360|iDev 2020 ONLINE

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

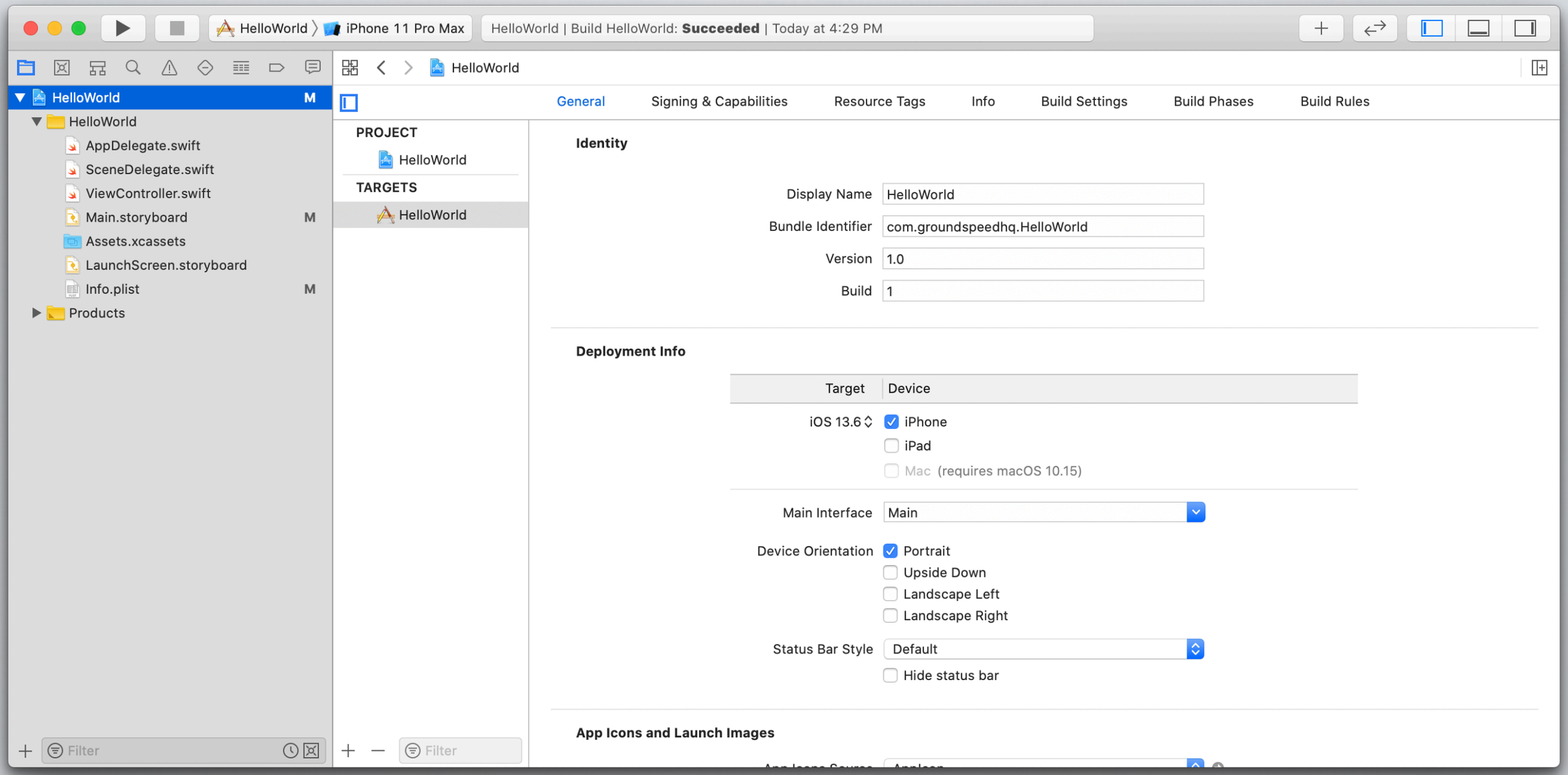
Storyboard Navigation

TERMINOLOGY

- iOS
- iPhone
- iPad
- MacBook
- iMac
- SDK
- Xcode

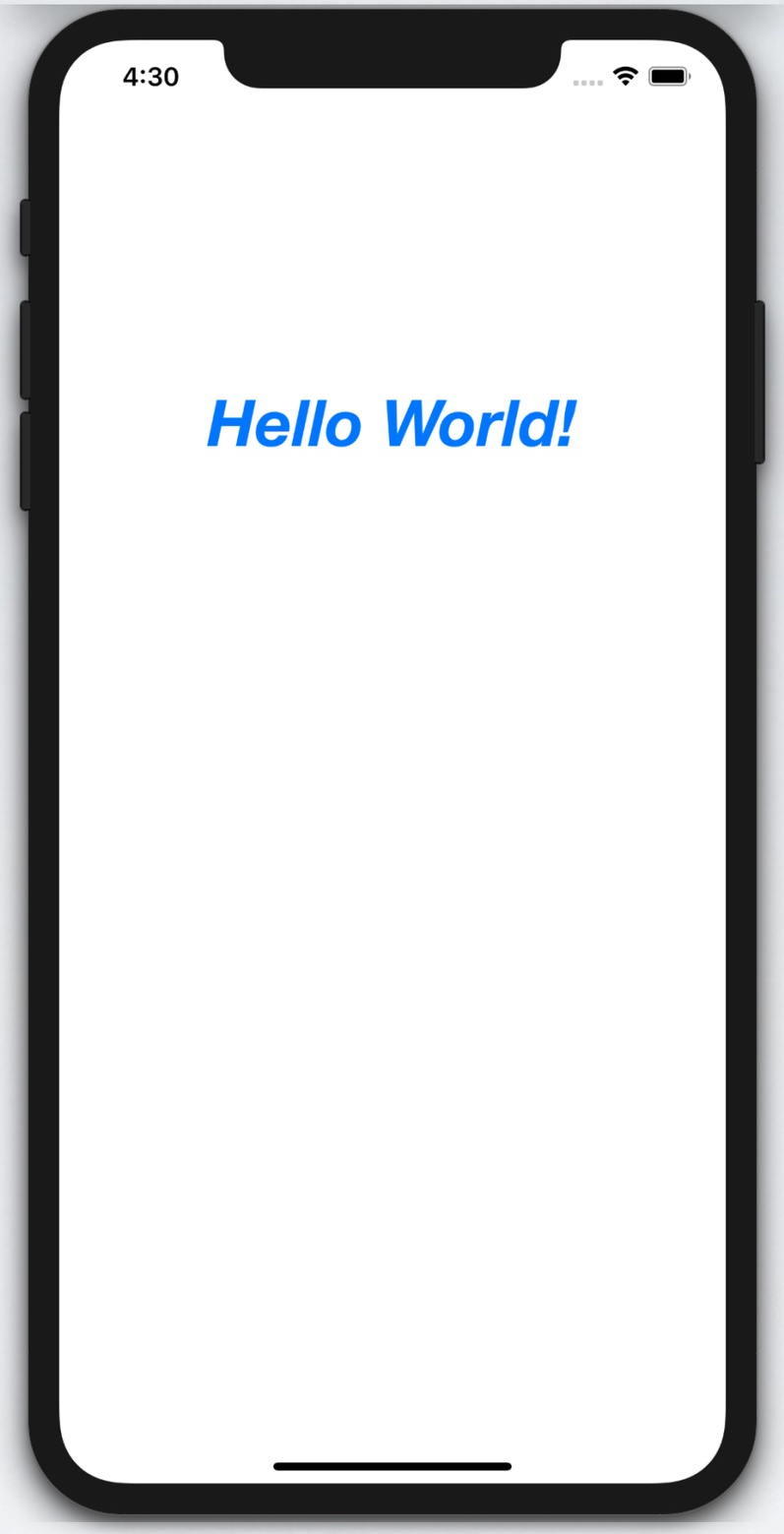
- OSX
- Catalina (10.15)
- Mojave (10.14)
- High Sierra (10.13)
- Sierra (10.12)
- El Capitan,
  Yosemite,
  Mavericks,
  Mountain Lion, etc.

XCODE 11.X & IOS 13.X



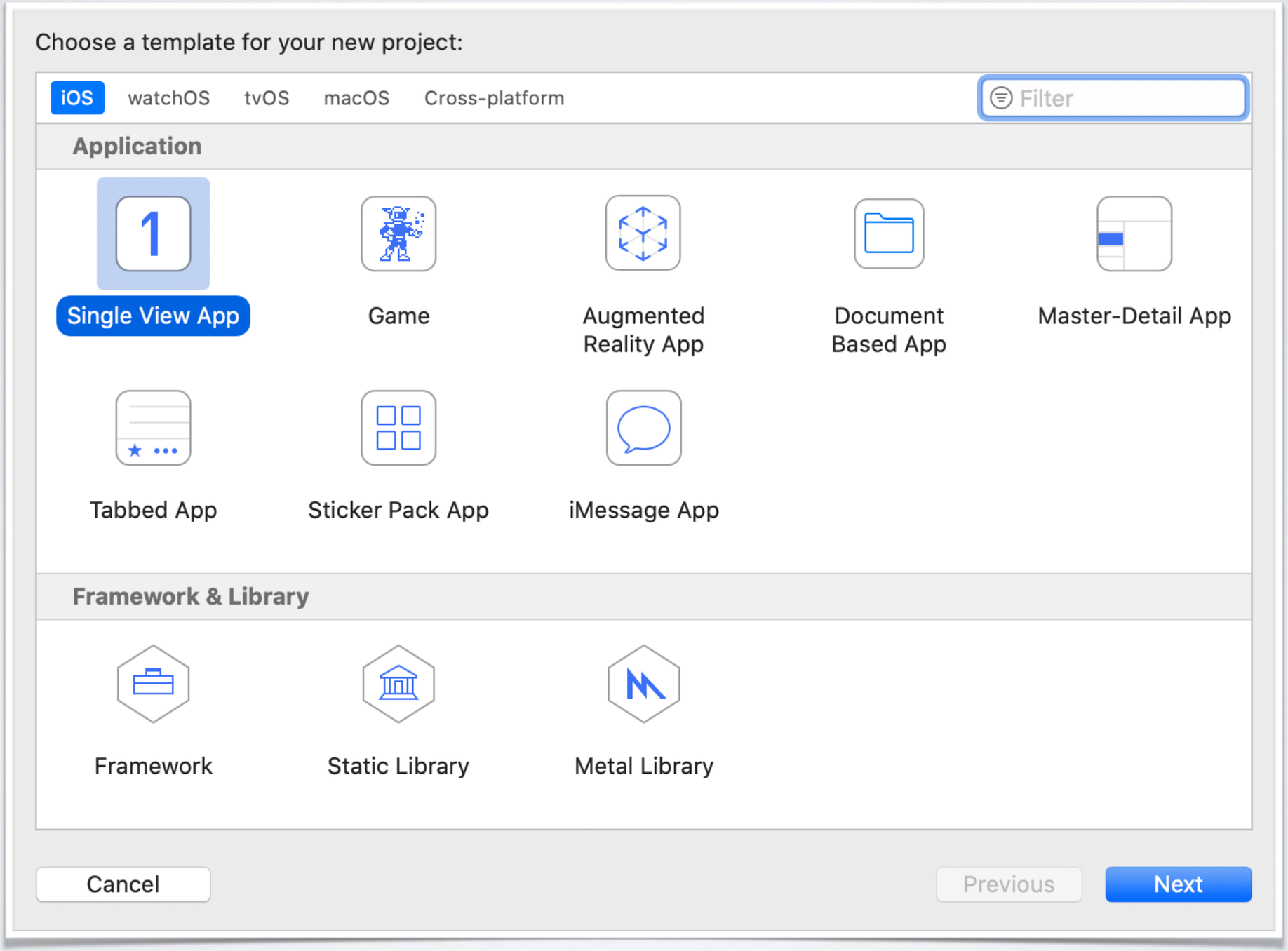
GroundSpeed™
rapid web + mobile software

HELLO WORLD!

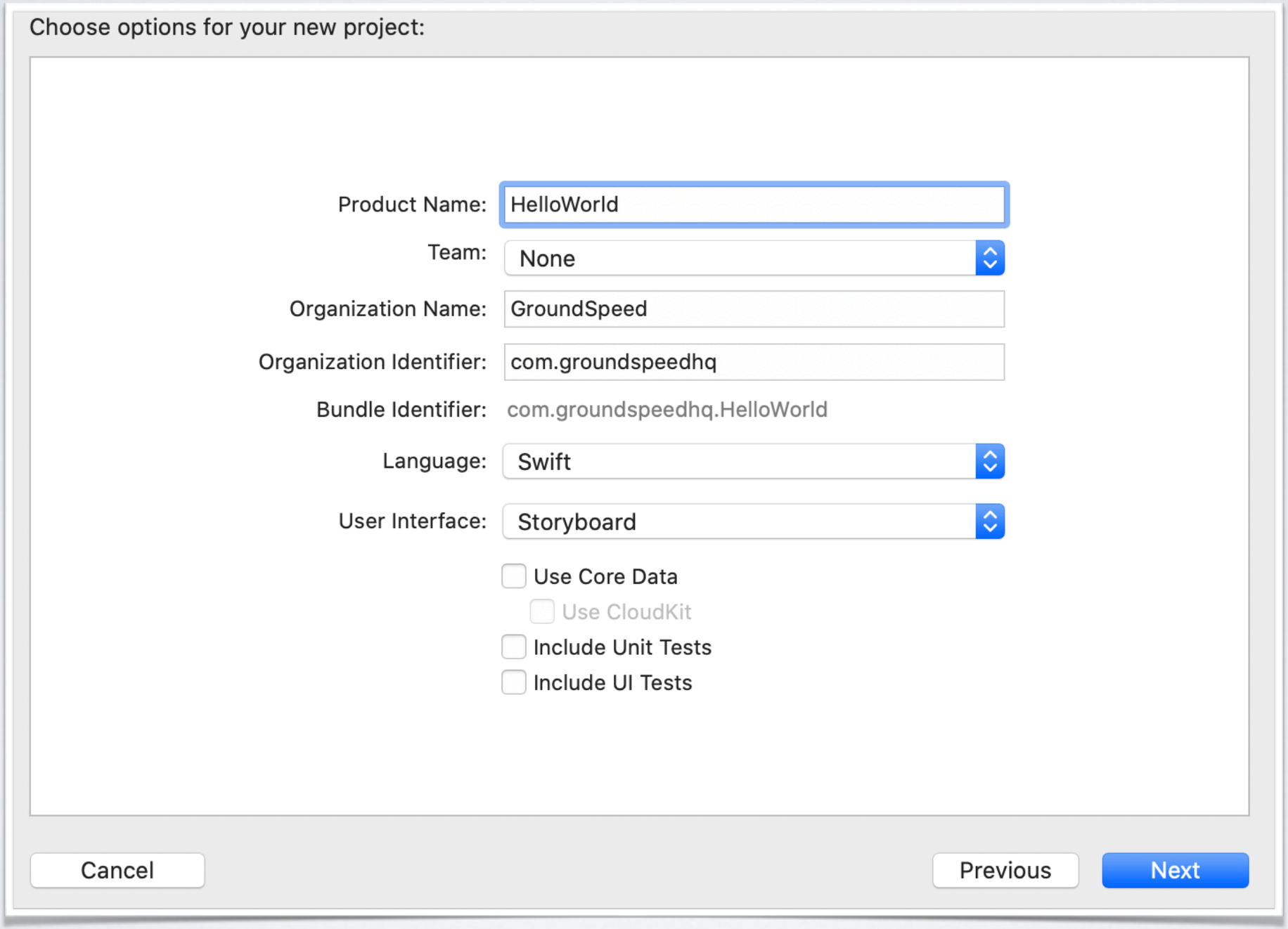


GroundSpeed™
rapid web + mobile software

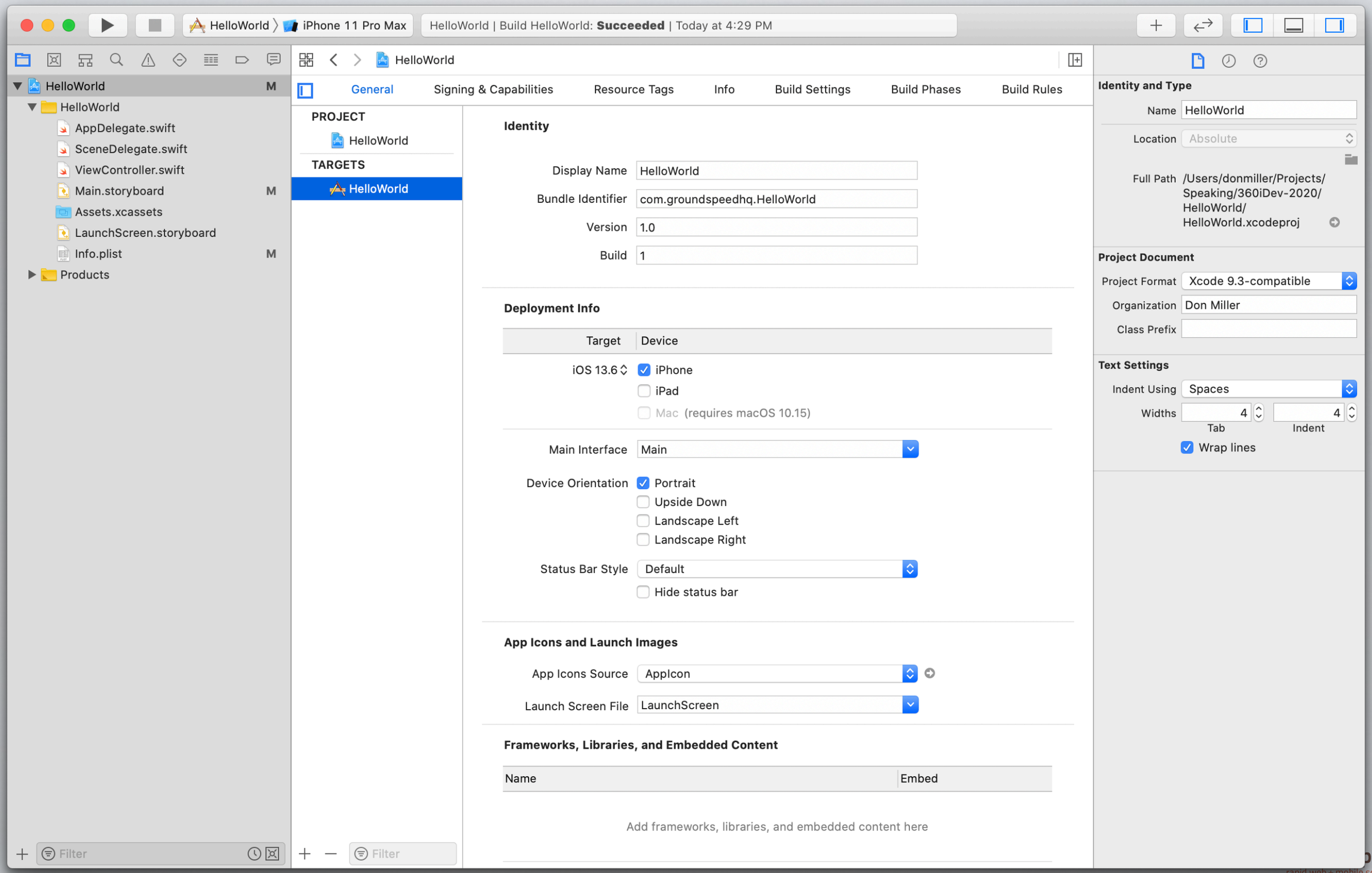
CREATE A NEW SINGLE VIEW APPLICATION



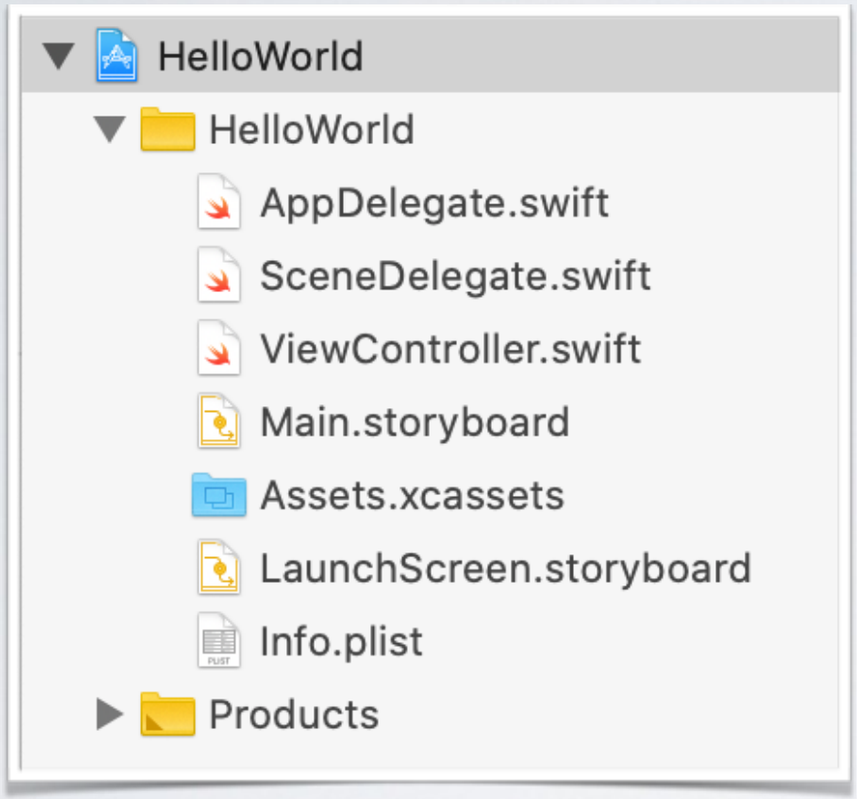
ENTER THE PRODUCT NAME



UPDATE THE GENERAL PROJECT ATTRIBUTES

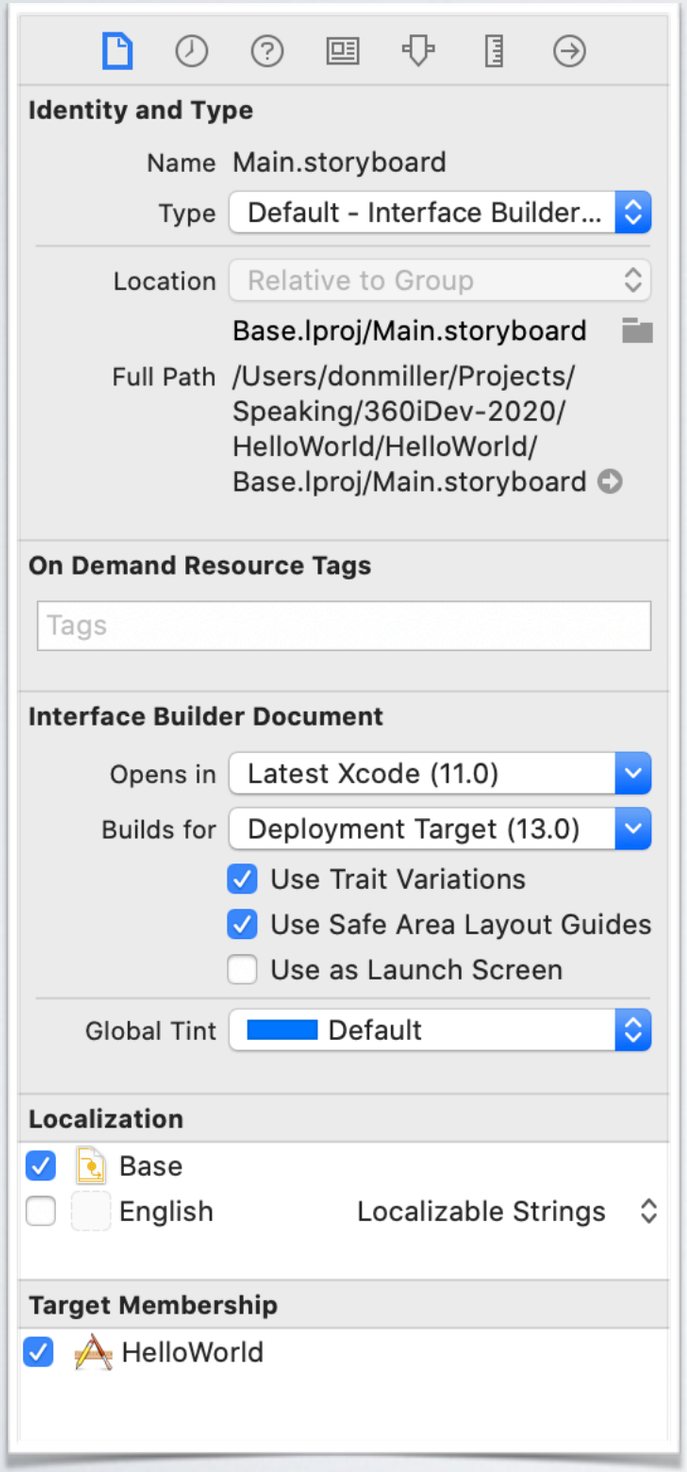


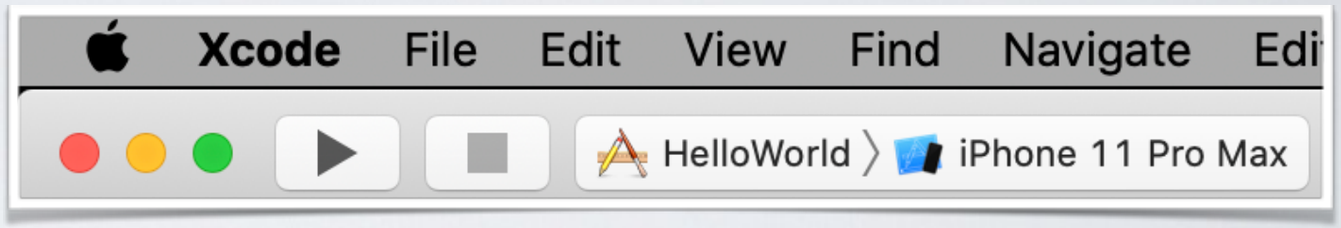
PROJECT STRUCTURE



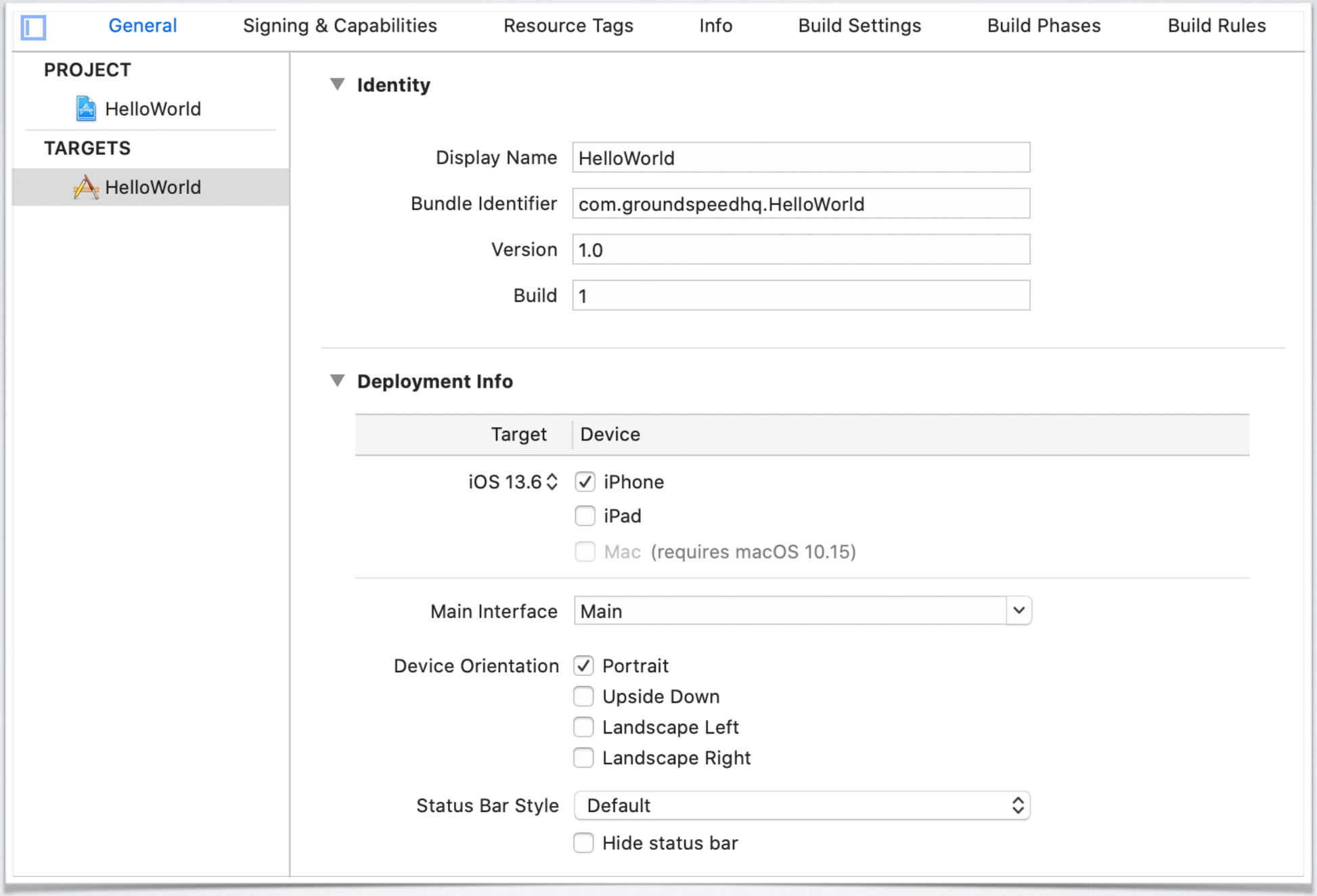
- Folders do not always match physical layout on file system.
- Actually you will not find any sub folder named like Supporting Files in your physical project folder.
- The folder hierarchy in the XCode's navigator is simply a logical grouping of the resources.
- At this stage, we would be interested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

