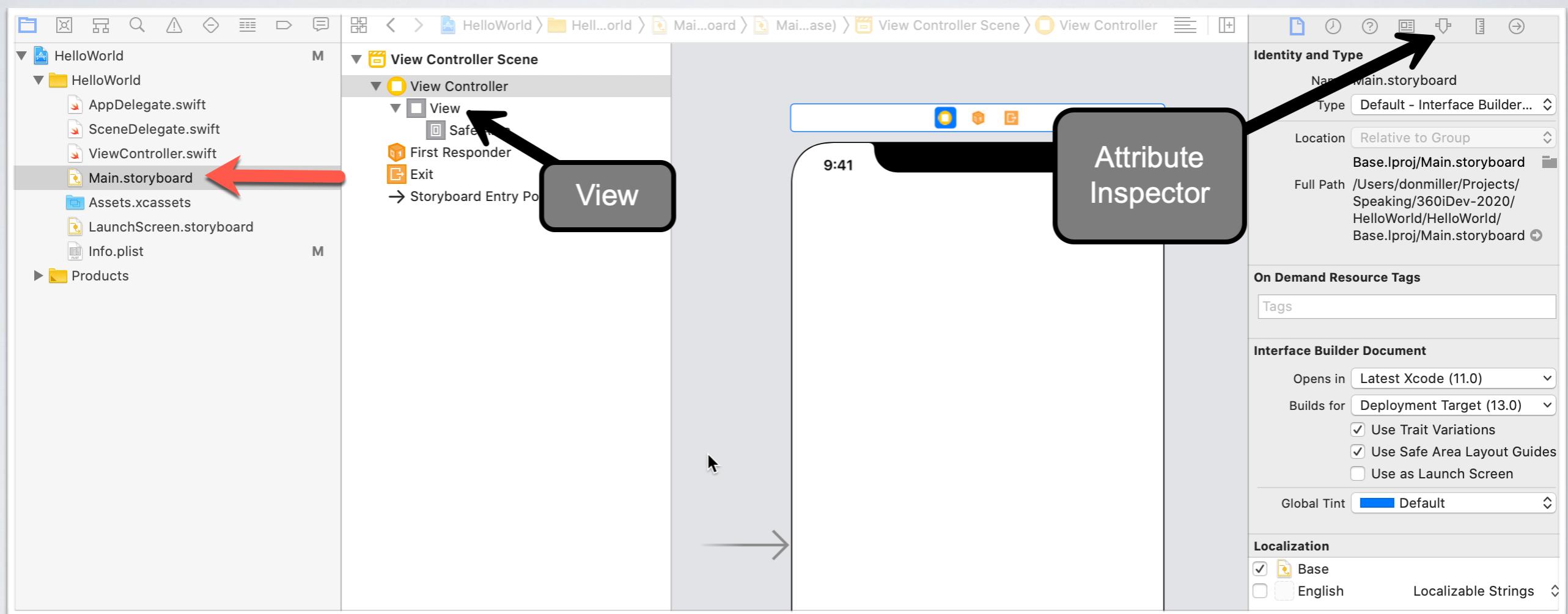


# RUN THE APPLICATION BY PRESSING PLAY

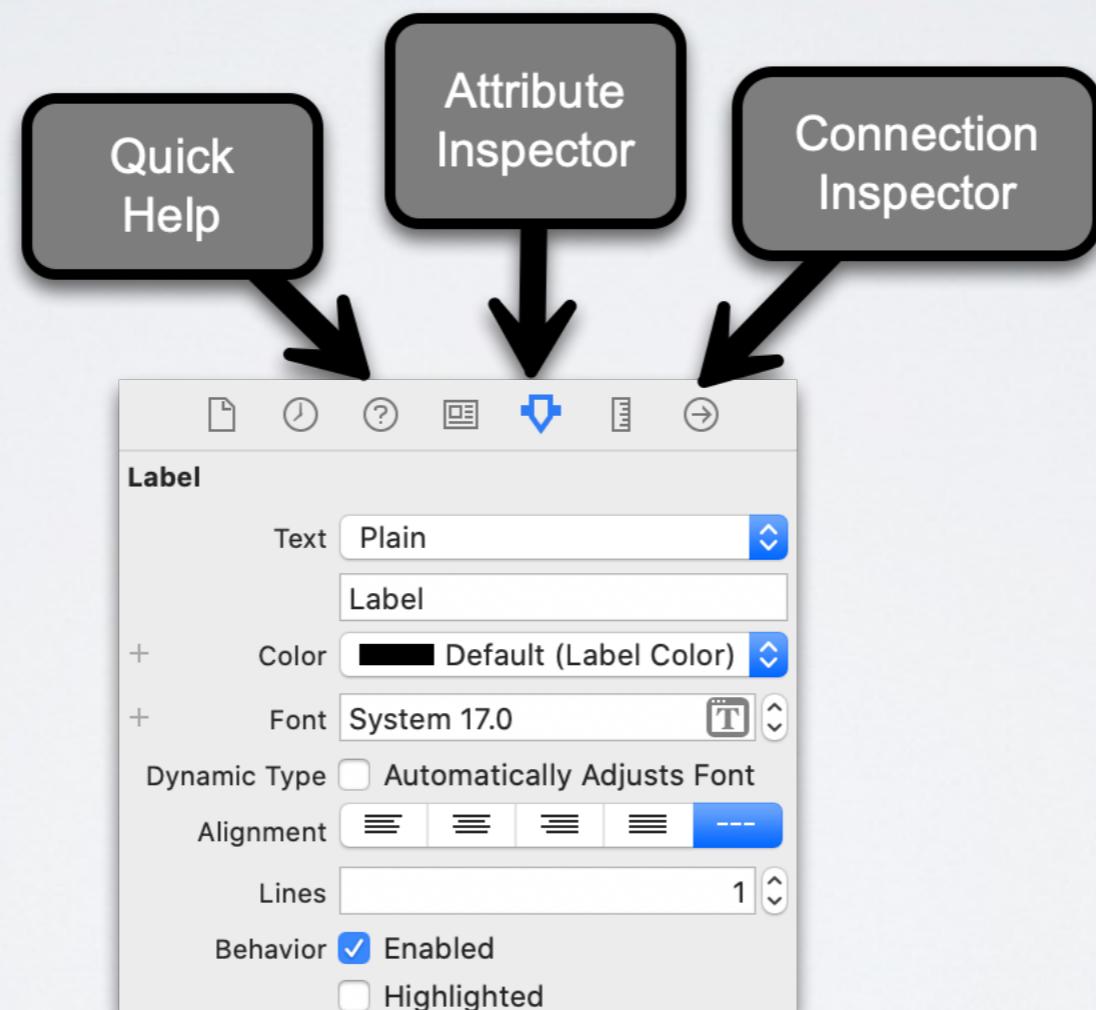


GroundSpeed™  
rapid web + mobile software

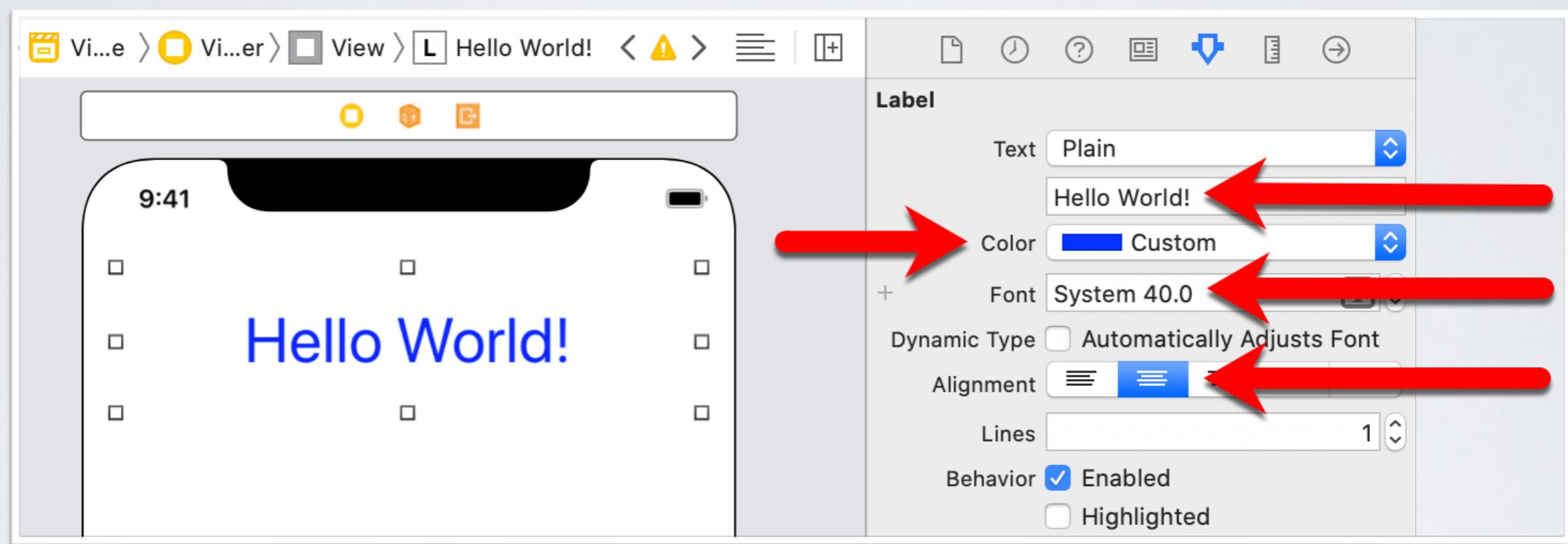
# ATTRIBUTE INSPECTOR



# INSPECTORS



# CREATE “HELLO WORLD!” LABEL



# Lab #1

## Hello World!

Take 15 Minutes to Build It

Break time afterwards



# SWIFT BASICS

Xcode Playgrounds

Variables & Constants (Mutable and Immutable)

Types

- Strings, Integers, Floats, Booleans

Operators

- Binary and Unary Operators

Collection Types

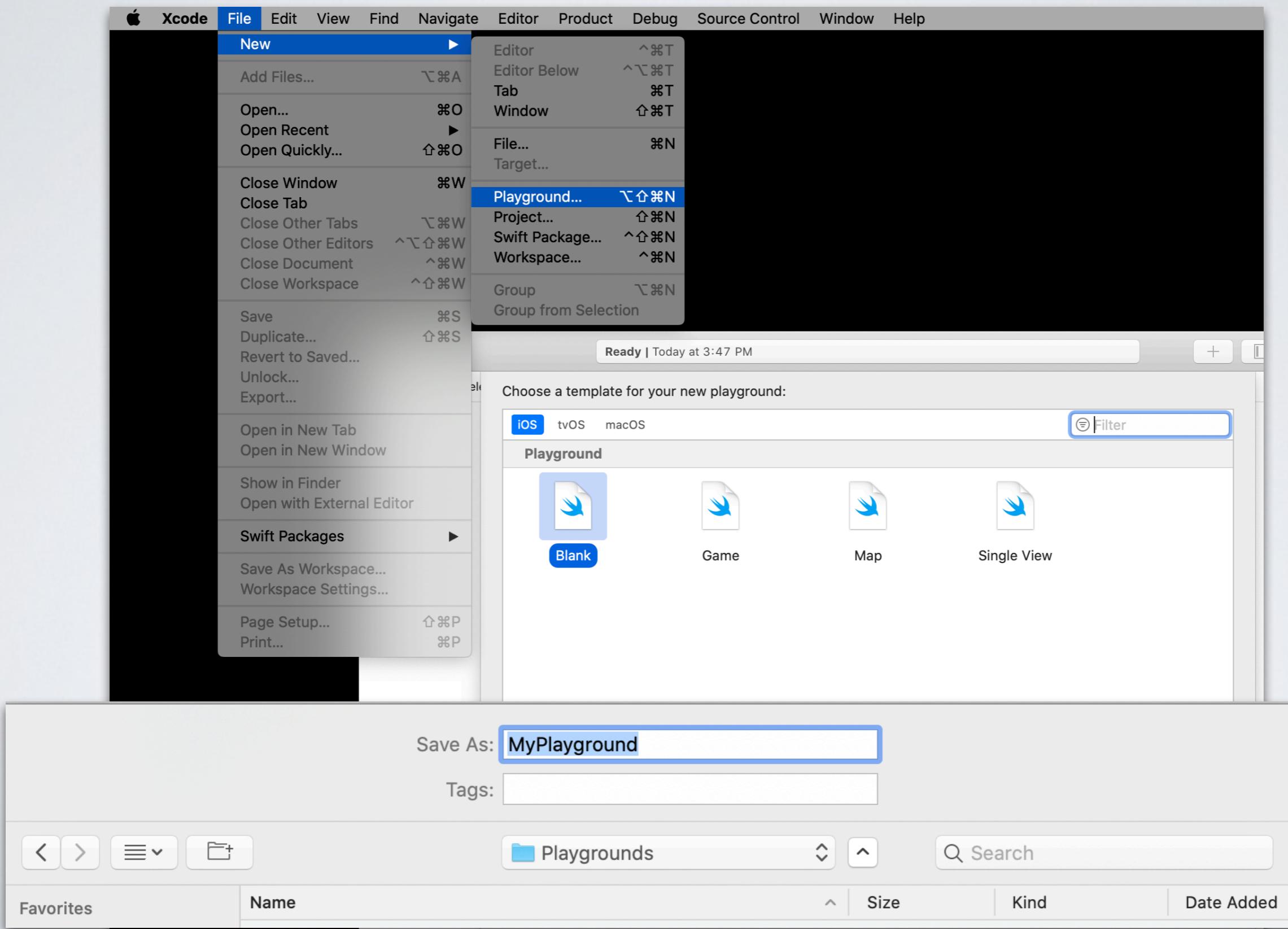
- Arrays and Dictionaries

Control Flow

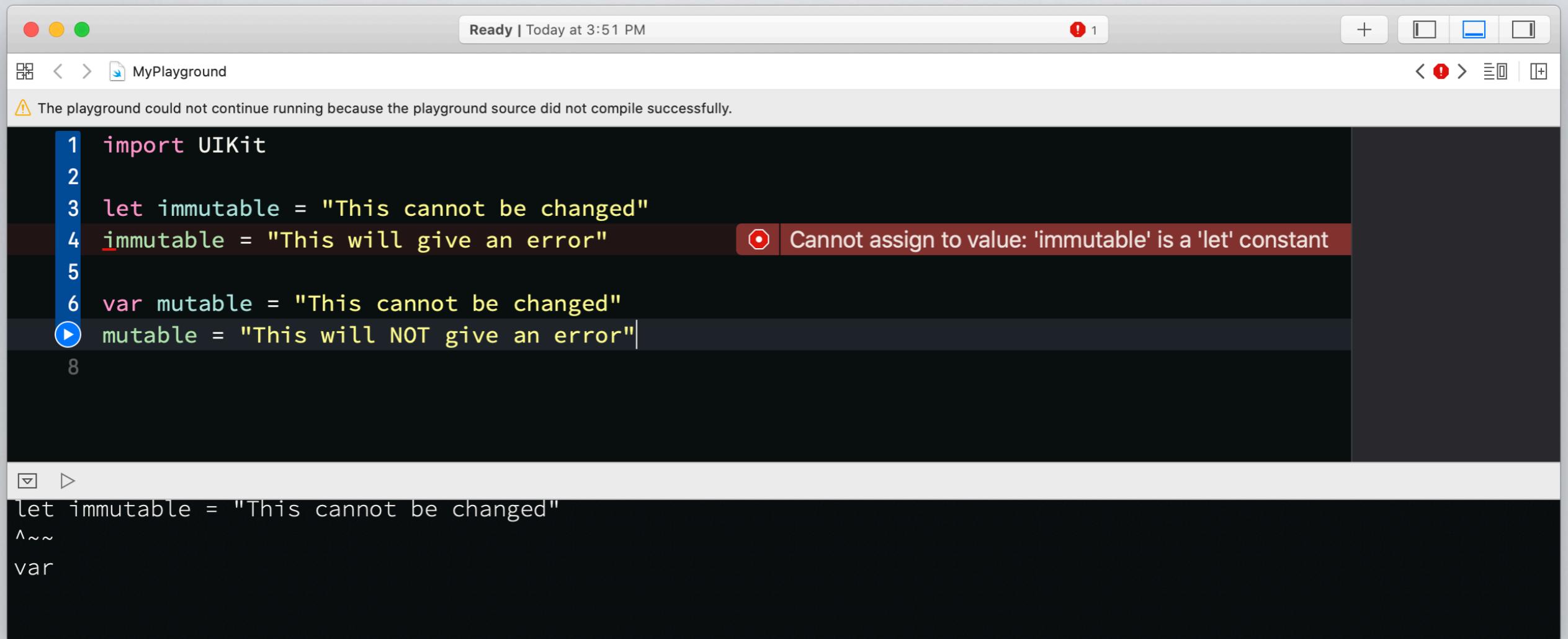
- For-In, While, For-Condition, If, Switch, Comparison and Logical Operators



# EXAMPLE: HAVE FUN IN THE PLAYGROUND



# ABOUT MUTABLE AND IMMUTABLE OBJECTS OR VARIABLES AND CONSTANTS



The screenshot shows an Xcode playground window titled "MyPlayground". The status bar at the top indicates "Ready | Today at 3:51 PM" and shows a red exclamation mark with the number "1", indicating one error. The playground code is as follows:

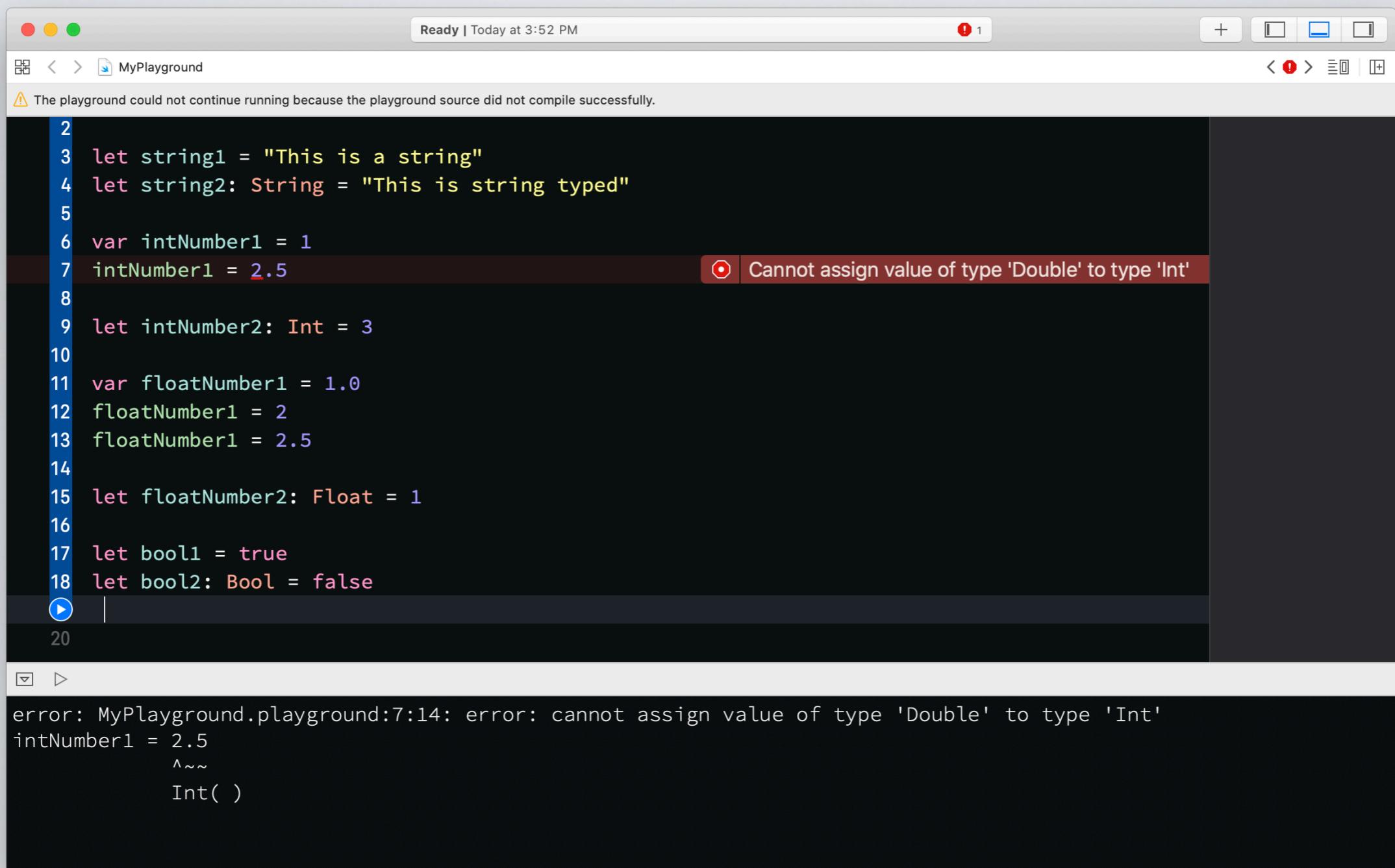
```
1 import UIKit
2
3 let immutable = "This cannot be changed"
4 immutable = "This will give an error"    ⚡ Cannot assign to value: 'immutable' is a 'let' constant
5
6 var mutable = "This cannot be changed"
7 mutable = "This will NOT give an error"  ⚡
8
```

A tooltip "Cannot assign to value: 'immutable' is a 'let' constant" is displayed over the fourth line. Below the playground, the results section shows the output of the code:

```
let immutable = "This cannot be changed"
^~~~  
var
```



# DECLARING TYPES: STRINGS, INTEGERS, FLOATS, AND BOOLEANS



The screenshot shows an Xcode playground window titled "MyPlayground". The code editor contains the following Swift code:

```
2
3 let string1 = "This is a string"
4 let string2: String = "This is string typed"
5
6 var intNumber1 = 1
7 intNumber1 = 2.5
8
9 let intNumber2: Int = 3
10
11 var floatNumber1 = 1.0
12 floatNumber1 = 2
13 floatNumber1 = 2.5
14
15 let floatNumber2: Float = 1
16
17 let bool1 = true
18 let bool2: Bool = false
19
20
```

A red callout box highlights the assignment `intNumber1 = 2.5`, indicating a type mismatch error. The message "Cannot assign value of type 'Double' to type 'Int'" is displayed next to the error icon.

The playground output pane at the bottom shows the error message:

```
error: MyPlayground.playground:7:14: error: cannot assign value of type 'Double' to type 'Int'
intNumber1 = 2.5
^~~
Int( )
```

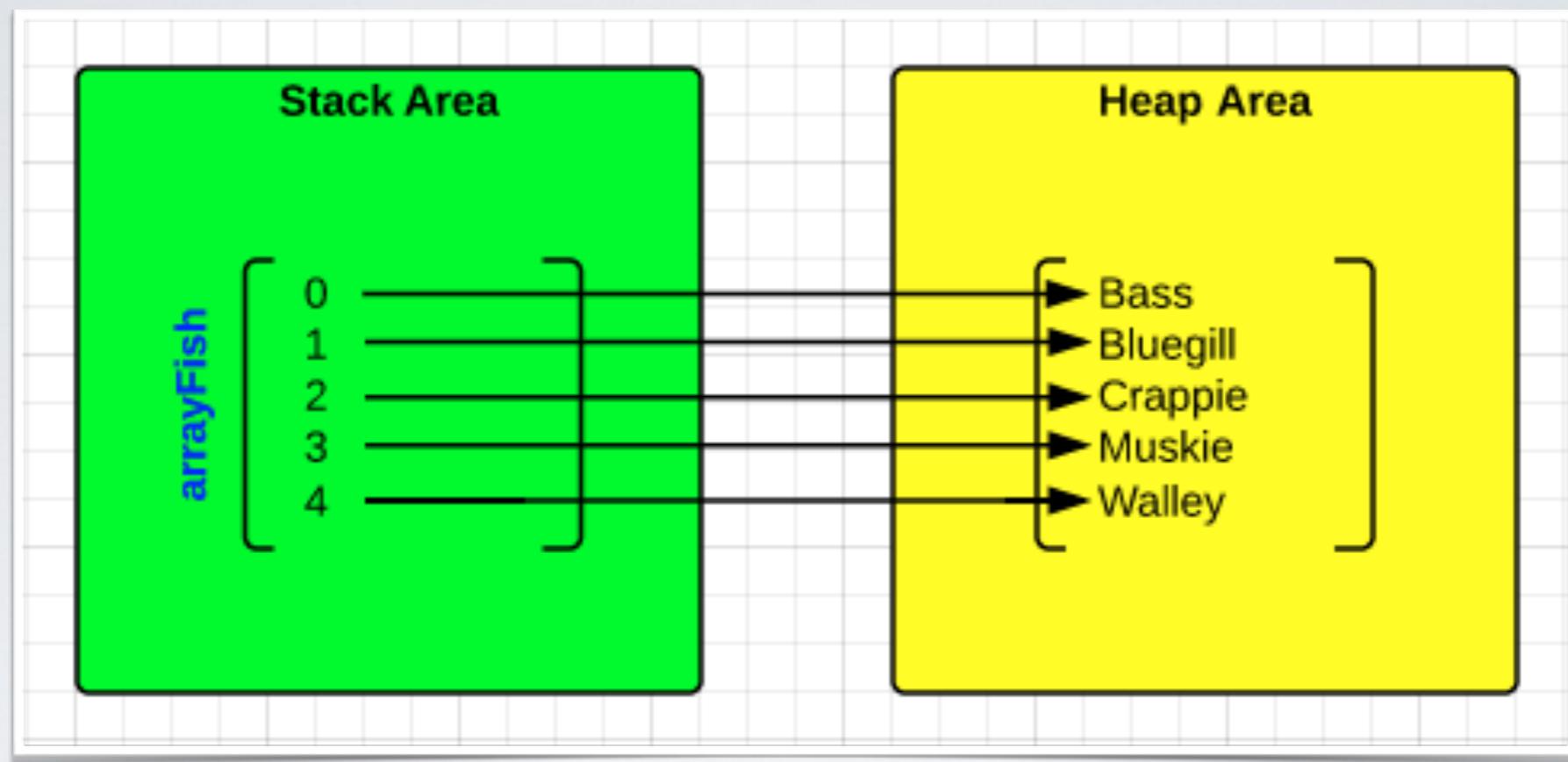


# OPERATORS: BINARY AND UNARY ...AND MAYBE A TERNARY TOO

```
1 // binary                                1
2 var a = 1                                 1
3 a = a + 1                               2
4 a = a - 1                               1
5 a = a / 1                                1
6 a = a * 4                                4
7 a = a % 3                                1
8
9 // unary                                  1
10 var b = 1                                1
11 b += 1                                 2
12 b += 1                                3
13 b                                     3
14 b -= 1                                 2
15 b                                     2
16
17 // ternary                               true
18 var bool = true                         true
19 var c = (bool ? 1 : 3)                  1
20 bool = false                           false
21 c                                     1
22 c = (bool ? 1 : 3)                  3
```



# ARRAYS



```
1  
2 var arrayFish = ["Bass", "Bluegill", "Crappie",  
  "Muskie", "Walleye"]  
3  
4 arrayFish.count  
5 arrayFish.first!  
6 arrayFish.last!  
7 arrayFish[3]  
8 arrayFish[5] ! error: Execution was interrupted, reason: E
```

```
["Bass", "Bluegill", "Crappie", "M... □  
5 □  
"Bass" □  
"Walleye" □  
"Muskie" □
```



# CREATING ARRAYS AND DICTIONARIES WITH SWIFT

```
1 // Arrays
2 let arrayFish = ["Bass", "Bluegill", "Crappie", "Muskie", "Walley"]
3
4 arrayFish.count
5 arrayFish[3]
6
7
8 // Dictionaries
9 let dictStates: Dictionary<String, String> = ["OH": "Ohio", "MI": "Michigan", "KY": "Kentucky", "IN": "Indiana", "IL": "Illinois", "PA": "Pennsylvania", "NY": "New York"]
11^ dictStates.count
12 dictStates["OH"]!
13
14 // For Loop through dictionary
15 for (stateCode, stateName) in dictStates {
16     print("\(stateCode): \(stateName)")  
                                (7 times)
17 }
18
19 for stateCode in dictStates.keys {
20     print("\(stateCode)")  
                                (7 times)
21 }
22
23 for stateName in dictStates.values {
24     print("\(stateName)")  
                                (7 times)
25 }
26
27 // Array of stateCodes
28 let stateCodes = [String](dictStates.keys)  
                                ["OH", "NY", "..."]
```



# CONTROL FLOW - LOOPING

```
1 // For-In
2 for index in 1...5 {
3     print("\(index) times 5 is \(index * 5)")
4 }
5
6 let arrayFish = ["Bass", "Bluegill", "Crappie", "Muskie", "Walleye"]
7 for fish in arrayFish {
8     print("Fish array: \(fish)")
9 }
10
11 // For-In with <
12 for index in 0 ..< 5 {
13     print("For Index is \(index)")
14 }
15
16 // While
17 var index = 0
18 var max = 5
19 while index < max {
20     print("While Index is \(index += 1)")
21 }
22
23 // Repeat-While
24 index = 0
25 max = 5
26 repeat {
27     print("Repeat-While Index is \(index += 1)")
28 } while index < max
```

## Console Output

```
1 times 5 is 5
2 times 5 is 10
3 times 5 is 15
4 times 5 is 20
5 times 5 is 25
Fish array: Bass
Fish array: Bluegill
Fish array: Crappie
Fish array: Muskie
Fish array: Walley
For Index is 0
For Index is 1
For Index is 2
For Index is 3
For Index is 4
While Index is ()
Repeat-While Index is ()
```



# CONTROL FLOW - DECISION

```
1 // IF
2 var temperatureInFahrenheit = 90
3 if temperatureInFahrenheit <= 32 {
4     print("It's very cold. Consider wearing a scarf.")
5 } else if temperatureInFahrenheit >= 86 {
6     print("It's really warm. Don't forget to wear sunscreen.")
7 } else {
8     print("It's not that cold. Wear a t-shirt.")
9 }
10
11 // SWITCH
12 let someCharacter: Character = "a"
13 switch someCharacter {
14 case "a", "e", "i", "o", "u":
15     print("\(someCharacter) is a vowel")
16 case "b", "c", "d", "f", "g", "h", "j", "k", "l", "m",
17 "n", "p", "q", "r", "s", "t", "v", "w", "x", "y", "z":
18     print("\(someCharacter) is a consonant")
19 default:
20     print("\(someCharacter) is not a vowel or a consonant")
21 }
```

90  
"It's really warm. Don't forget to wear s..  
"a"  
"a is a vowel\n"



# CLASS ILLUSTRATIONS

```
1 class Person {  
2     var ssn: String!  
3     var name: String!  
4     var city: String!  
5     var state: String!  
6 }  
  
8 class Employee: Person {  
9     var empId: String!  
10    var department: String!  
11    var healthInsurance: String!  
  
13    func printChecks() {  
14        print("Printing Checks")  
15    }  
16 }  
  
18 class SalariedEmployee: Employee {  
19     var salary: String!  
20 }  
  
22 class HourlyEmployee: Employee {  
23     var hourlyRate: String!  
24 }
```

```
30 let hourlyEmployee = HourlyEmployee()  
31 hourlyEmployee.name = "Don Miller"  
32 hourlyEmployee.empId = "1234"  
33 hourlyEmployee.hourlyRate = "9.99"  
34  
35 var output = "Name: \(hourlyEmployee.name!) \r"  
36 output += "Employee ID is \(hourlyEmployee.empId!) \r"  
37 output += "Hourly wage is \(hourlyEmployee.hourlyRate!) \r"  
38  
39 print("\(output)")  
40 hourlyEmployee.printChecks()
```

## Console Output

```
Name: Don Miller  
Employee ID is 1234  
Hourly wage is 9.99  
Printing Checks
```



# CLASSES VS STRUCTS

```
1 struct Person {  
2     var ssn: String!  
3     var name: String!  
4     var city: String!  
5     var state: String!  
6 }
```

```
1 class Person {  
2     var ssn: String!  
3     var name: String!  
4     var city: String!  
5     var state: String!  
6 }
```

## Structs

- Value Types
- Cannot inherit
- Can be extended
- Can conform to protocols

## Classes

- Reference Types
- Can inherit
- Can be extended
- Can conform to protocols

Great reference for learning more here:

<https://learnappmaking.com/struct-vs-class-swift-how-to>



GroundSpeed™  
rapid web + mobile software

# Lab #2

# Swift Playground!

Take a Few Minutes to Build  
the Previous Class Model  
Using Structs

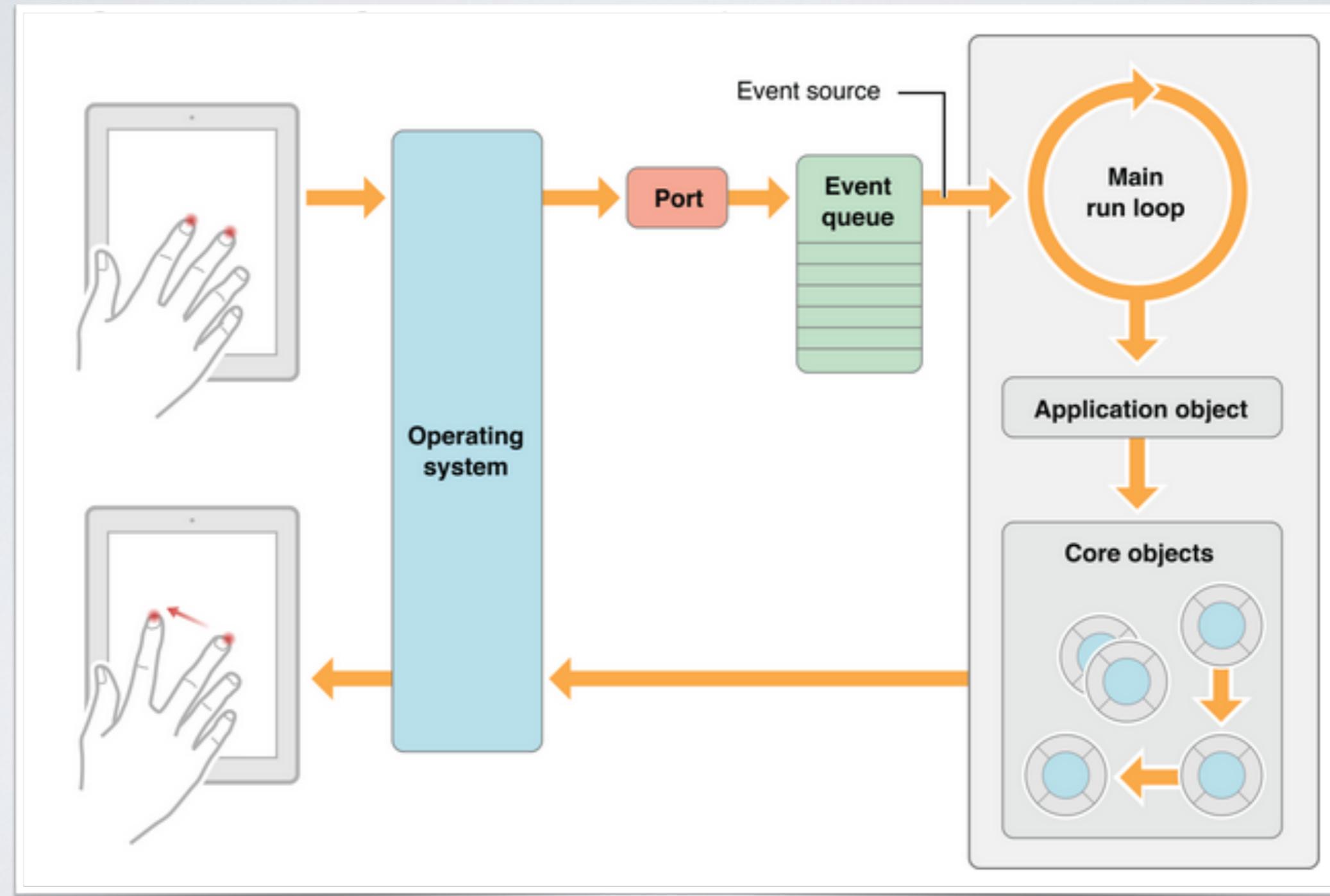
STARTING AGAIN at 11:15a MT



# MVC, Storyboards, and Events



# IOS APPLICATION RUNTIME CYCLE

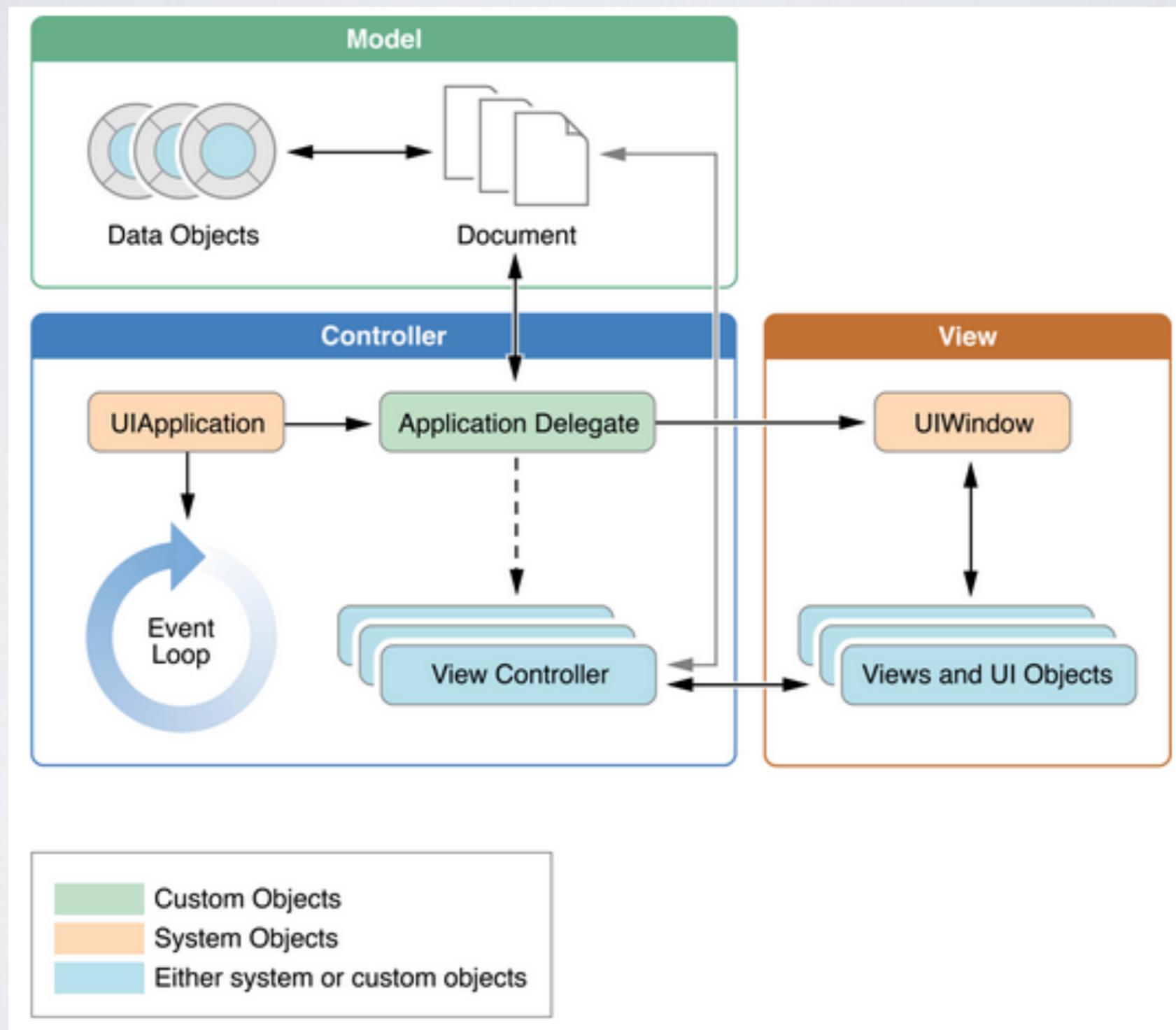


# COMMON TYPES OF EVENTS

Event type	Delivered to...	Notes
Touch	The view object in which the event occurred	Views are responder objects. Any touch events not handled by the view are forwarded down the responder chain for processing.
Remote control Shake motion events	First <a href="#">responder object</a>	Remote control events are for controlling media playback and are generated by headphones and other accessories.
Accelerometer Magnetometer Gyroscope	The object you designate	Events related to the accelerometer, magnetometer, and gyroscope hardware are delivered to the object you designate.
Location	The object you designate	You register to receive location events using the Core Location framework. For more information about using Core Location, see <a href="#">Location and Maps Programming Guide</a> .
Redraw	The view that needs the update	Redraw events do not involve an event object but are simply calls to the view to draw itself. The drawing architecture for iOS is described in <a href="#">Drawing and Printing Guide for iOS</a> .



# MODEL - VIEW - CONTROLLER



# MODEL - VIEW - CONTROLLER

**The Model:** The M in MVC stands for the Model: For the Model, we will need to develop an independent class (or a set of classes) that will serve the data requirements and standard business procedures.

**The View:** In this context the View is the UI screen's layout (the canvas). Here, we configure many User Interface items (also known as UI controls on the screen).

**The Controller:** This is a class that connects the view with the model. In its simplest form, it contains the name of the UIControls (that would be used in the app), and it also includes the signatures and the detail implementations of event handlers.



# STORYBOARD OVERVIEW

Storyboards are graphic organizers displayed in sequence for the purpose of pre-visualizing a motion picture or animation. The storyboarding process, in the form it is known today, is used by developers to give the designer, developer, and end user an opportunity to see what the application or web site will look like at a high level.

In iOS, a Storyboard is a container of screens that may have embedded view controllers, navigation controllers, and tab bar controllers. It allows the user to graphically connect screens to one another through embedded objects on the screen. These connections are called segues (pronounced seg-wey). The storyboard eliminates the need for xib files and has one file that contains all of your screens and transitions. It is possible to connect two storyboard files with one another; however, unlikely with smaller applications.



# PROS OF STORYBOARDS

## Pros of using Storyboards:

- With a storyboard you have a better conceptual overview of all the screens in your app and the connections between them. It's easier to keep track of everything because the entire design is in a single file, rather than spread out over many separate nibs.
- The storyboard describes the transitions between the various screens. These transitions are called “segues” and you create them by simply ctrl-dragging from one view controller to the next. Thanks to segues you need less code to take care of your UI.
- Storyboards make working with table views a lot easier with the new prototype cells and static cells features. You can design your table views almost completely in the storyboard editor, something else that cuts down on the amount of code you have to write.



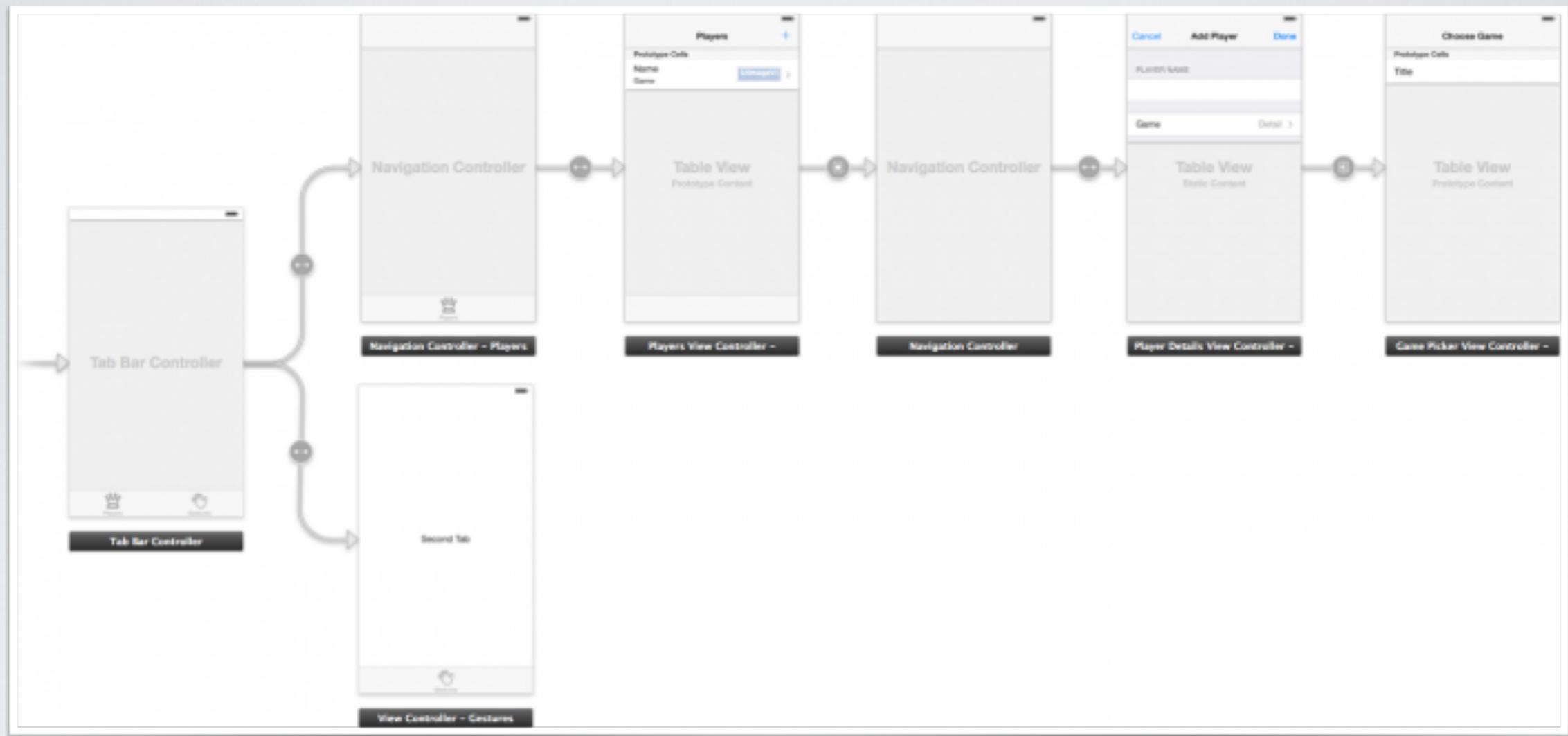
# CONS OF STORYBOARDS

## Cons of using Storyboards:

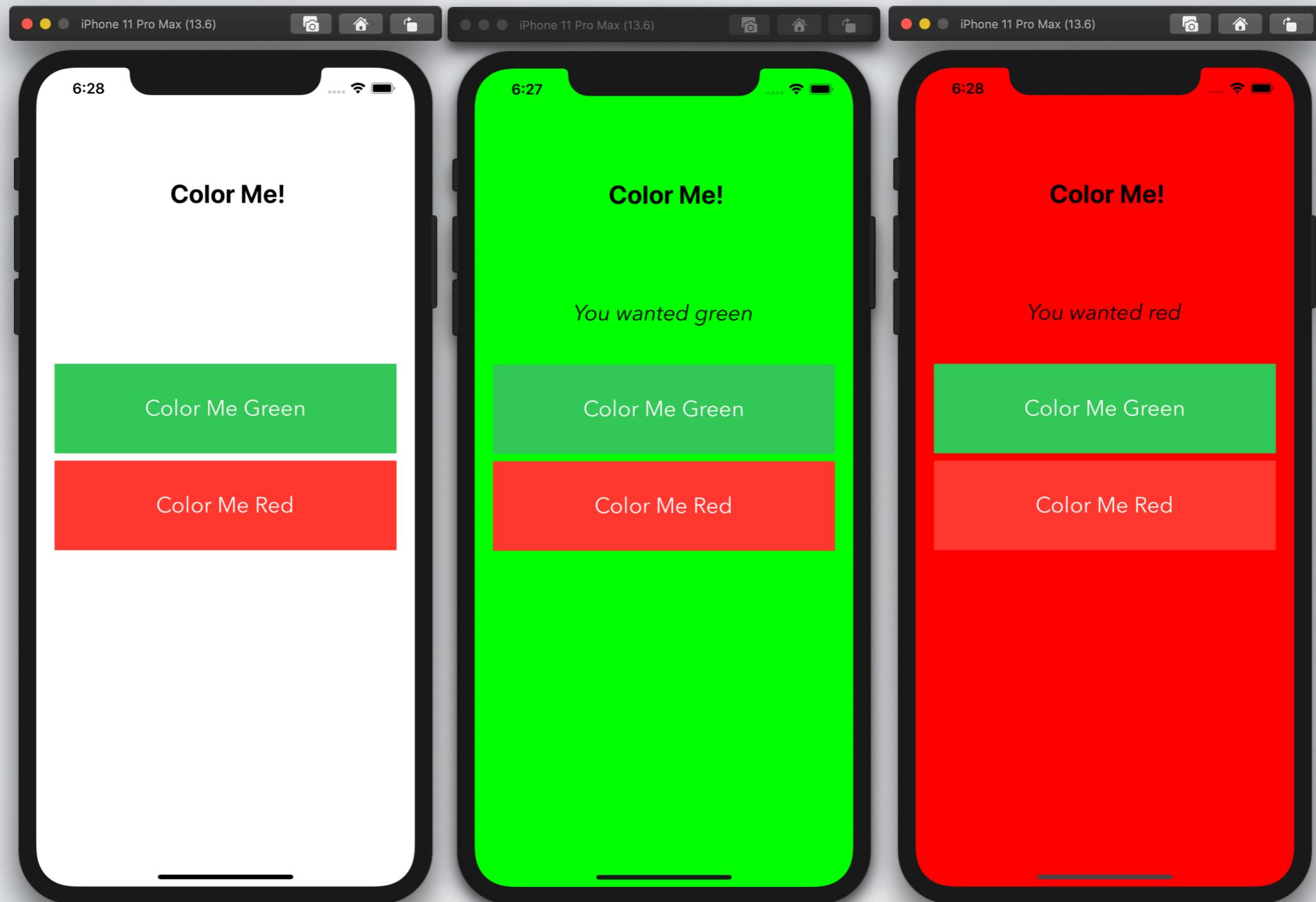
- It's not easy to work with storyboards in a team, since only one participant can work on the storyboard at once (because it's one binary file).
- If you need to do things storyboard doesn't offer, it's not quite easy to get storyboard mixed with programmatically created views)
- Since all objects are in one file, a larger project could cause the storyboard to be very slow and unresponsive. It loads the entire thing into memory. It loads the entire item into memory.