FILE INSPECTOR & SCHEMA SETTINGS

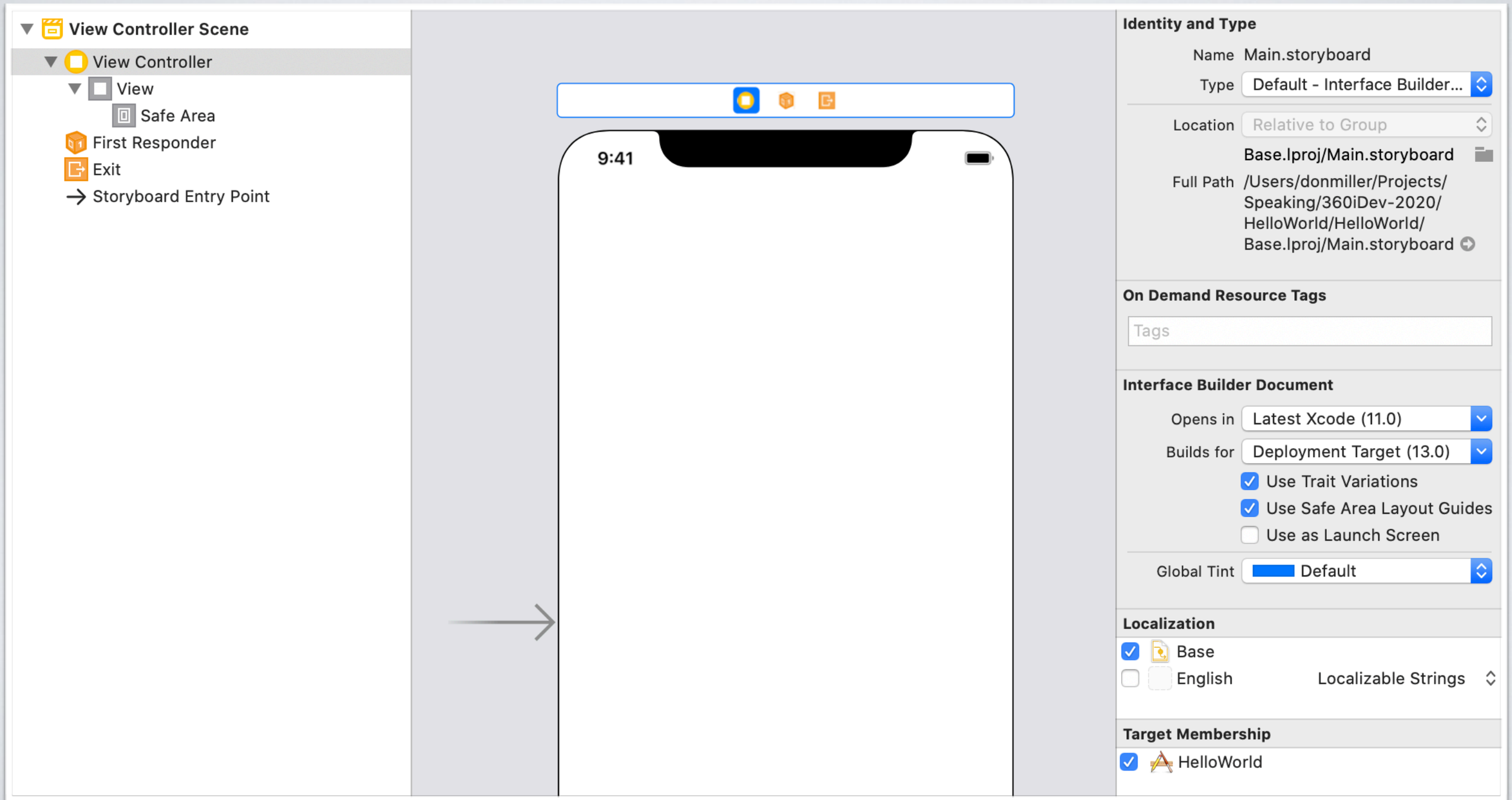




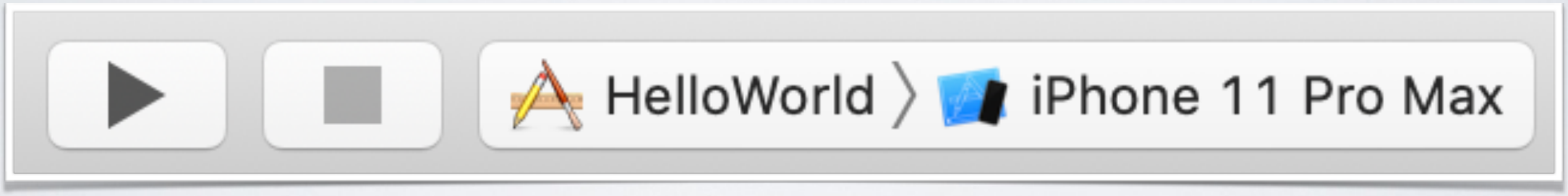
DEVICES & TARGETS



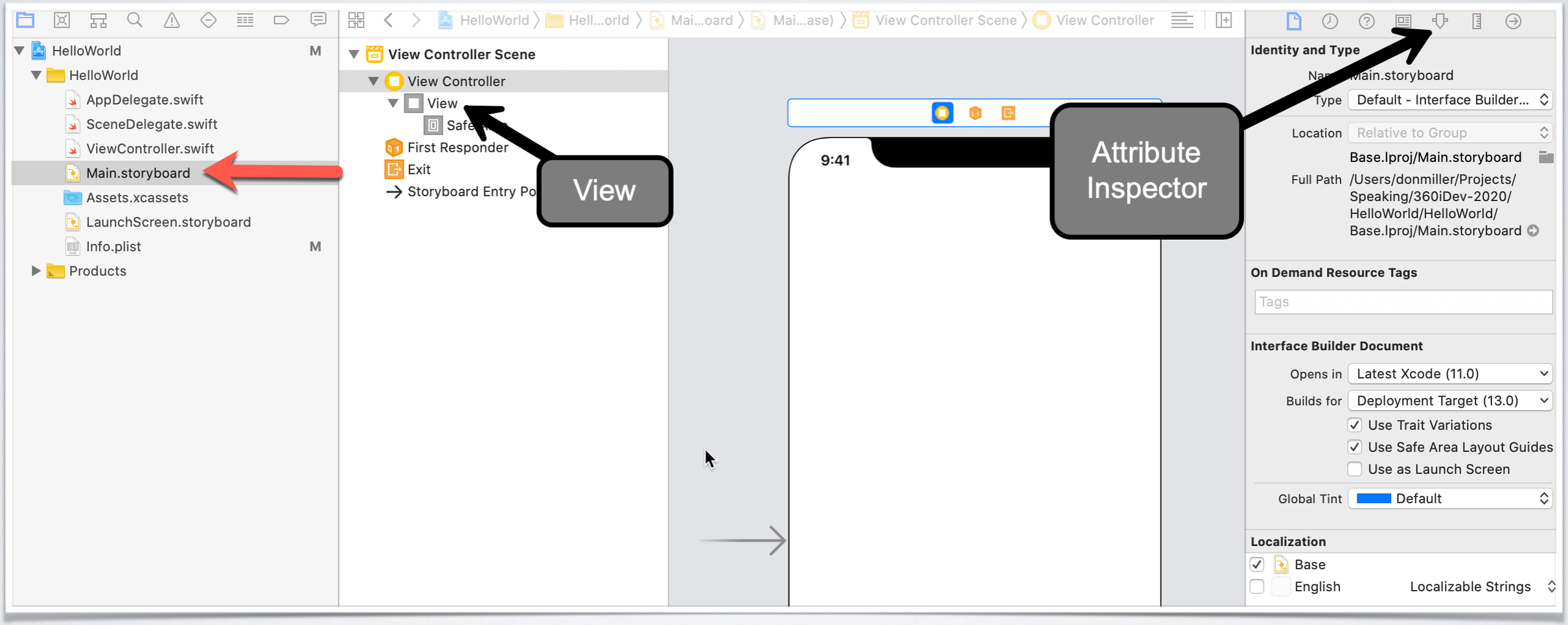
FILE OWNER, VIEW, & FIRST RESPONDER



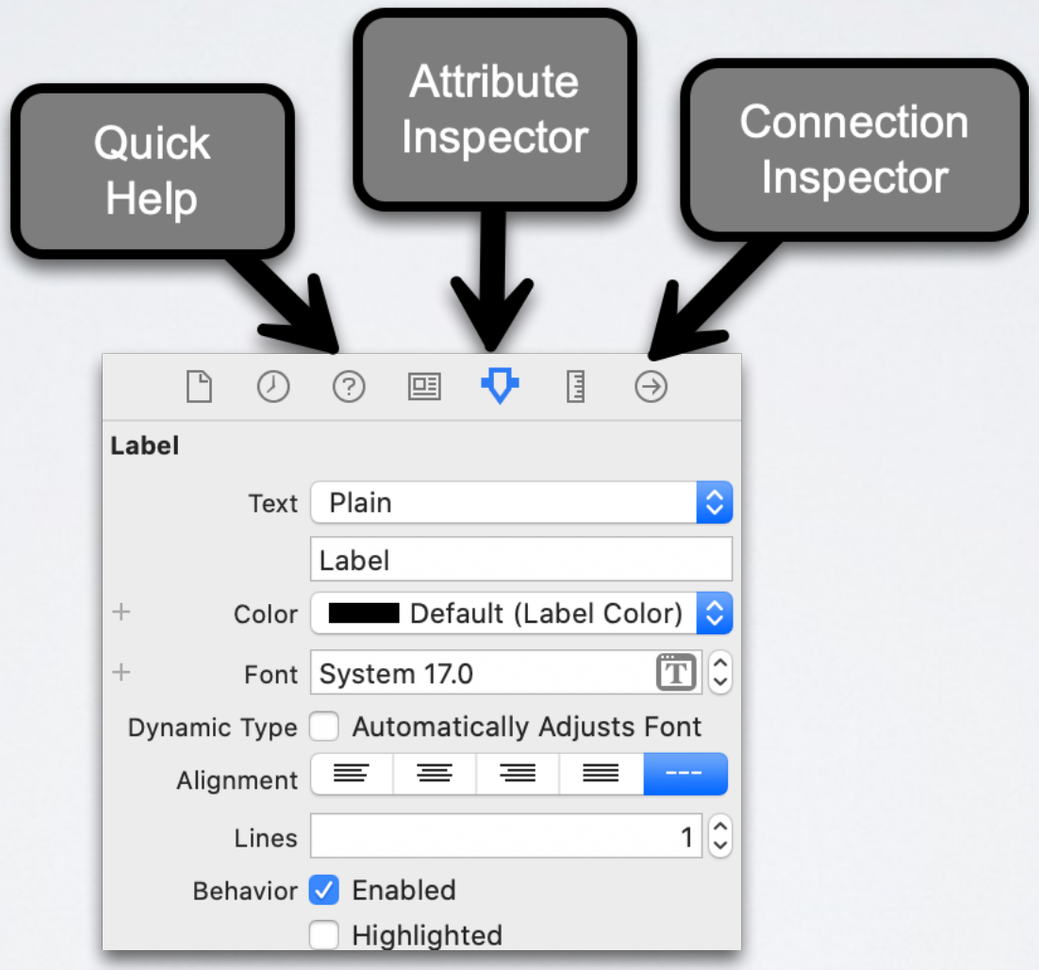
RUN THE APPLICATION BY PRESSING PLAY



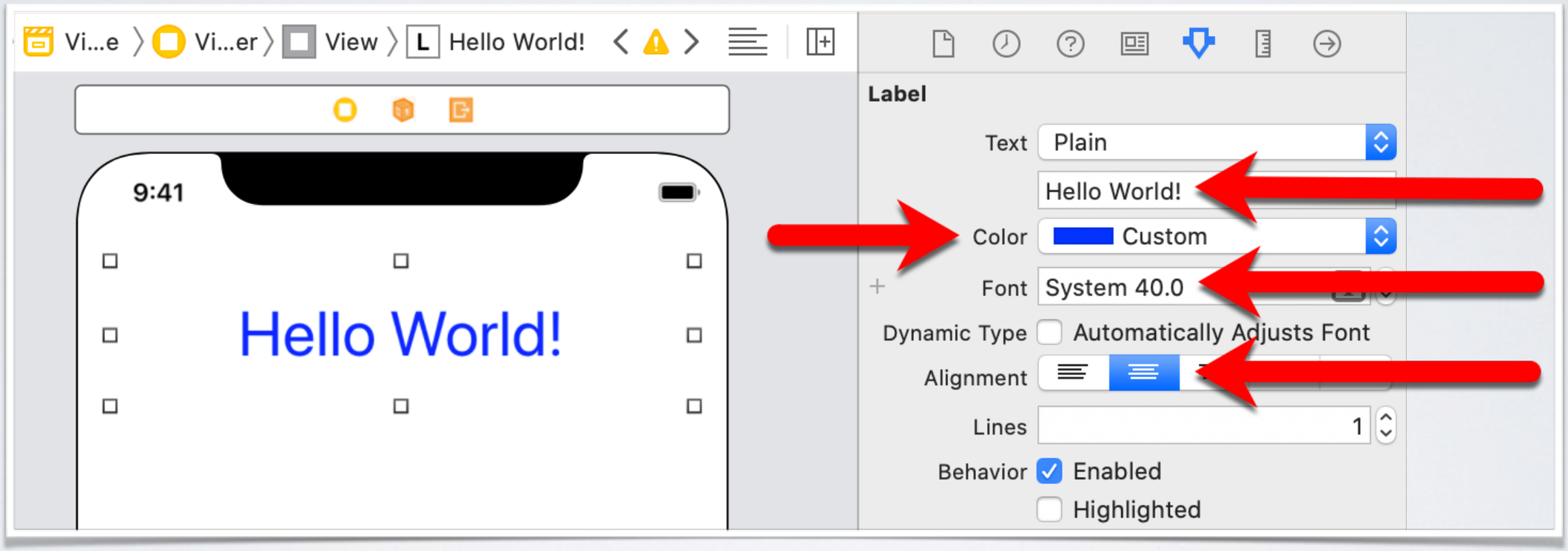
ATTRIBUTE INSPECTOR



INSPECTORS



CREATE "HELLO WORLD!" LABEL



Lab #1

Hello World!

Take 15 Minutes to Build It

Break time afterwards

GroundSpeed™
rapid web + mobile software

SWIFT BASICS

Xcode Playgrounds

Variables & Constants (Mutable and Immutable)

Types

- Strings, Integers, Floats, Booleans

Operators

- Binary and Unary Operators

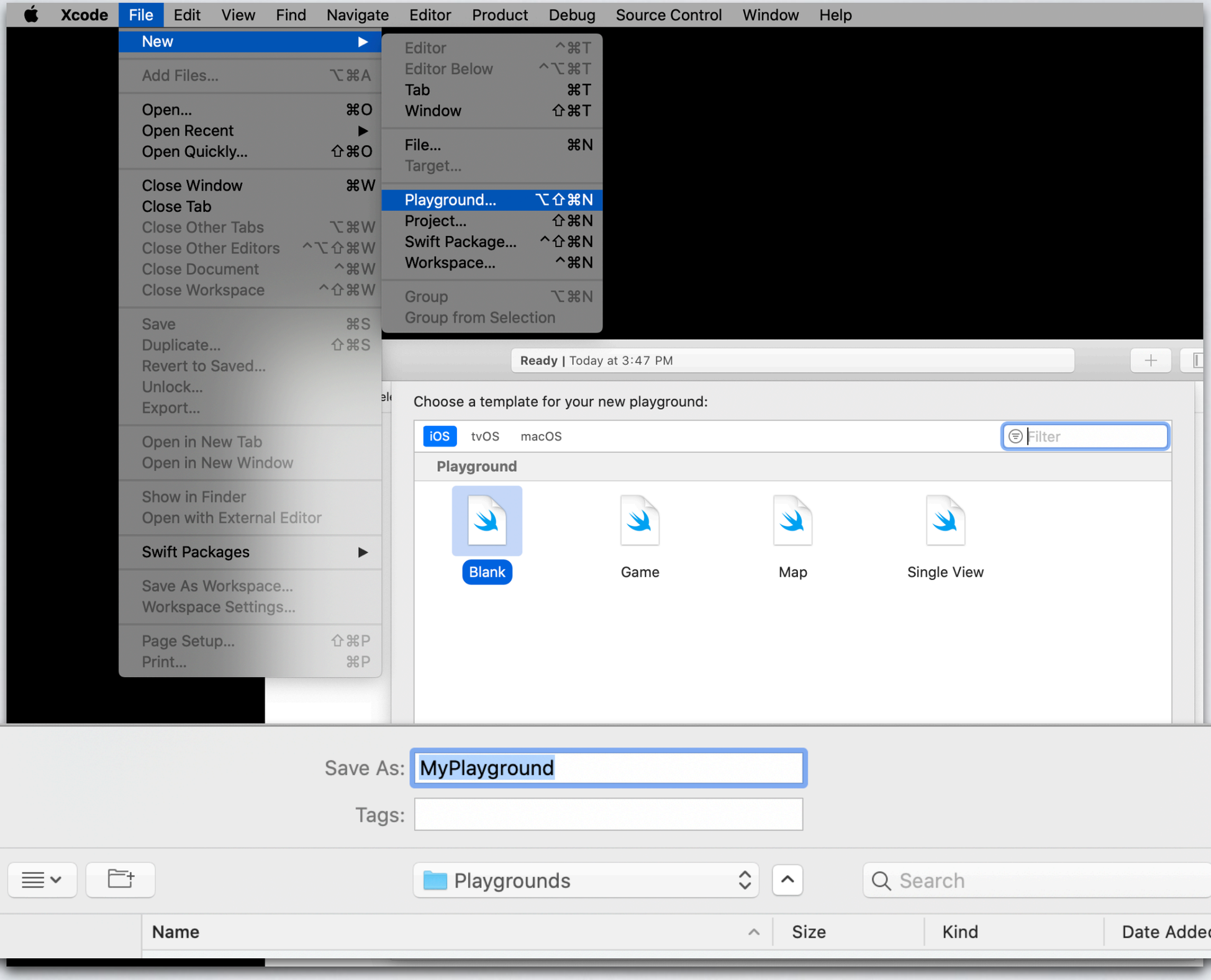
Collection Types

- Arrays and Dictionaries

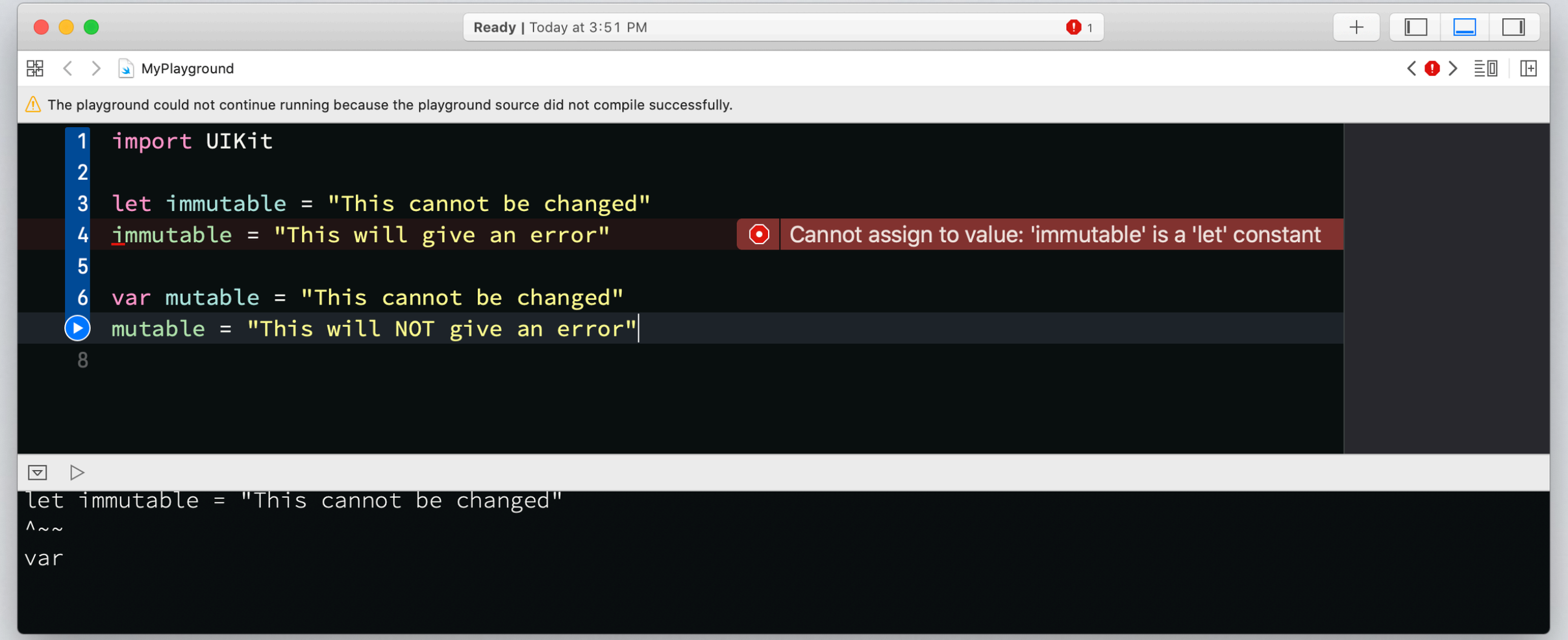
Control Flow

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

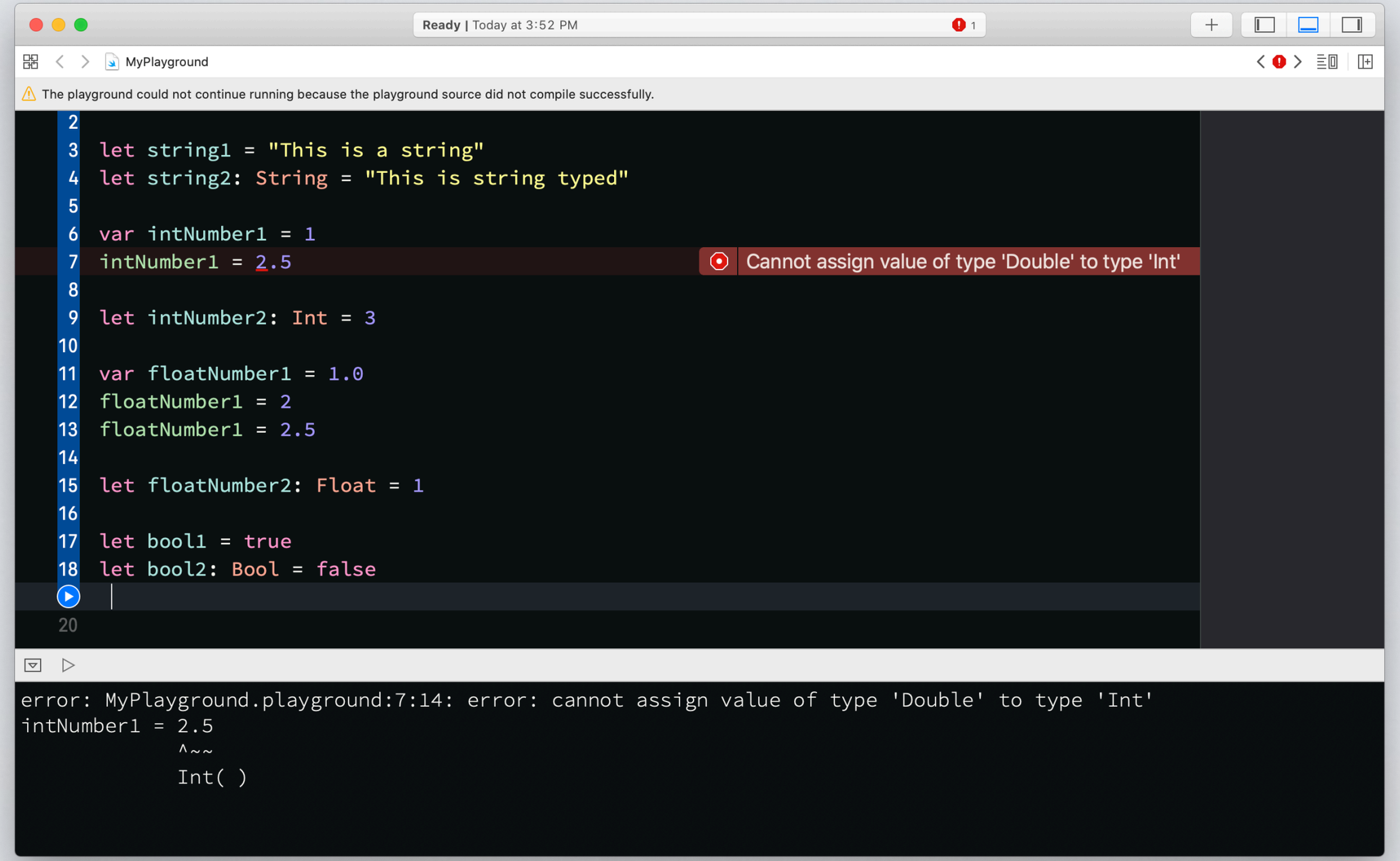
EXAMPLE: HAVE FUN IN THE PLAYGROUND



ABOUT MUTABLE AND IMMUTABLE OBJECTS OR VARIABLES AND CONSTANTS



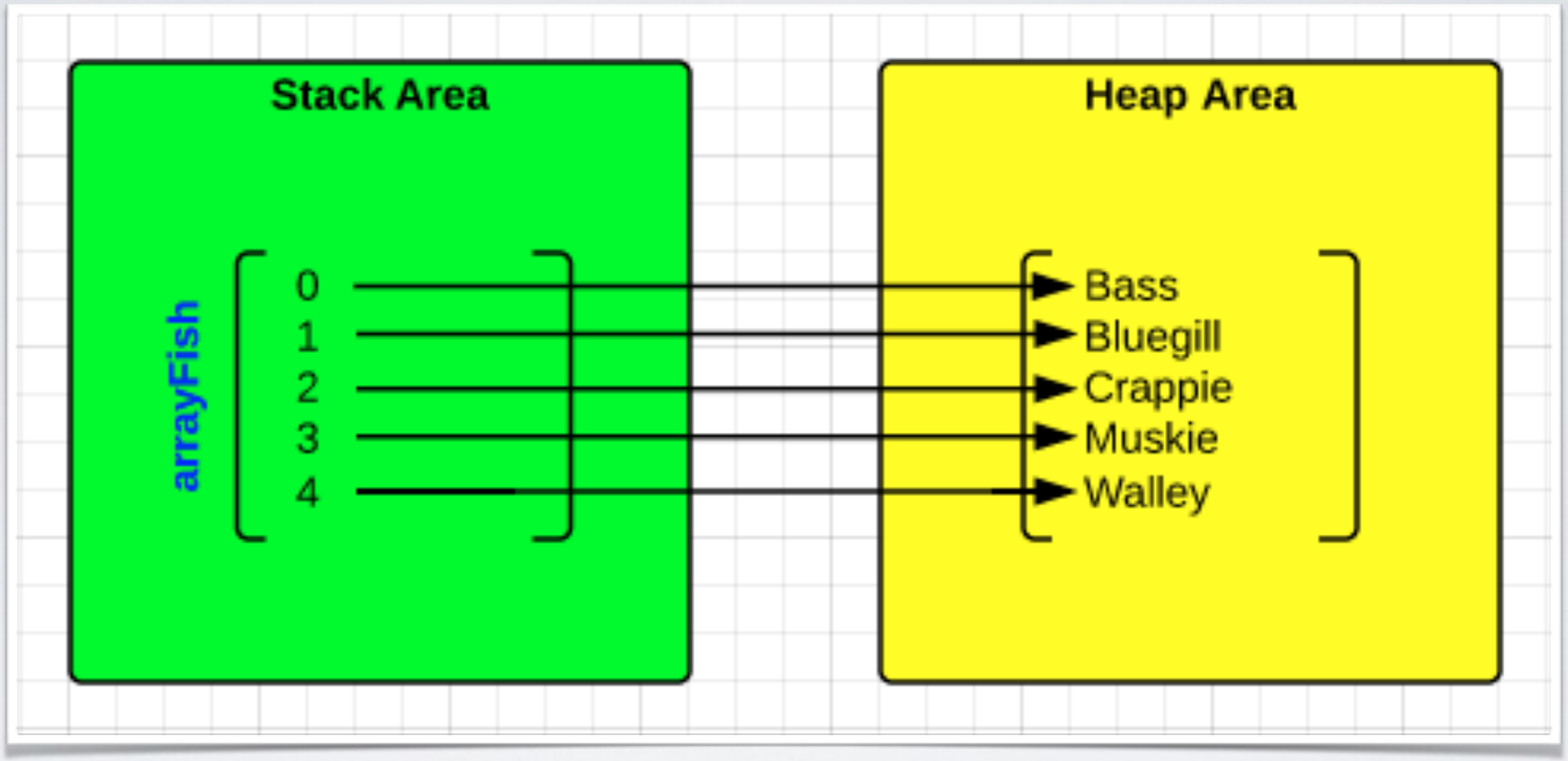
DECLARING TYPES: STRINGS, INTEGERS, FLOATS, AND BOOLEANS



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

```
1  // binary                                  
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                                   
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                                 
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