# STORYBOARD OVERVIEW



# ANOTHER QUICK DEMO - [COLOR ME]



# CREATE A SINGLEVIEW APPLICATION

Choose a template for your new project:

iOS watchOS tvOS macOS Cross-platform Filter

**Application**

 Single View App	 Game	 Augmented Reality App	 Document Based App	 Master-Detail App	
 Tabbed App	 Sticker Pack App	 iMessage App			

**Framework & Library**

 Framework	 Static Library	 Metal Library
---	--	---

Cancel Previous Next

Choose options for your new project:

Product Name: ColorMe  
Team: None

Organization Name: GroundSpeed  
Organization Identifier: com.groundspeedhq  
Bundle Identifier: com.groundspeedhq.ColorMe

Language: Swift  
User Interface: Storyboard

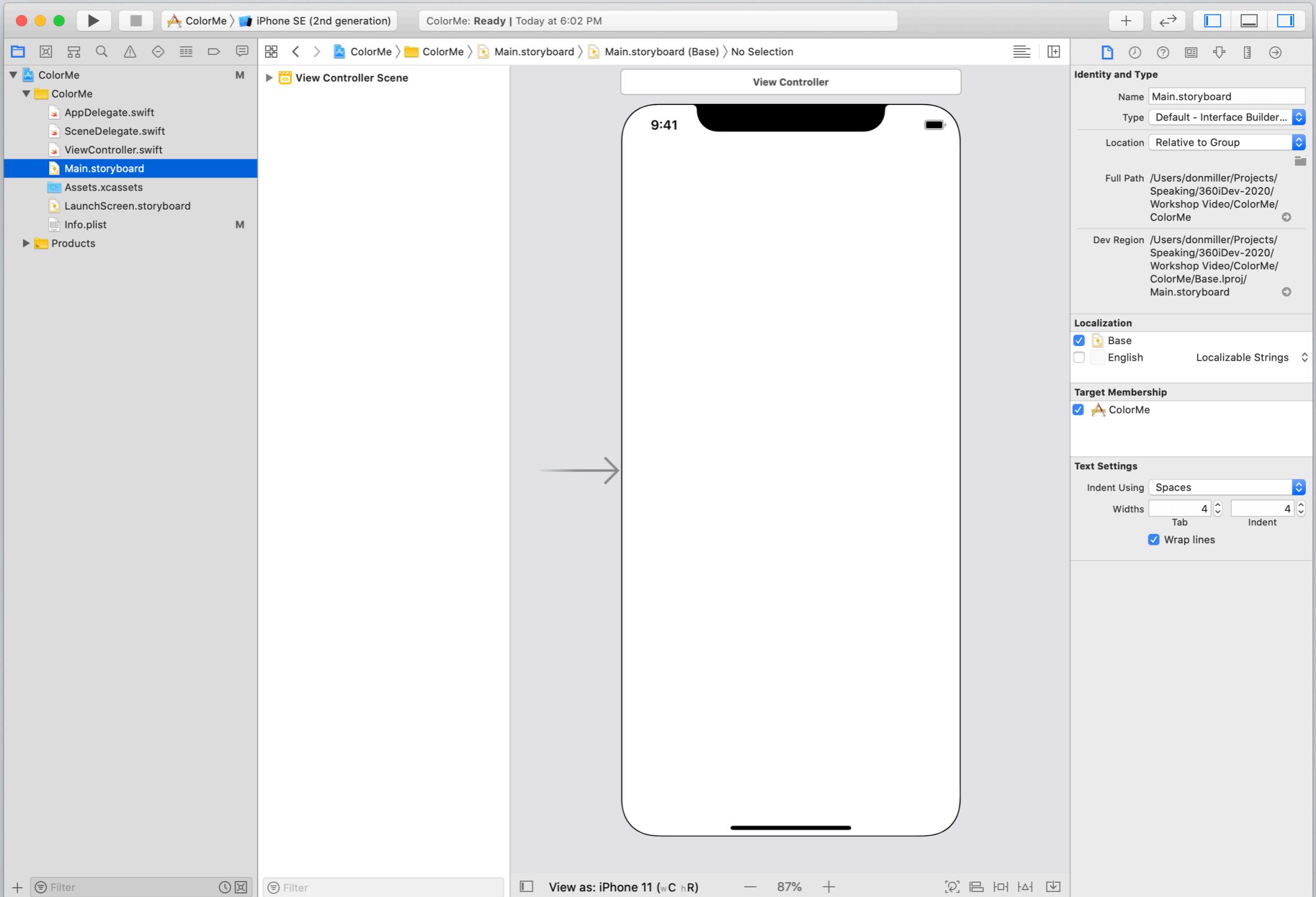
Use Core Data  
 Use CloudKit  
 Include Unit Tests  
 Include UI Tests

Cancel Previous Next

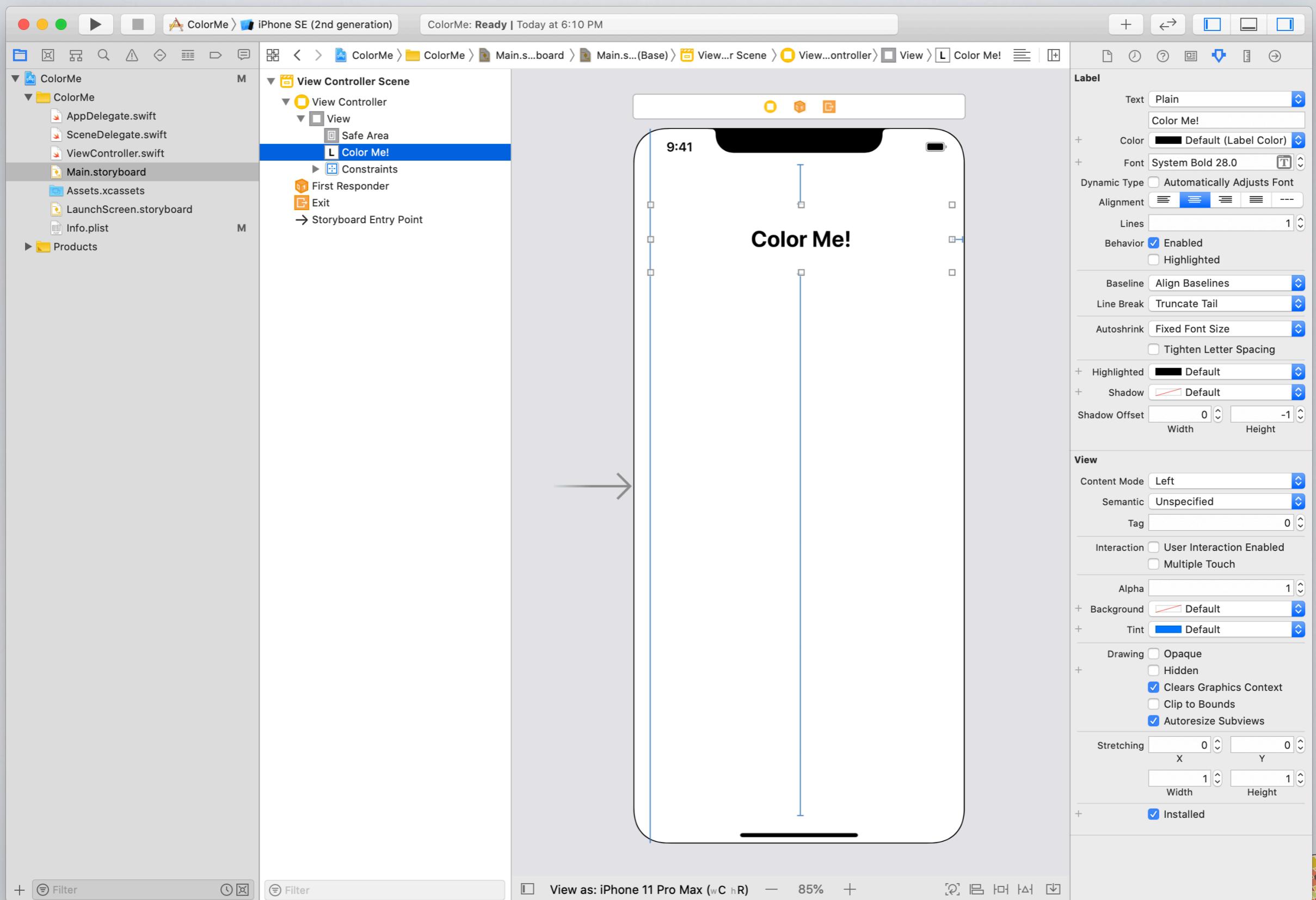


GroundSpeed™  
rapid web + mobile software

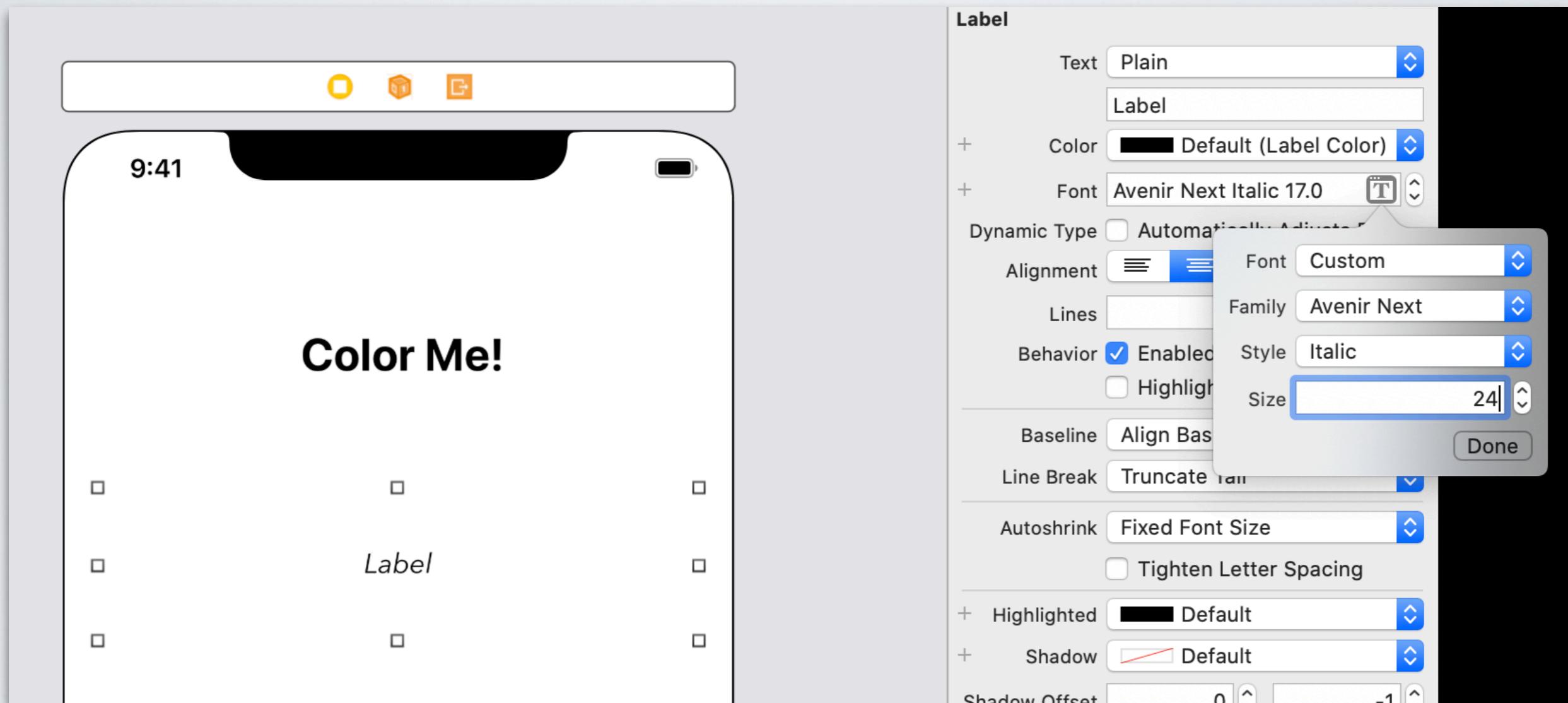
# YOUR STORYBOARD SHOULD LOOK LIKE THIS



# DRAG A LABEL TO THE STORYBOARD



# CREATE A BLANK LABEL FOR OUTPUT





**Button**

Type: System  
State Config: Default  
Title: Plain  
Color: Color Me Red  
Font: Avenir Next Regular 24.0  
Text Color: White Color  
Shadow Color: Default  
Image: Default Image  
Background: Default Background I...  
Default Symbol Configuration  
Configuration: Unspecified  
Scale: Unspecified  
Weight: Unspecified  
Pointer:  Interaction Enabled  
Accessibility:  Adjusts Image Size  
Shadow Offset: 0 0 Width Height  
 Reverses On Highlight  
Drawing:  Shows Touch On Highlight  
 Highlighted Adjusts Image  
 Disabled Adjusts Image  
Line Break: Truncate Middle  
Drag and Drop:  Spring Loaded

**Control**

Alignment: Horizontal Vertical  
State:  Selected  
 Enabled  
 Highlighted

**View**

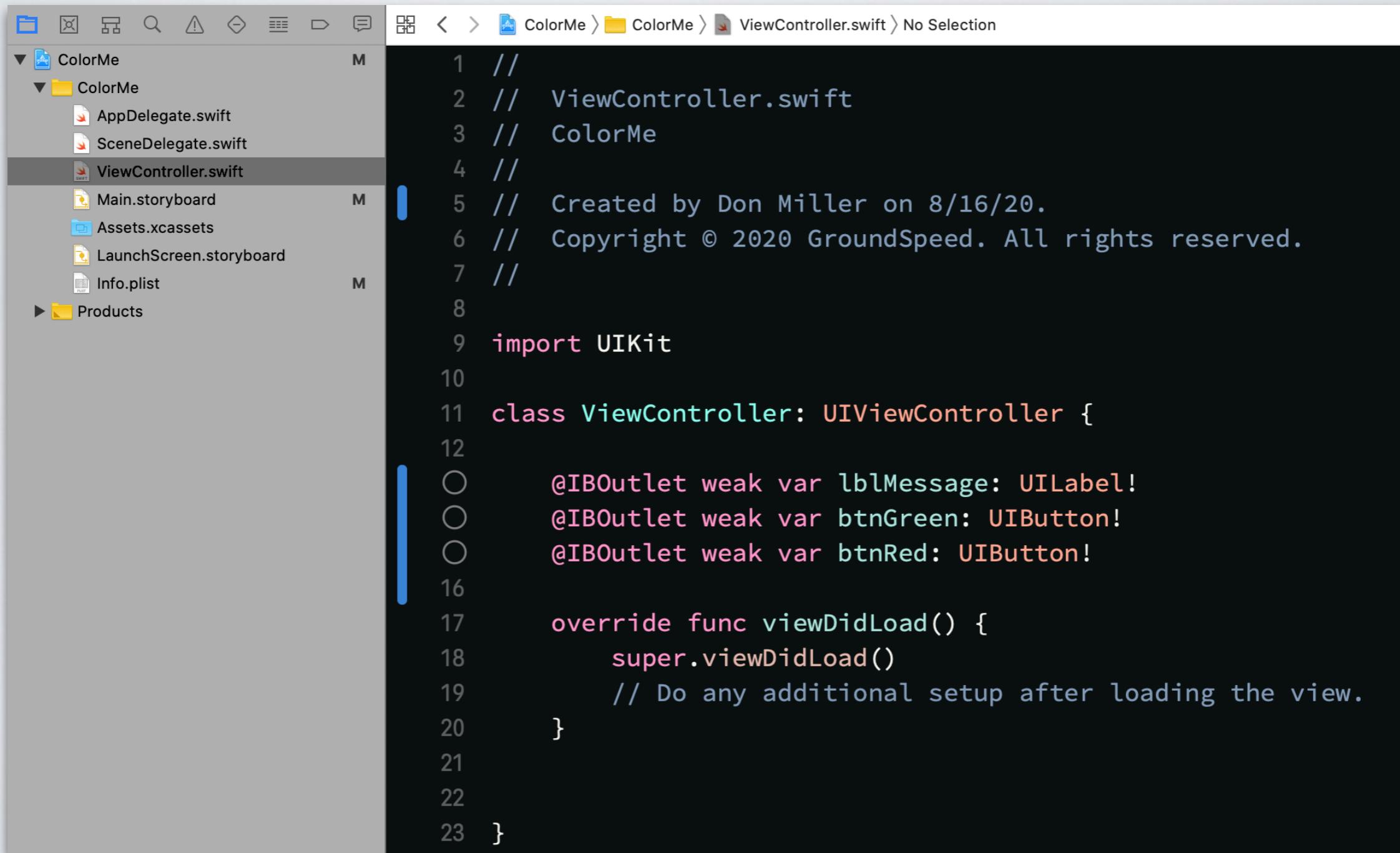
Content Mode: Scale To Fill  
Semantic: Unspecified  
Tag: 0  
Interaction:  User Interaction Enabled  
 Multiple Touch  
Alpha: 1  
Background: System Red Color

# CREATE TWO BUTTONS



GroundSpeed™  
rapid web + mobile software

# ADD OUTLETS TO CONTROLLER

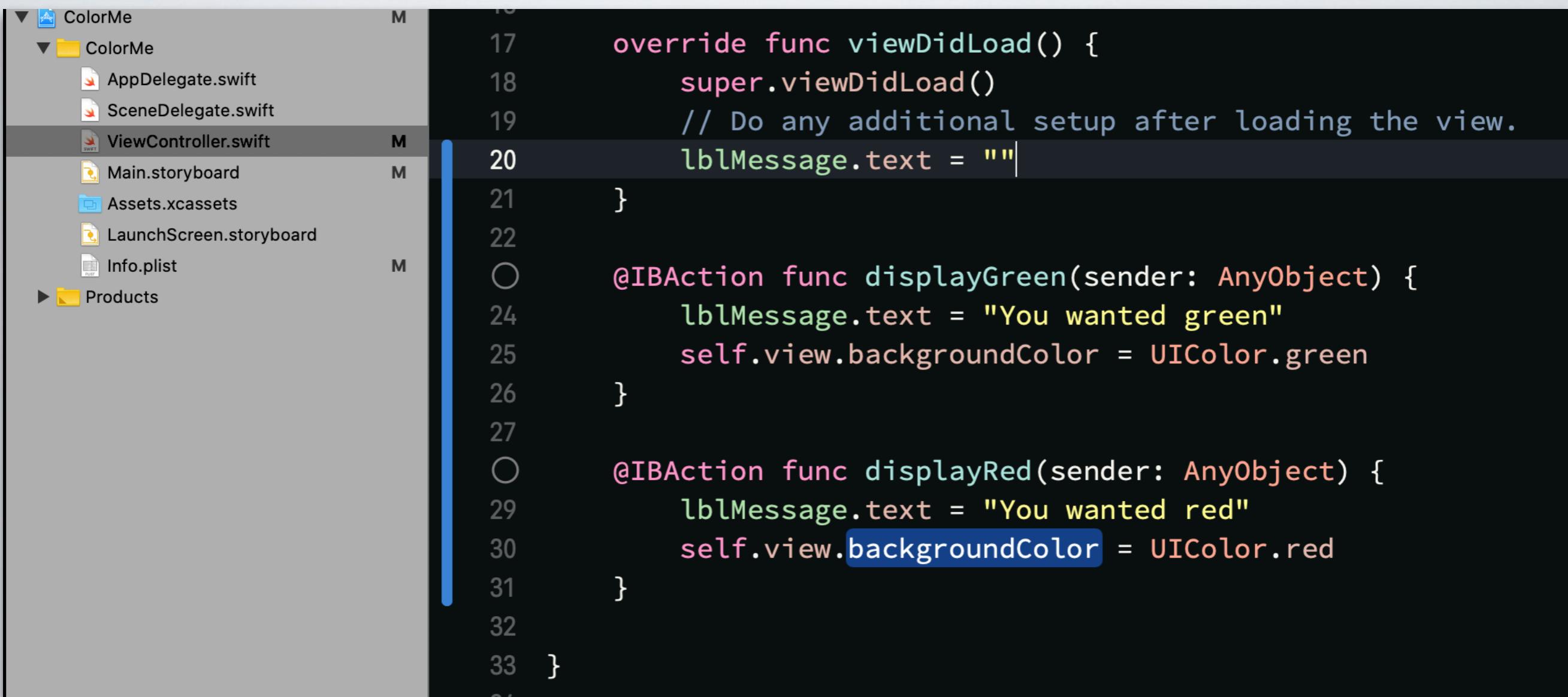


The screenshot shows the Xcode interface with the project navigation bar at the top. The left sidebar displays the project structure under 'ColorMe': 'ColorMe' folder contains 'AppDelegate.swift', 'SceneDelegate.swift', and 'ViewController.swift'. Below that are 'Main.storyboard', 'Assets.xcassets', 'LaunchScreen.storyboard', and 'Info.plist'. A 'Products' section is also visible. The main editor area shows the code for 'ViewController.swift'.

```
1 //  
2 //  ViewController.swift  
3 //  ColorMe  
4 //  
5 //  Created by Don Miller on 8/16/20.  
6 //  Copyright © 2020 GroundSpeed. All rights reserved.  
7 //  
8  
9 import UIKit  
10  
11 class ViewController: UIViewController {  
12  
13     @IBOutlet weak var lblMessage: UILabel!  
14     @IBOutlet weak var btnGreen: UIButton!  
15     @IBOutlet weak var btnRed: UIButton!  
16  
17     override func viewDidLoad() {  
18         super.viewDidLoad()  
19         // Do any additional setup after loading the view.  
20     }  
21  
22  
23 }
```



# ADD ACTIONS TO CONTROLLER AND UPDATE VIEWDIDL LOAD

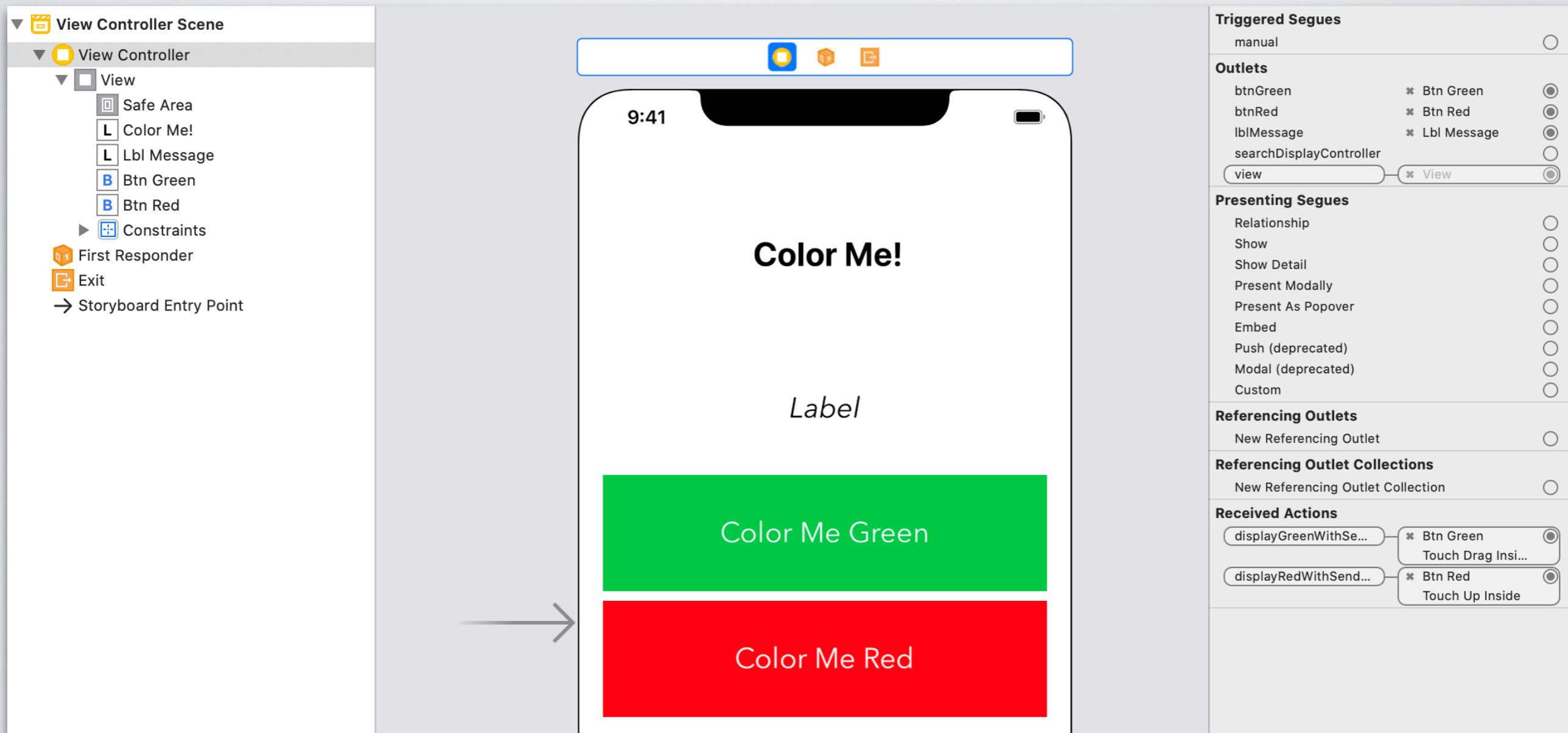


The screenshot shows the Xcode interface with the project navigation bar on the left and the code editor on the right. The code editor displays Swift code for a ViewController. A blue vertical selection bar is visible on the left side of the code editor.

```
16
17     override func viewDidLoad() {
18         super.viewDidLoad()
19         // Do any additional setup after loading the view.
20         lblMessage.text = ""
21     }
22
23 @IBAction func displayGreen(sender: AnyObject) {
24     lblMessage.text = "You wanted green"
25     self.view.backgroundColor = UIColor.green
26 }
27
28 @IBAction func displayRed(sender: AnyObject) {
29     lblMessage.text = "You wanted red"
30     self.view.backgroundColor = UIColor.red
31 }
32
33 }
```



# CONNECTING THE OUTLETS AND ACTIONS





iPhone 11 Pro Max (13.6)



6:28

**Color Me!**

Color Me Green

Color Me Red



iPhone 11 Pro Max (13.6)



6:27

**Color Me!**

*You wanted green*

Color Me Green

Color Me Red



iPhone 11 Pro Max (13.6)



6:28

**Color Me!**

*You wanted red*

Color Me Green

Color Me Red



GroundSpeed™  
rapid web + mobile software

# Lab #3 - Create Color Me Storyboard App



# STORYBOARD DEMO

## STEP 1: CREATE A NEW SINGLE VIEW APPLICATION

Choose a template for your new project:

iOS watchOS tvOS macOS Cross-platform

**Application**

 Single View App	 Game	 Augmented Reality App	 Document Based App	 Master-Detail App
 Tabbed App	 Sticker Pack App	 iMessage App	Choose options for your new project:	

**Framework & Library**

 Framework	 Static Library	 Metal Library
--	---	--

**Product Name:** Storyboard  
**Team:** None  
**Organization Name:** GroundSpeed  
**Organization Identifier:** com.groundspeedhq  
**Bundle Identifier:** com.groundspeedhq.Storyboard  
**Language:** Swift  
**User Interface:** Storyboard

Use Core Data  
 Use CloudKit  
 Include Unit Tests  
 Include UI Tests

 **GroundSpeed™** + mobile software

# STEP 2: OBSERVE THE SUMMARY PAGE

▼ **Identity**

Display Name

Bundle Identifier

Version

Build

---

▼ **Deployment Info**

Target	Device
iOS 13.6	<input checked="" type="checkbox"/> iPhone <input type="checkbox"/> iPad <input type="checkbox"/> Mac (requires macOS 10.15)

Main Interface

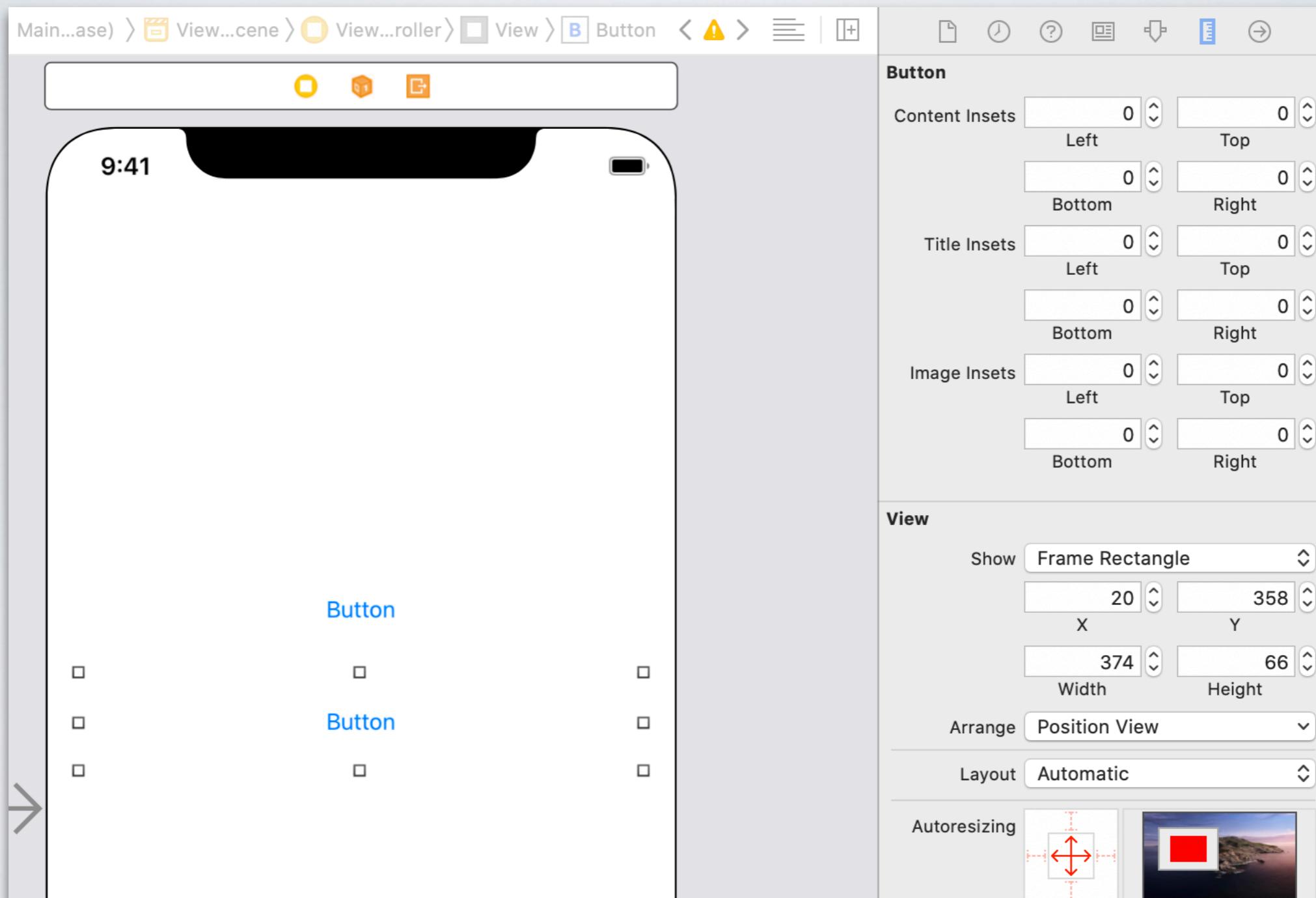
Device Orientation  Portrait  
 Upside Down  
 Landscape Left  
 Landscape Right

Status Bar Style   
 Hide status bar



# STEP 3: ADD 2 BUTTONS TO THE VIEW CONTROLLER

# STEP 4: LET'S MAKE THEM THE SAME WIDTH IN SIZE INSPECTOR



# STEP 5: CHANGE THE BUTTON LABELS TEXT TO RED AND GREEN

The screenshot shows a storyboard interface in Xcode. On the left, a modal view is displayed on an iPhone screen. The modal has a white background and contains two buttons: one labeled "Red Push" in red text and another labeled "Green Modal" in green text. The storyboard on the right shows the configuration for the "Red Push" button. The "Type" is set to "System". Under "Title", it is set to "Plain" and "Red Push". The "Font" is "System 15.0" and the "Text Color" is "System Red Color". The "Background" is "Default Background". In the "Default Symbol Configuration" section, the "Weight" is set to "Unspecified". Under "Drawing", the "Highlighted Adjusts Image" and "Disabled Adjusts Image" checkboxes are checked. The "Line Break" is set to "Truncate Middle". A large gray arrow points from the "Green Modal" button towards the "Red Push" button, indicating a transition or relationship between them.

**Button**

Type System

State Config Default

Title Plain

Red Push

+ Font System 15.0

Text Color System Red Color

Shadow Color Default

Image Default Image

Background Default Background I...

**Default Symbol Configuration**

Configuration Unspecified

Scale Unspecified

Weight Unspecified

Pointer  Interaction Enabled

Accessibility  Adjusts Image Size

Shadow Offset 0 0

Width Height

Reverses On Highlight

Drawing  Shows Touch On Highlight

Highlighted Adjusts Image

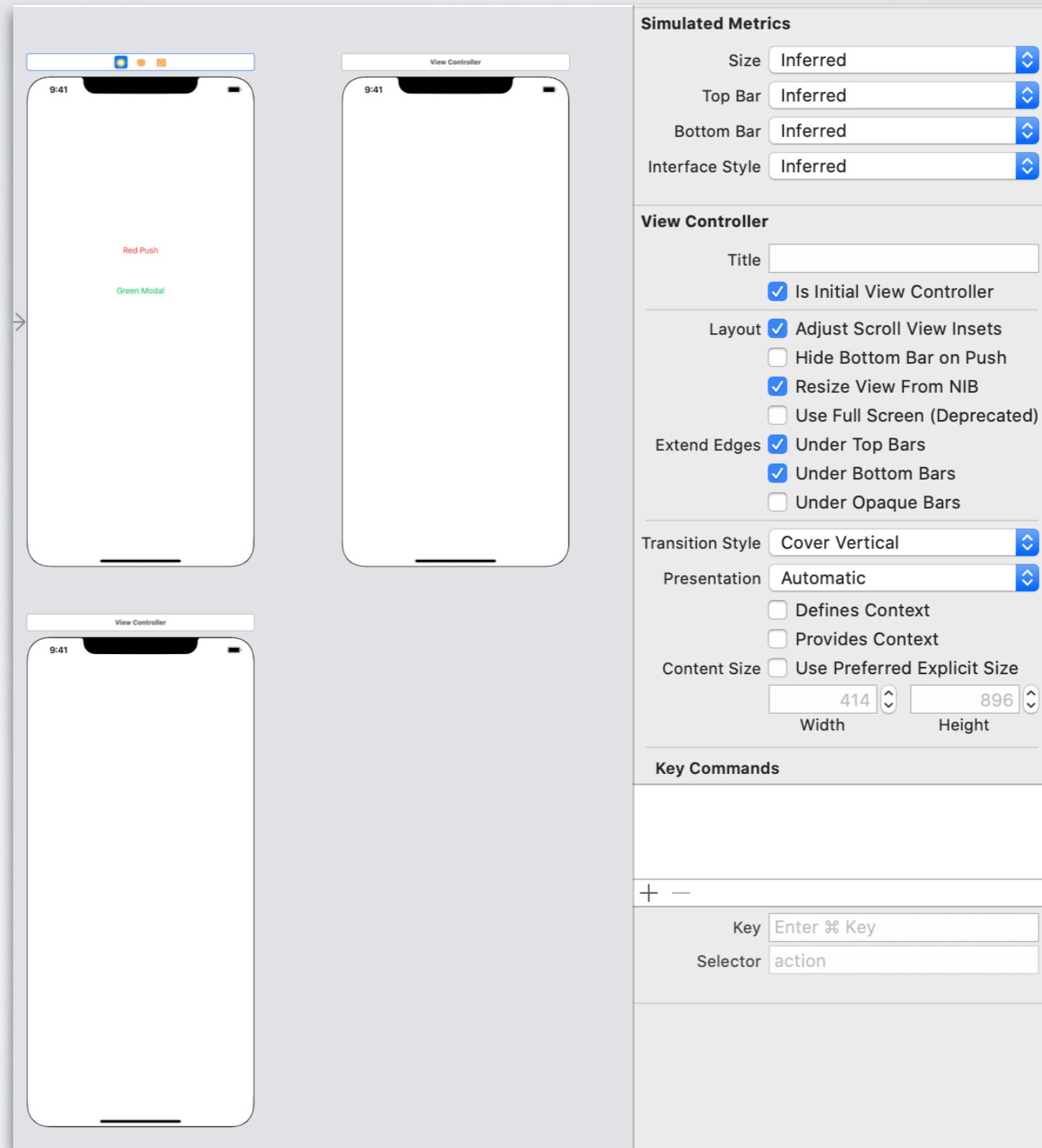
Disabled Adjusts Image

Line Break Truncate Middle

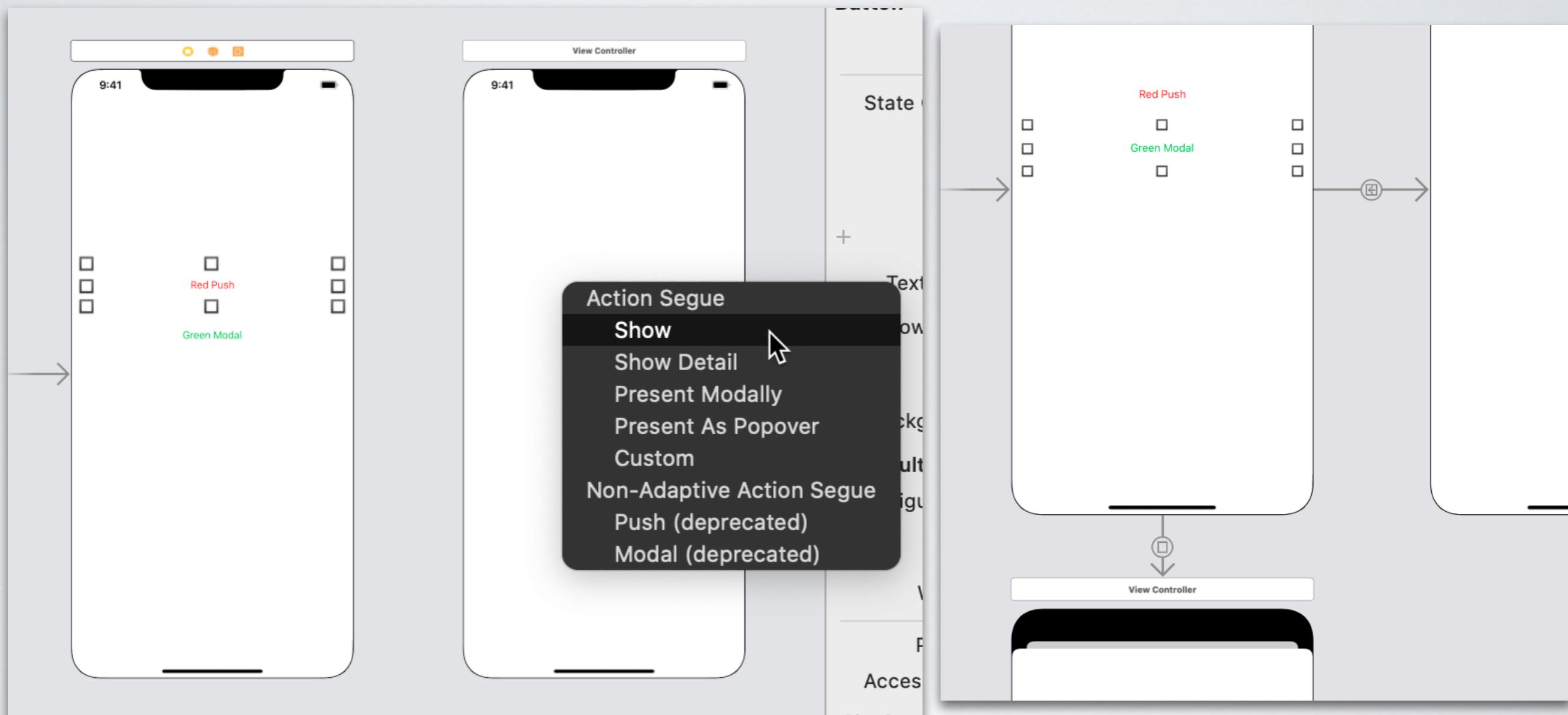
Drag and Drop  Spring Loaded



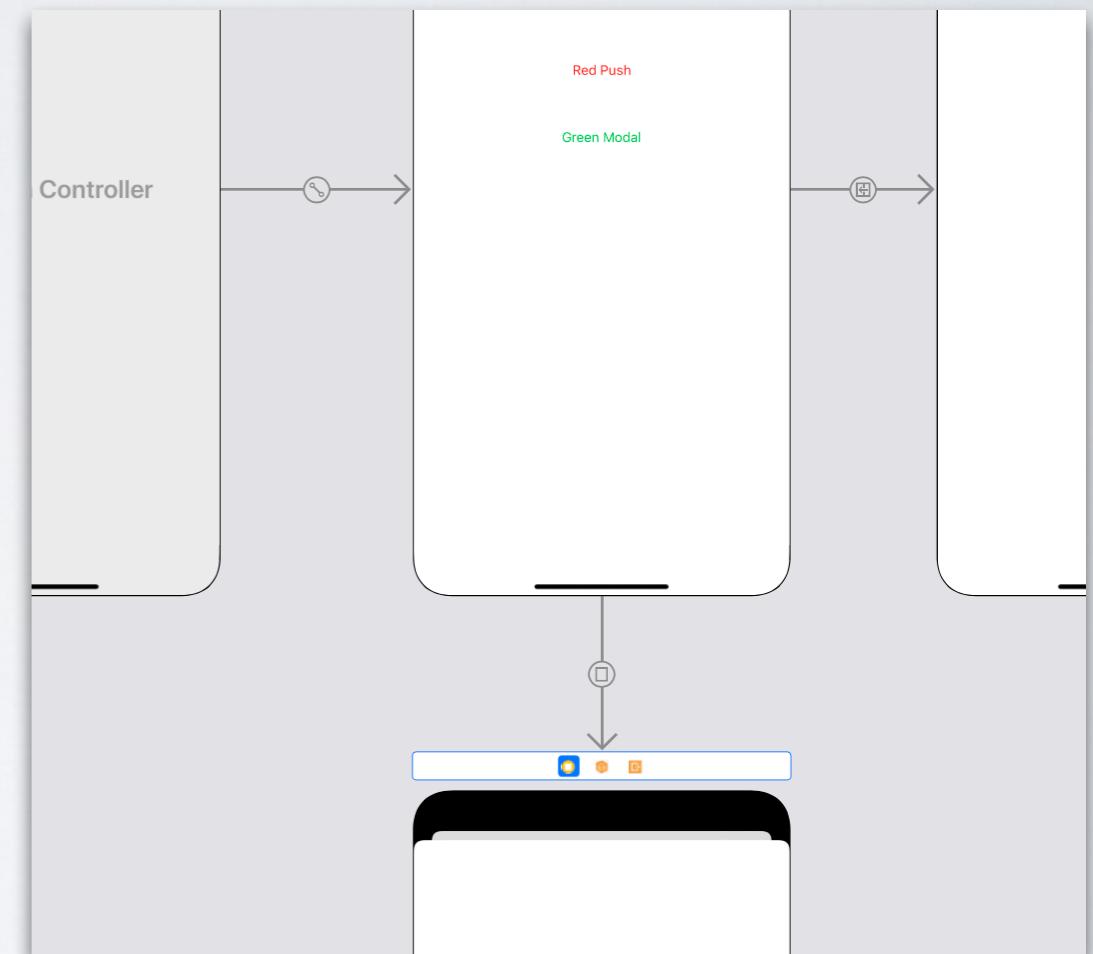
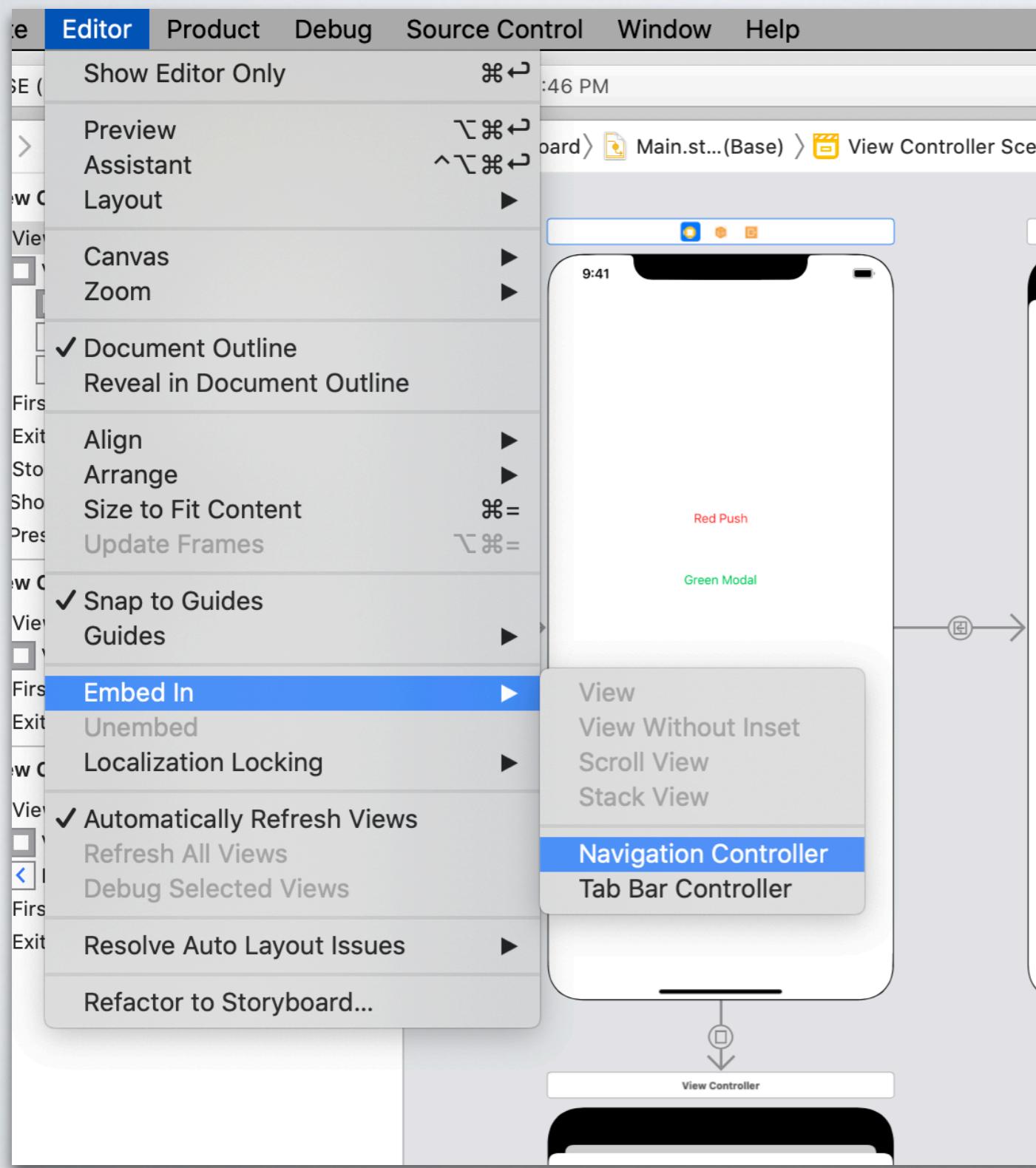
# STEP 6: ADD TWO VIEW CONTROLLERS TO THE STORYBOARD BY DRAGGING THEM OVER



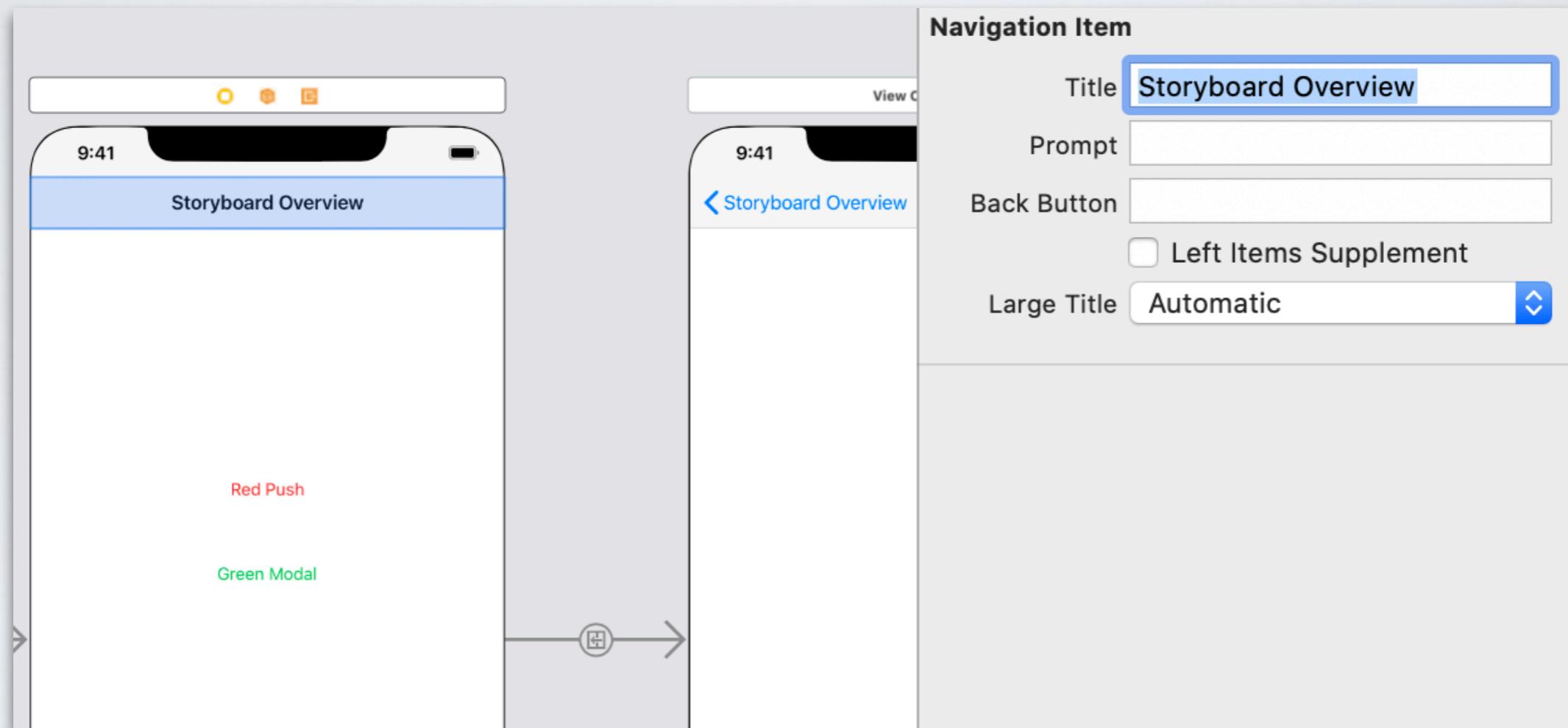
# STEP 7: CONNECT THE BUTTONS TO THE NEW VIEW CONTROLLERS BY HOLDING DOWN THE [CONTROL] KEY AND DRAGGING FROM THE BUTTON TO THE NEWVIEW CONTROLLER



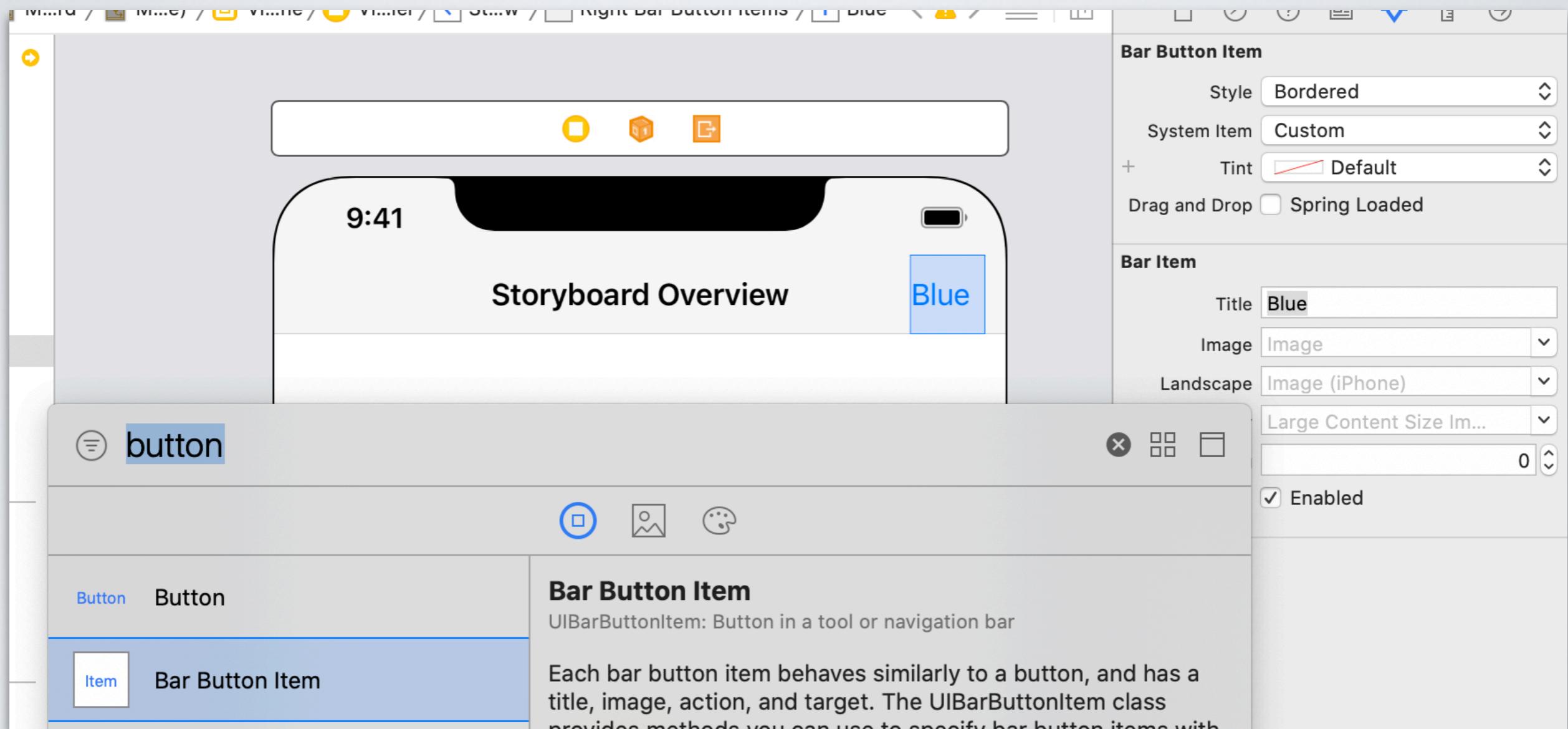
# STEP 8: EMBED A NAVIGATION CONTROLLER ON YOUR PRIMARY VIEW



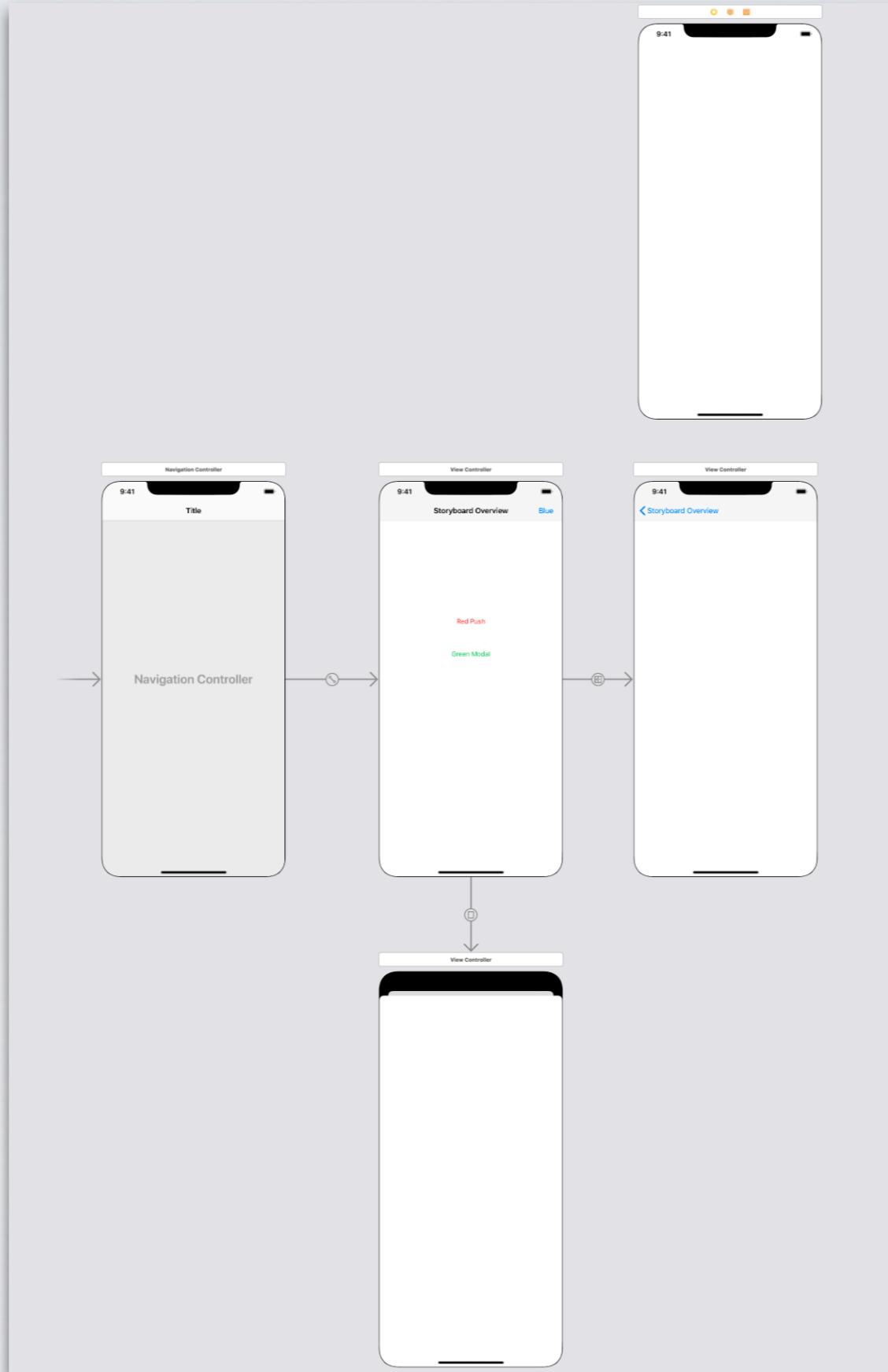
# STEP 9: CHANGE THE TITLE OF THE NAVIGATION BAR