```swift
// Arrays
let arrayFish = ["Bass", "Bluegill", "Crappie", "Muskie", "Walley"]

arrayFish.count
arrayFish[3]


// Dictionaries
let dictStates: Dictionary<String,String> = ["OH": "Ohio", "MI": "Michigan", "KY":
    "Kentucky", "IN": "Indiana", "IL": "Illinois", "PA": "Pennsylvania", "NY": "New
    York"]
dictStates.count
dictStates["OH"]!

// For Loop through dictionary
for (stateCode, stateName) in dictStates {
    print("\(stateCode): \(stateName)")
}

for stateCode in dictStates.keys {
    print("\(stateCode)")
}

for stateName in dictStates.values {
    print("\(stateName)")
}

// Array of stateCodes
let stateCodes = [String](dictStates.keys)
```

```
["Bass", "Blue...
5
"Muskie"
["OH": "Ohio"....
7
"Ohio"
(7 times)
(7 times)
(7 times)
["OH", "NY", "...
```

CONTROL FLOW - LOOPING

```swift
// For-In
for index in 1...5 {
    print("\(index) times 5 is \(index * 5)")
}

let arrayFish = ["Bass", "Bluegill", "Crappie", "Muskie", "Walleye"]
for fish in arrayFish {
    print("Fish array: \(fish)")
}

// For-In with <
for index in 0 ..< 5 {
    print("For Index is \(index)")
}

// While
var index = 0
var max = 5
while index < max {
    print("While Index is \(index += 1)")
}

// Repeat-While
index = 0
max = 5
repeat {
    print("Repeat-While Index is \(index += 1)")
} 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 ()
While Index is ()
While Index is ()
While Index is ()
While Index is ()
Repeat-While Index is ()
Repeat-While Index is ()
Repeat-While Index is ()
Repeat-While Index is ()
Repeat-While Index is ()
```

CONTROL FLOW - DECISION

```swift
// IF
var temperatureInFahrenheit = 90
if temperatureInFahrenheit <= 32 {
    print("It's very cold. Consider wearing a scarf.")
} else if temperatureInFahrenheit >= 86 {
    print("It's really warm. Don't forget to wear sunscreen.")
} else {
    print("It's not that cold. Wear a t-shirt.")
}

// SWITCH
let someCharacter: Character = "a"
switch someCharacter {
case "a", "e", "i", "o", "u":
    print("\(someCharacter) is a vowel")
case "b", "c", "d", "f", "g", "h", "j", "k", "l", "m",
"n", "p", "q", "r", "s", "t", "v", "w", "x", "y", "z":
    print("\(someCharacter) is a consonant")
default:
    print("\(someCharacter) is not a vowel or a consonant")
}
```

```
90

"It's really warm. Don't forget to wear s..

"a"

"a is a vowel\n"
```

CLASS ILLUSTRATIONS

```swift
class Person {
    var ssn: String!
    var name: String!
    var city: String!
    var state: String!
}

class Employee: Person {
    var empId: String!
    var department: String!
    var healthInsurance: String!

    func printChecks() {
        print("Printing Checks")
    }
}

class SalariedEmployee: Employee {
    var salary: String!
}

class HourlyEmployee: Employee {
    var hourlyRate: String!
}
```

```swift
let hourlyEmployee = HourlyEmployee()
hourlyEmployee.name = "Don Miller"
hourlyEmployee.empId = "1234"
hourlyEmployee.hourlyRate = "9.99"

var output = "Name: \(hourlyEmployee.name!) \r"
output += "Employee ID is \(hourlyEmployee.empId!) \r"
output += "Hourly wage is \(hourlyEmployee.hourlyRate!) \r"

print("\(output)")
hourlyEmployee.printChecks()
```

Console Output

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

CLASSES VS STRUCTS

```swift
struct Person {
    var ssn: String!
    var name: String!
    var city: String!
    var state: String!
}
```