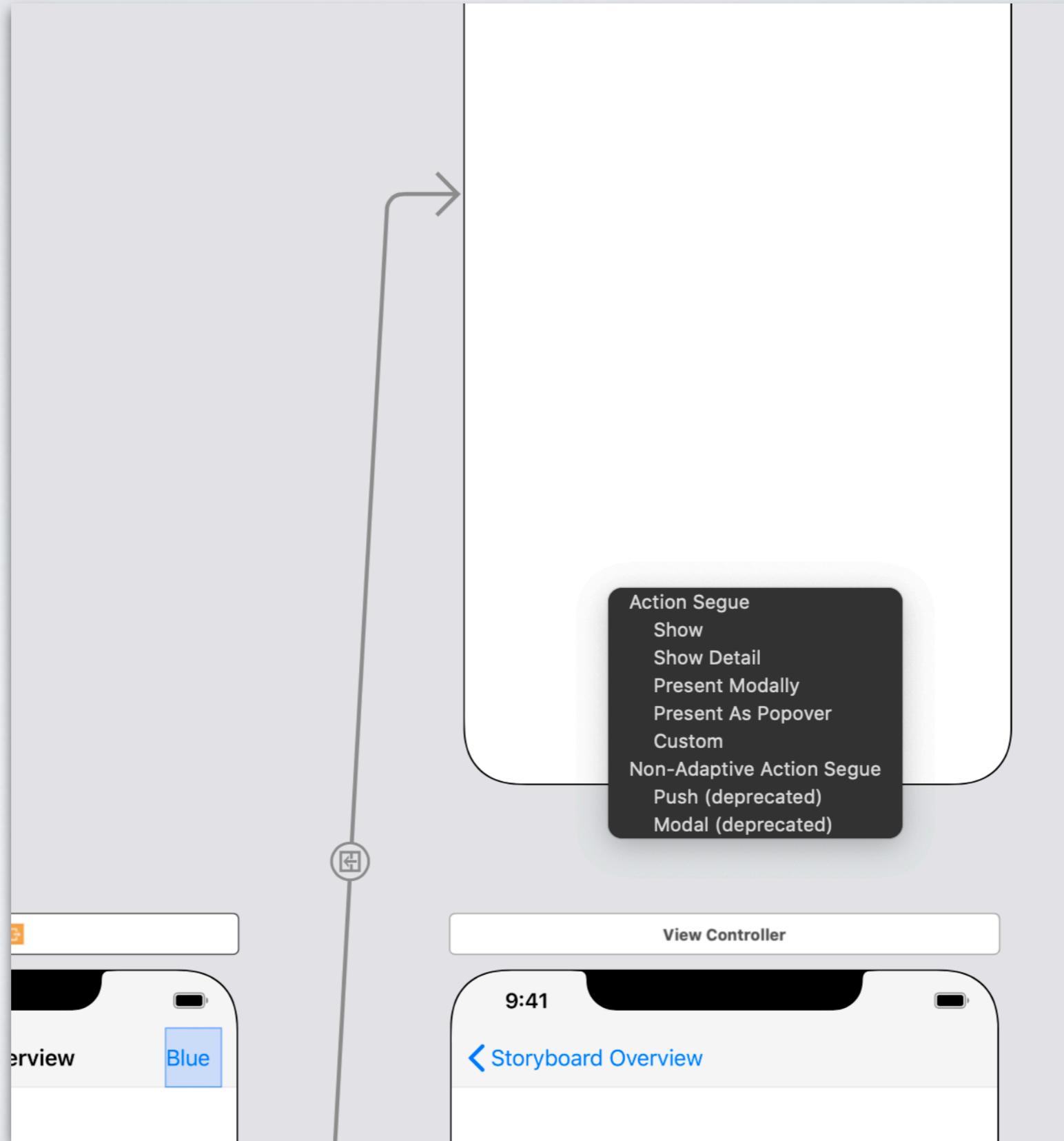
# STEP 10: ADD A BAR BUTTON ITEM TO THE NAVIGATION BAR AND CHANGE THE TITLE TO BLUE



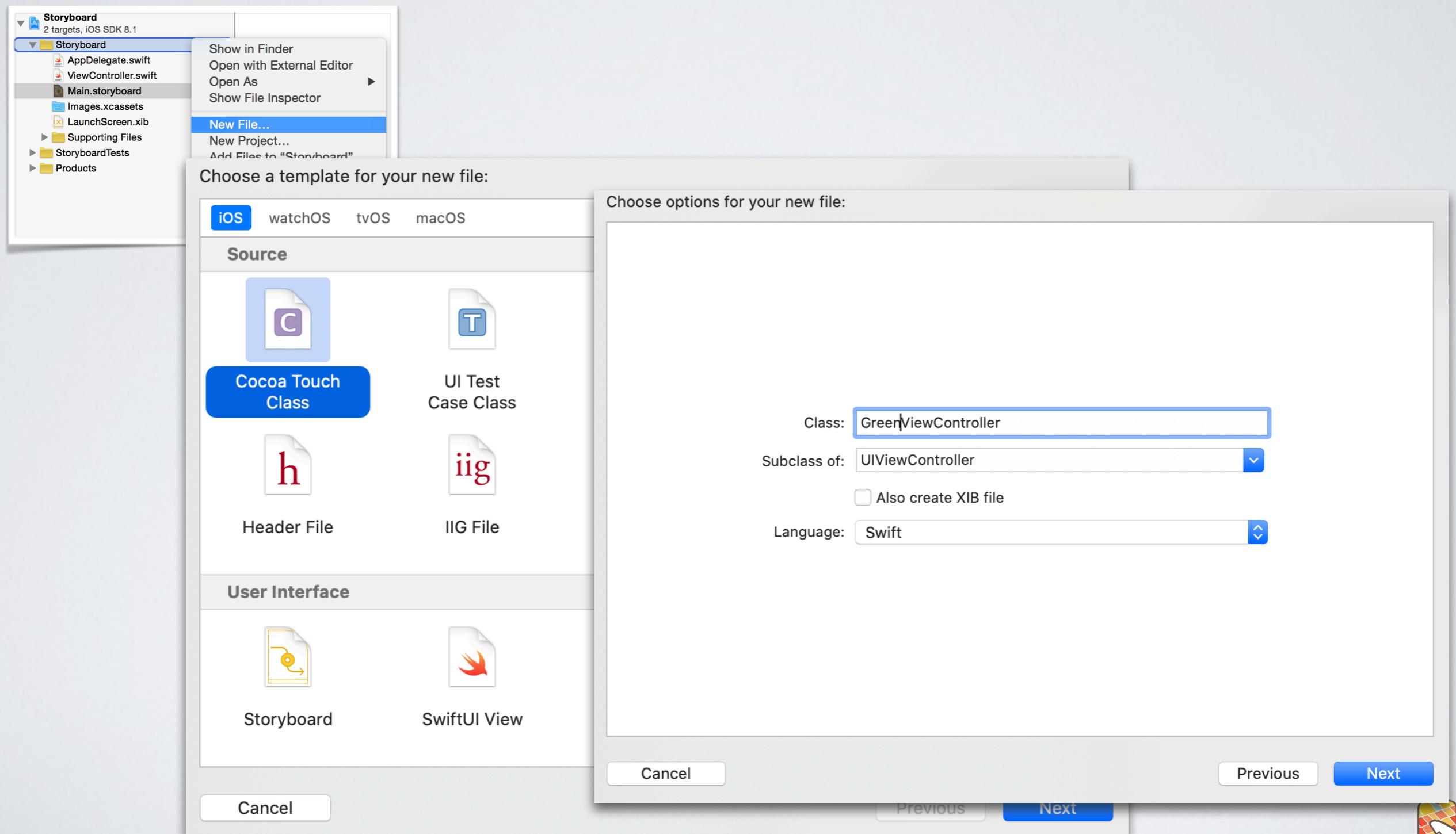
# STEP III: ADD OUR THIRD AND FINAL VIEW CONTROLLER TO THE STORYBOARD



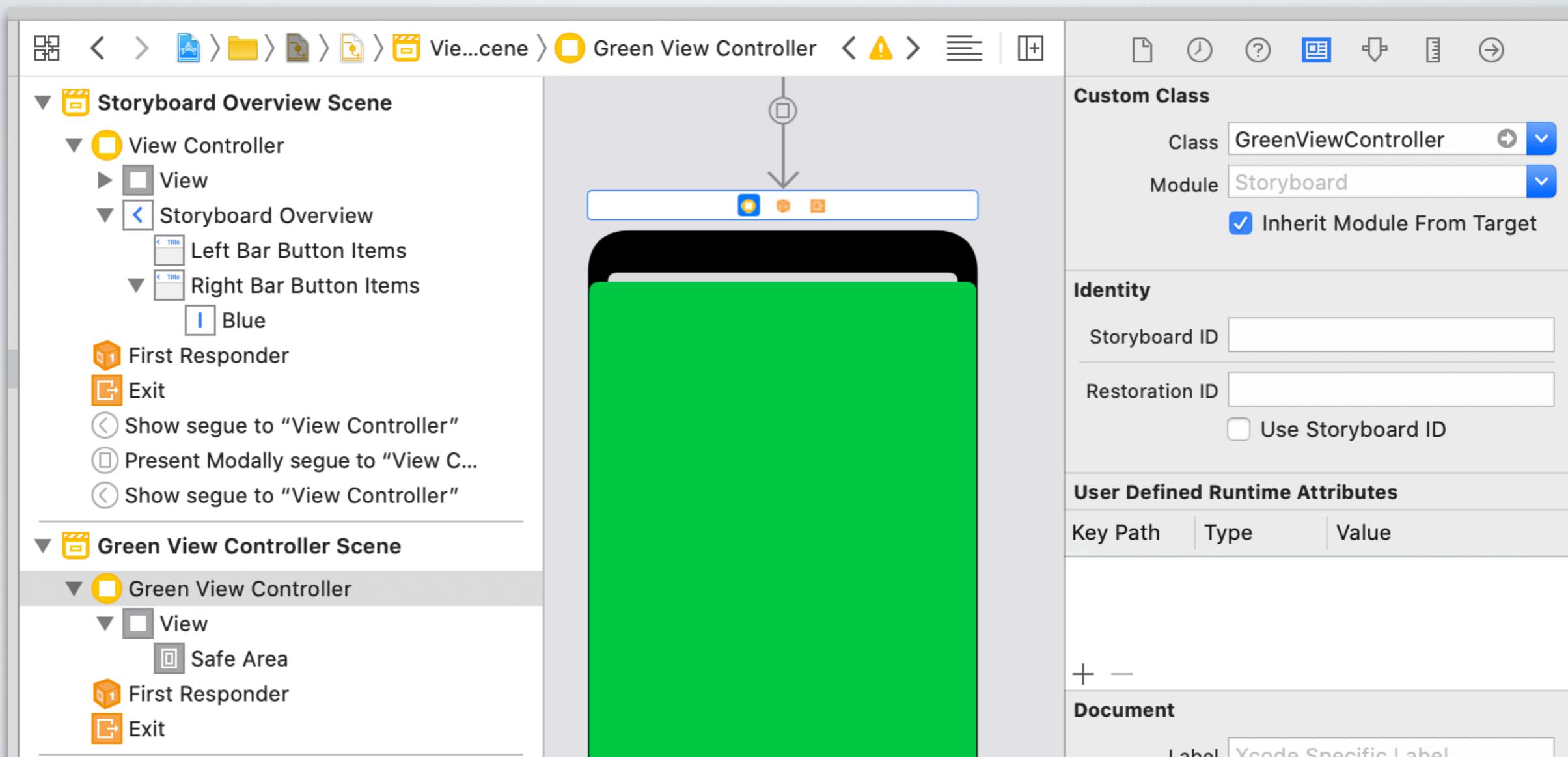
STEP 12: CONNECT OUR BAR BUTTON ITEM TO THE NEW VIEW CONTROLLER AND SELECT SHOW. REMEMBER TO HOLD DOWN THE [CONTROL] BUTTON WHILE CLICKING AND DRAGGING



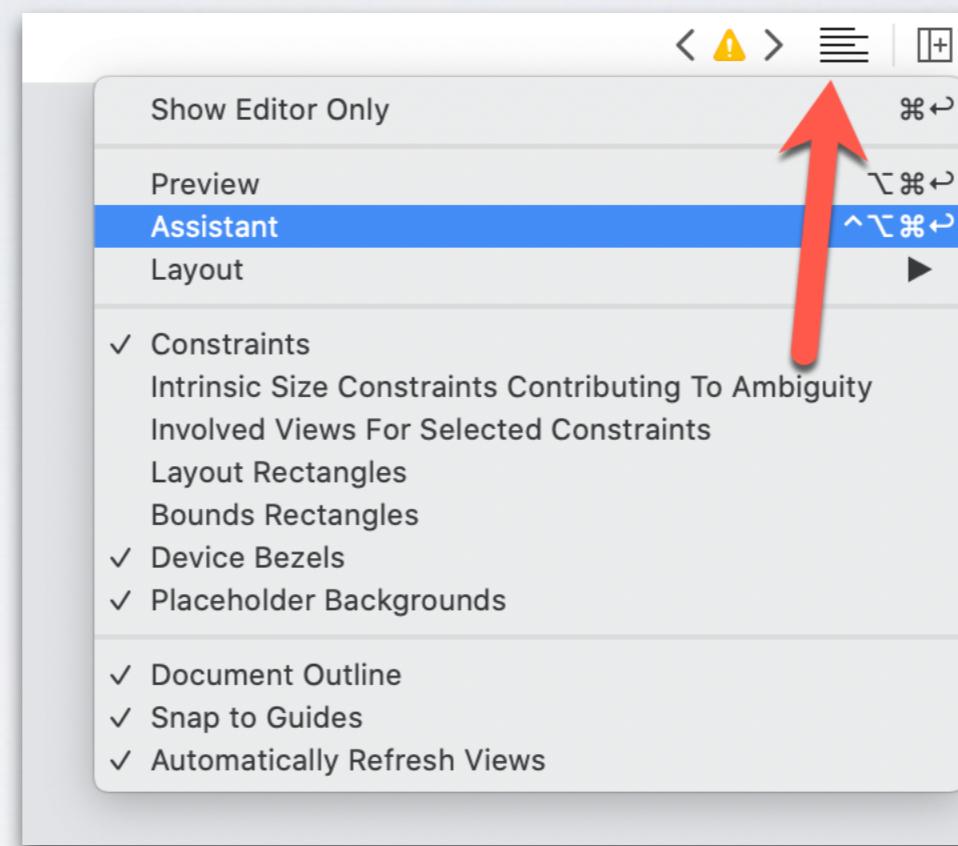
# STEP 13: NOW WE NEED TO DO A LITTLE PROGRAMMING. ADD A CONTROLLER OBJECT AND NAME IT “GREENVIEWCONTROLLER” AS SHOWN BELOW



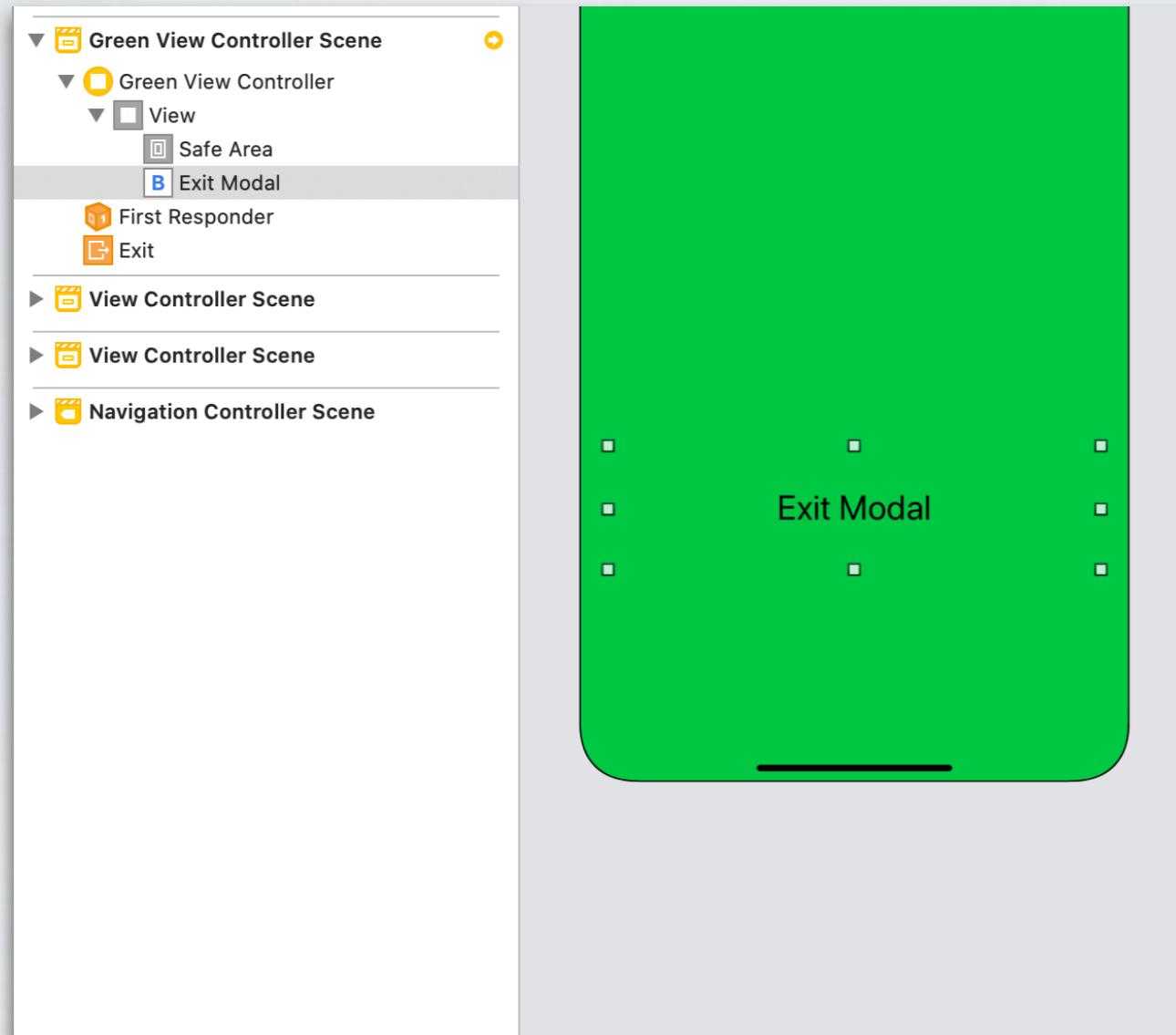
# STEP 14: ASSIGN YOUR VIEW IN THE STORYBOARD TO THE NEW OBJECTVIEW CONTROLLER YOU JUST CREATED



# STEP 15: SHOW THE ASSISTANT EDITOR TO HAVE OUR CODE GENERATED FOR US BY CLICKING



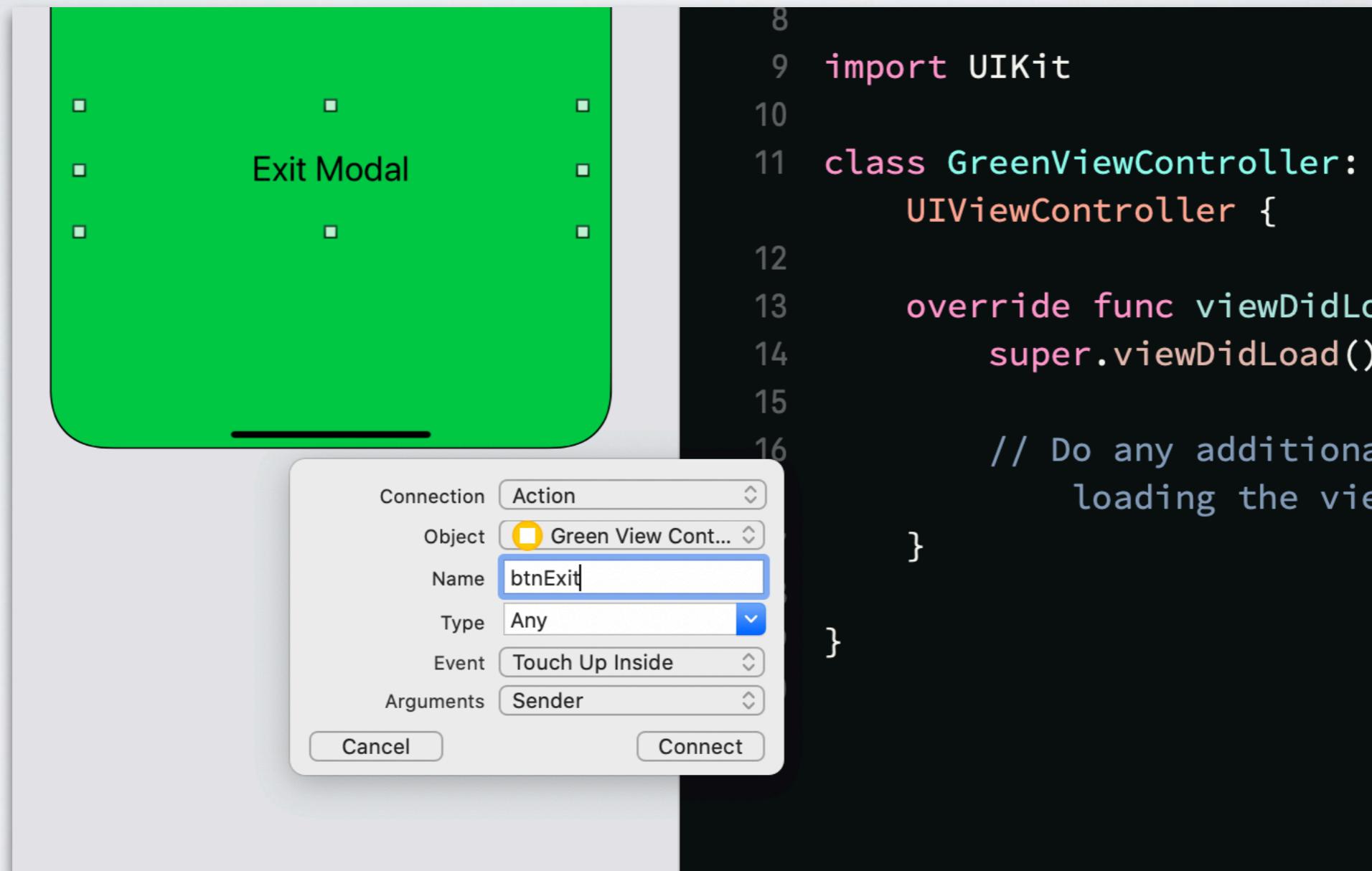
# STEP 16: ADD A BUTTON TO THE GREEN VIEW



```
2 //  GreenViewController.swift
3 //  Storyboard
4 //
5 //  Created by Don Miller on 8/16/20.
6 //  Copyright © 2020 GroundSpeed. All
7 //  rights reserved.
8
9 import UIKit
10
11 class GreenViewController:
12     UIViewController {
13
14     override func viewDidLoad() {
15         super.viewDidLoad()
16
17         // Do any additional setup after
18         // loading the view.
19     }
20 }
```



STEP 17: HOLD DOWN THE CONTROL WHEN CLICKING FROM THE BUTTON TO YOU HEADER FILE AND DRAG IT UNDER THE INTERFACE LINE. CHANGE THE CONNECTION TO ACTION AND THE NAME TO BTNEXTIT