```swift
class Person {
    var ssn: String!
    var name: String!
    var city: String!
    var state: String!
}
```

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

Lab #2

Swift Playground!

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

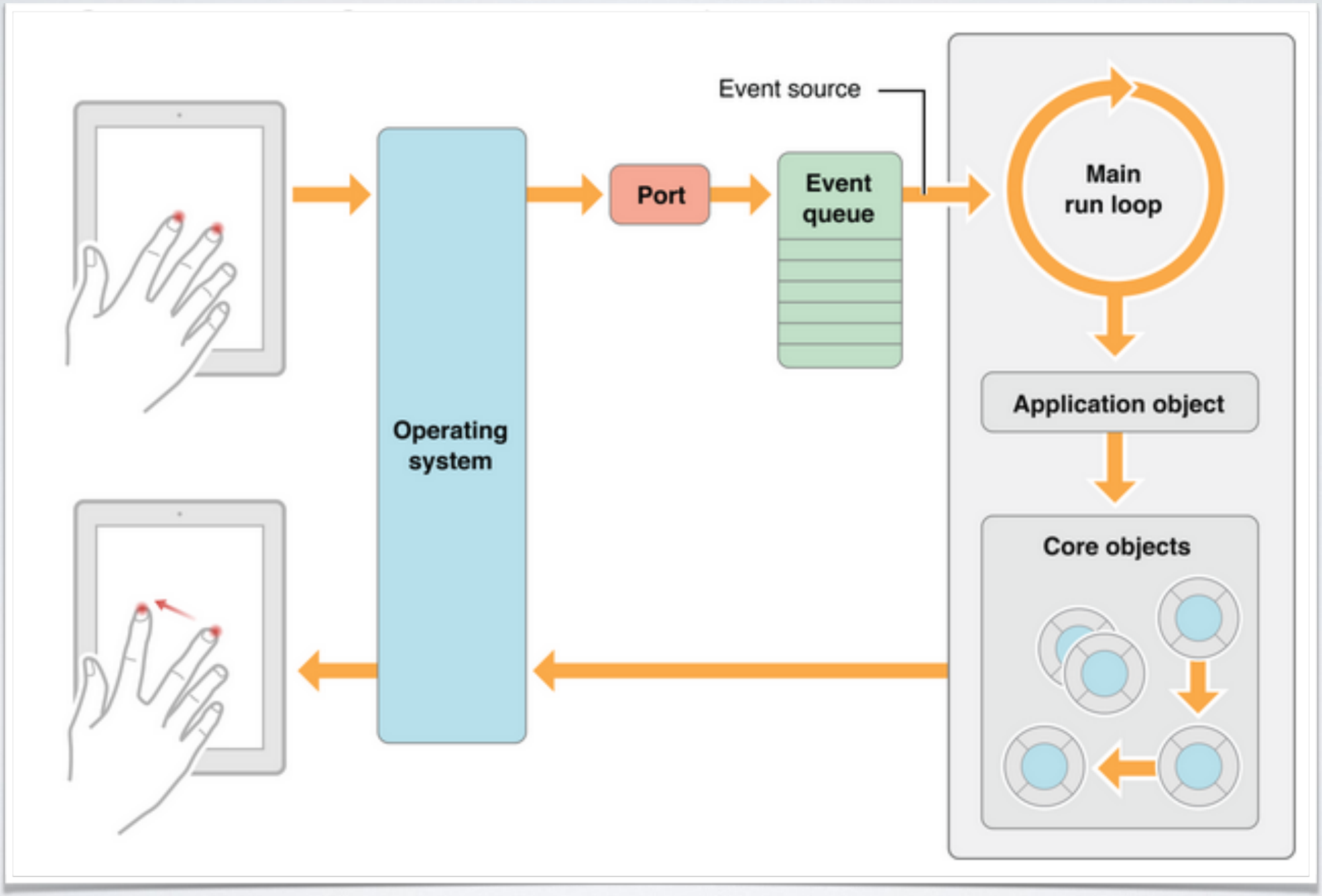
STARTING AGAIN at 11:15a MT

GroundSpeed™
rapid web + mobile software

MVC, Storyboards, and Events

GroundSpeed™
rapid web + mobile software

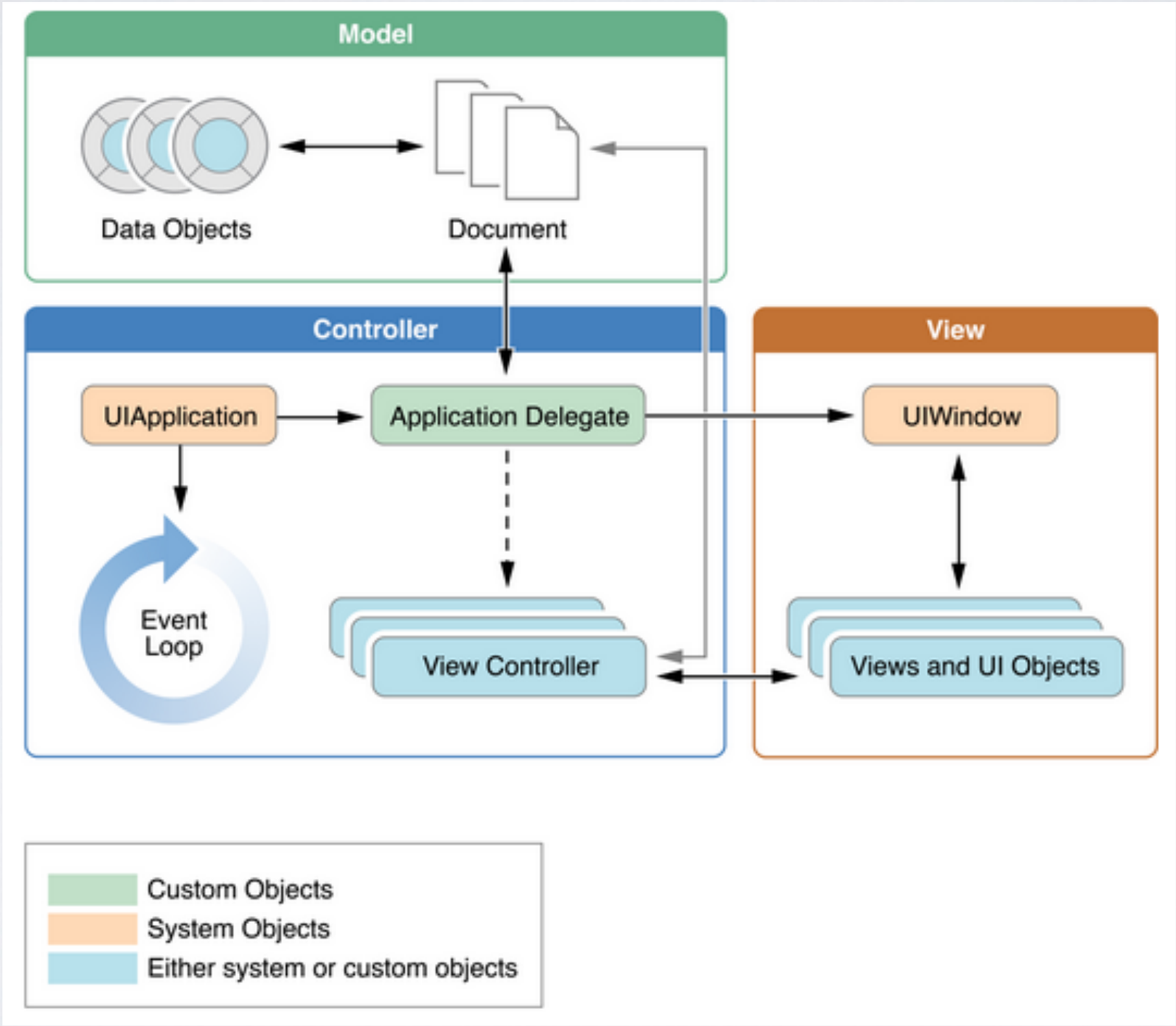
IOS APPLICATION RUNTIME CYCLE



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 responder object	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 *Location and Maps Programming Guide*.
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 *Drawing and Printing Guide for iOS*.

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

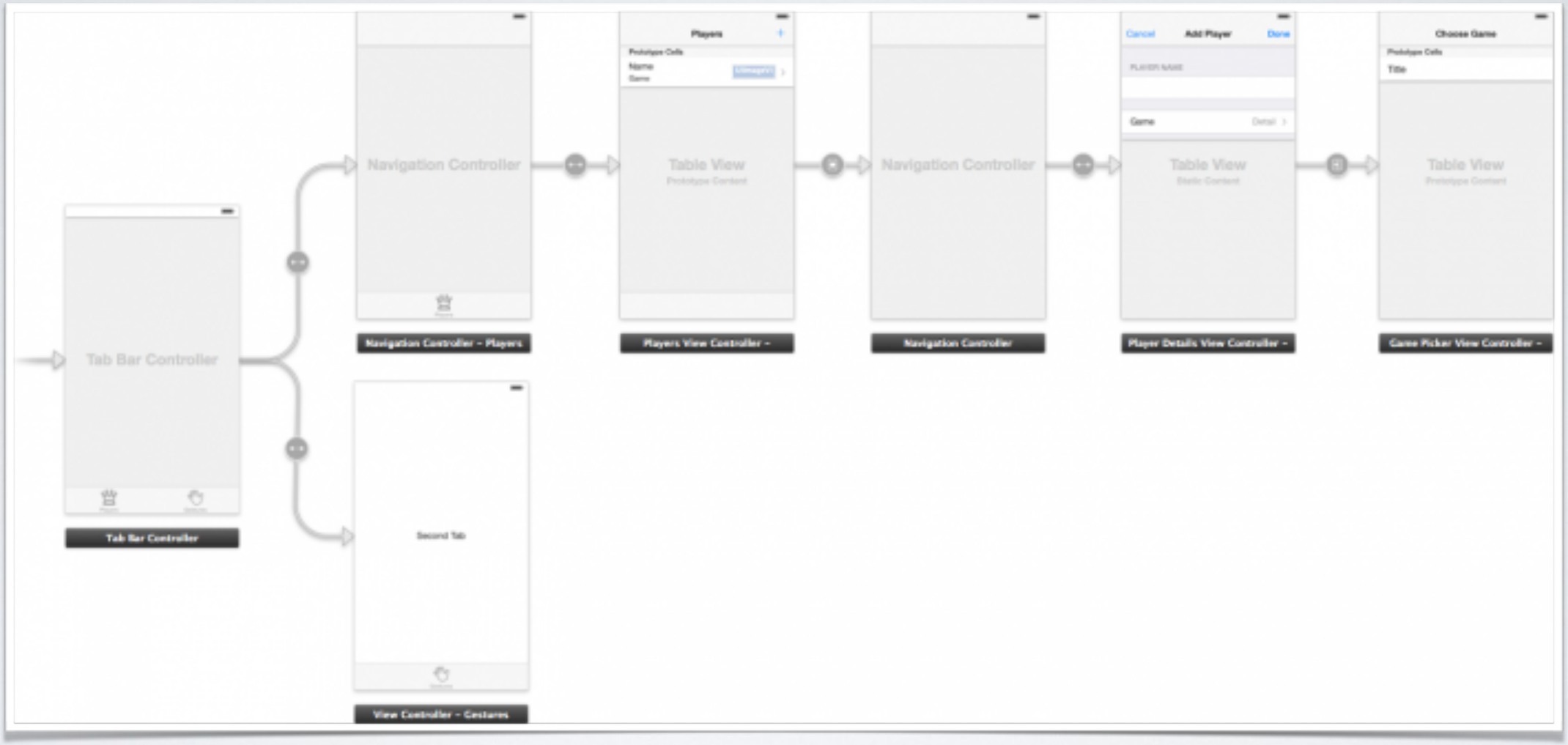
Cons of using Storyboards:

- 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).

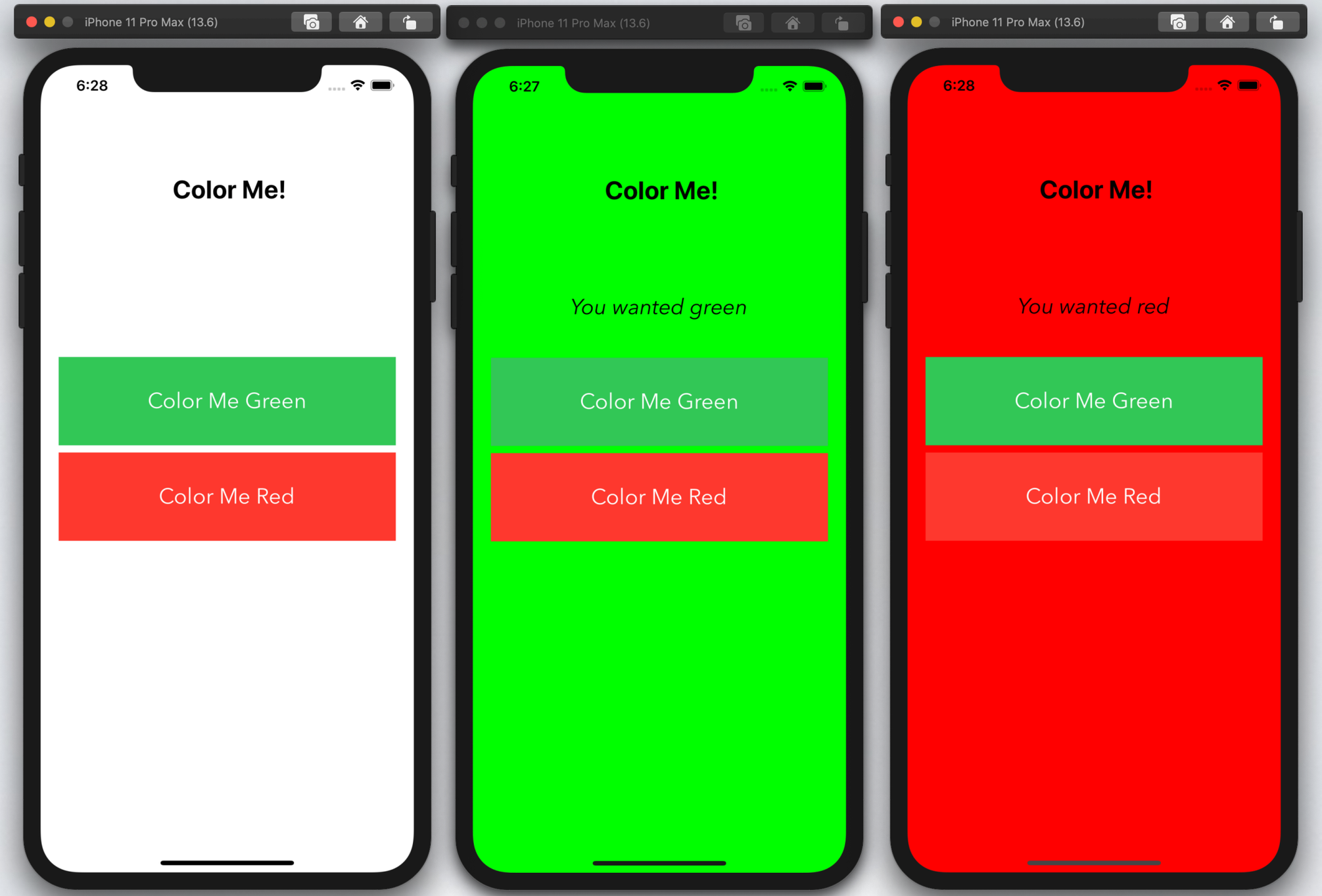
- If you need to do things storyboard doesn't offer, it's not quite easy to get storyboard mixed with programmatically created views)

- 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.

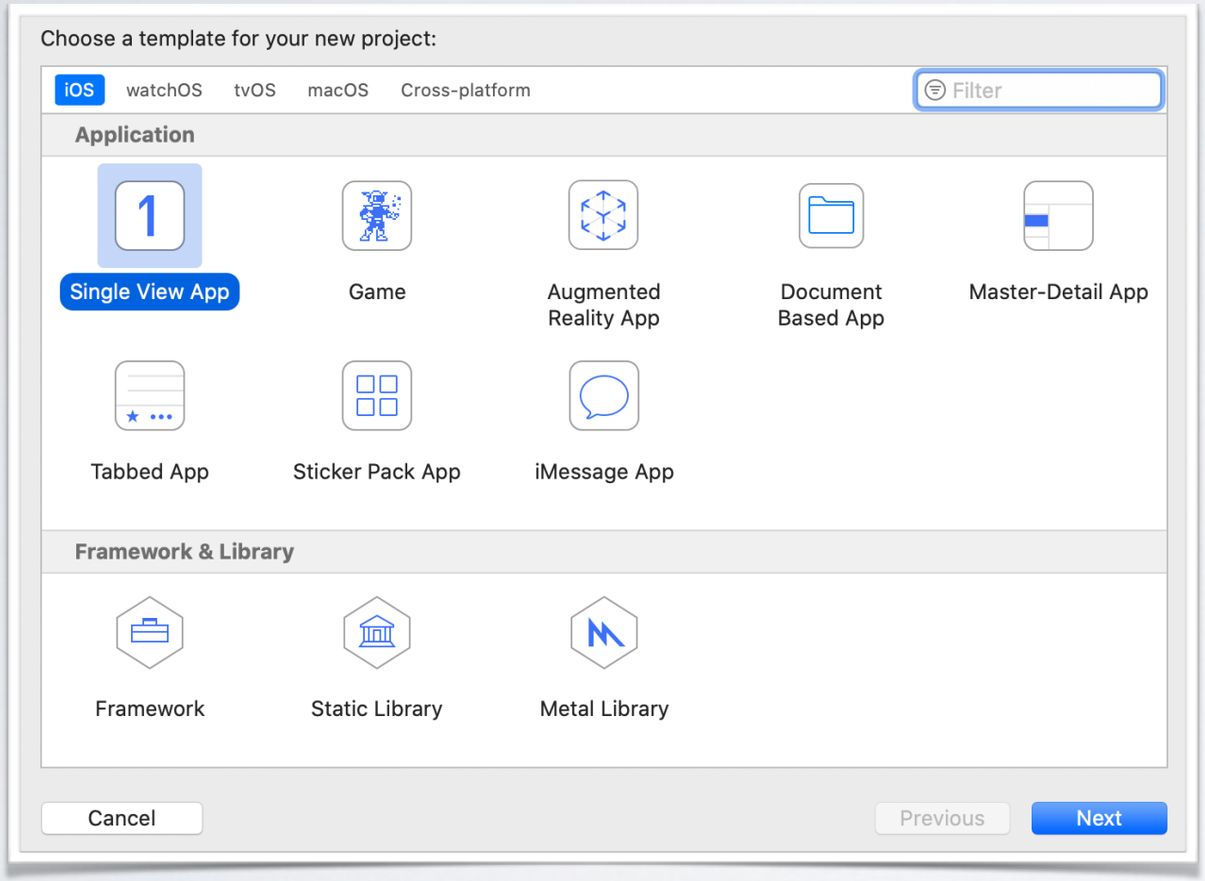
STORYBOARD OVERVIEW

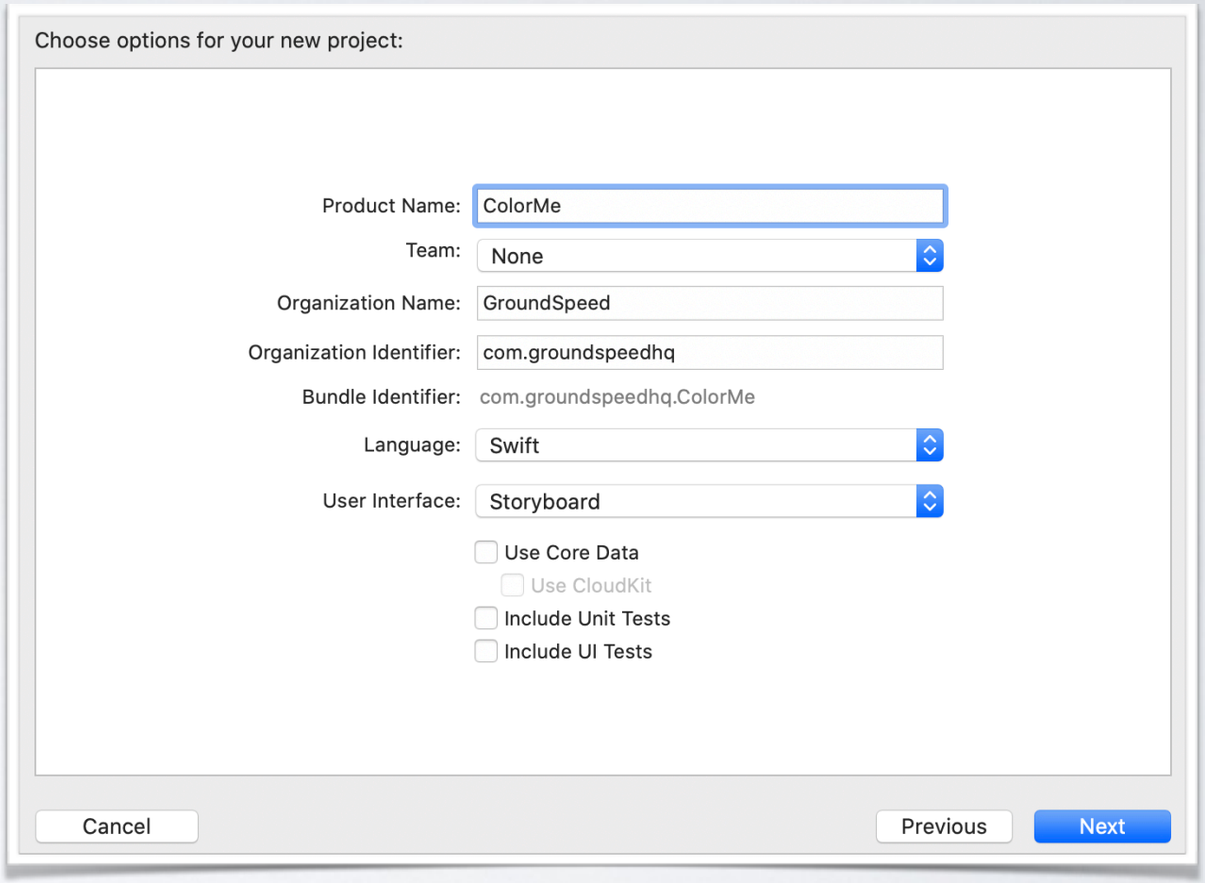


ANOTHER QUICK DEMO - [COLOR ME]



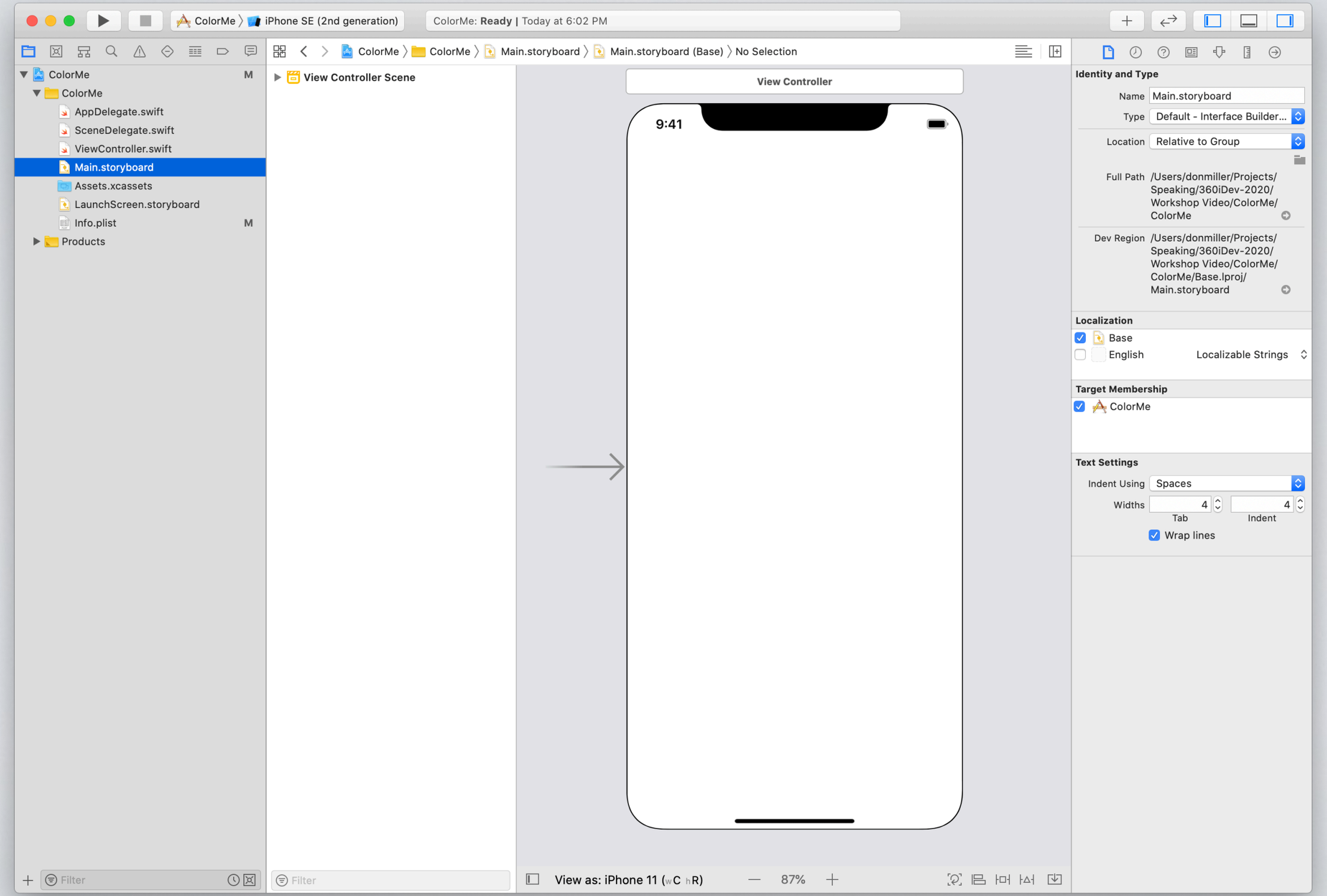
CREATE A SINGLE VIEW APPLICATION



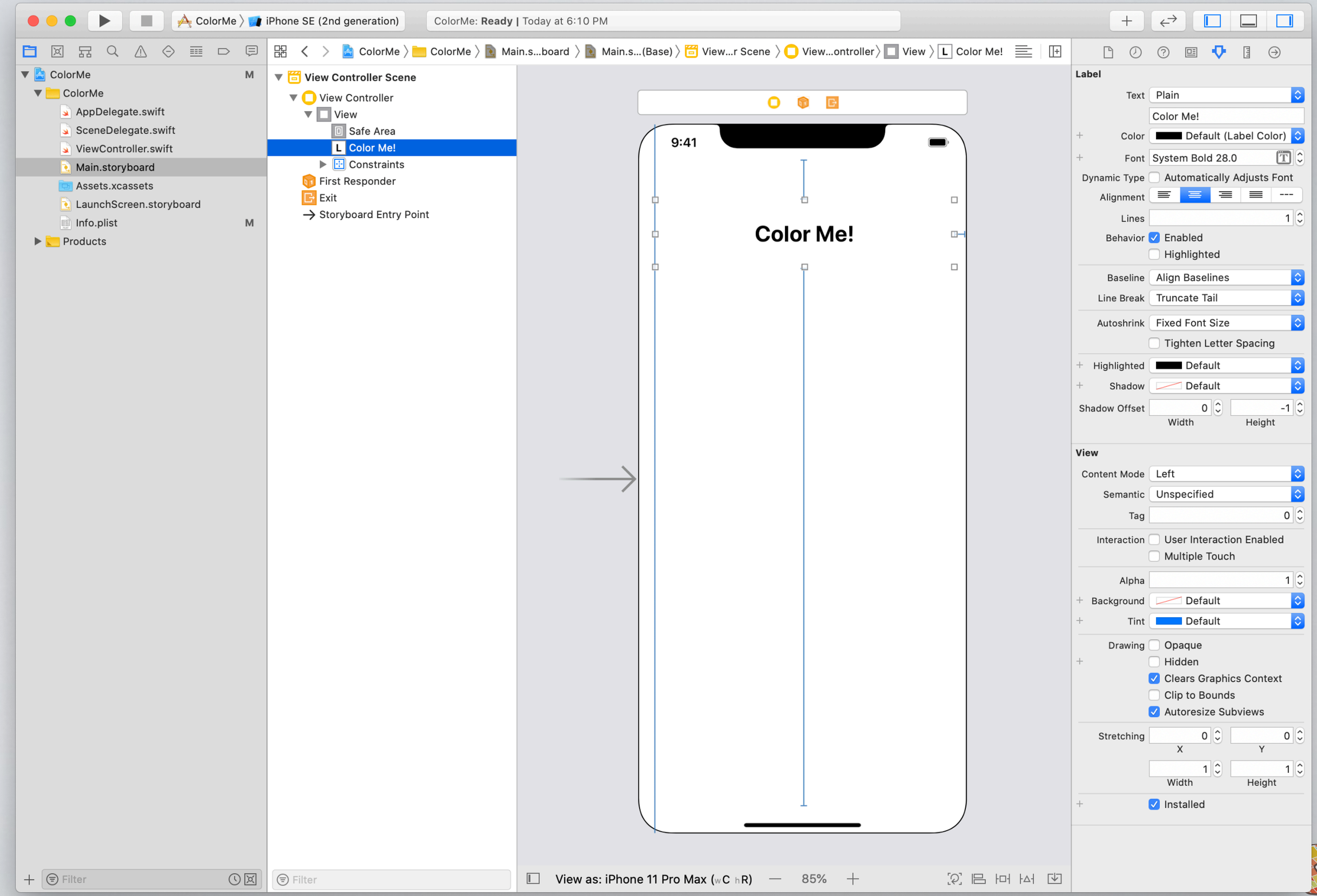


GroundSpeed™
rapid web + mobile software

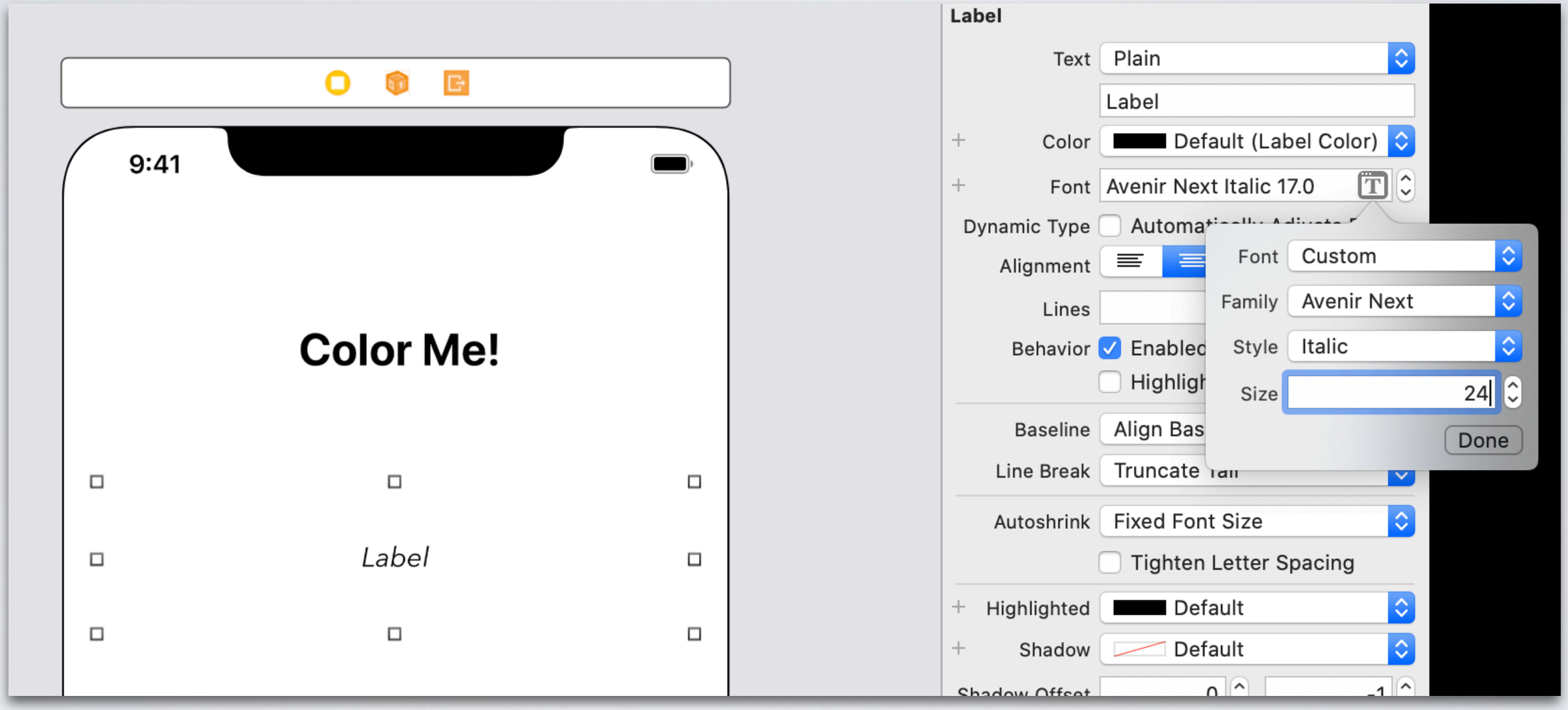
YOUR STORYBOARD SHOULD LOOK LIKE THIS

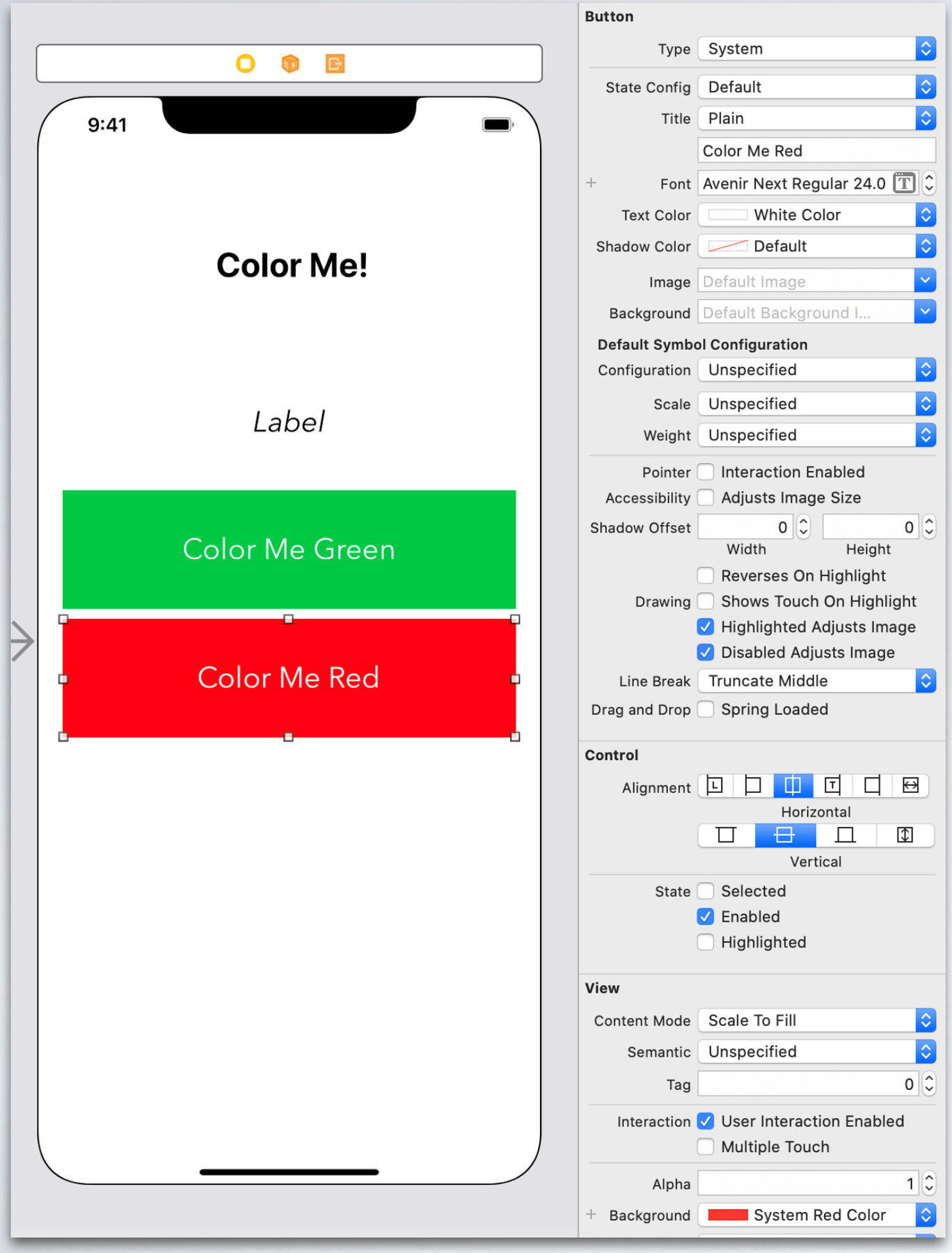


DRAG A LABEL TO THE STORYBOARD



CREATE A BLANK LABEL FOR OUTPUT

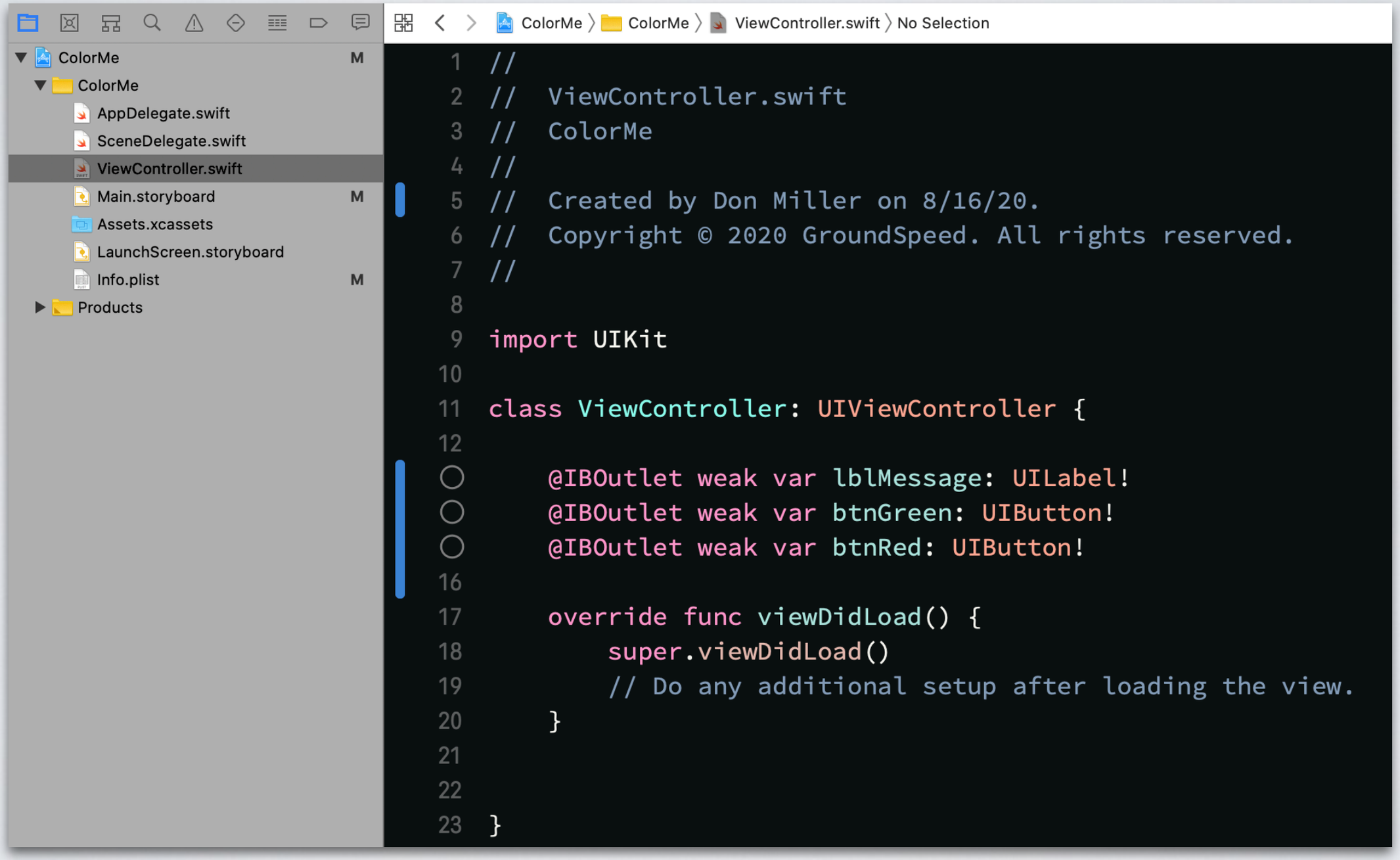




CREATE TWO BUTTONS

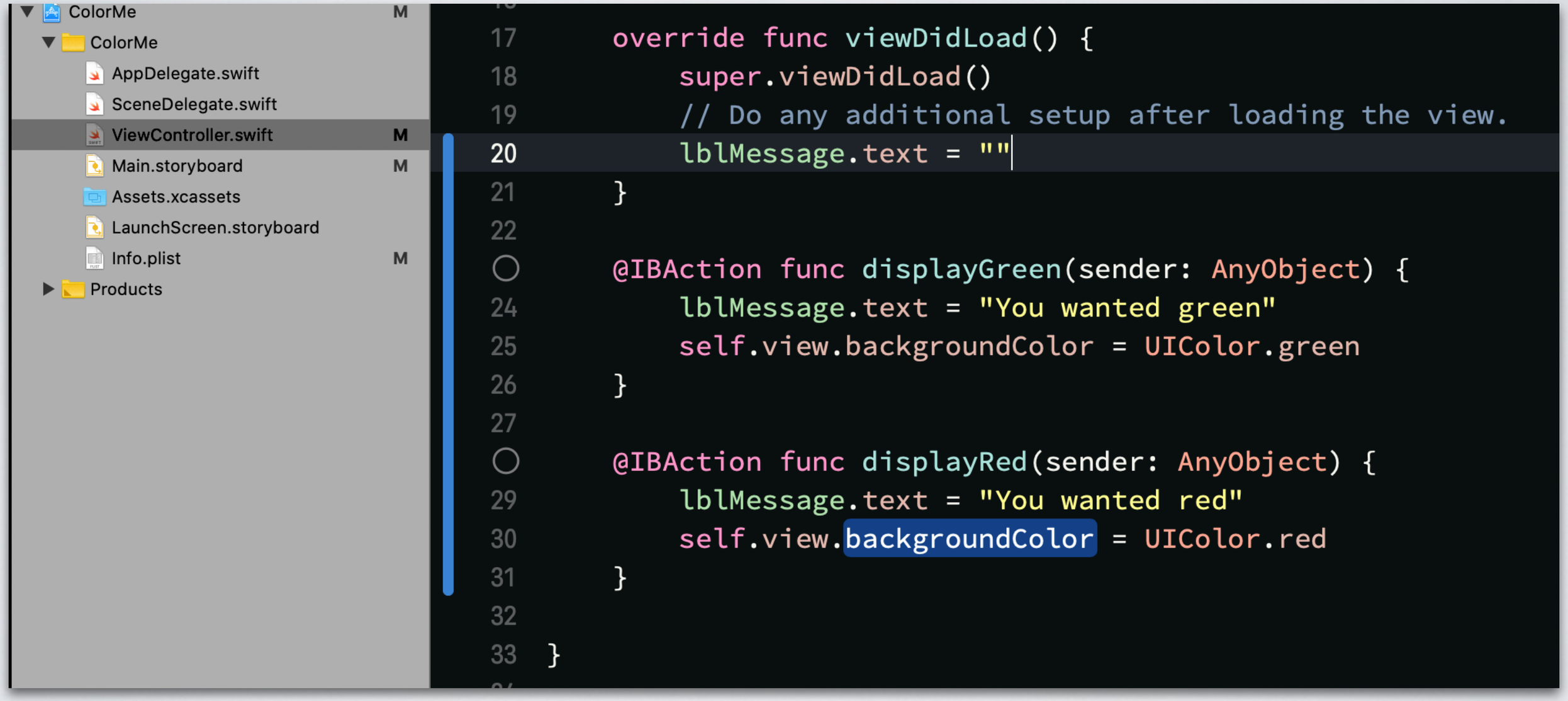
GroundSpeed™
rapid web + mobile software

ADD OUTLETS TO CONTROLLER

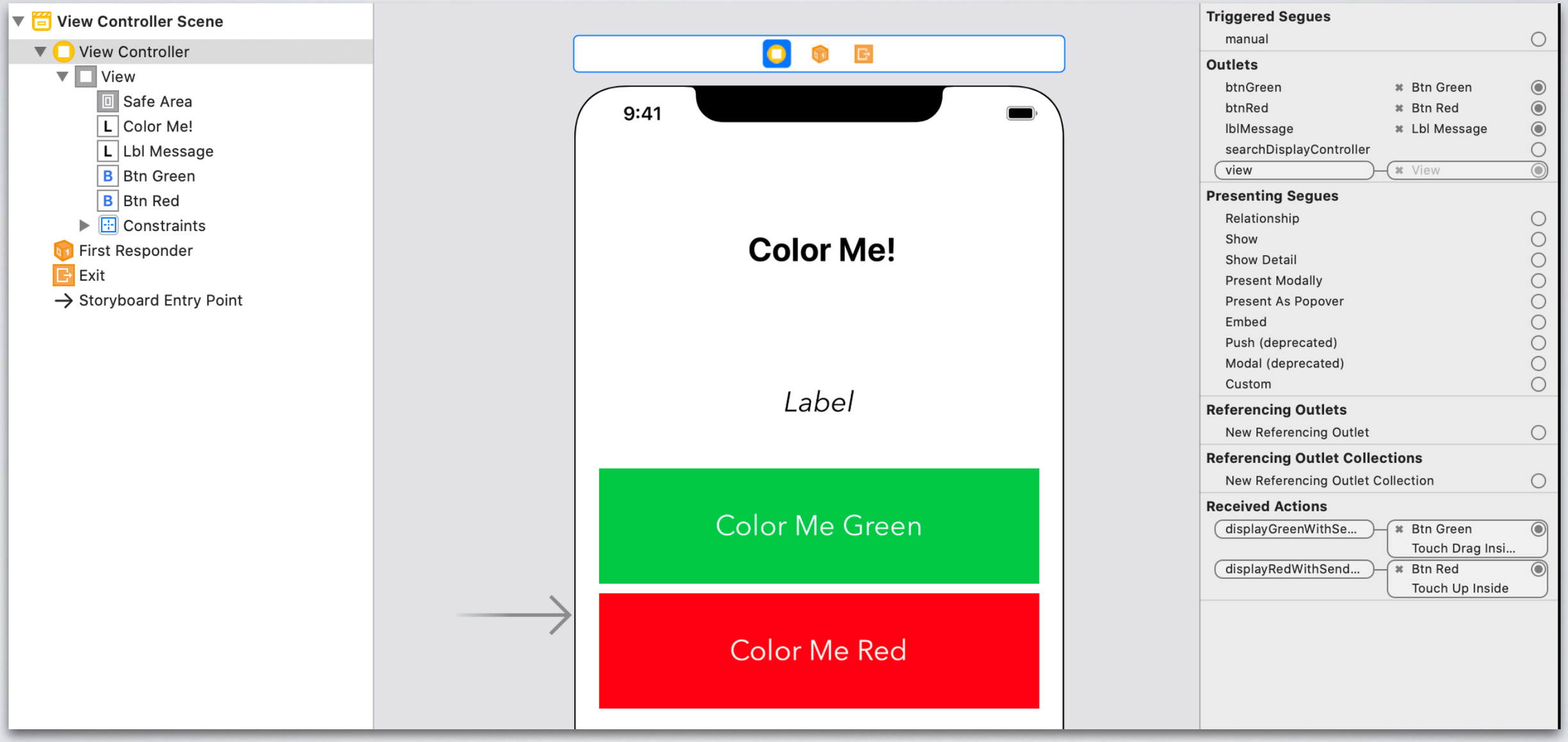


GroundSpeed™
rapid web + mobile software

ADD ACTIONS TO CONTROLLER AND UPDATE VIEWDIDLOAD



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

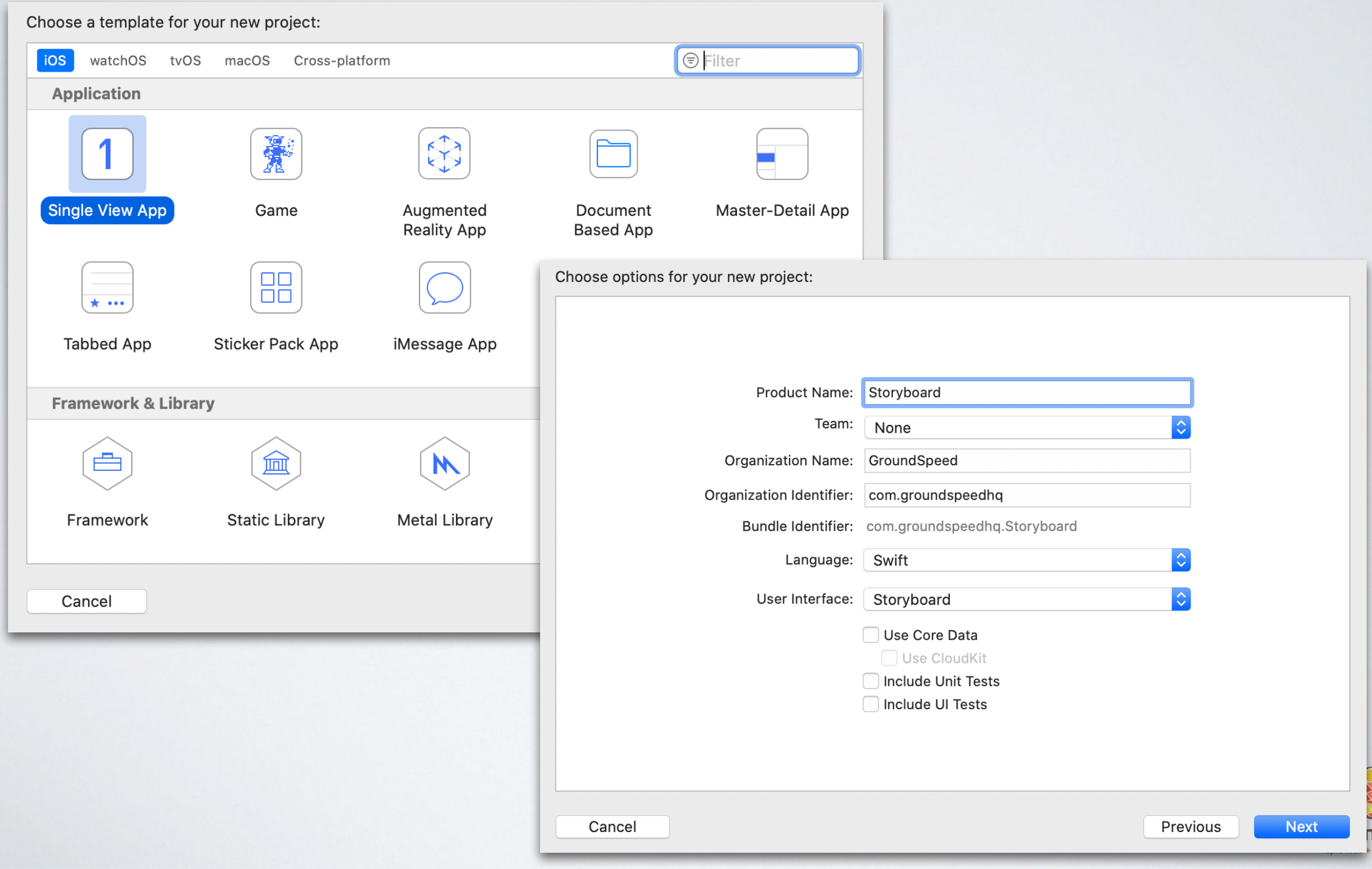
Color Me Green

Color Me Red

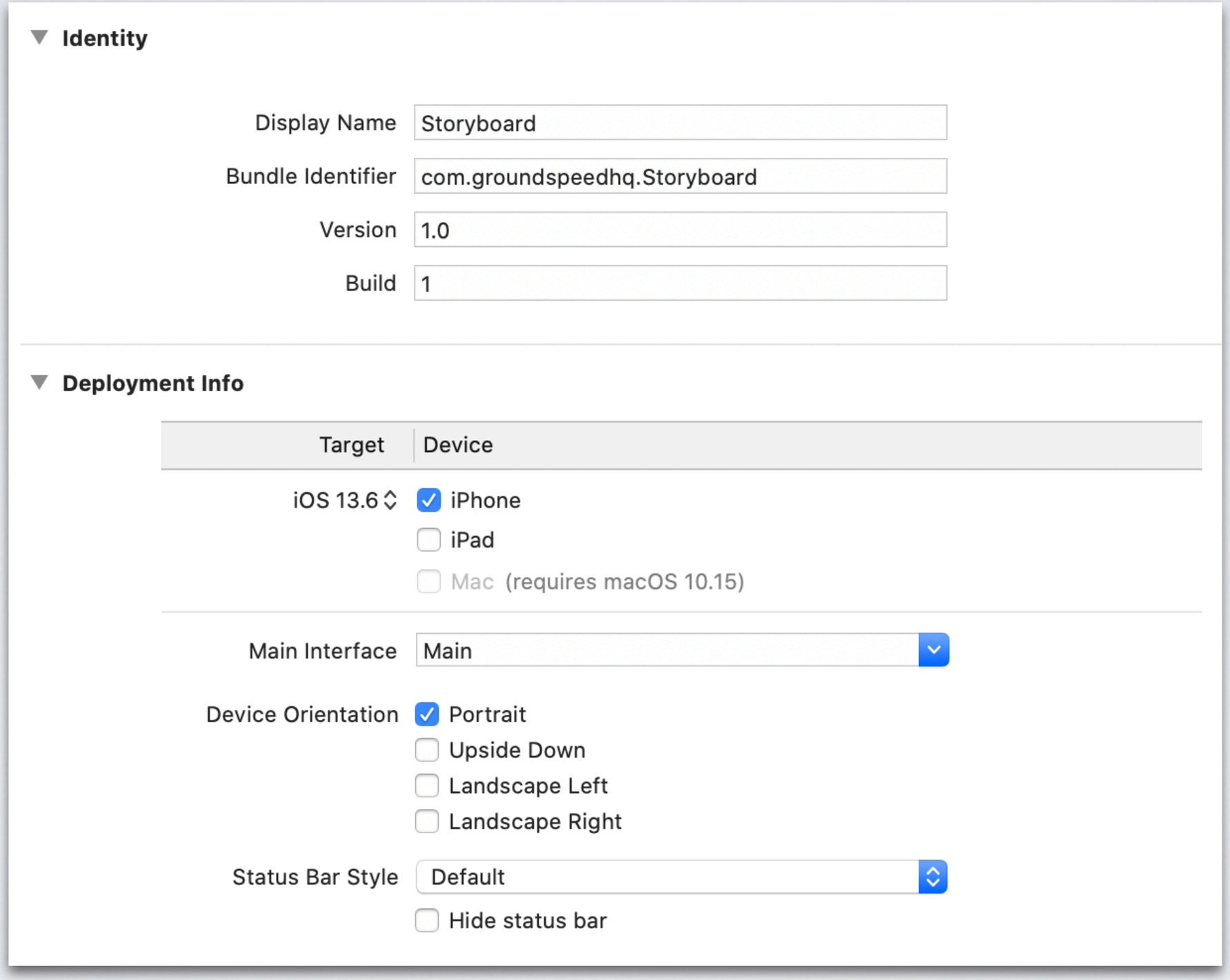
Lab #3 - Create Color Me Storyboard App

GroundSpeed™
rapid web + mobile software

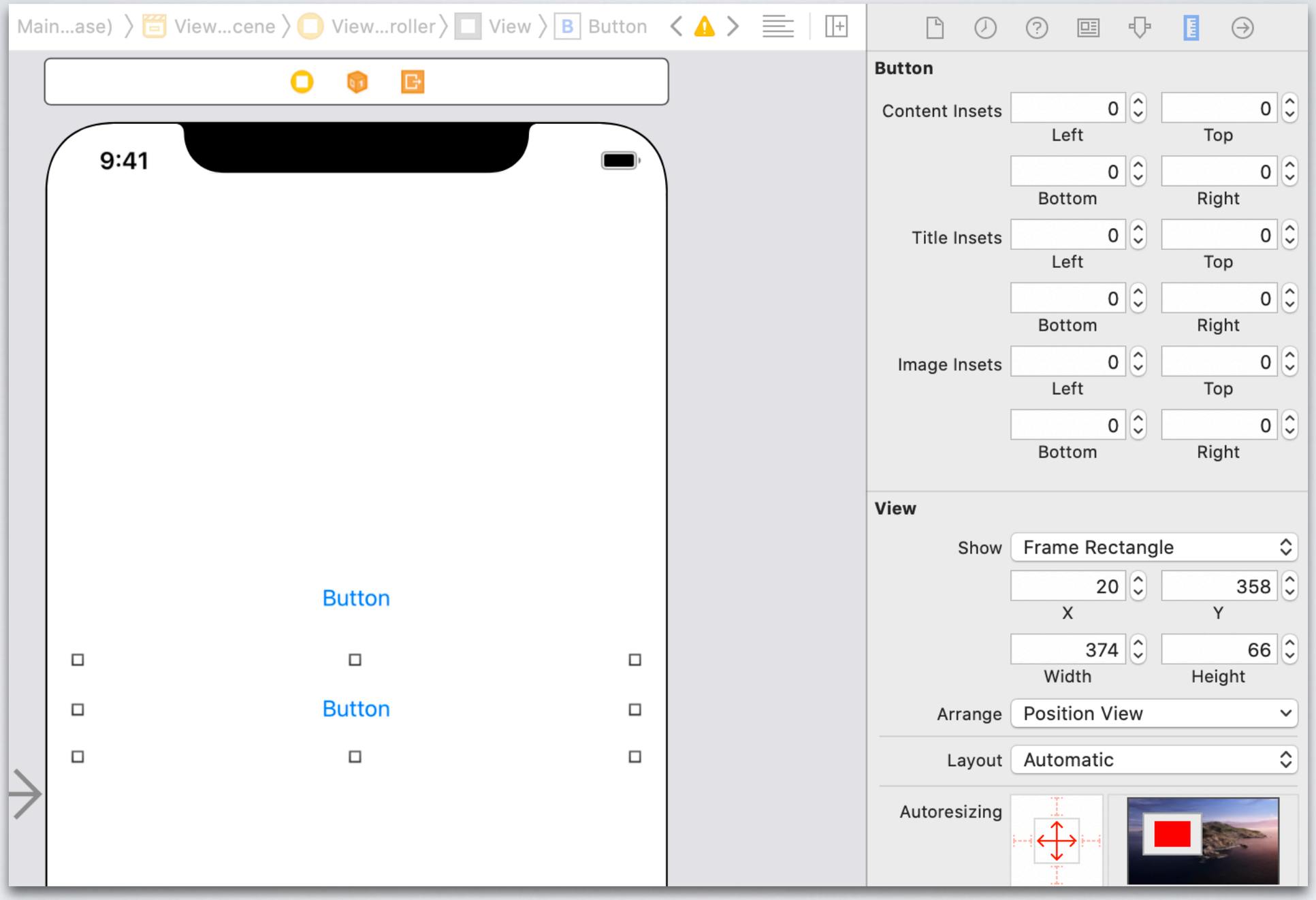
STORYBOARD DEMO
STEP 1: CREATE A NEW SINGLE VIEW APPLICATION



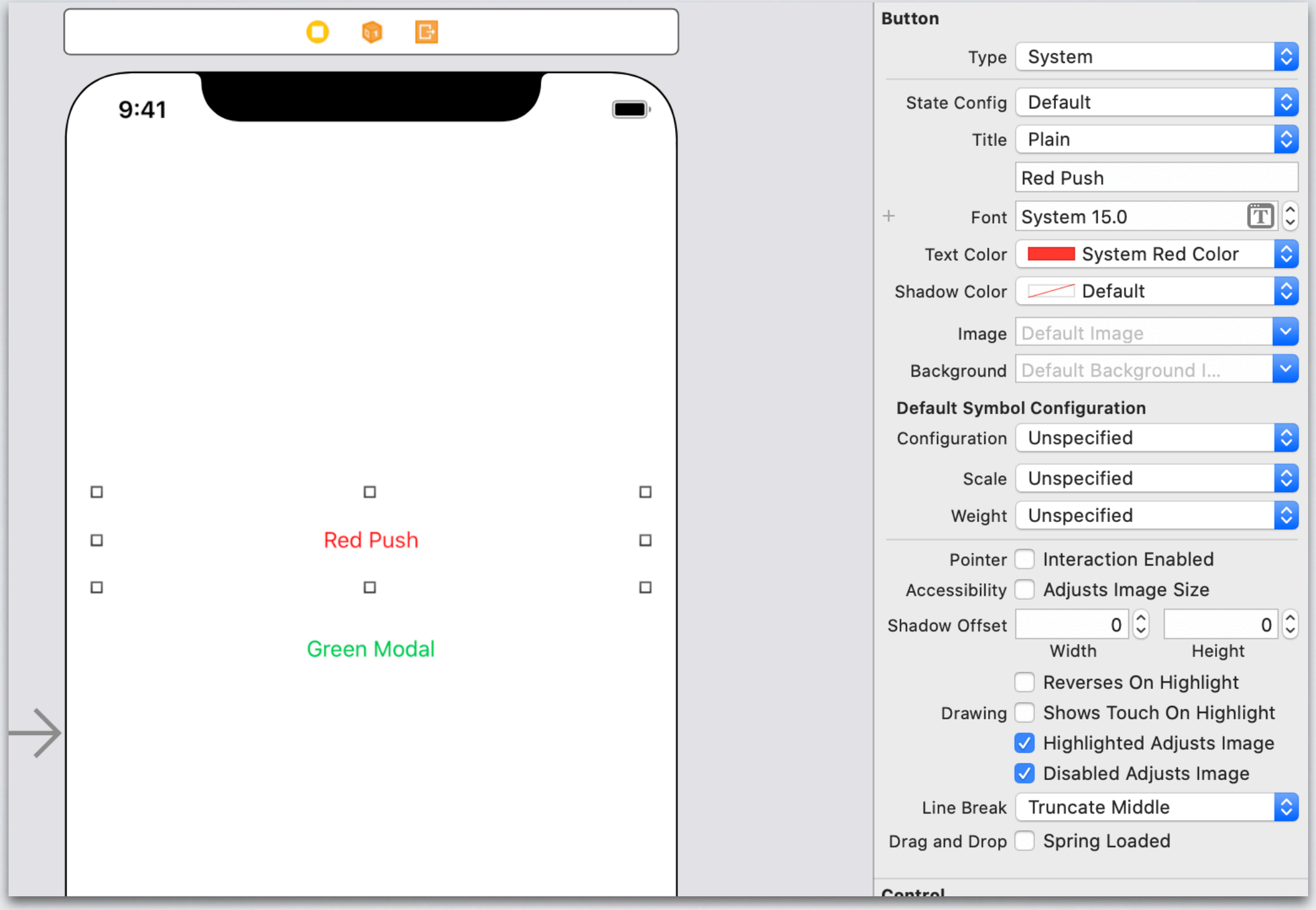
STEP 2: OBSERVE THE SUMMARY PAGE



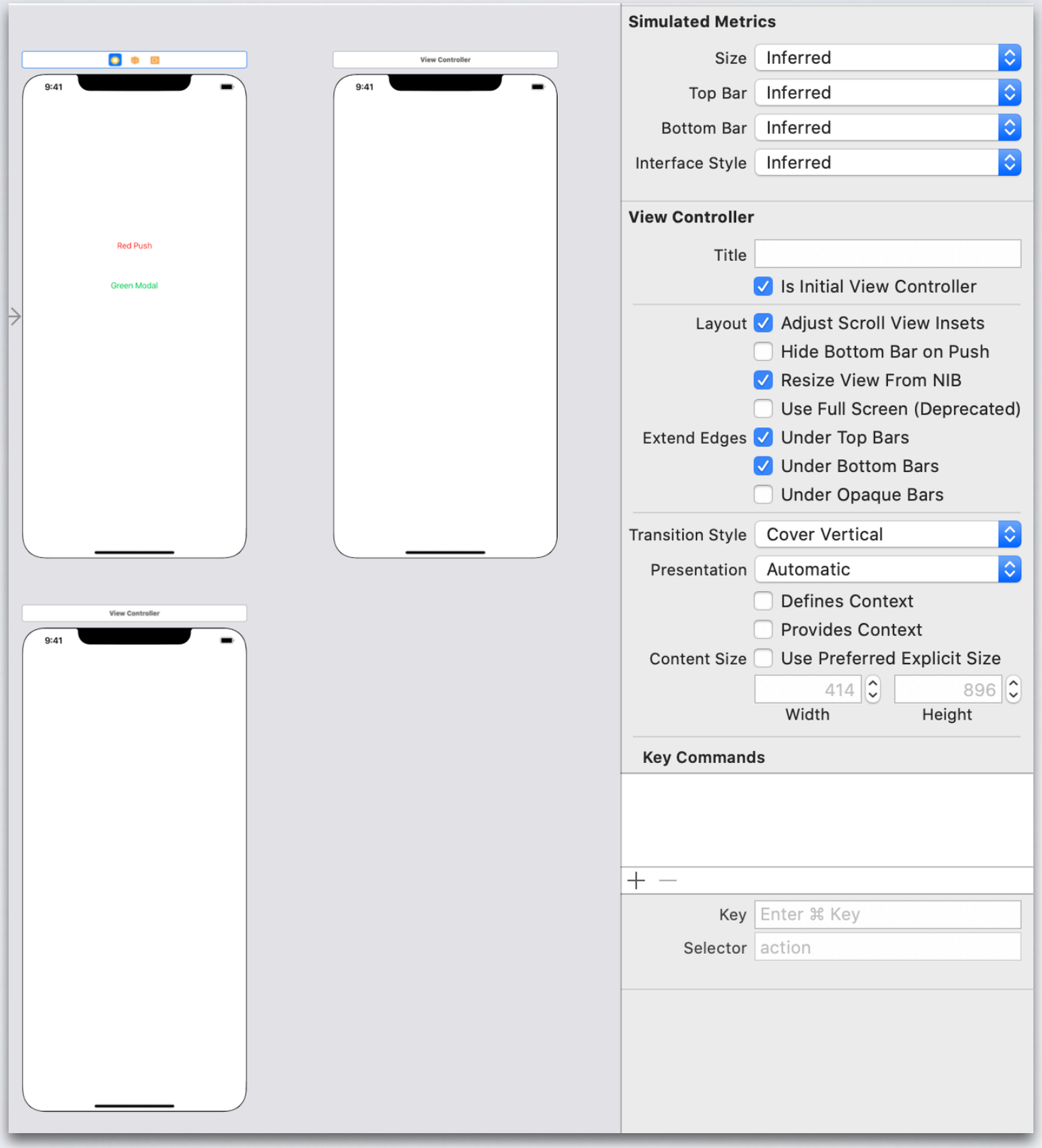
STEP 3: ADD 2 BUTTONS TO THE VIEW CONTROLLER
STEP 4: LET'S MAKE THEM THE SAME WIDTH IN SIZE INSPECTOR



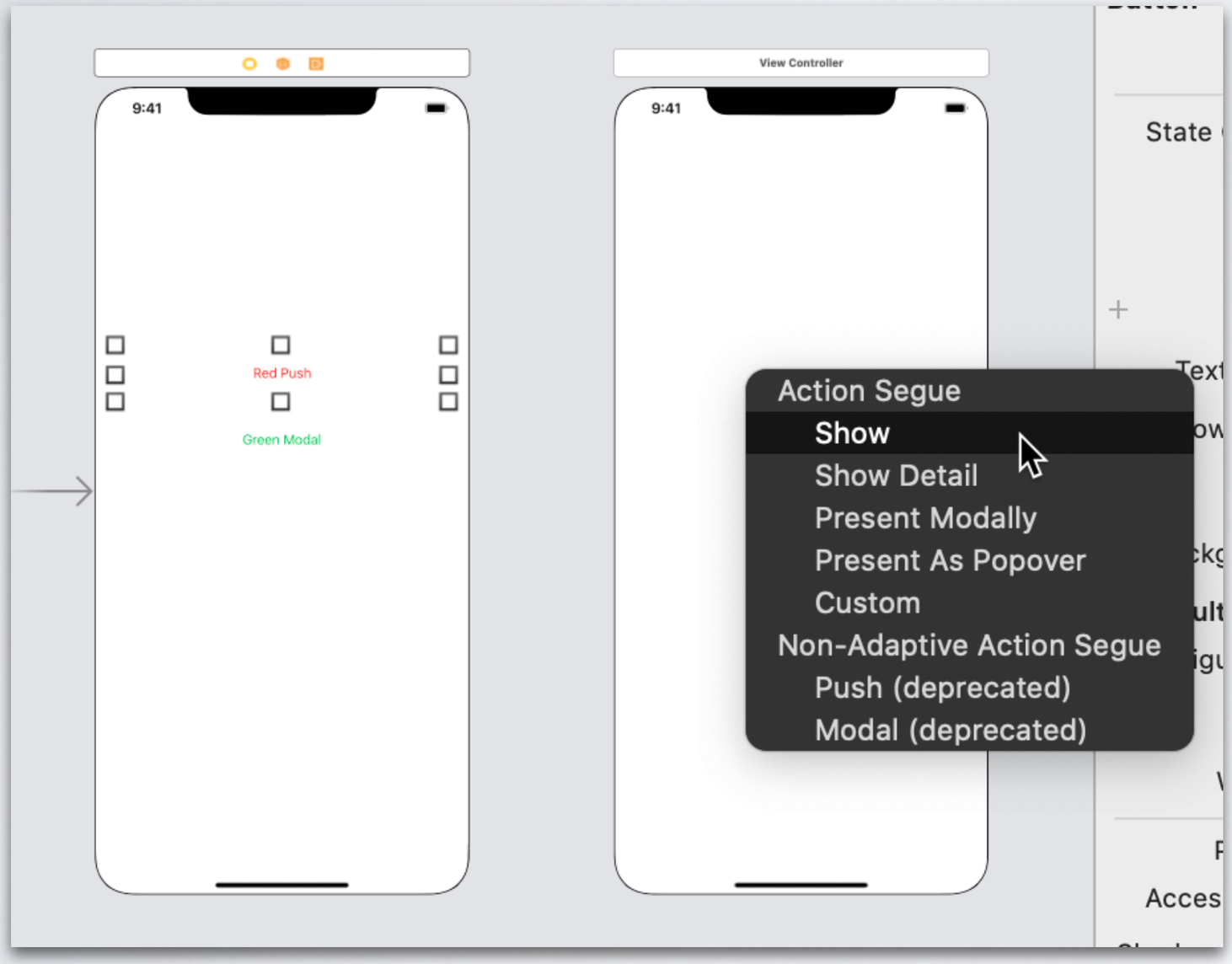
STEP 5: CHANGE THE BUTTON LABELS TEXT TO RED AND GREEN

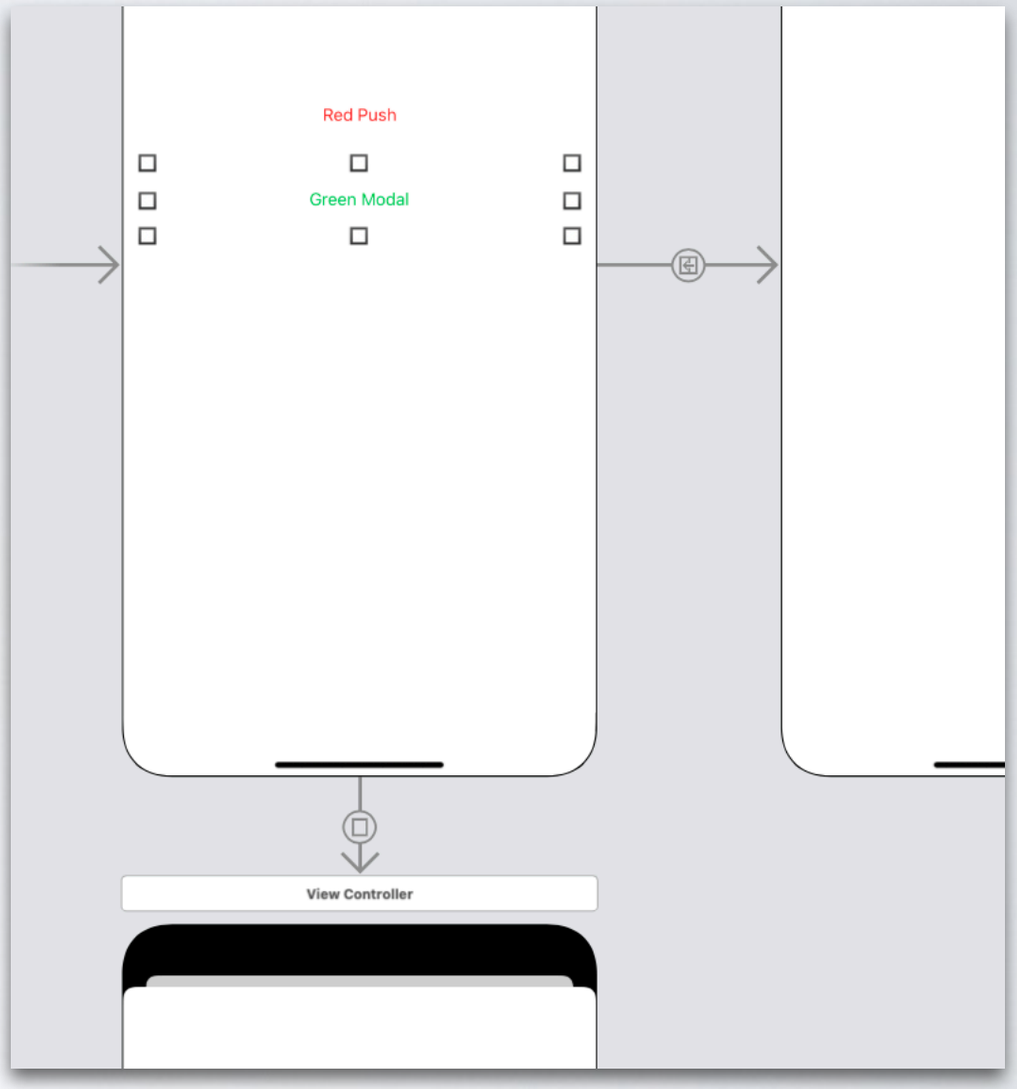


STEP 6: ADD TWO VIEW CONTROLLERS TO THE STORYBOARD BY DRAGGING THEM OVER