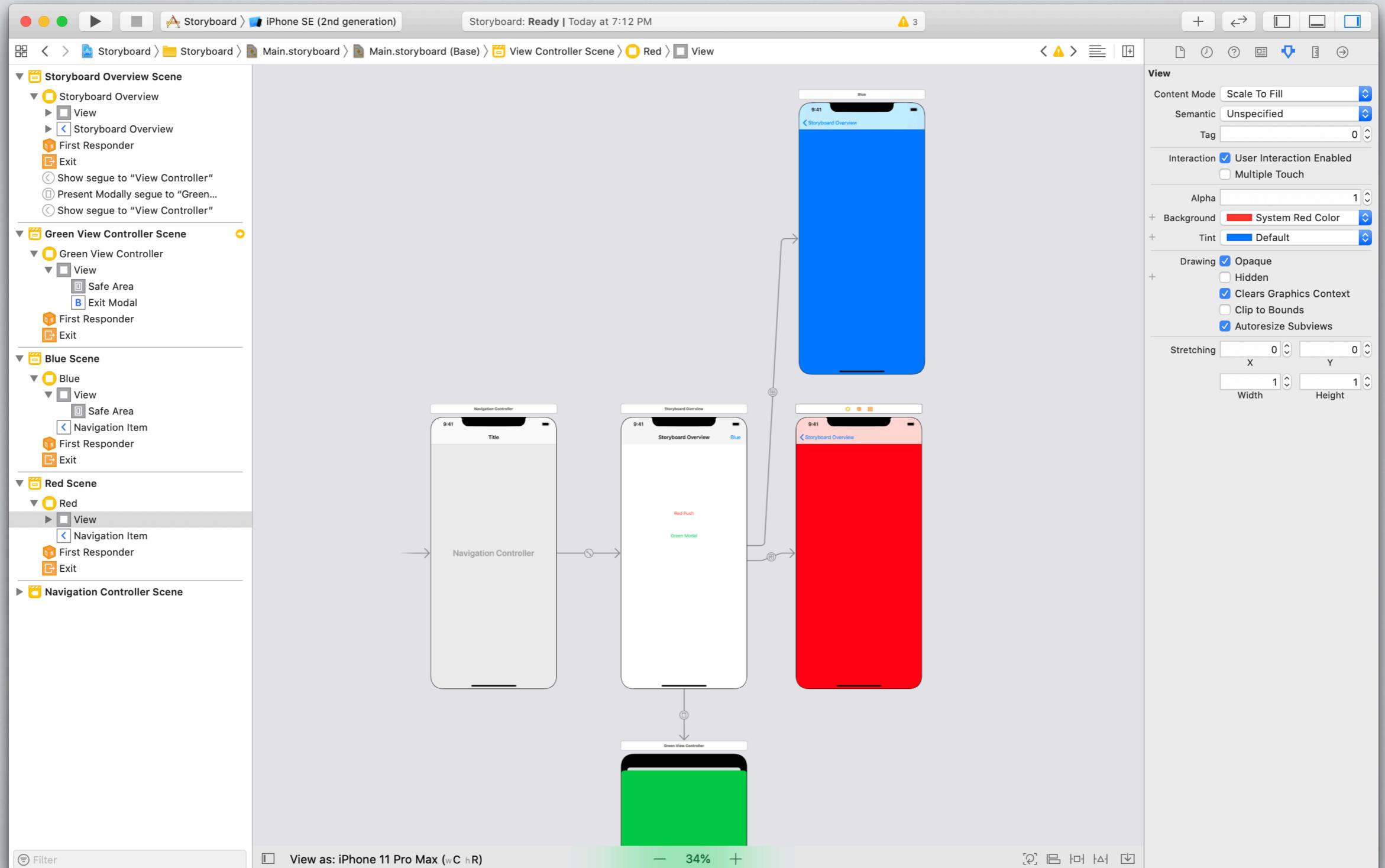


# STEP 18: ADD THE FOLLOWING LINE OF CODE IN THE BTNEXTIT METHOD TO DISMISS THE MODAL WINDOW

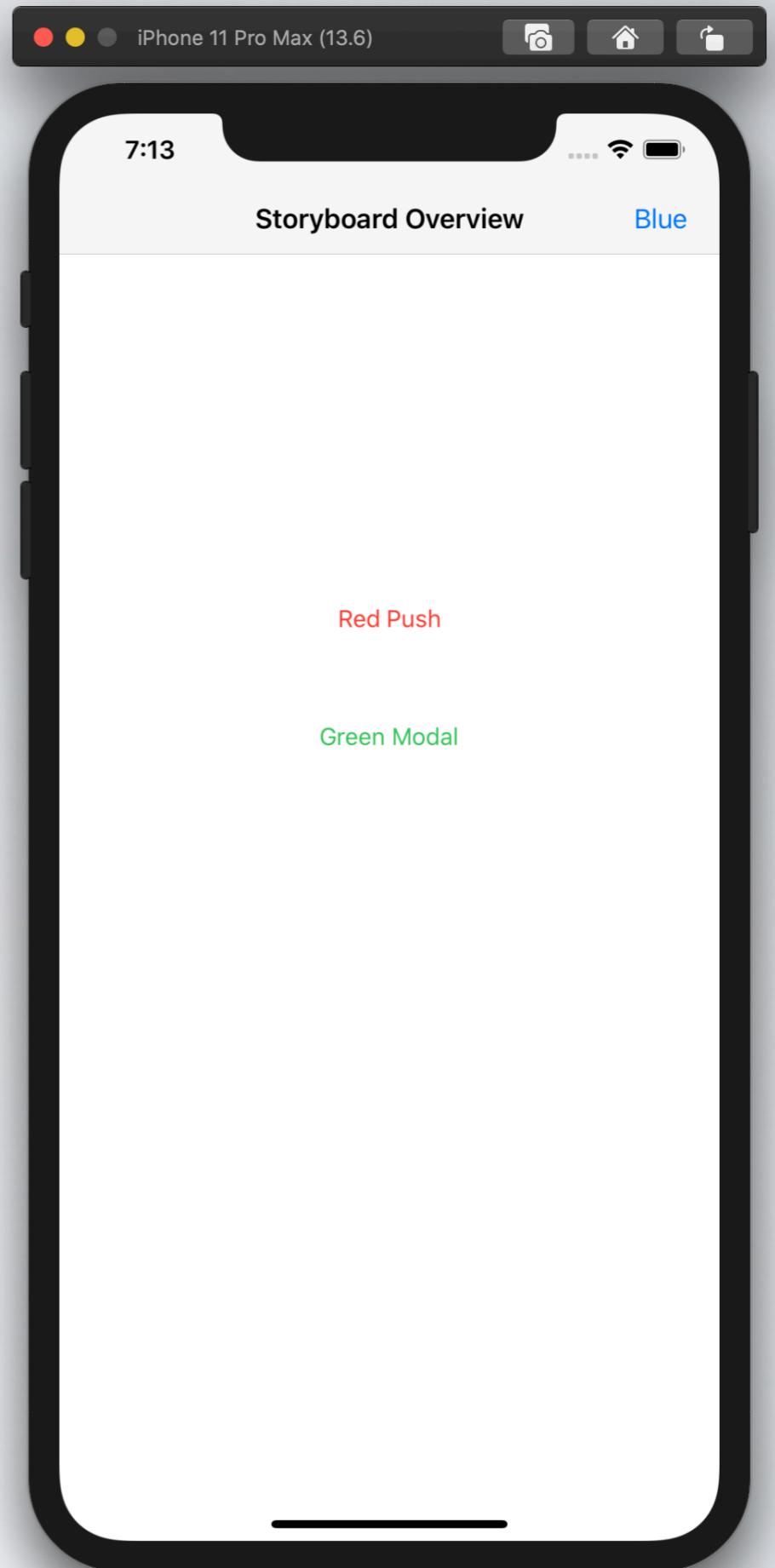
```
8  
9 import UIKit  
10  
11 class GreenViewController: UIViewController {  
12  
13     override func viewDidLoad() {  
14         super.viewDidLoad()  
15  
16         // Do any additional setup after loading the view.  
17     }  
18  
19     @IBAction func btnExit(_ sender: Any) {  
20         self.dismiss(animated: true, completion: nil)  
21     }  
22 }  
23
```



# STEP 19: FINAL TOUCHES. CHANGE THE VIEW CONTROLLER TITLES AND BACKGROUNDS TO REFLECT THE COLOR DESCRIBED IN THE CALLING BUTTON



# STEP 20: RUN IT!



GroundSpeed™  
rapid web + mobile software

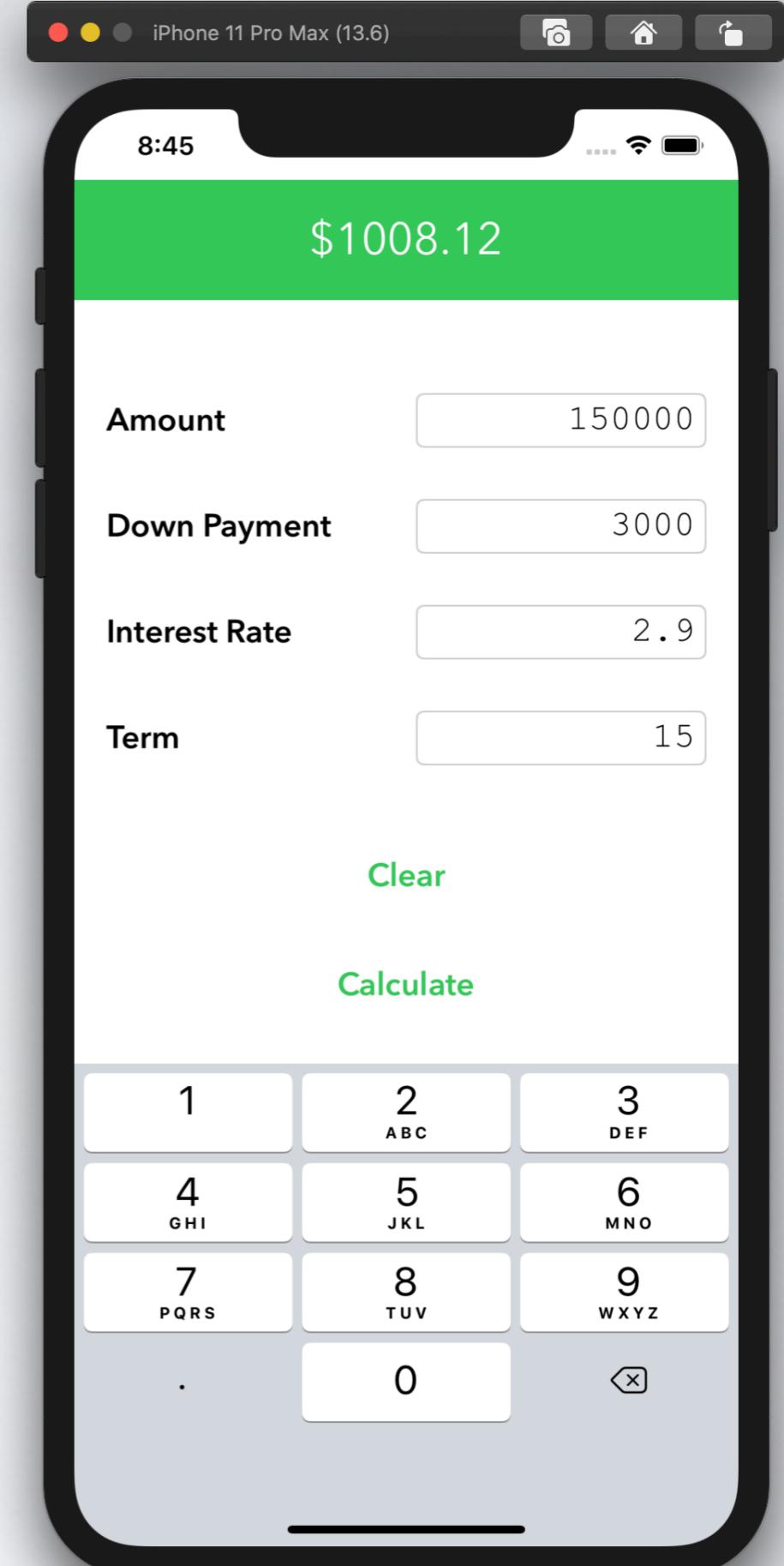
# Lab #4 - Create Amortize Payment App



# CREATE AN APP TO CALCULATE AN AMORTIZATION PAYMENT

1. Create 5 Labels
2. Create 4 Text Boxes
3. Create 2 Buttons
  - A. Clear clears all text boxes and total label
  - B. Calculate creates payment amount on top label

$$P = A \cdot \frac{1 - \left(\frac{1}{1+r}\right)^n}{r}$$



```

func getMonthlyPayment() {
    // A = payment Amount per period
    // P = initial Principal (loan amount)
    // r = interest rate per period
    // n = total number of payments or periods

    let principal : Float = Float(txtAmount.text!)! - Float(txtDownPayment.text!)!
    let payments = Float(txtTerm.text!)!*12
    let rate = Float(txtInterestRate.text!)!/12/100
    let amount = calculatPMTWithRatePerPeriod(ratePerPeriod: rate, numberOfPayments: payments, loanAmount:
        principal, futureValue: 0, type: 0)

    if (amount.isNaN || amount.isInfinite)
    {
        lblPayment.font = UIFont.boldSystemFont(ofSize: 18)
        lblPayment.textColor = UIColor.red
        lblPayment.text = "You must enter all required fields."
    }
    else
    {
        lblPayment.font = UIFont(name: "Avenir Next", size: 28)
        lblPayment.textColor = UIColor.white
        lblPayment.text = String(format: "$%.02f", amount)
    }
}

func calculatPMTWithRatePerPeriod (ratePerPeriod: Float, numberOfPayments: Float, loanAmount: Float, futureValue:
    Float, type: Float) -> Float {

    var q : Float

    q = pow(1 + ratePerPeriod, numberOfPayments)

    let returnValue = (ratePerPeriod * (futureValue + (q * loanAmount))) / ((-1 + q) * (1 + ratePerPeriod *
        (type)))

    return returnValue
}

```

$$P = A \cdot \frac{1 - \left(\frac{1}{1+r}\right)^n}{r}$$

# Tab Bar and Navigation Controllers



# CREATE A NEW APPLICATION

Choose a template for your new project:

iOS watchOS tvOS macOS Cross-platform Filter

**Application**

 Single View App	 Game	 Augmented Reality App	 Document Based App	 Master-Detail App
 Tabbed App	 Sticker Pack App	 iMessage App		

**Framework & Library**

 Framework	 Static Library	 Metal Library
--	--	--

Cancel Previous Next



# CREATE A NEW APPLICATION

Choose options for your new project:

Product Name:  None

Organization Name:

Organization Identifier:

Bundle Identifier: com.groundspeedhq.NavDemo

Language:  Storyboard

User Interface: Storyboard

Use Core Data

Use CloudKit

Include Unit Tests

Include UI Tests

Cancel

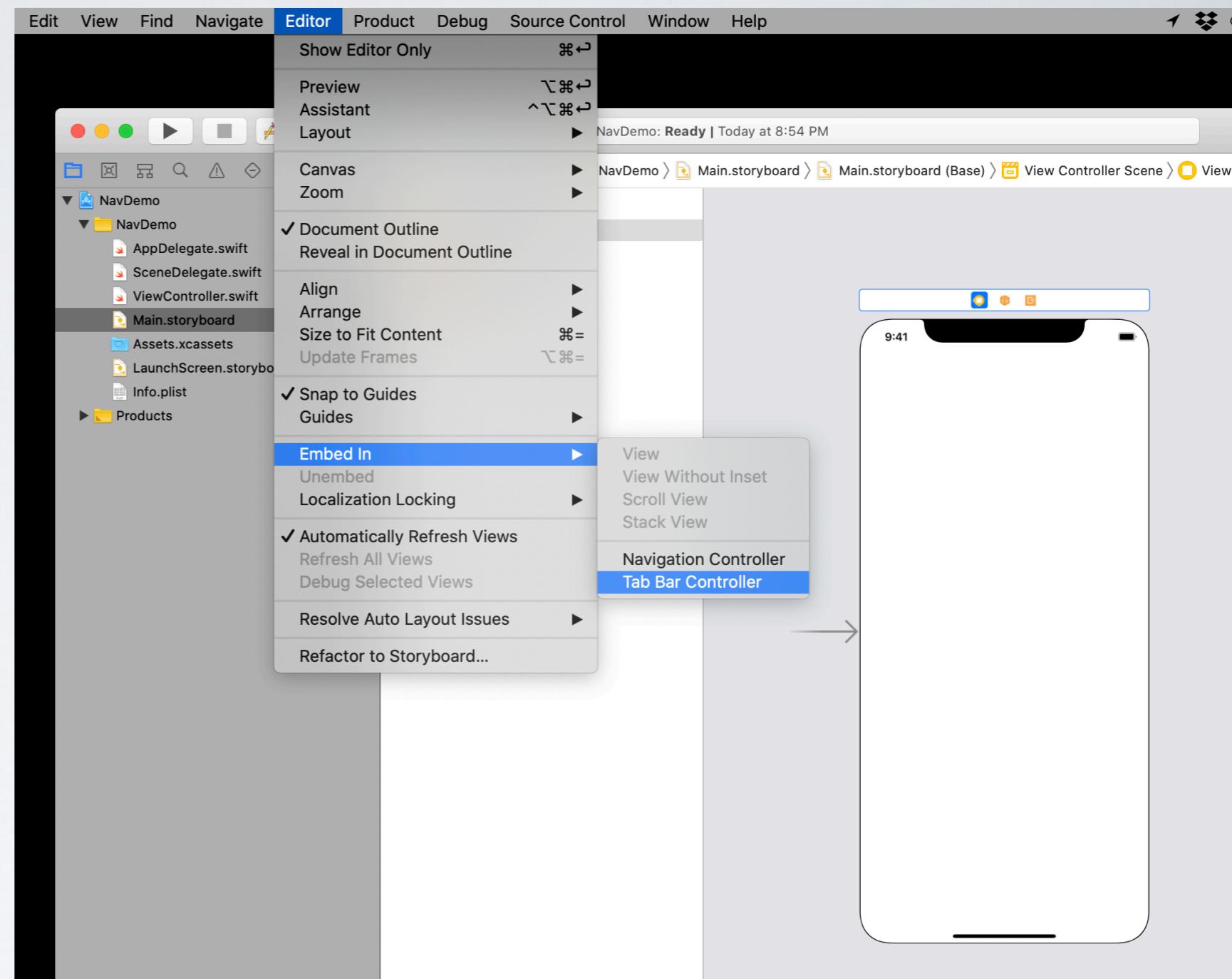
Previous

Next

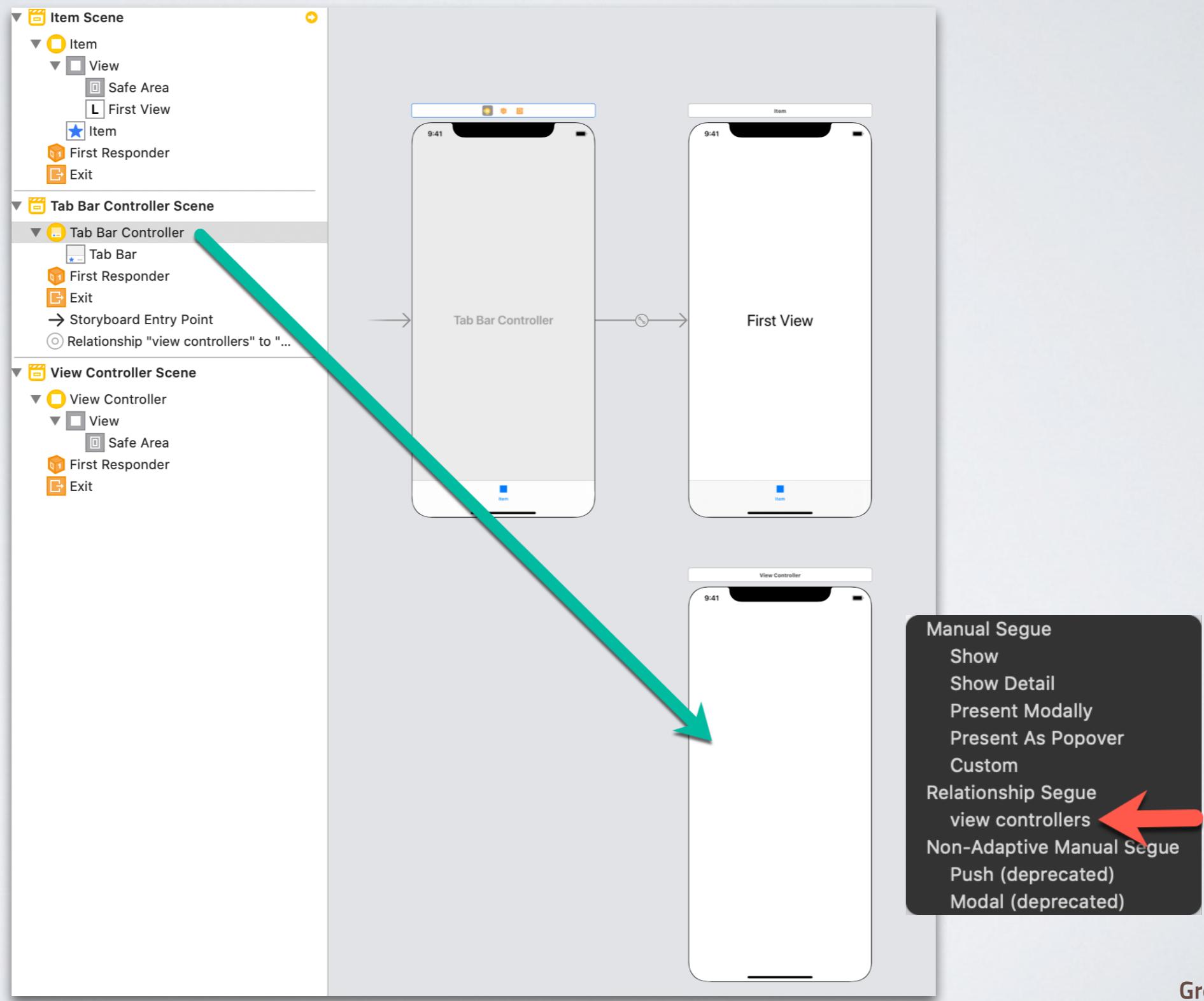


GroundSpeed™  
rapid web + mobile software

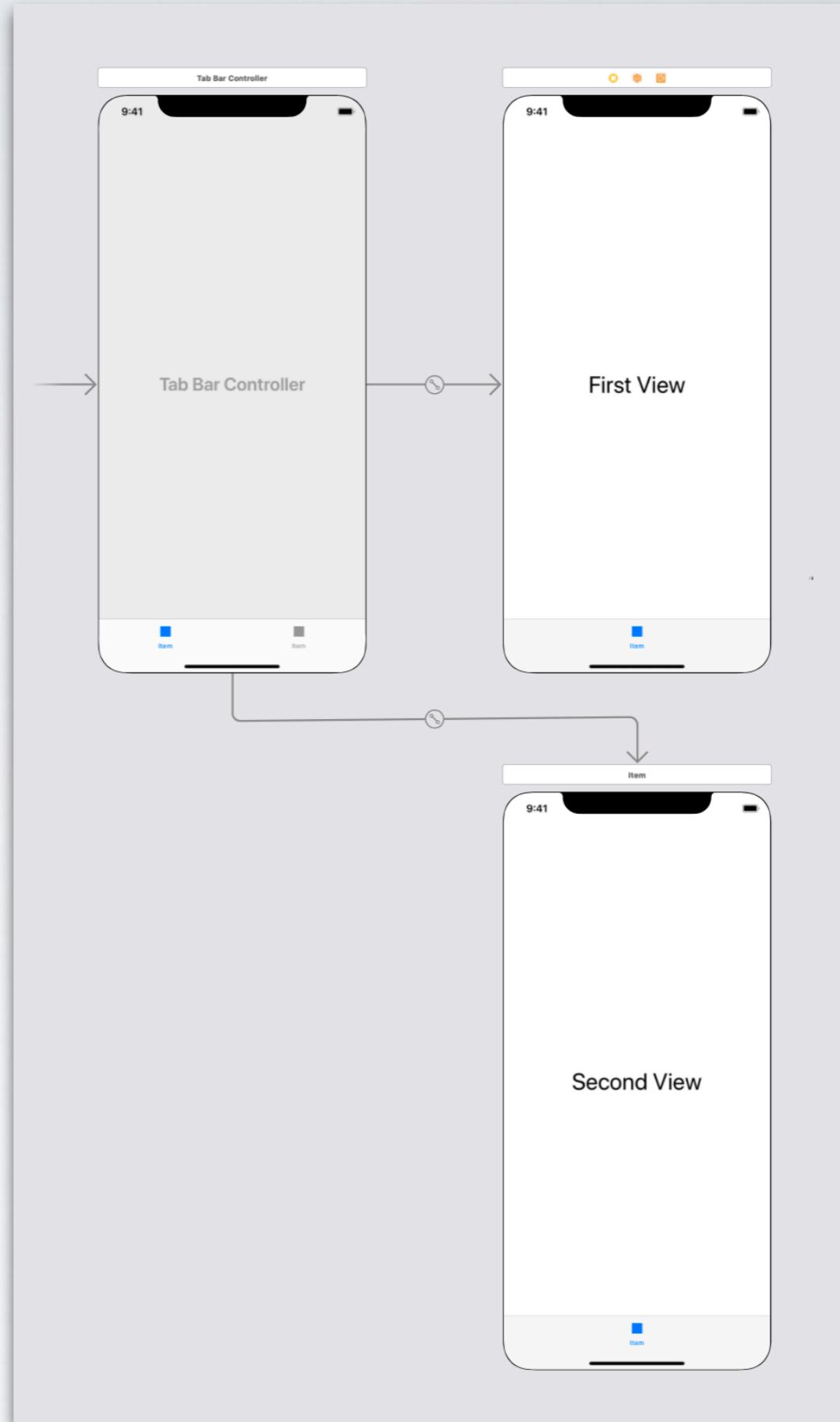
# INSIDE THE STORYBOARD, LET'S EMBED OUR INITIAL CONTROLLER WITHIN A TAB BAR CONTROLLER



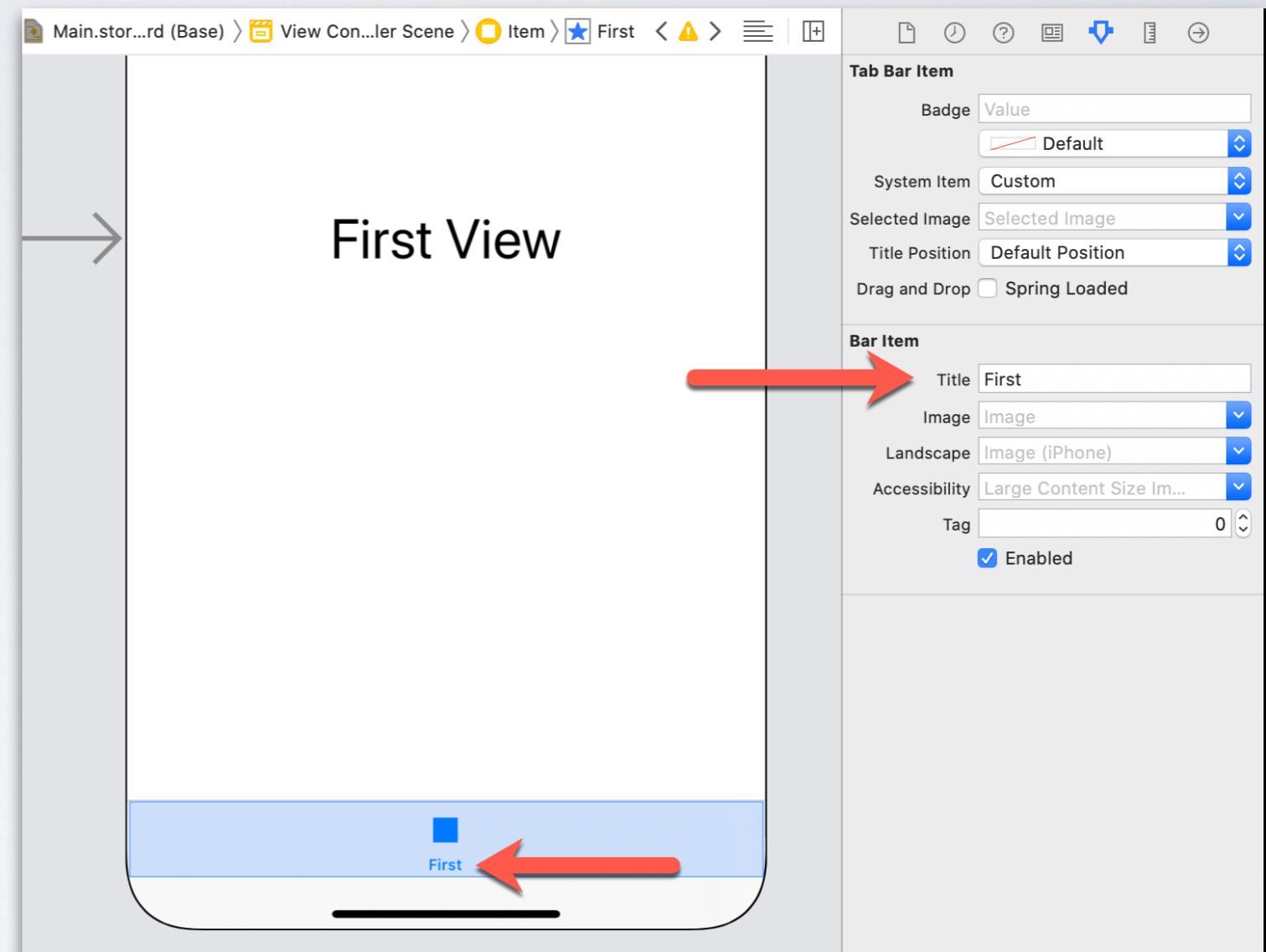
ADD A SECOND VIEWCONTROLLER THEN HOLD CONTROL BUTTON AND DRAG FROM TAB BAR CONTROLLER TO THE NEWVIEW CONTROLLER



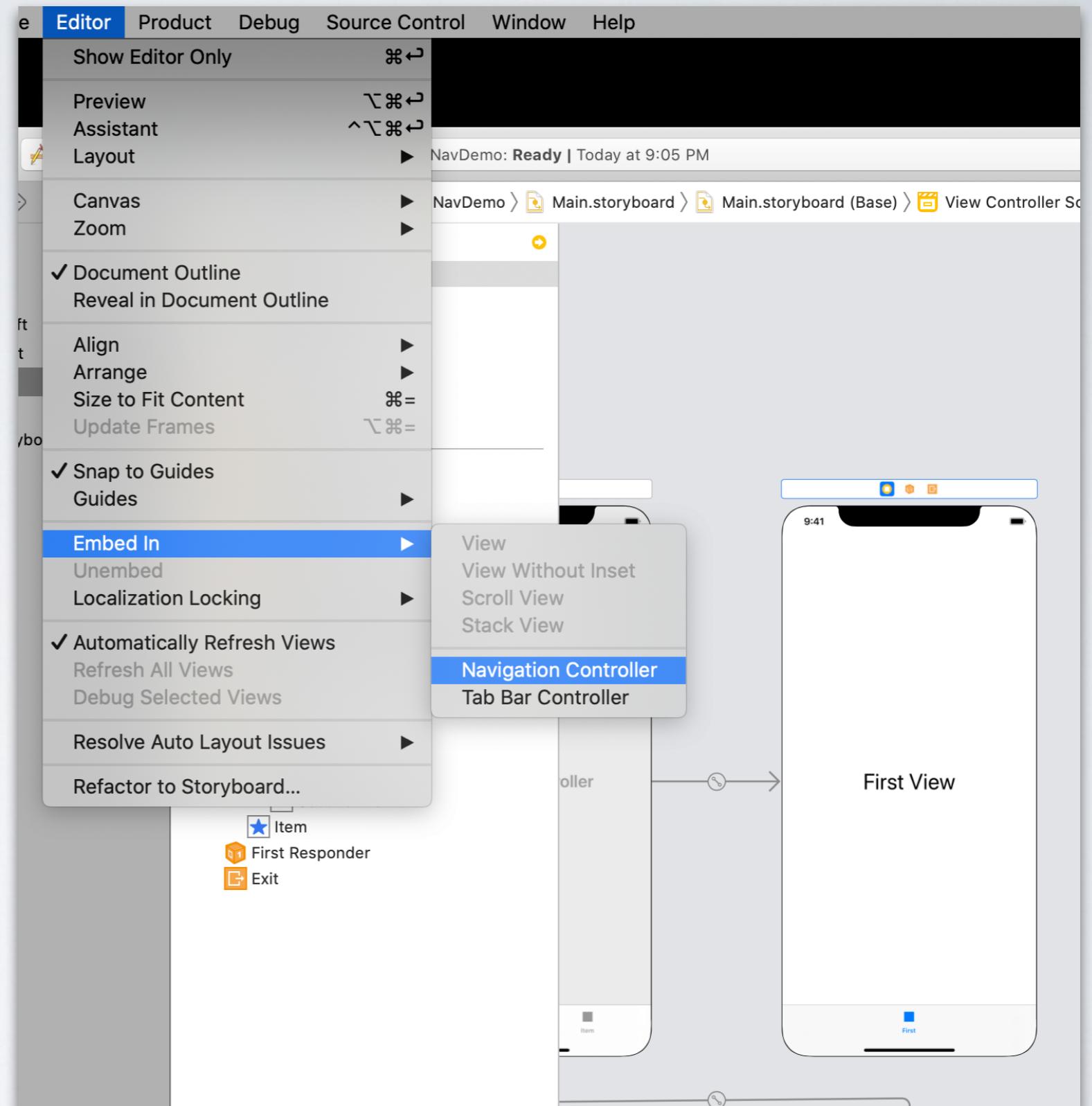
YOUR  
STORYBOARD  
SHOULD LOOK  
LIKE THIS...



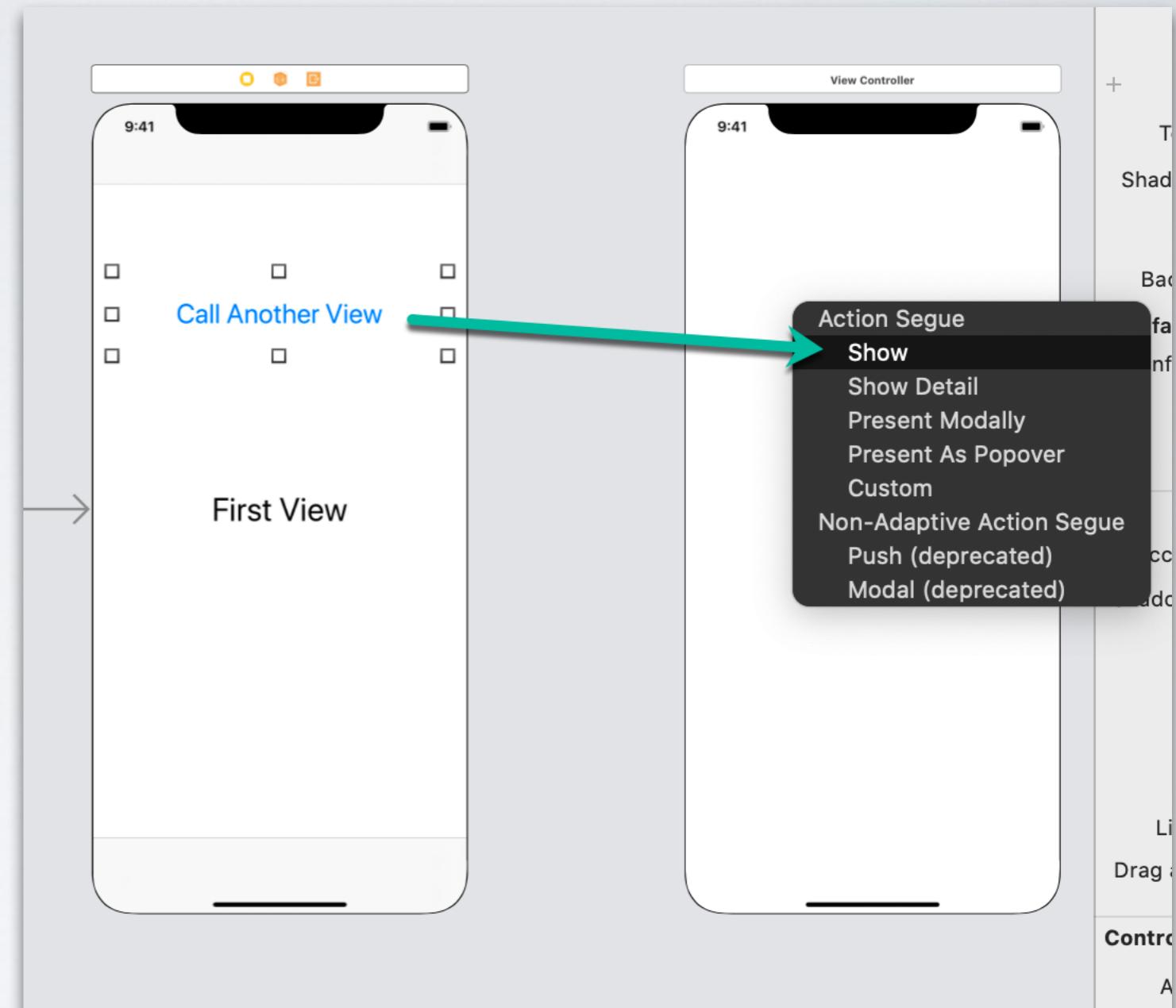
THE IMAGES OF  
THE TAB BAR  
BUTTONS ARE  
CONFIGURED  
ON EACH  
INDIVIDUAL VIEW  
CONTROLLER



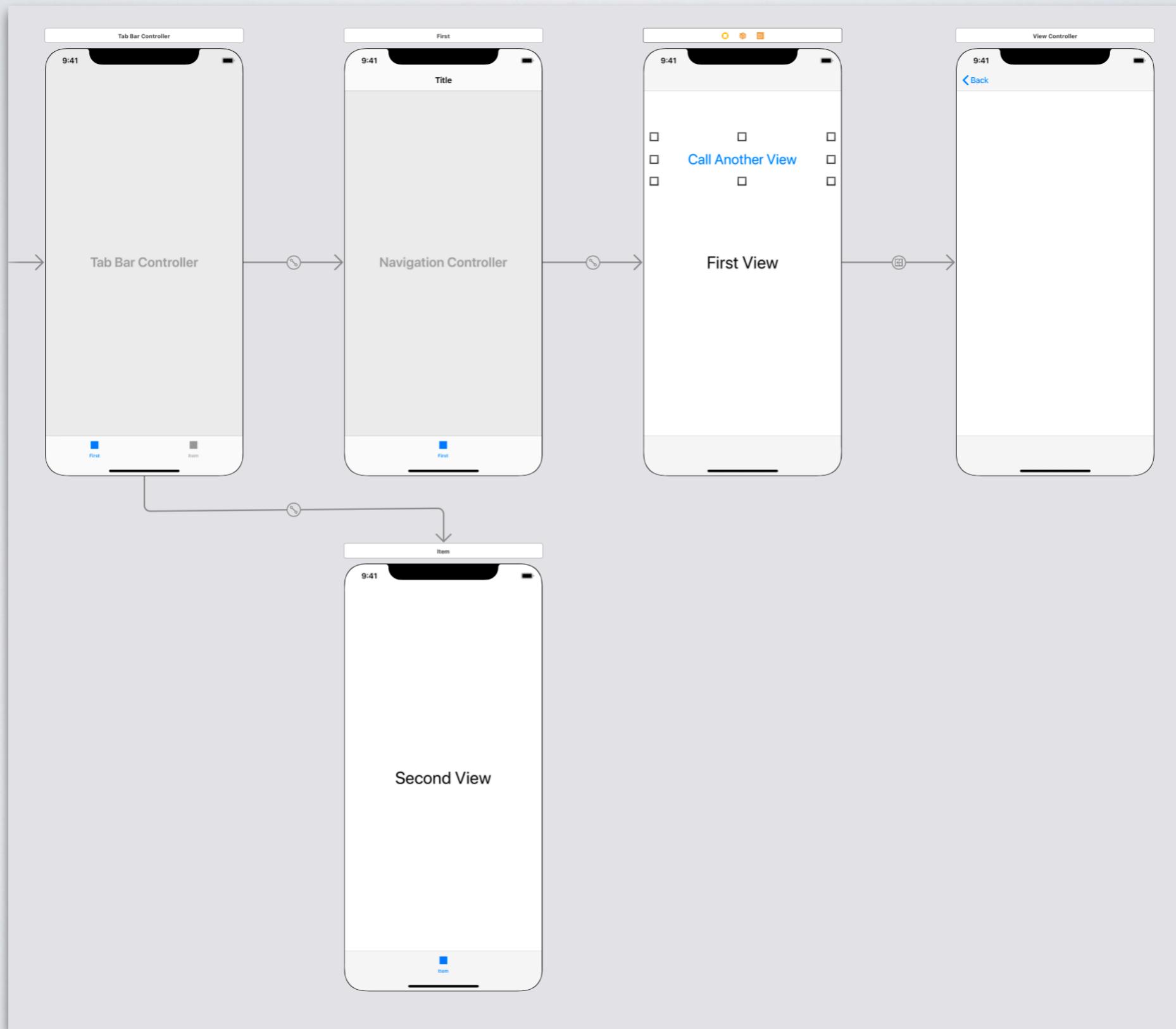
# THE TAB VIEW CAN CALL ANOTHER FORM BY EMBEDDING IT INTO A NAVIGATION CONTROLLER

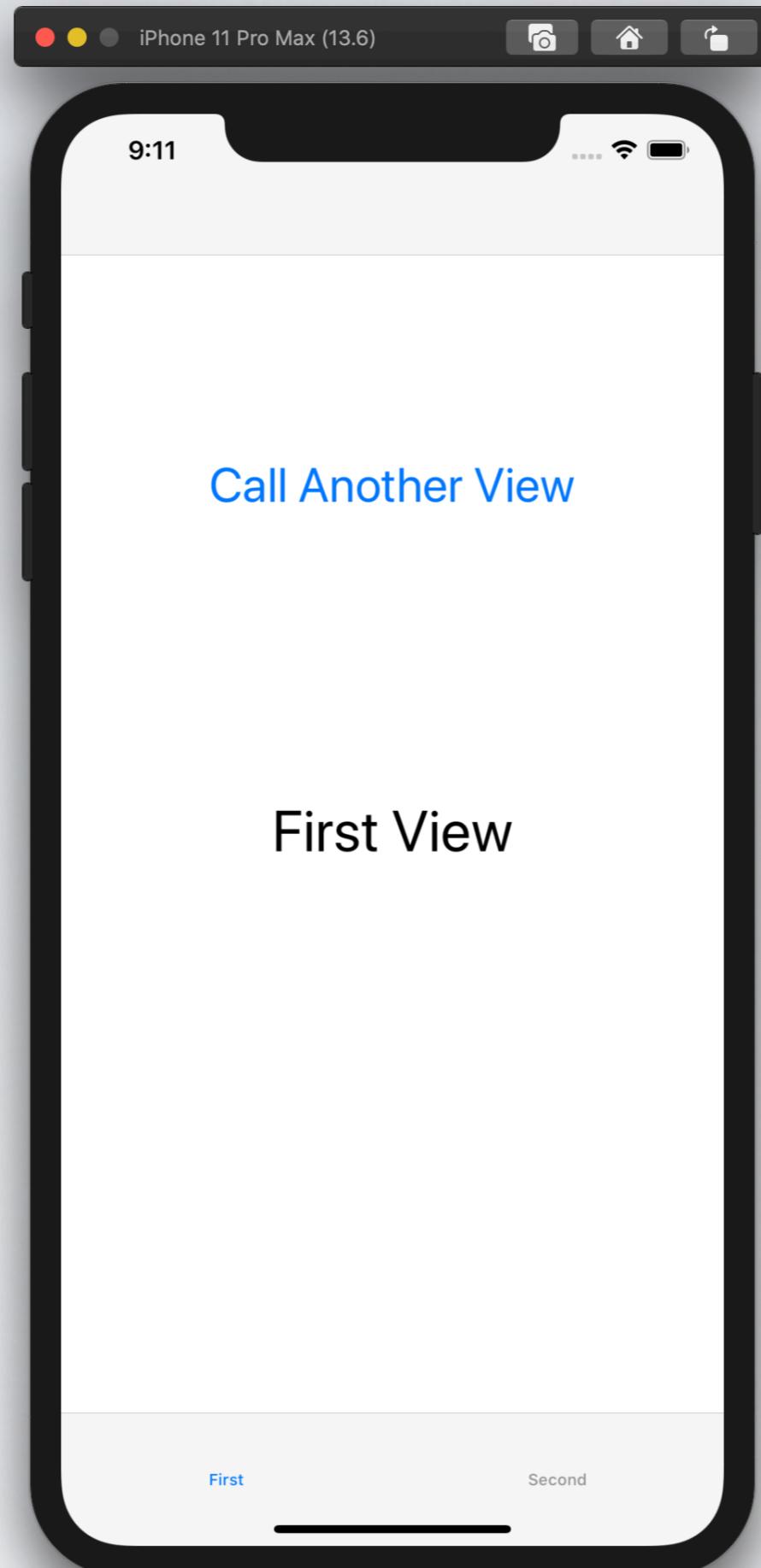


NAVIGATION TO  
THE NEW FORM  
CAN BE  
HANDLED BY  
CREATING A  
SEGUE FROM THE  
BUTTON TO THE  
NEW FORM  
(CONTROL-  
CLICK-DRAG)



# STORYBOARD VIEW





**GroundSpeed™**  
rapid web + mobile software

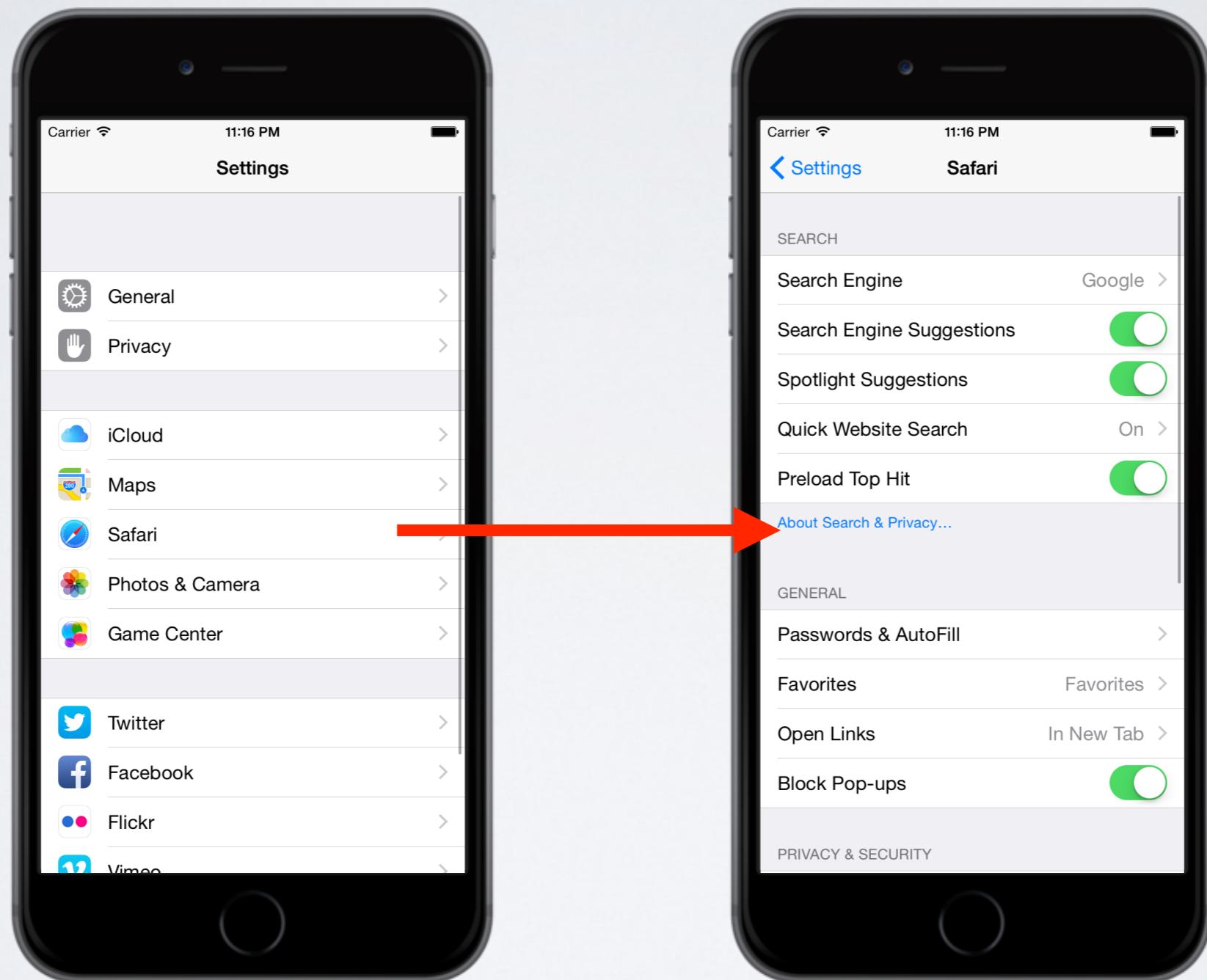
# Lab #5 - Convert Payment App to Tabbed View



# UITableViews

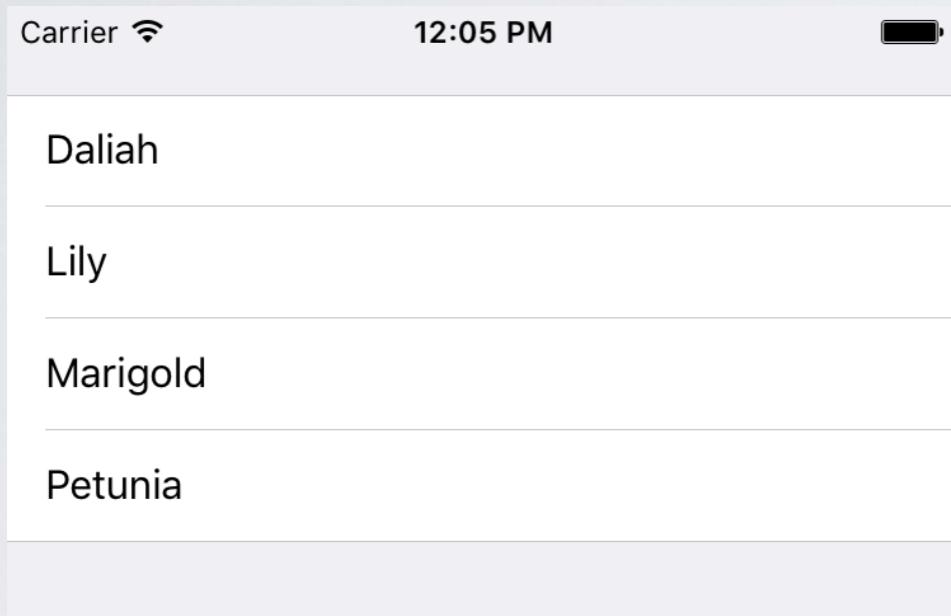


# UITABLEVIEW OVERVIEW



# UITABLEVIEW OVERVIEW

## A Simple UITableView with 4 Rows



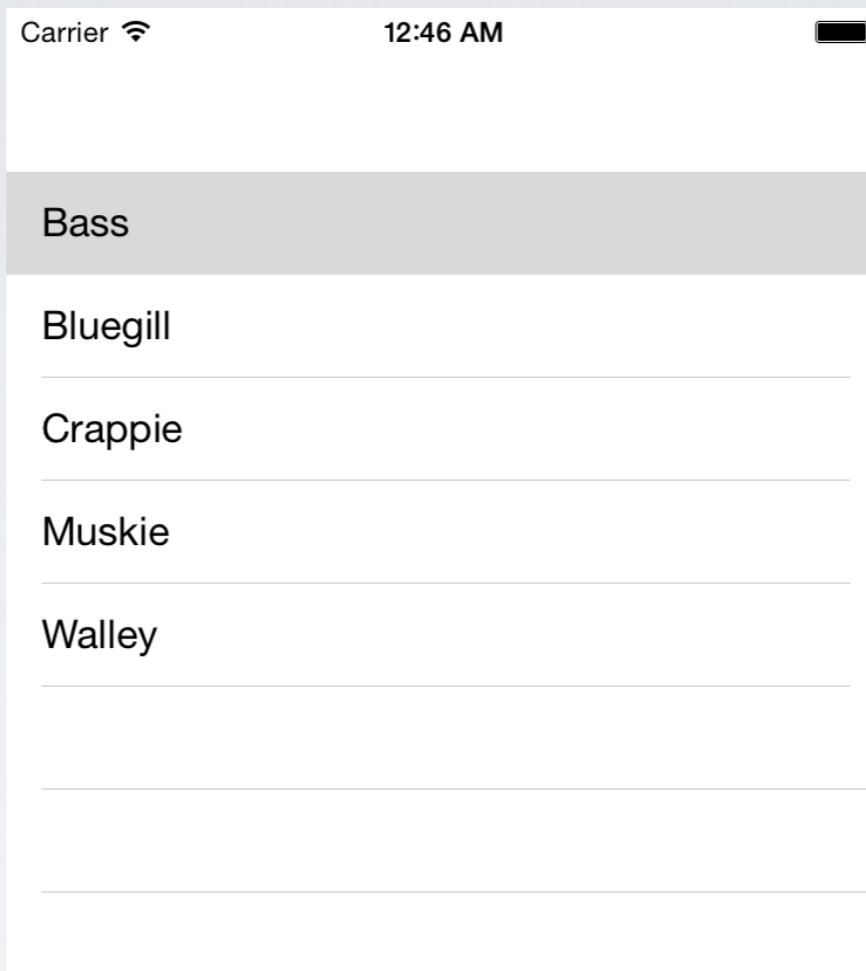
## A Sectioned UITableView

TODAY	
30 Year Fixed	3.78%
15 Year Fixed	3.01%
5/1 ARM	2.96%
LAST WEEK	
30 Year Fixed	3.80%
15 Year Fixed	2.99%
5/1 ARM	2.99%

- Programmatically, the UIKit identifies rows and sections through their index number.
- Sections are numbered 0 through n-1 from the top of a table
- Rows are numbered 0 through n-1 within a section
- As Mentioned earlier, the table view comes in one of two basic styles: plain or grouped.