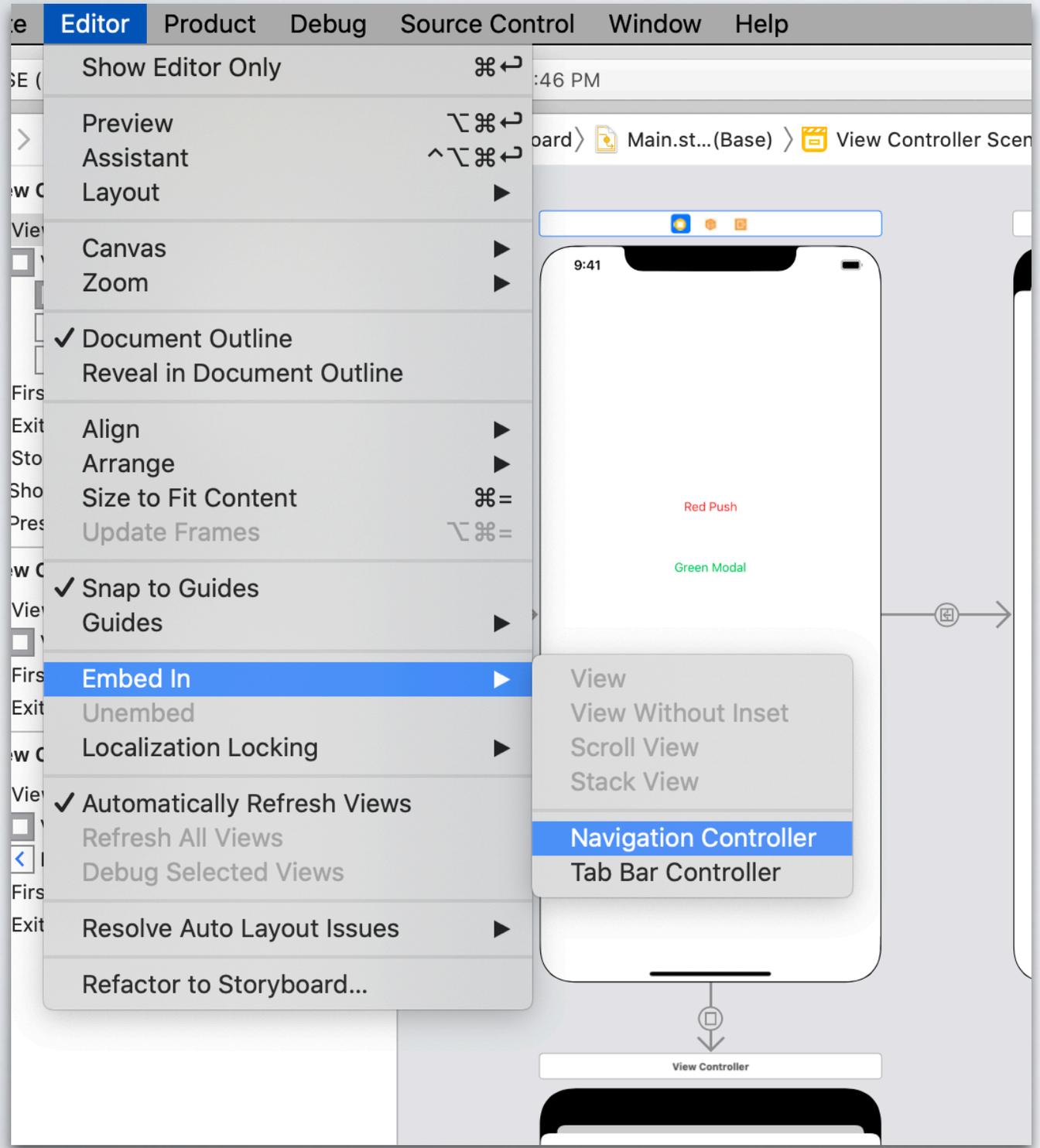


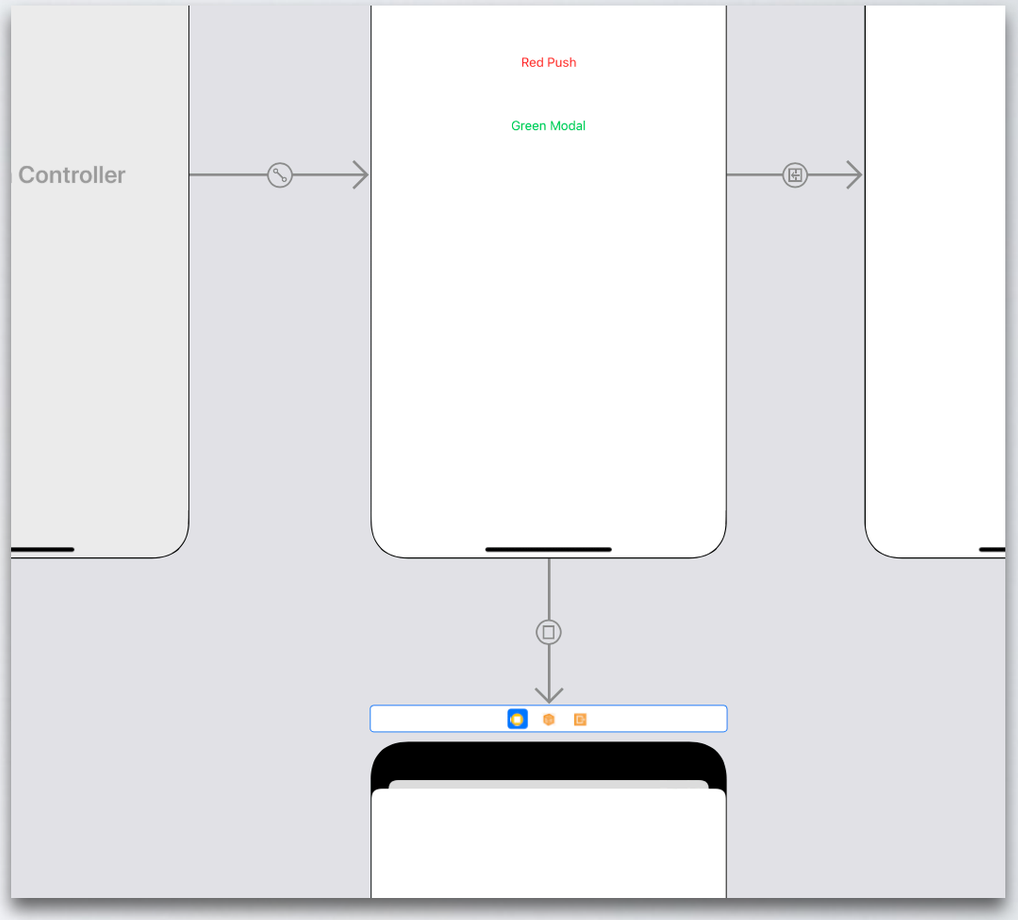
STEP 7: CONNECT THE BUTTONS TO THE NEW VIEW CONTROLLERS BY HOLDING DOWN THE [CONTROL] KEY AND DRAGGING FROM THE BUTTON TO THE NEW VIEW CONTROLLER





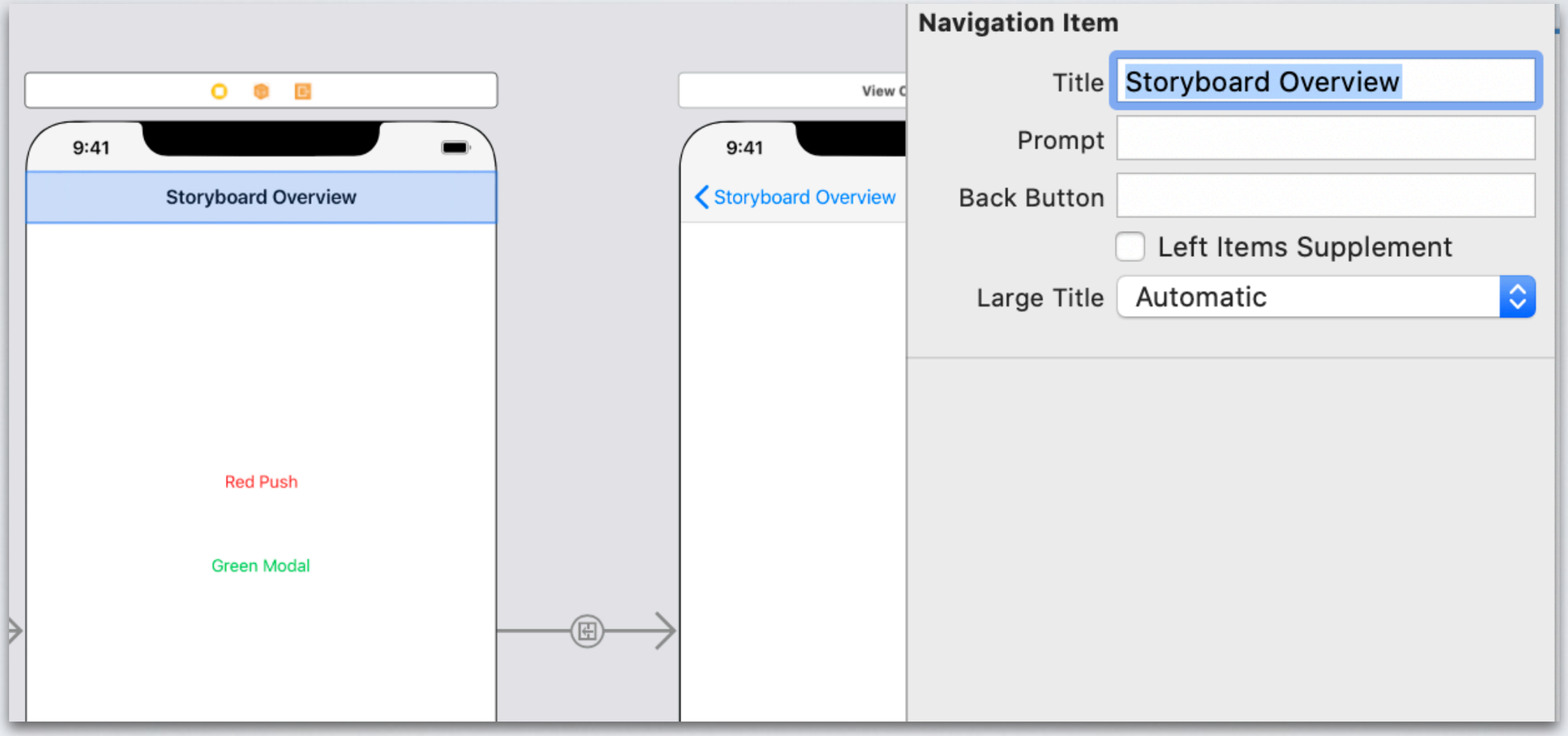
STEP 8: EMBED A NAVIGATION CONTROLLER ON YOUR PRIMARY VIEW



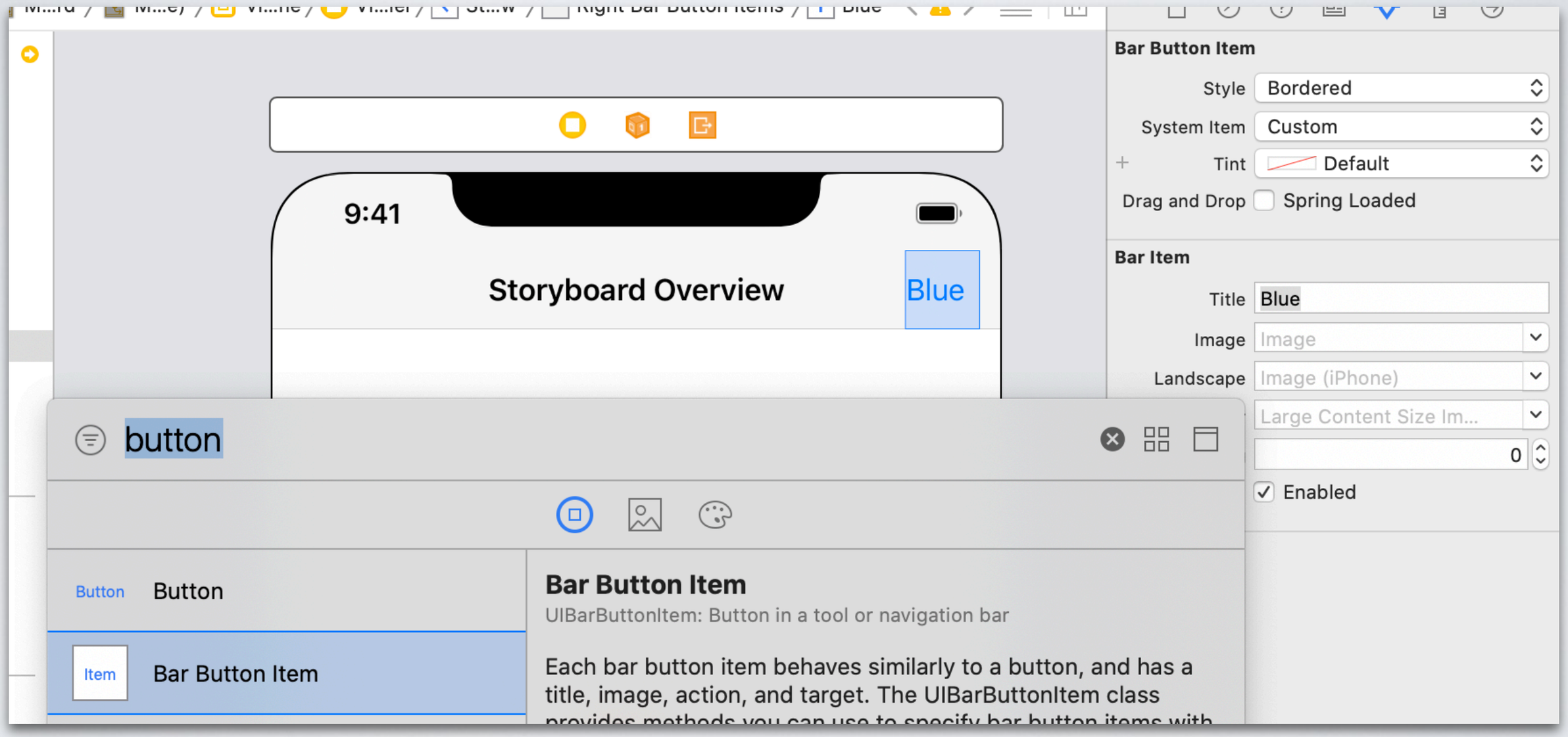


GroundSpeed™
rapid web + mobile software

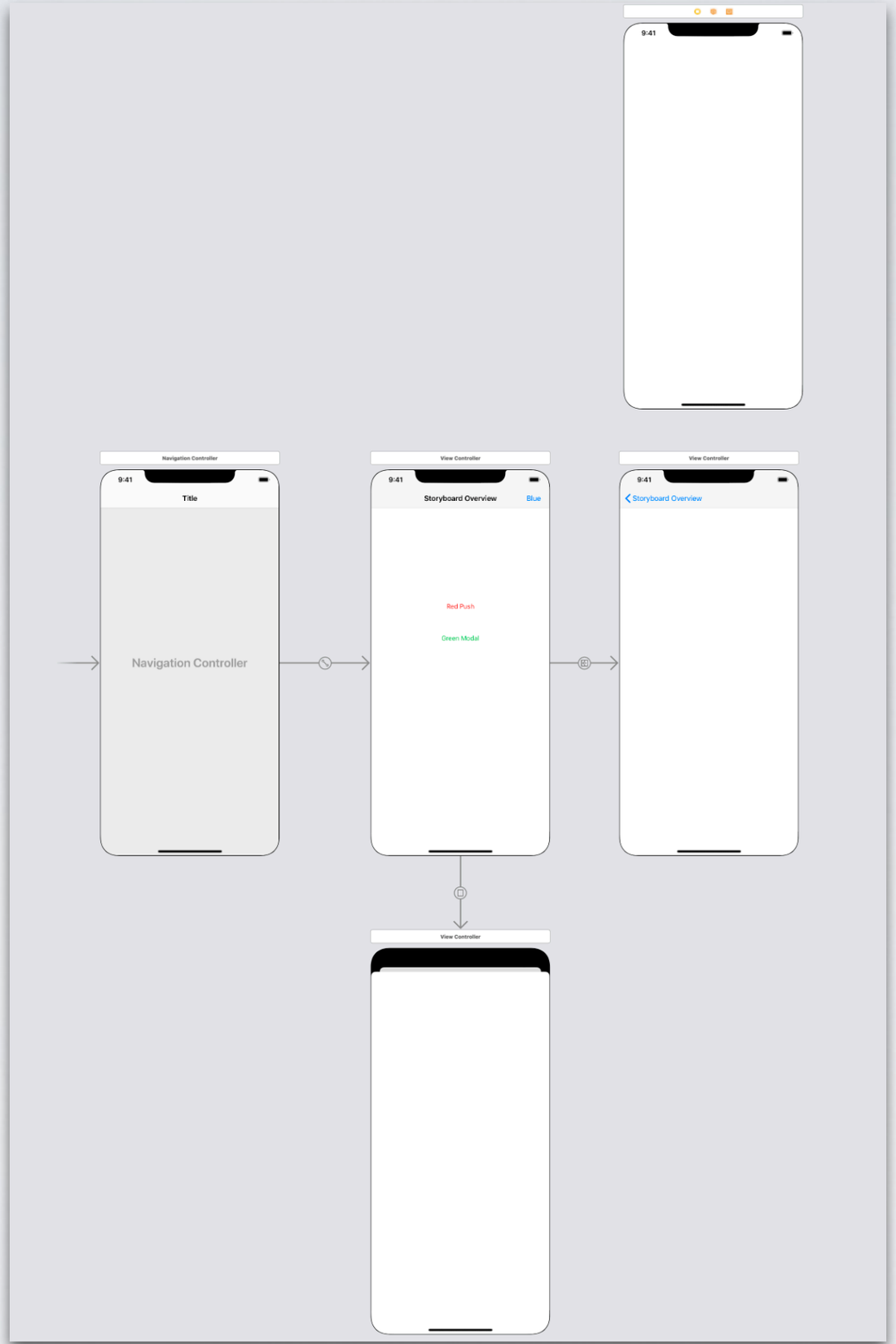
STEP 9: CHANGE THE TITLE OF THE NAVIGATION BAR



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

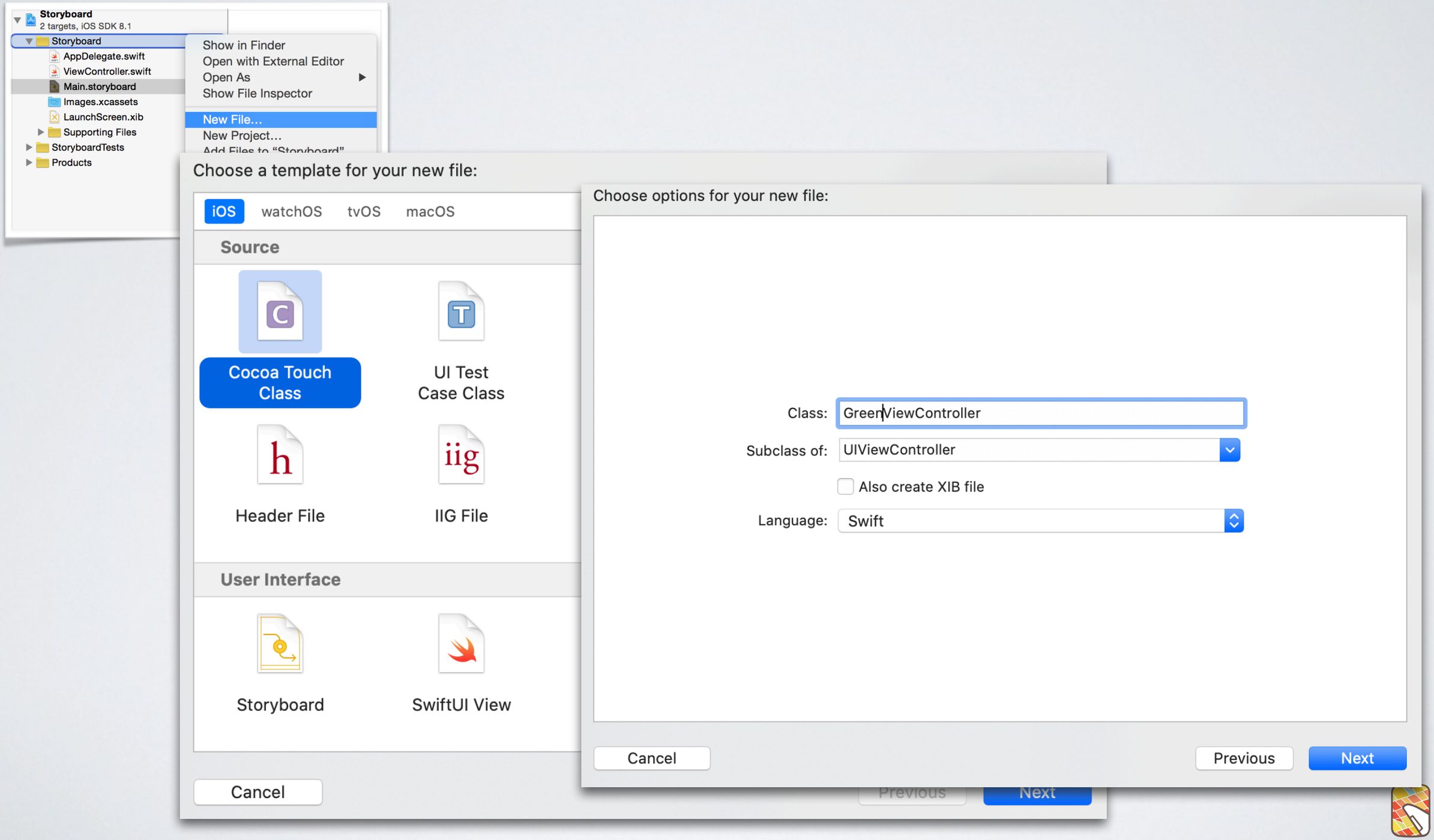


STEP 11: ADD OUR THIRD AND FINAL VIEW CONTROLLER TO THE STORYBOARD

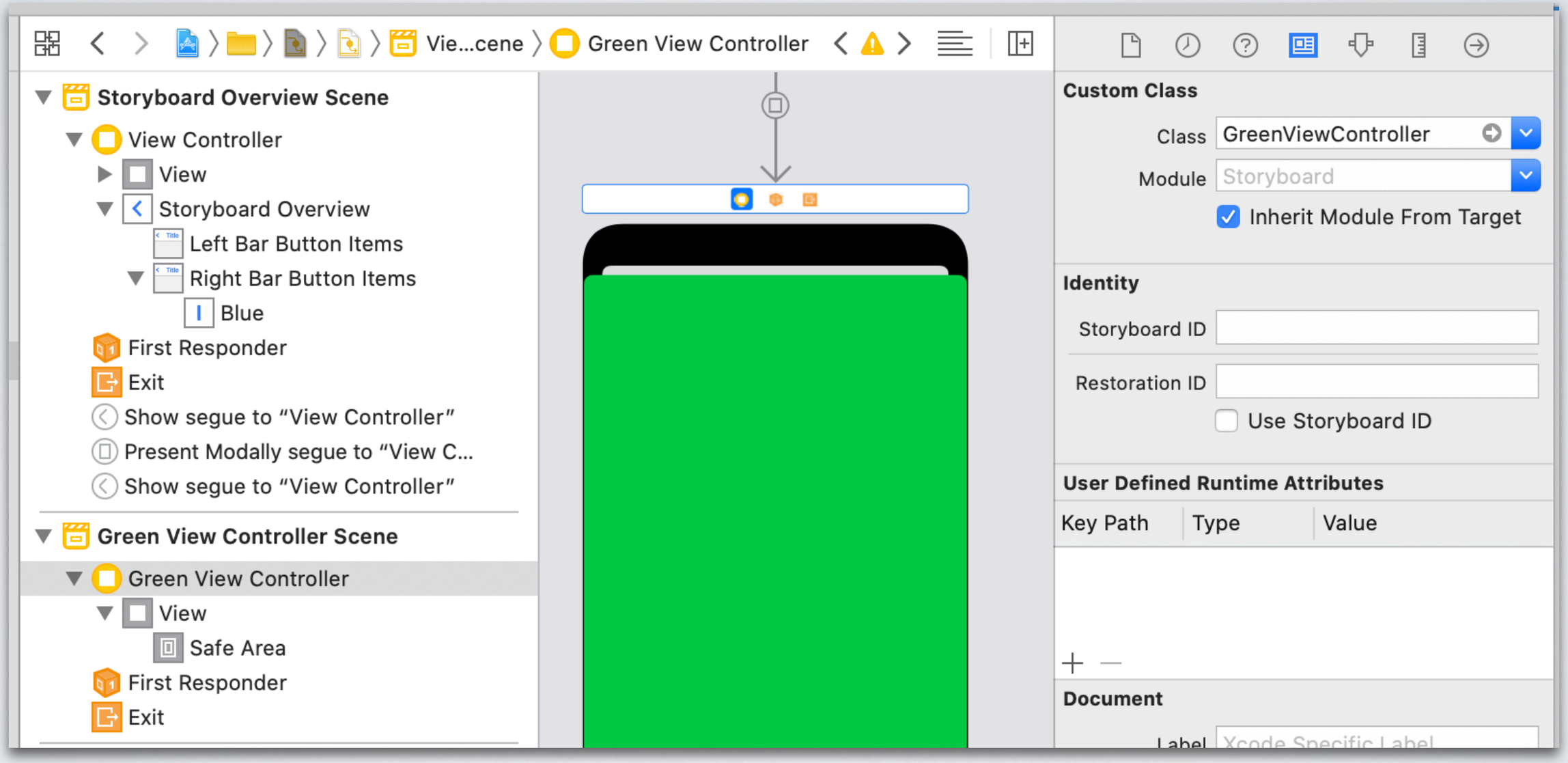


STEP 12: CONNECT OUR BAR BUTTON ITEM TO THE NEW VIEW CONTROLLER AND SELECT SHOW. REMEMBER TO HOLD DOWN THE [CONTROL] BUTTON WHILE CLICKING AND DRAGGING

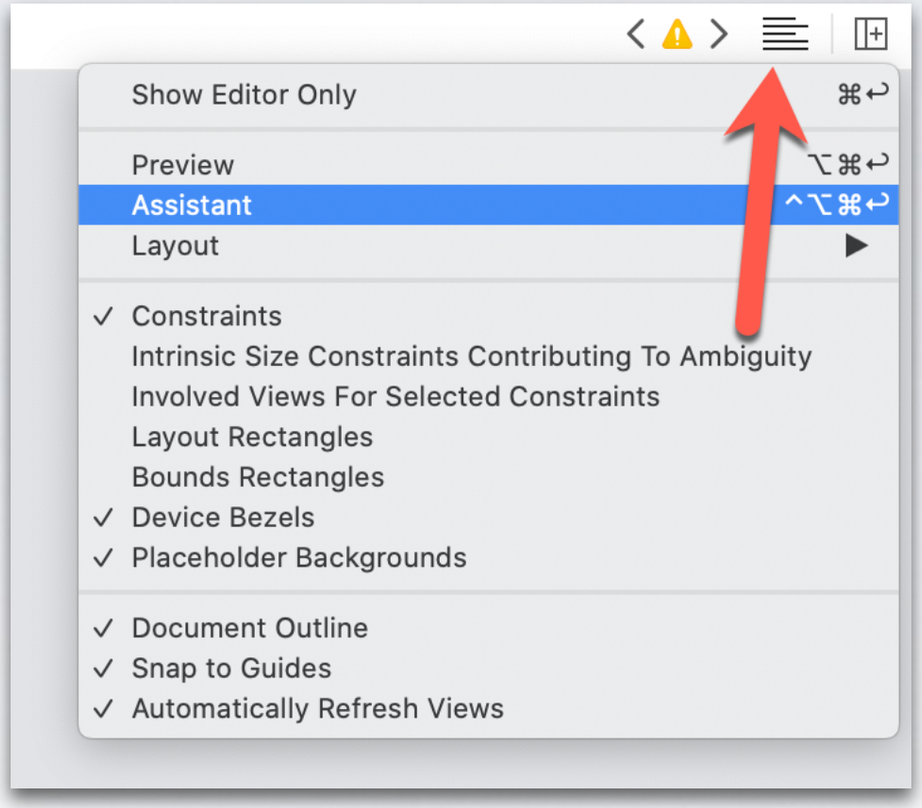
STEP 13: NOW WE NEED TO DO A LITTLE PROGRAMMING. ADD A CONTROLLER OBJECT AND NAME IT "GREENVIEWCONTROLLER" AS SHOWN BELOW



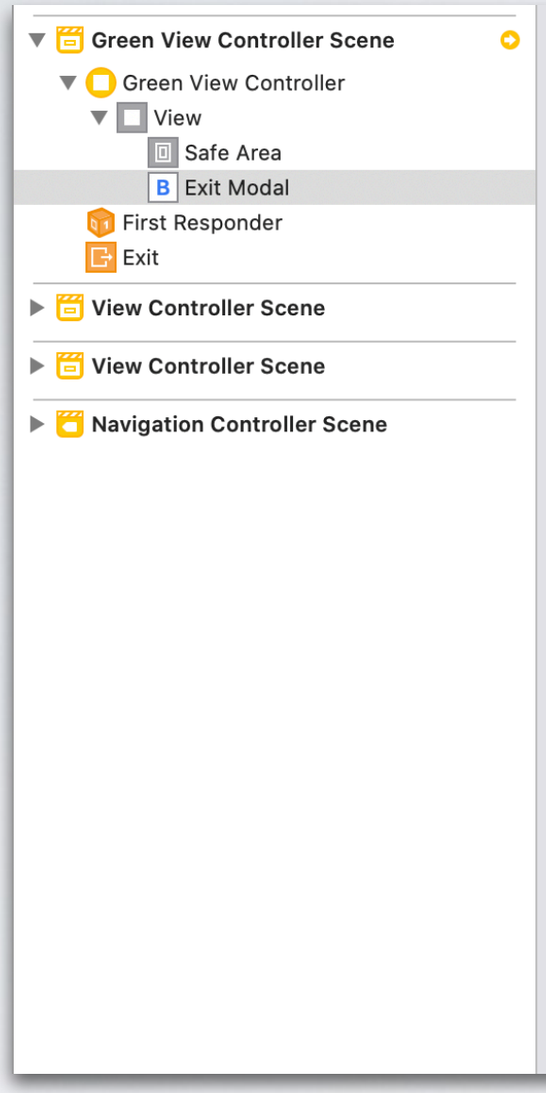
STEP 14: ASSIGN YOUR VIEW IN THE STORYBOARD TO THE NEW OBJECT VIEW CONTROLLER YOU JUST CREATED

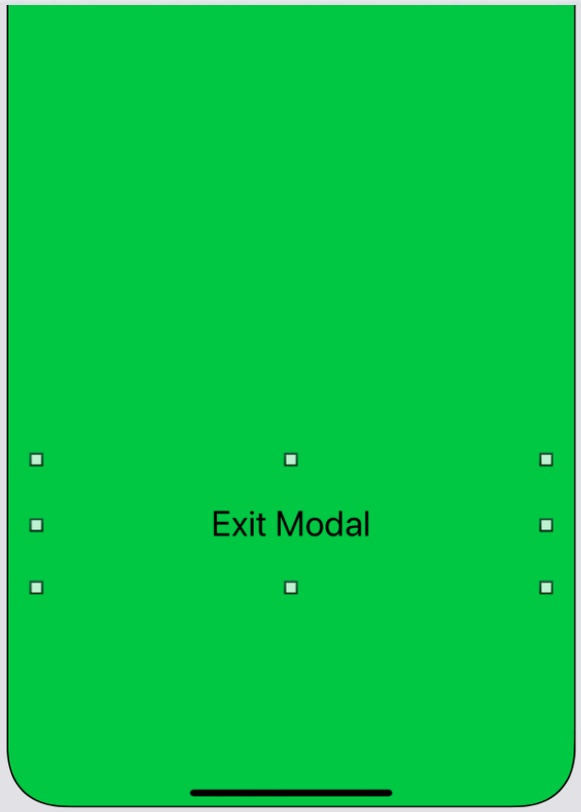


STEP 15: SHOW THE ASSISTANT EDITOR TO HAVE OUR CODE GENERATED FOR US BY CLICKING



STEP 16: ADD A BUTTON TO THE GREEN VIEW

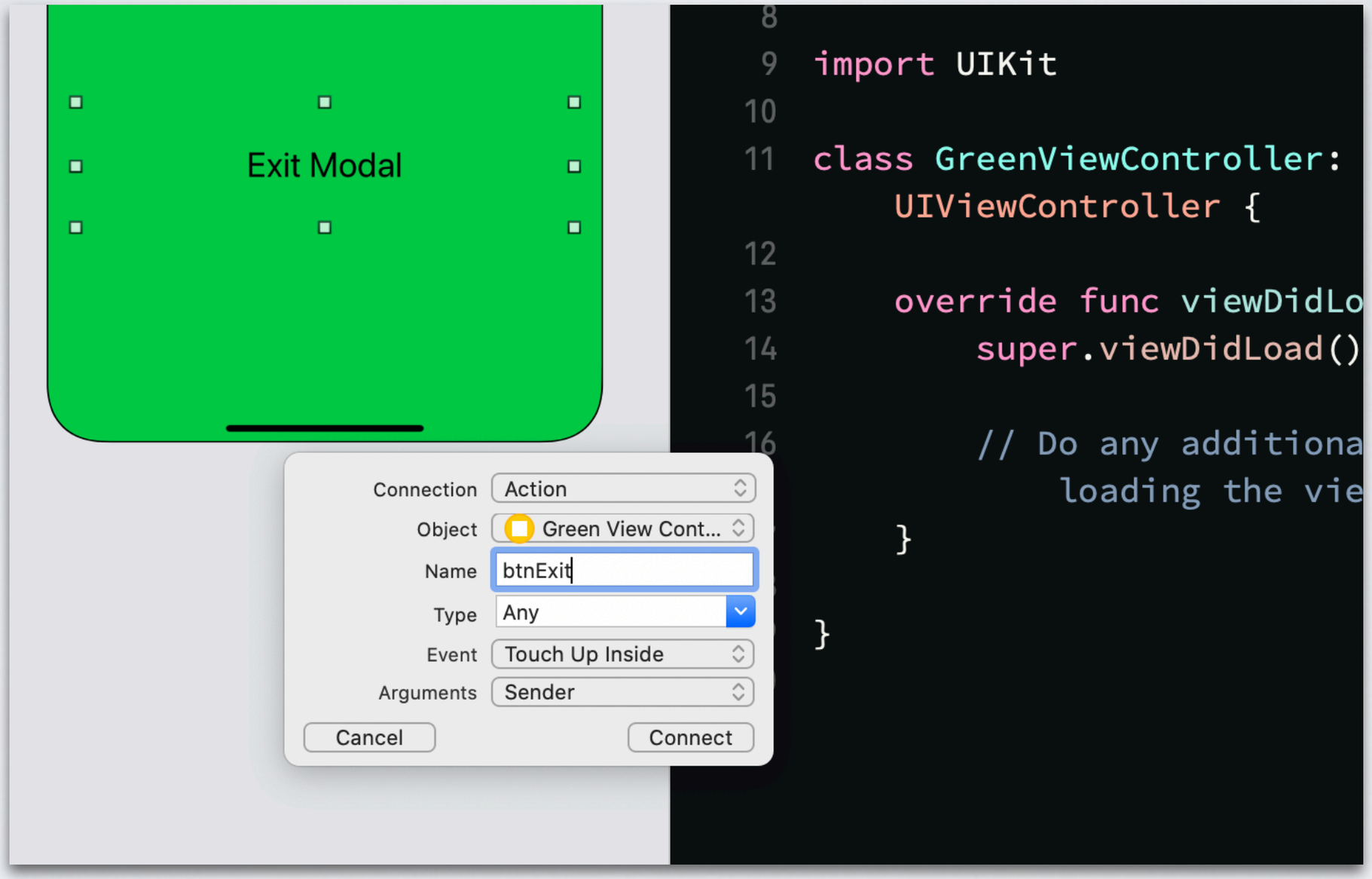