# TABLE VIEW EXAMPLE

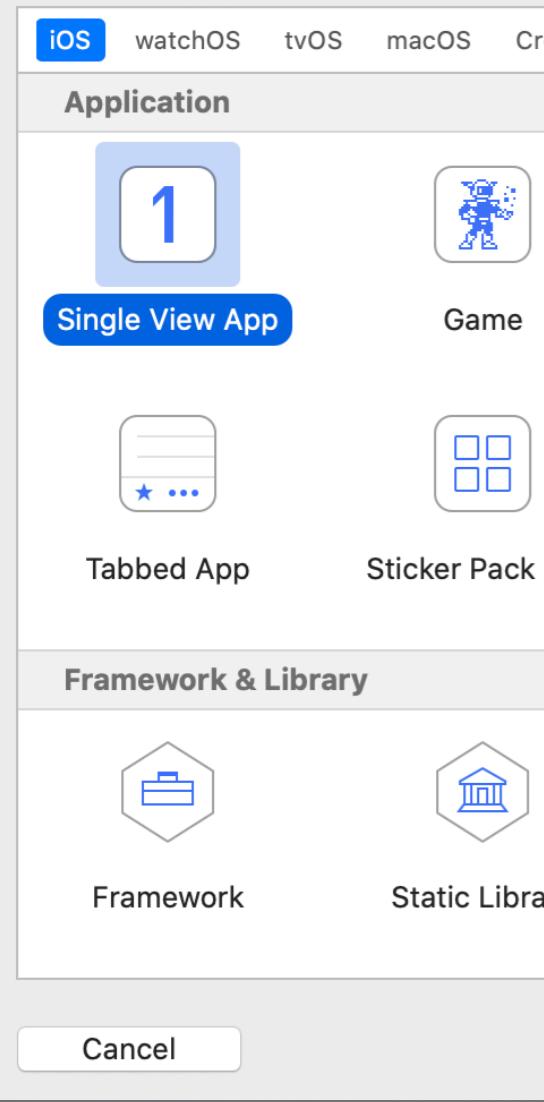


In this example, we will develop a simple table view to display the names of our fish from the example earlier. The completed project will run as shown.



# CREATE A SINGLE VIEW BASED APP.

Choose a template for your new project:



The screenshot shows the Xcode New Project Assistant. On the left, there's a sidebar with categories: iOS, watchOS, tvOS, macOS, and Cross-platform. Under the Application category, 'Single View App' is selected and highlighted with a blue border. Other options include Game, Tabbed App, Sticker Pack, Framework & Library, Framework, and Static Library. At the bottom of the sidebar is a 'Cancel' button. To the right, a large modal window titled 'Choose options for your new project:' displays the text 'Let's name it: TblVu1'. It contains fields for Product Name (Tb1Vu1), Team (None), Organization Name (GroundSpeed), Organization Identifier (com.groundspeedhq), Bundle Identifier (com.groundspeedhq.Tb1Vu1), Language (Swift), and User Interface (Storyboard). Below these are four unchecked checkboxes: 'Use Core Data', 'Use CloudKit', 'Include Unit Tests', and 'Include UI Tests'. At the bottom of the modal are 'Cancel', 'Previous', and 'Next' buttons. The 'Next' button is highlighted in blue.

Choose a template for your new project:

iOS watchOS tvOS macOS Cross-platform

Application

Single View App

Game

Tabbed App

Sticker Pack

Framework & Library

Framework

Static Library

Cancel

Choose options for your new project:

## Let's name it: Tb1Vu1

Product Name: Tb1Vu1

Team: None

Organization Name: GroundSpeed

Organization Identifier: com.groundspeedhq

Bundle Identifier: com.groundspeedhq.Tb1Vu1

Language: Swift

User Interface: Storyboard

Use Core Data

Use CloudKit

Include Unit Tests

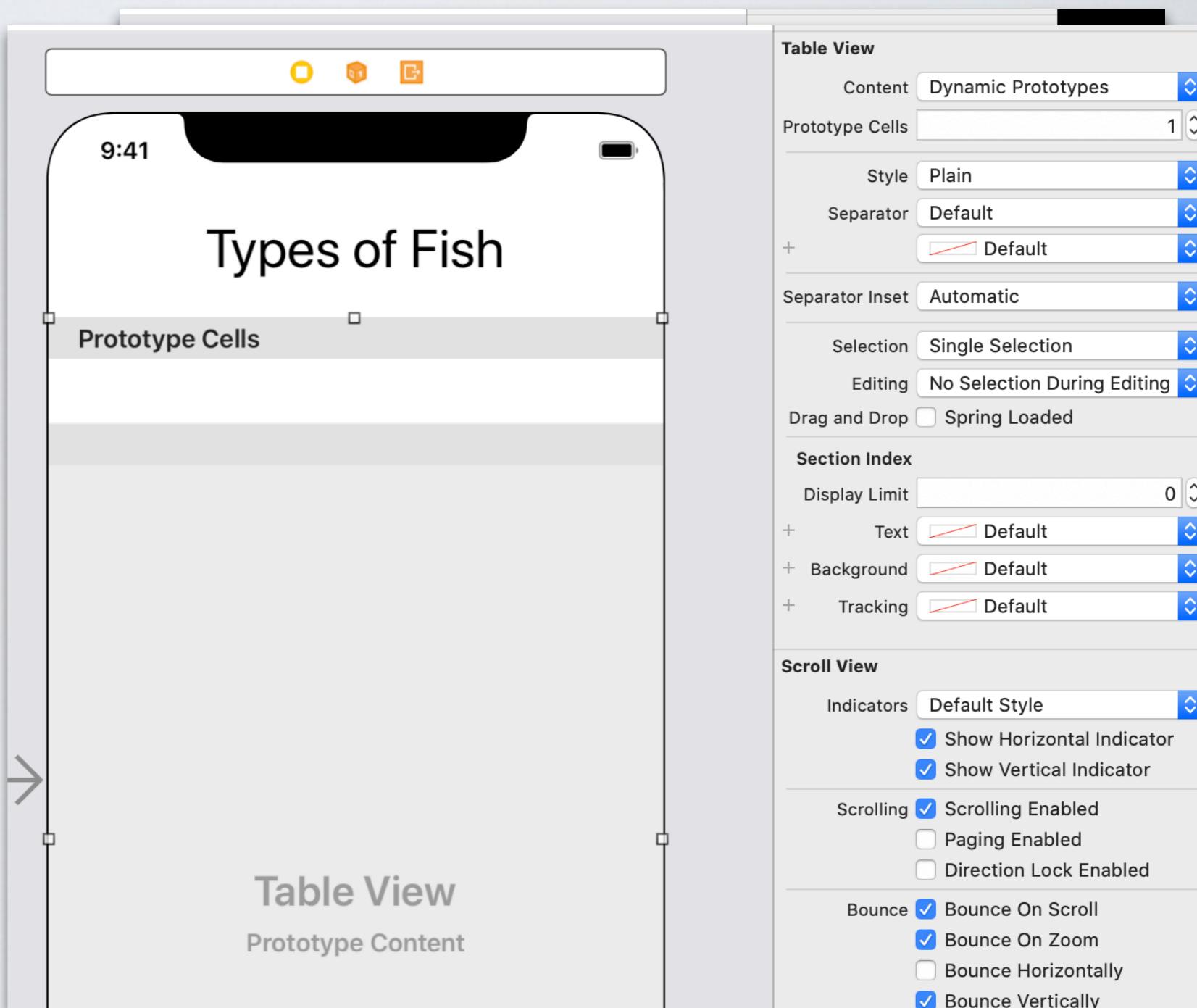
Include UI Tests

Cancel Previous Next

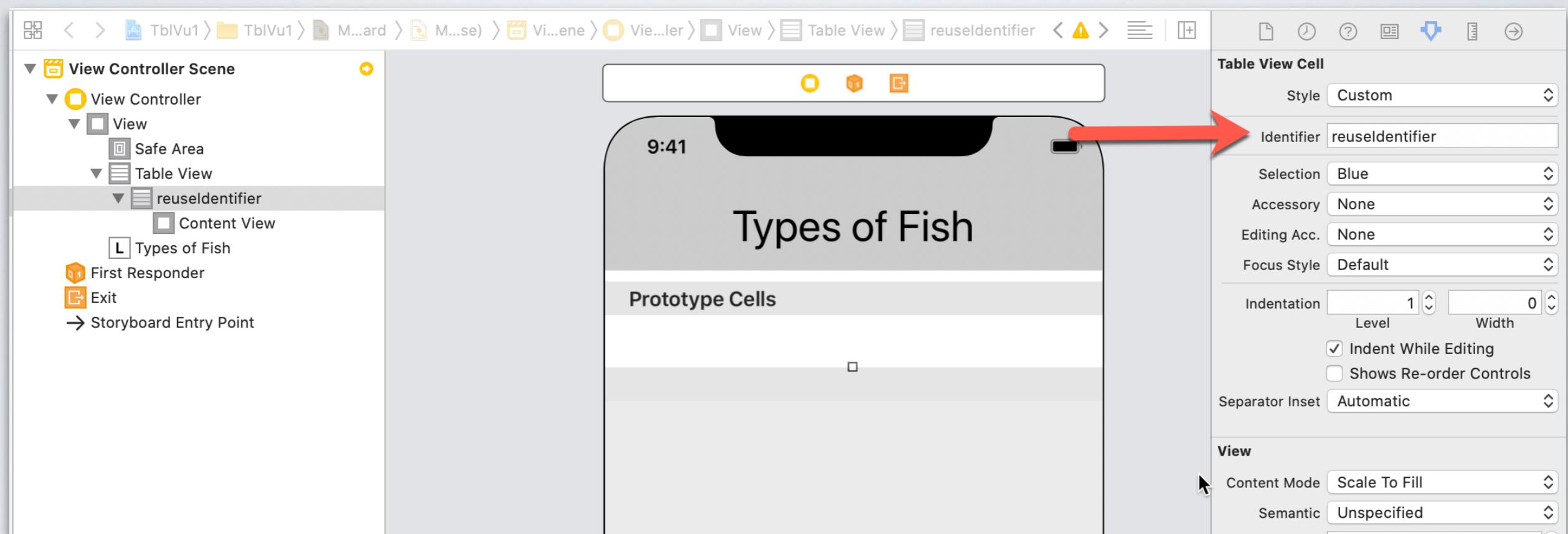


# DRAW TABLE VIEW

- Open the Main.storyboard file.
- Draw a label and change its text to "Types of Fish".
- Now draw a TableView as shown

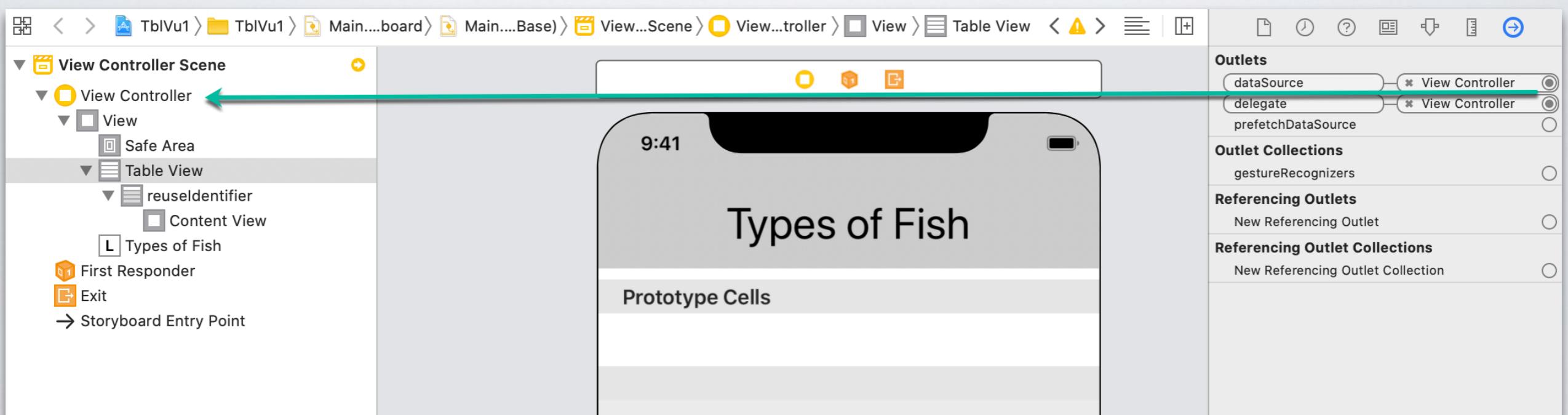


# UPDATE PROTOTYPE CELL WITH REUSE IDENTIFIER



# DATASOURCE & DELEGATE

**Open the Property Inspector of the Table View, then drag its dataSource and delegate outlets and drop those on the View Controller.**



# OPEN THE VIEWCONTROLLER.SWIFT FILE AND INSERT THE FOLLOWING CODE

```
class ViewController: UIViewController, UITableViewDelegate, UITableViewDataSource {

    let arrayFish = ["Bass", "Bluegill", "Crappie", "Muskie", "Walleye"]

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view.
    }

    func numberOfSections(in tableView: UITableView) -> Int {
        return 1
    }

    func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        return arrayFish.count
    }

    func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
        let cell = tableView.dequeueReusableCell(withIdentifier: "reuseIdentifier", for: indexPath)

        cell.textLabel?.text = arrayFish[indexPath.row]

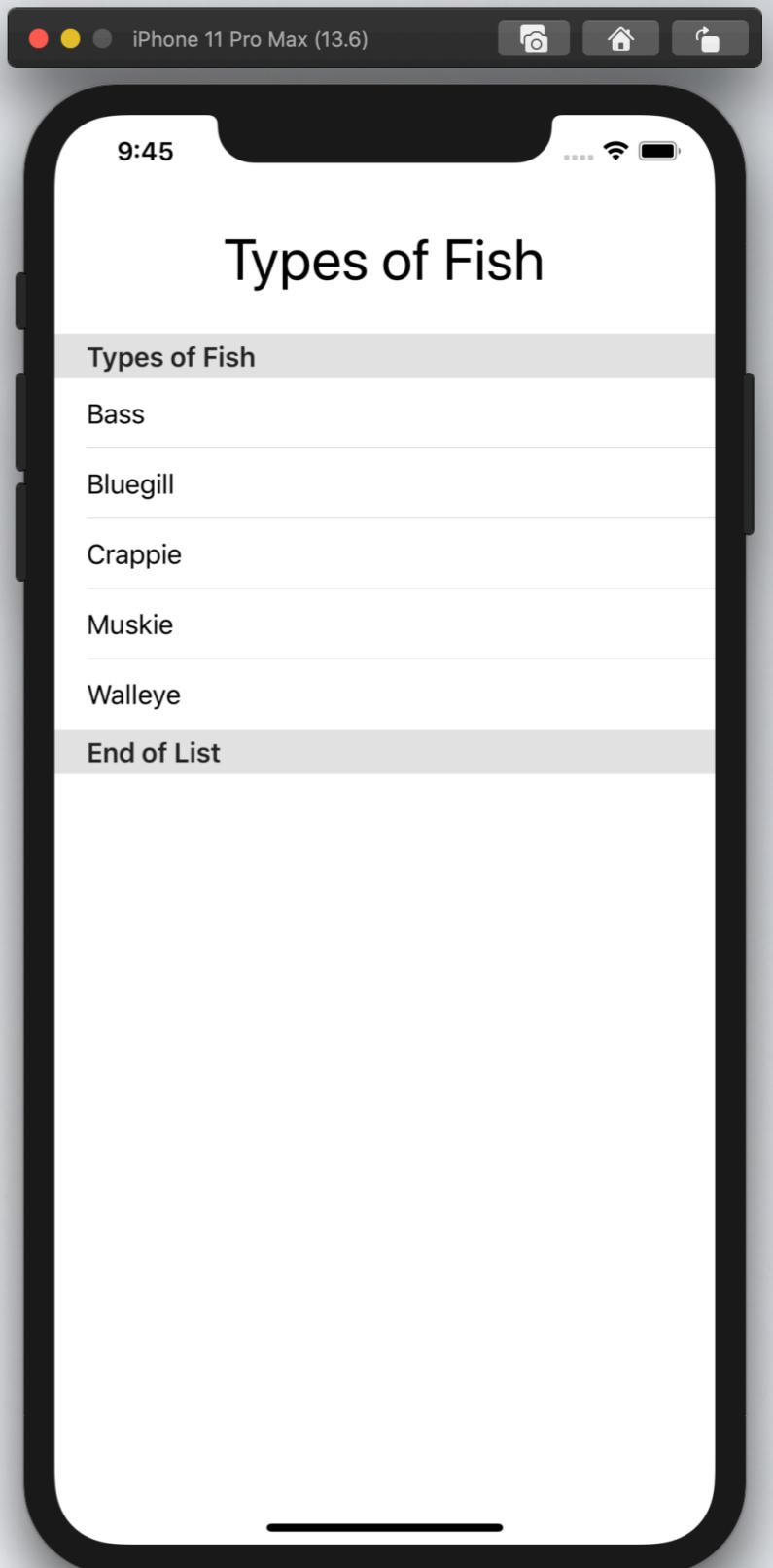
        return cell
    }
}
```

# OPTIONAL HEADER AND FOOTER TITLE OF SECTIONS: YOU MAY IMPLEMENT THE FOLLOWING METHODS FOR THE HEADER AND FOOTER TITLES

```
func tableView(_ tableView: UITableView, titleForHeaderInSection section: Int) -> String? {  
    return "Types of Fish"  
}  
  
func tableView(_ tableView: UITableView, titleForFooterInSection section: Int) -> String? {  
    return "End of List"  
}
```



# YOU ARE DONE!

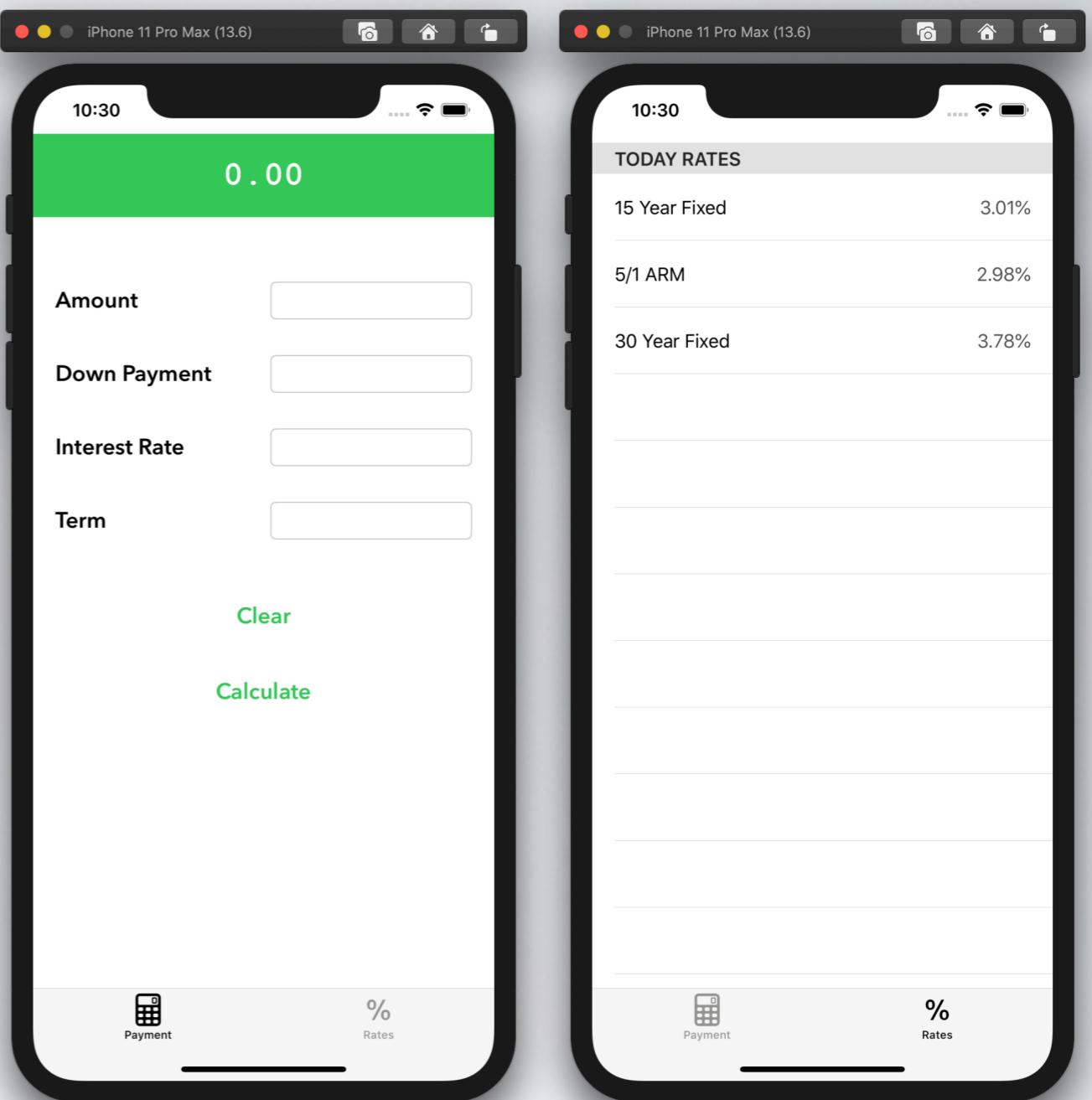


Lab #6 -  
Prepare a static  
table to handle  
the rates data



# ADD A TAB BAR CONTROLLER TO THE AMORTIZATION PAYMENT APP

1. Open Amortized App
2. Open storyboard and drag the tab controller to it
3. Connect tab controller to amortized app
4. Add TableView Controller and connect tab controller
5. Change the images and labels on each of the tab bar items

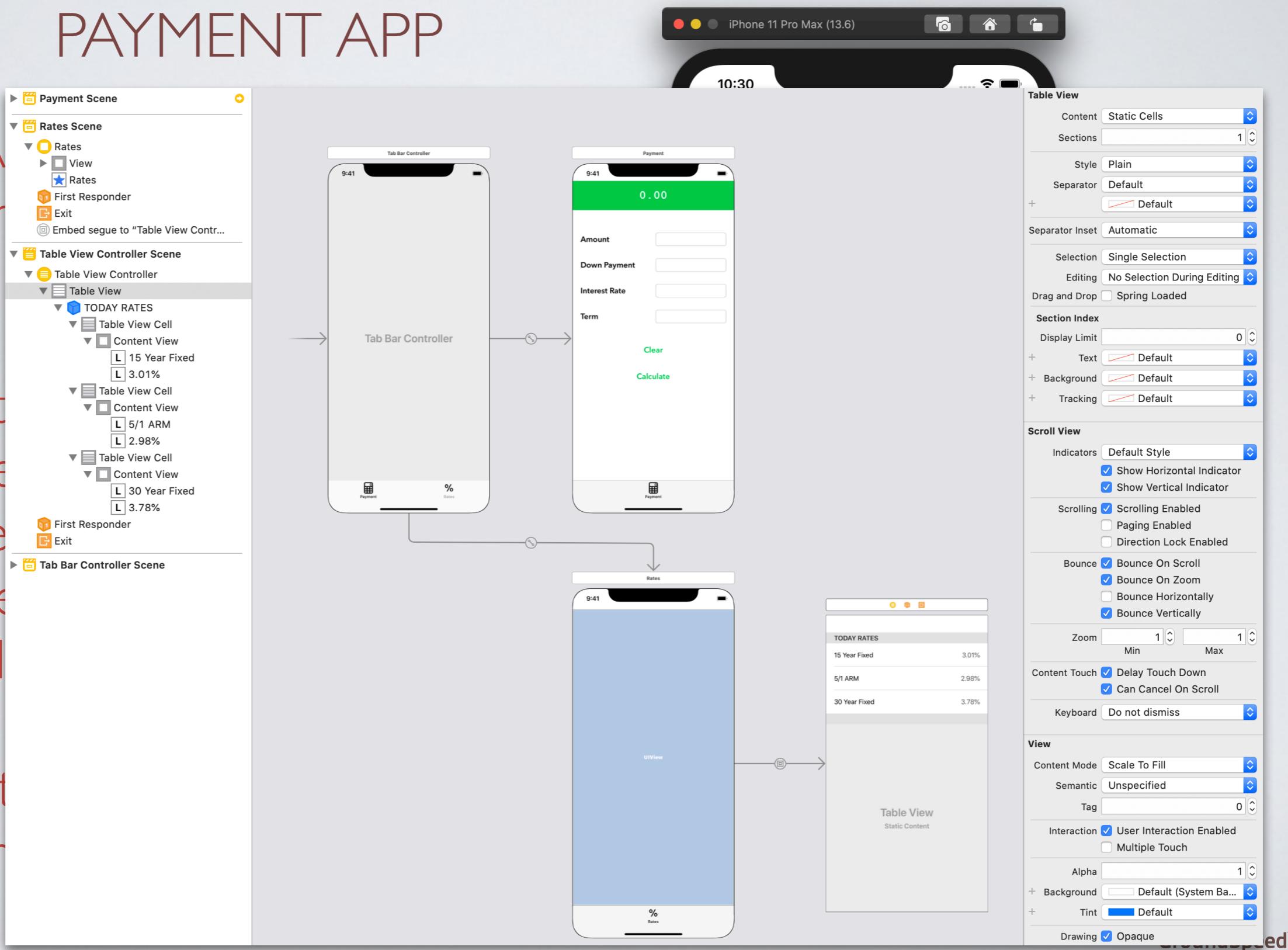


Download Images At:  
<http://sl.gshq.co/percent>  
<http://sl.gshq.co/calculator>



# ADD A TAB BAR CONTROLLER TO THE AMORTIZATION PAYMENT APP

1. Open App
2. Select the Rates Scene
3. Change the Content View cells
4. Set Sections
5. Set Style
6. Add the content view of the selected section
7. Create Interface to each section on the right
8. Add some styling



# Lab #7 -

## GitHub

# Walkthrough



**Don Miller**

**don@GroundSpeedHQ.com**

**@donmiller**

**<http://github.com/donmiller>**

**<http://github.com/groundspeed>**