```
2  //  GreenViewController.swift
3  //  Storyboard
4  //
5  //  Created by Don Miller on 8/16/20.
6  //  Copyright © 2020 GroundSpeed. All
       rights reserved.
7  //
8
9  import UIKit
10
11 class GreenViewController:
       UIViewController {
12
13     override func viewDidLoad() {
14        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

GroundSpeed™
rapid web + mobile software

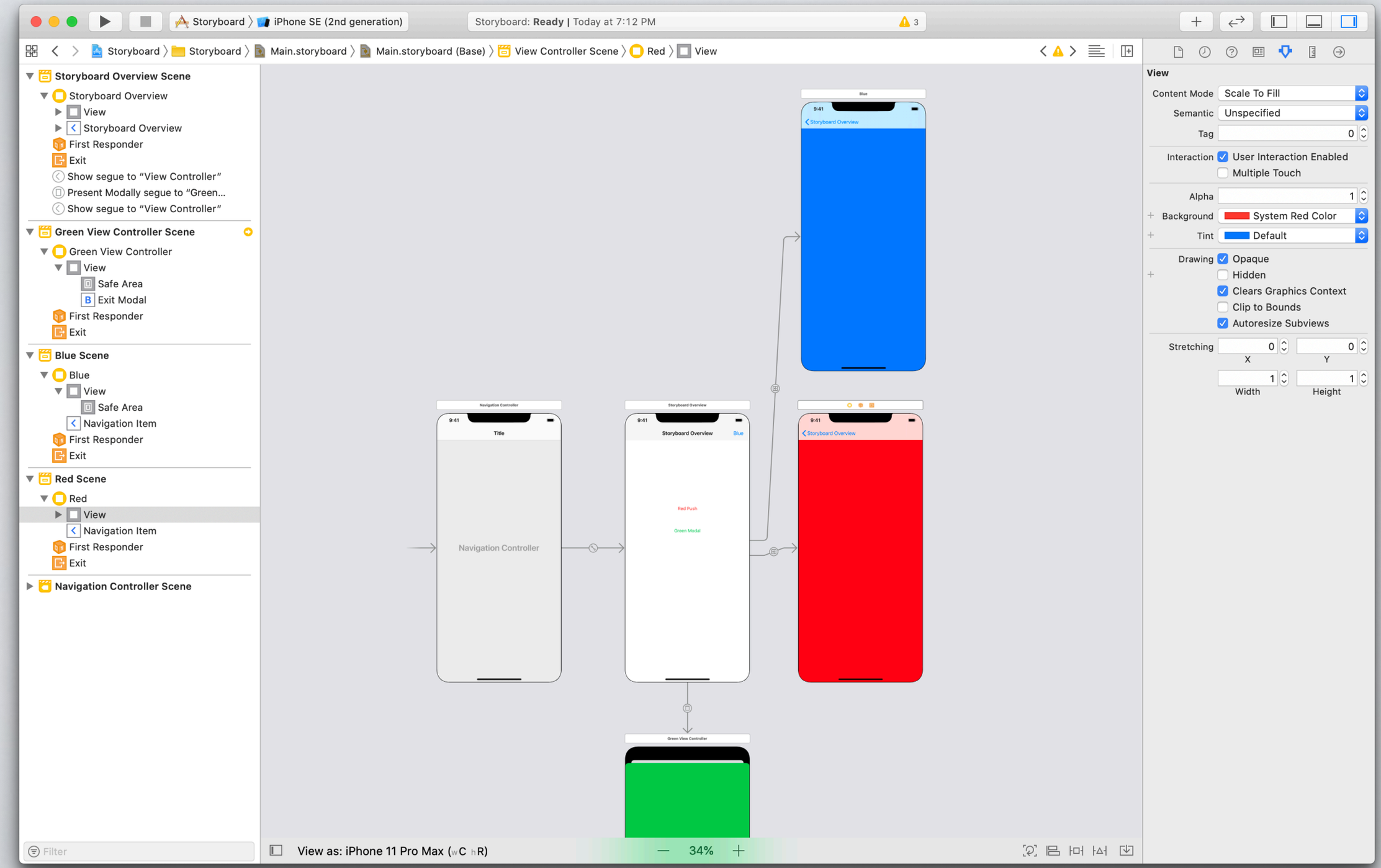
STEP 17: HOLD DOWN THE CONTROL WHEN CLICKING FROM THE BUTTON TO YOU HEADER FILE AND DRAG IT UNDER THE INTERFACE LINE. CHANGE THE CONNECTION TO ACTION AND THE NAME TO BTNEXIT



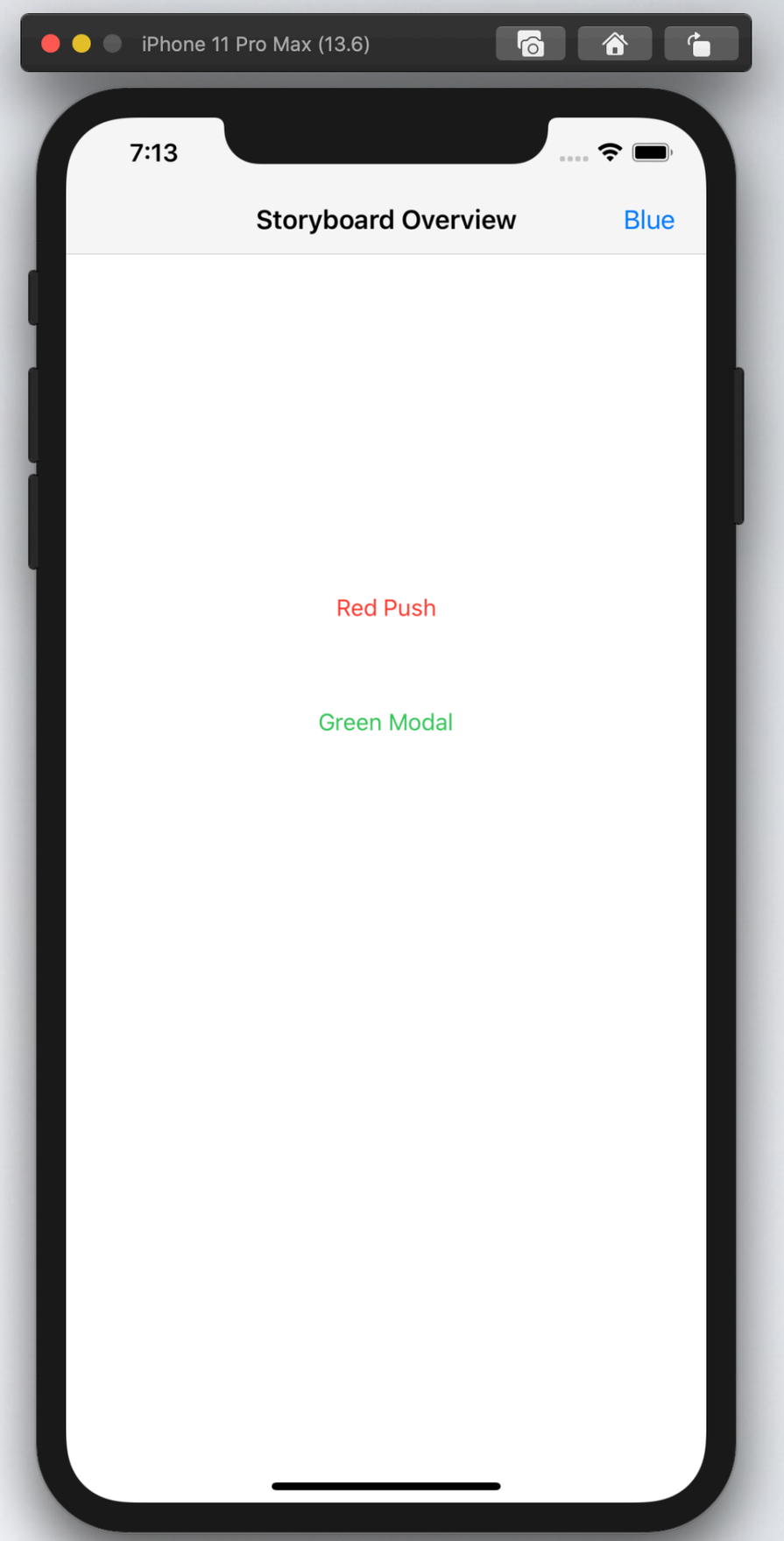
STEP 18: ADD THE FOLLOWING LINE OF CODE IN THE BTNEXIT METHOD TO DISMISS THE MODAL WINDOW

```
import UIKit

class GreenViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()

        // Do any additional setup after loading the view.
    }

    @IBAction func btnExit(_ sender: Any) {
        self.dismiss(animated: true, completion: nil)
    }
}

```

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



STEP 20: RUN IT!



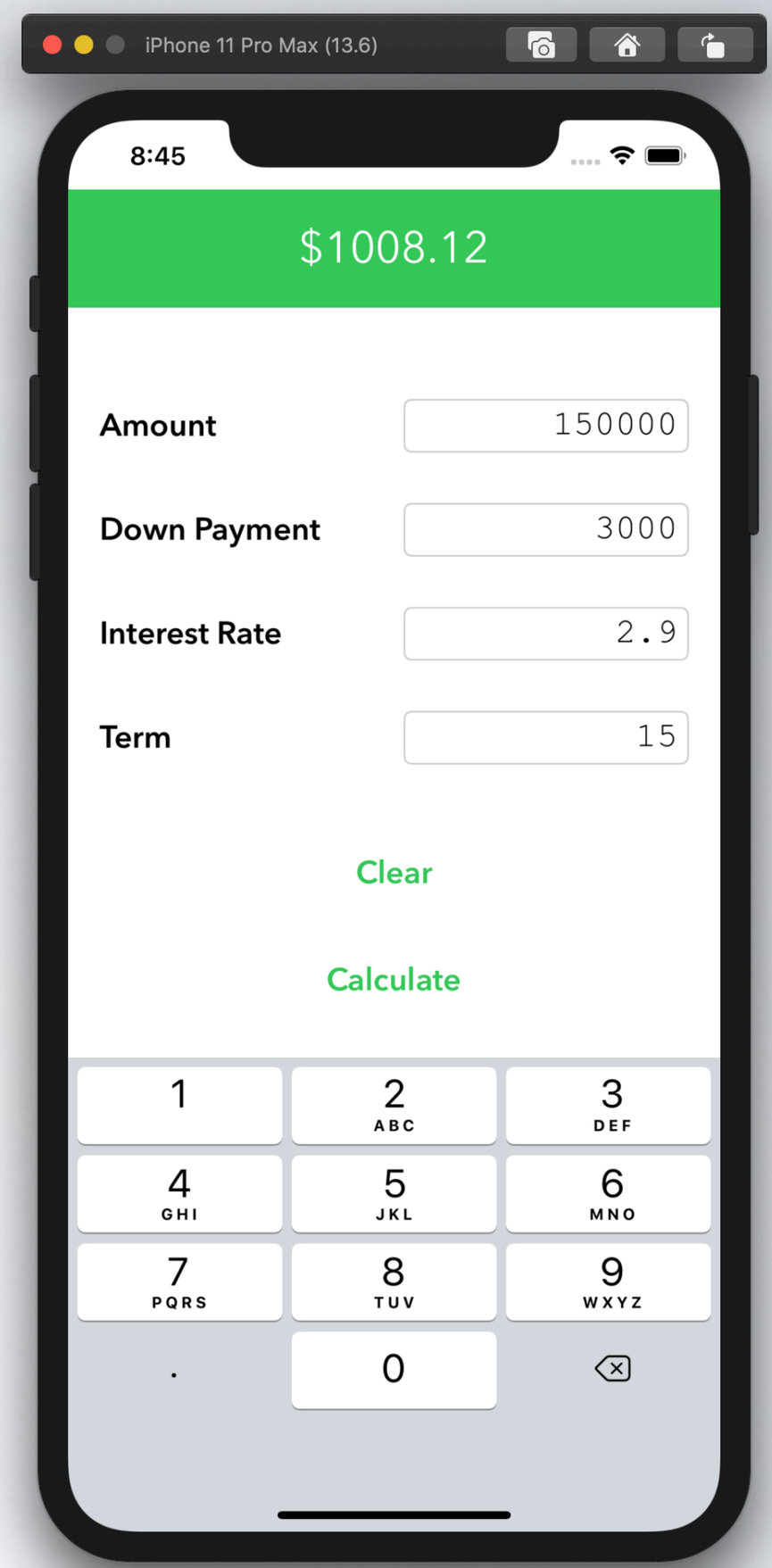
Lab #4 - Create Amortize Payment App

GroundSpeed™
rapid web + mobile software

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

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

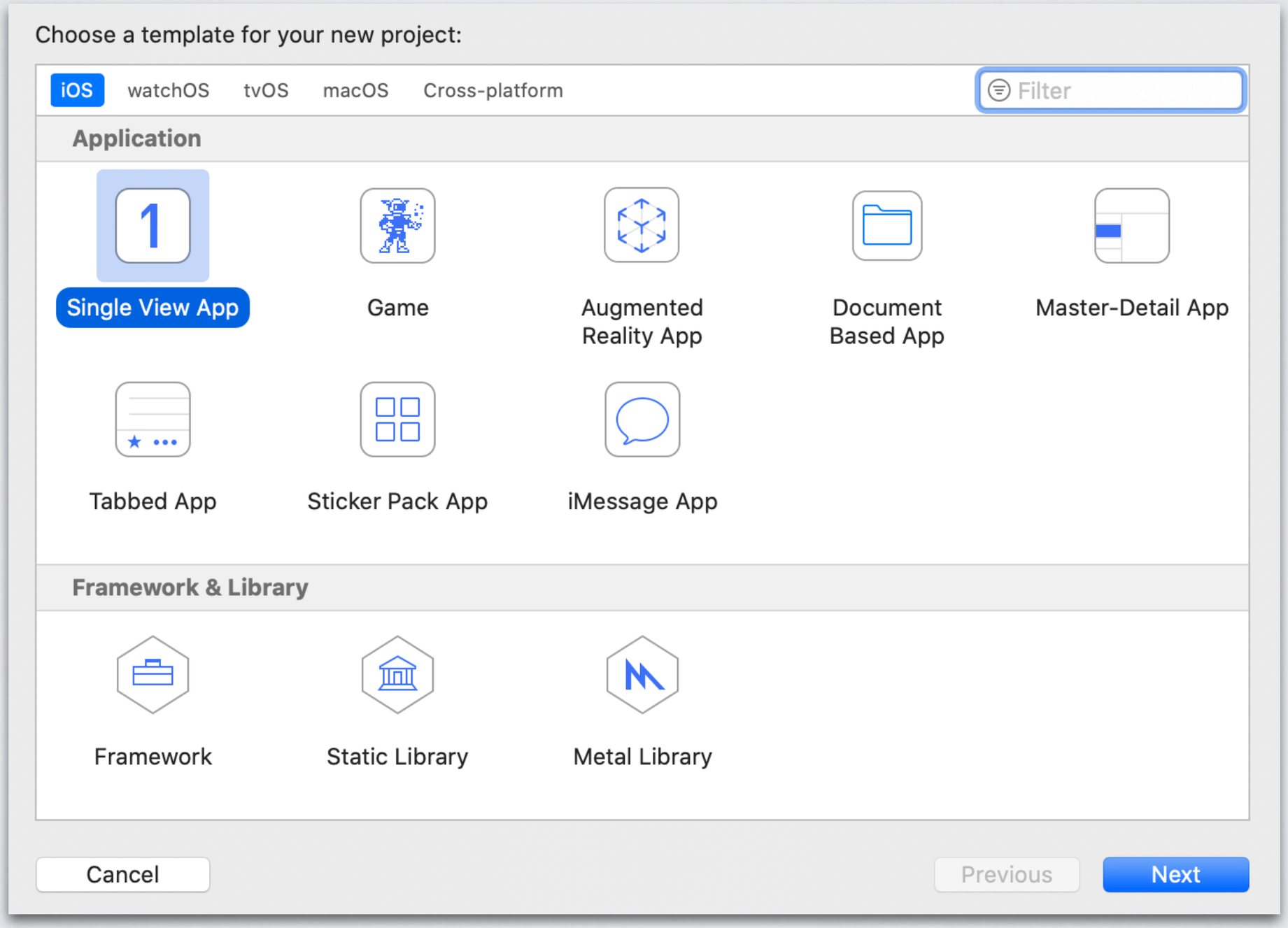


$$P = A \cdot \frac{1 - \left(\frac{1}{1+r}\right)^n}{r}$$

```
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

}
```

Tab Bar and Navigation Controllers

CREATE A NEW APPLICATION



CREATE A NEW APPLICATION

Choose options for your new project:

Product Name: NavDemo

Team: None

Organization Name: GroundSpeed

Organization Identifier: com.groundspeedhq

Bundle Identifier: com.groundspeedhq.NavDemo

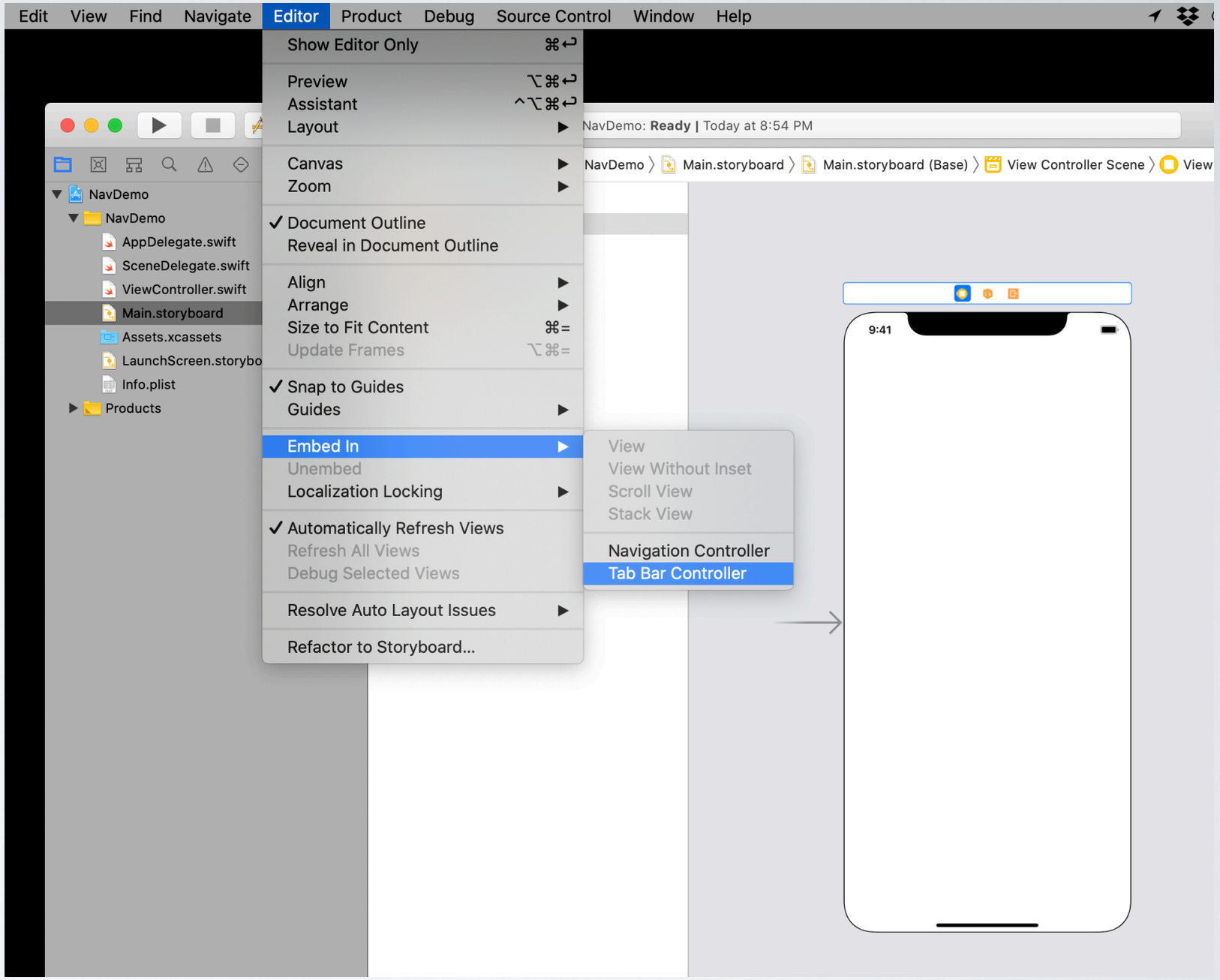
Language: Swift

User Interface: Storyboard

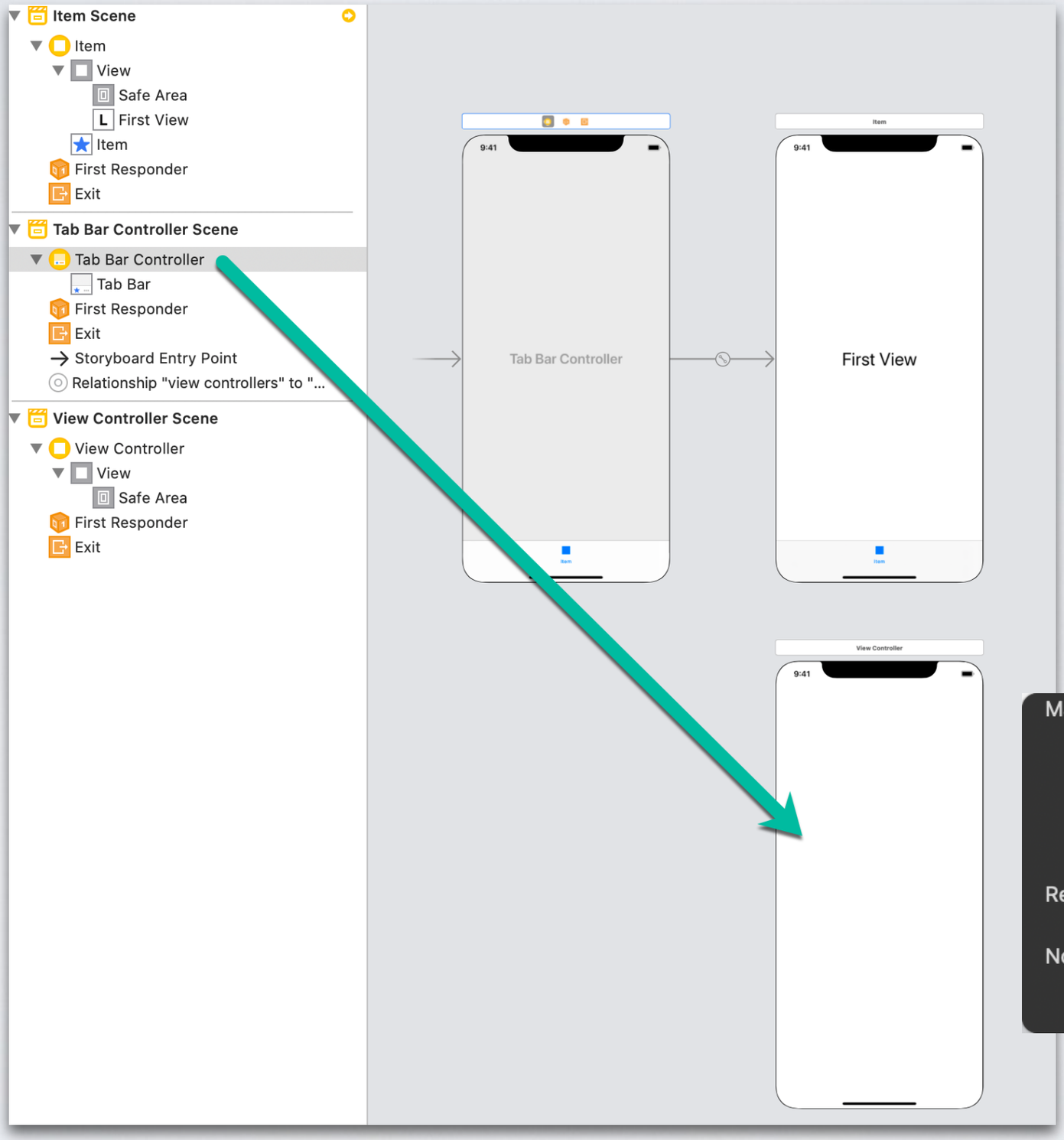
- [ ] Use Core Data
  - [ ] Use CloudKit
- [ ] Include Unit Tests
- [ ] Include UI Tests

Cancel | Previous | Next

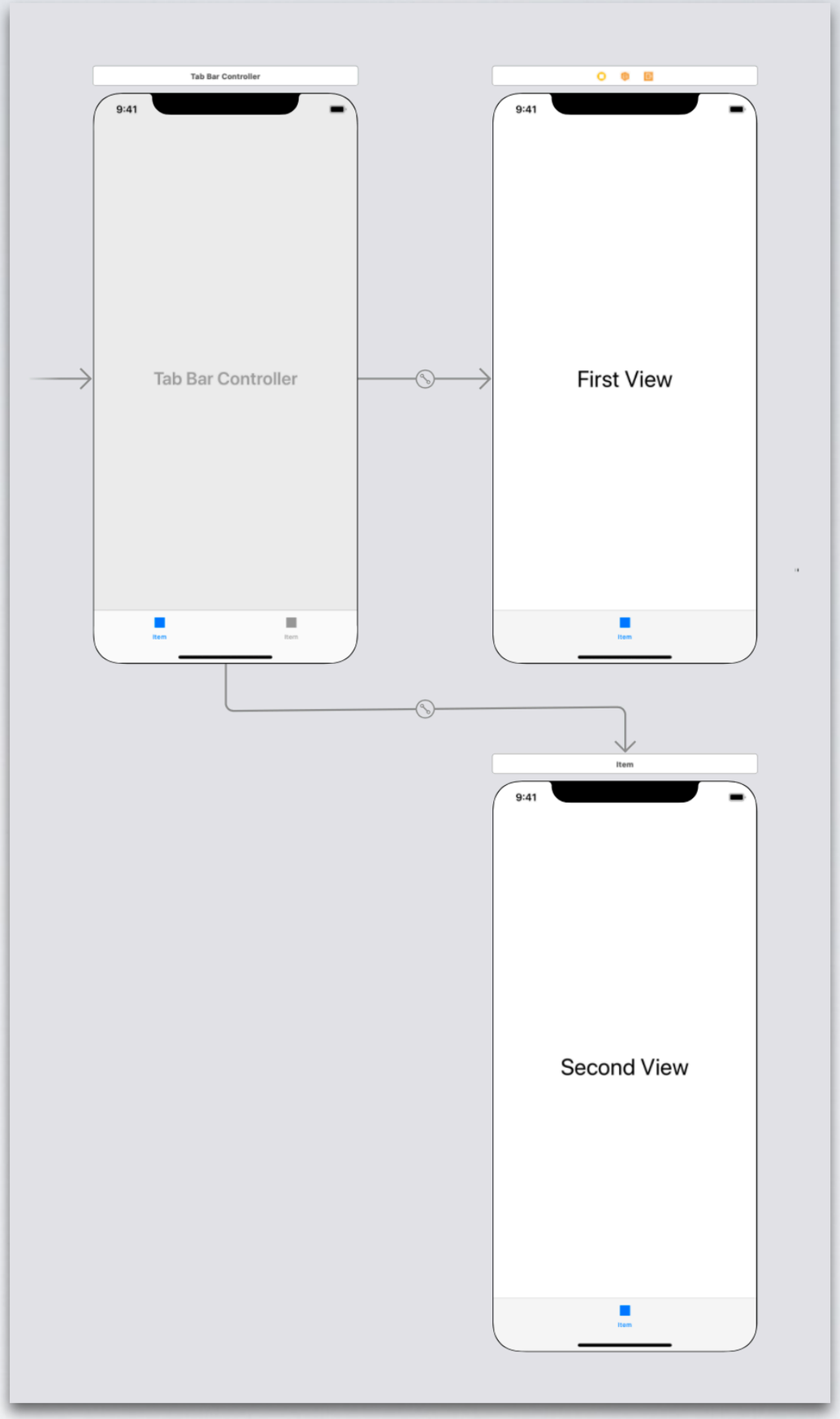
INSIDE THE STORYBOARD, LET'S EMBED OUR INITIAL CONTROLLER WITHIN A TAB BAR CONTROLLER



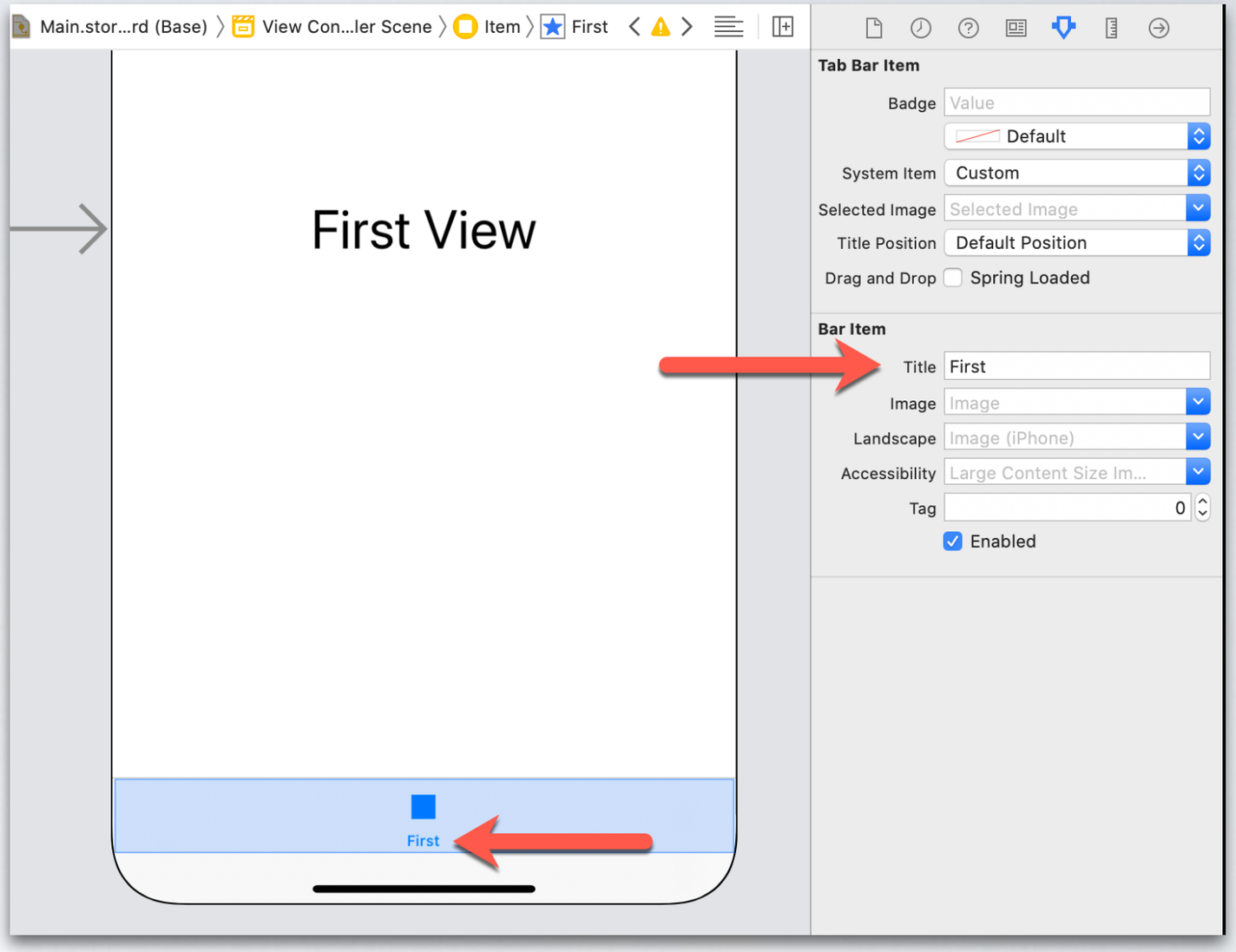
ADD A SECOND VIEWCONTROLLER. THEN HOLD CONTROL BUTTON AND DRAG FROM TAB BAR CONTROLLER TO THE NEW VIEW CONTROLLER



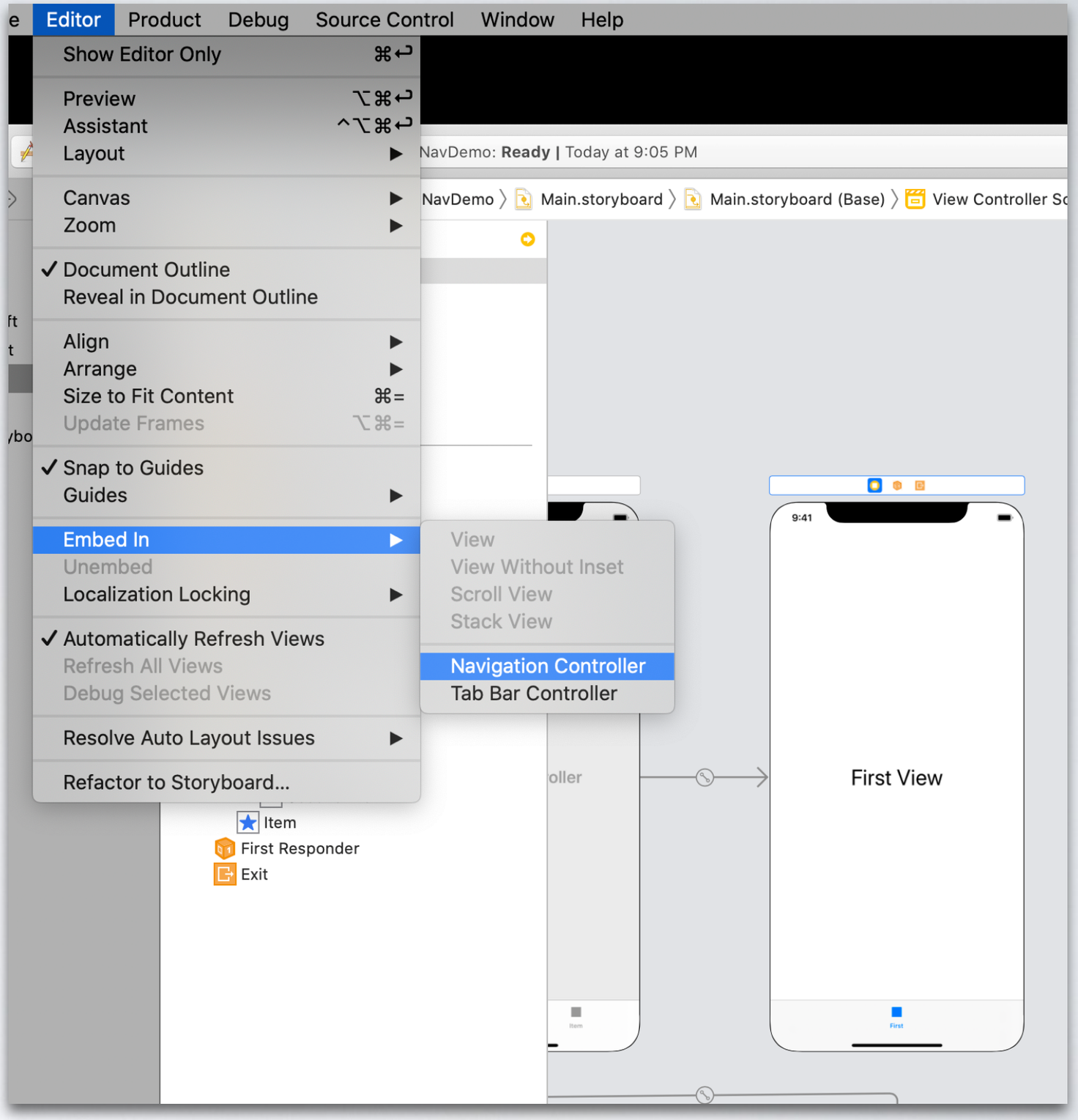
YOUR STORYBOARD SHOULD LOOK LIKE THIS…



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

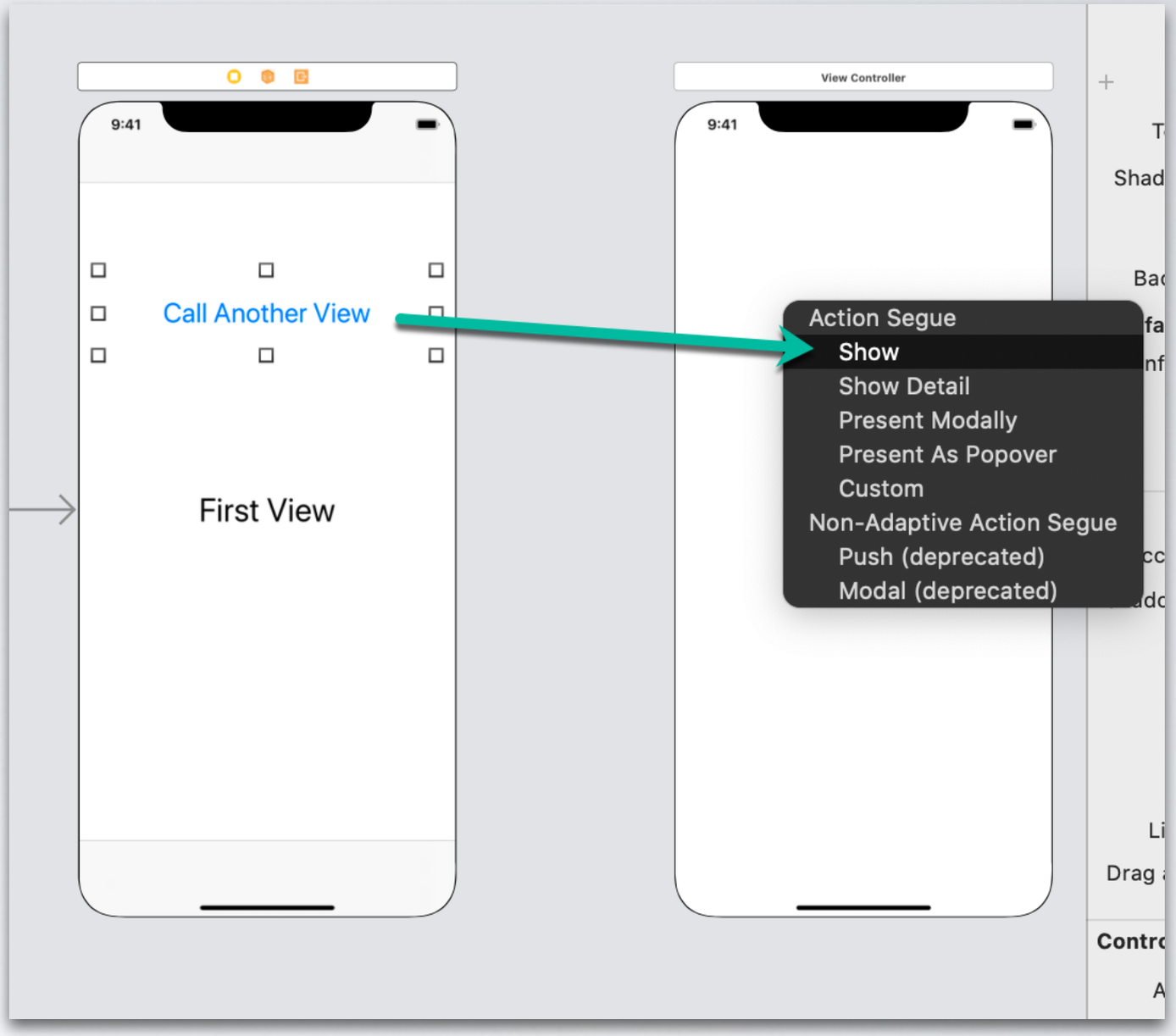


THE TAB VIEW CAN CALL ANOTHER FORM BY EMBEDDING IT INTO A NAVIGATION CONTROLLER

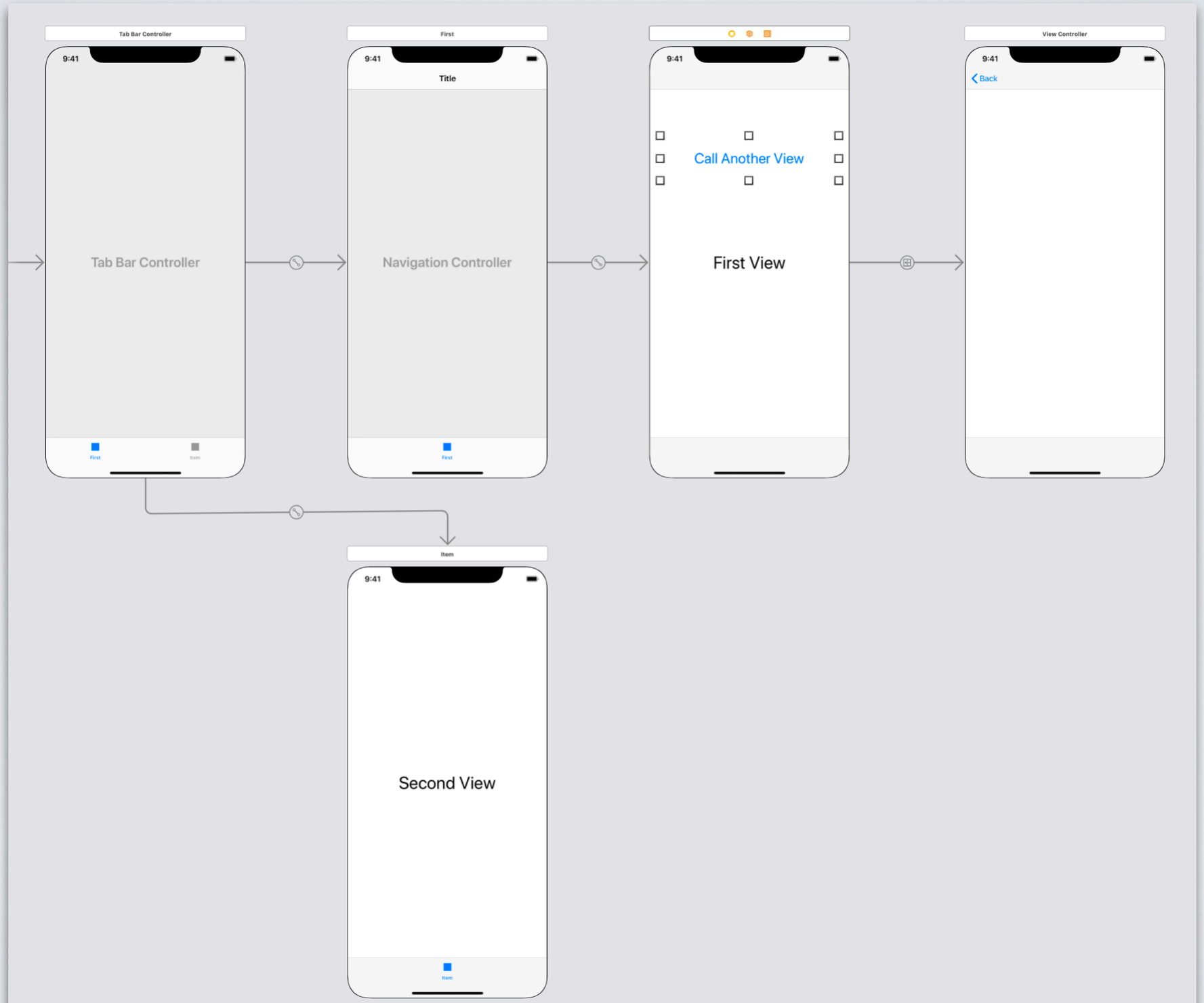


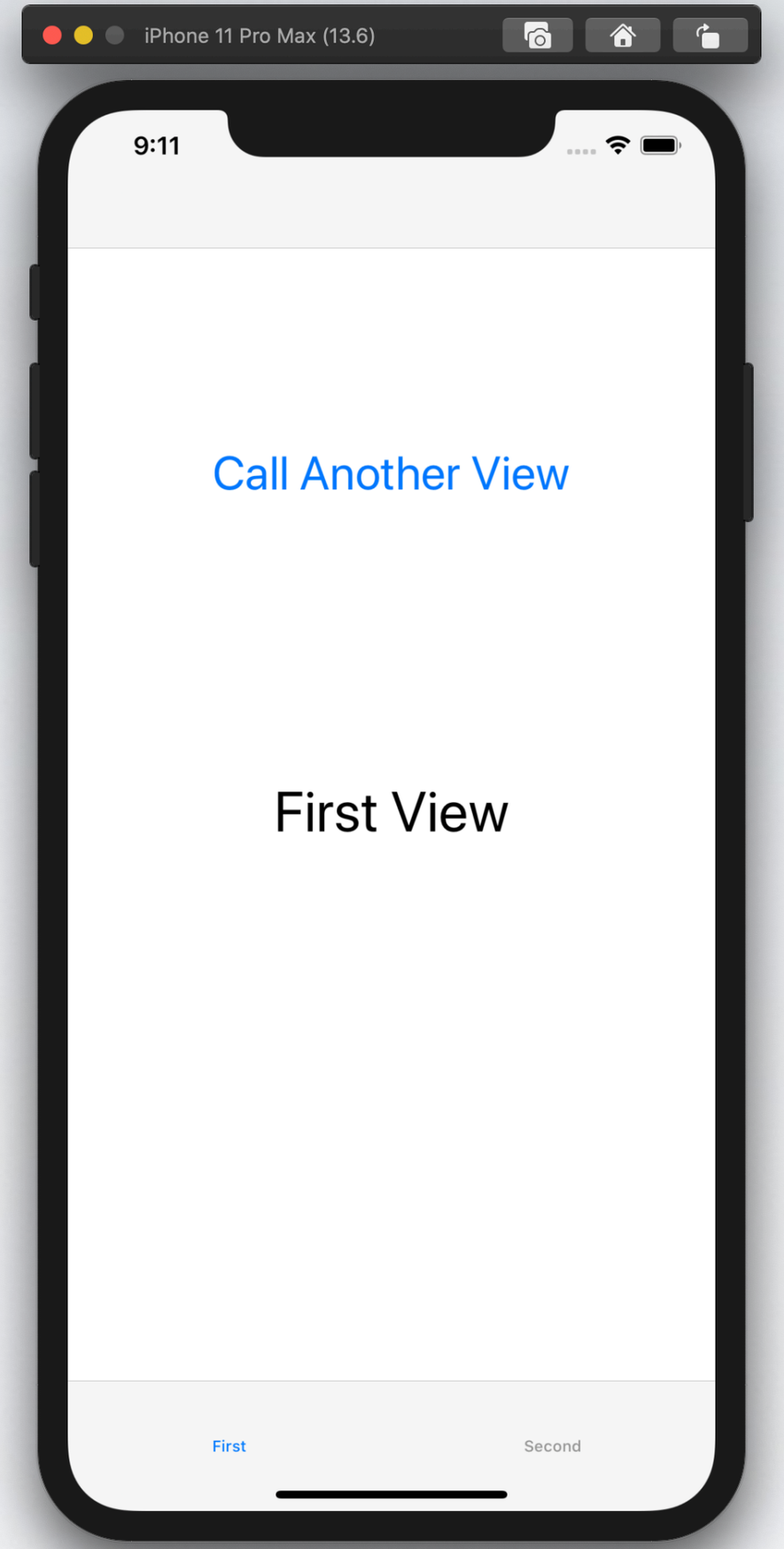
GroundSpeed™
rapid web + mobile software

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



STORYBOARD VIEW



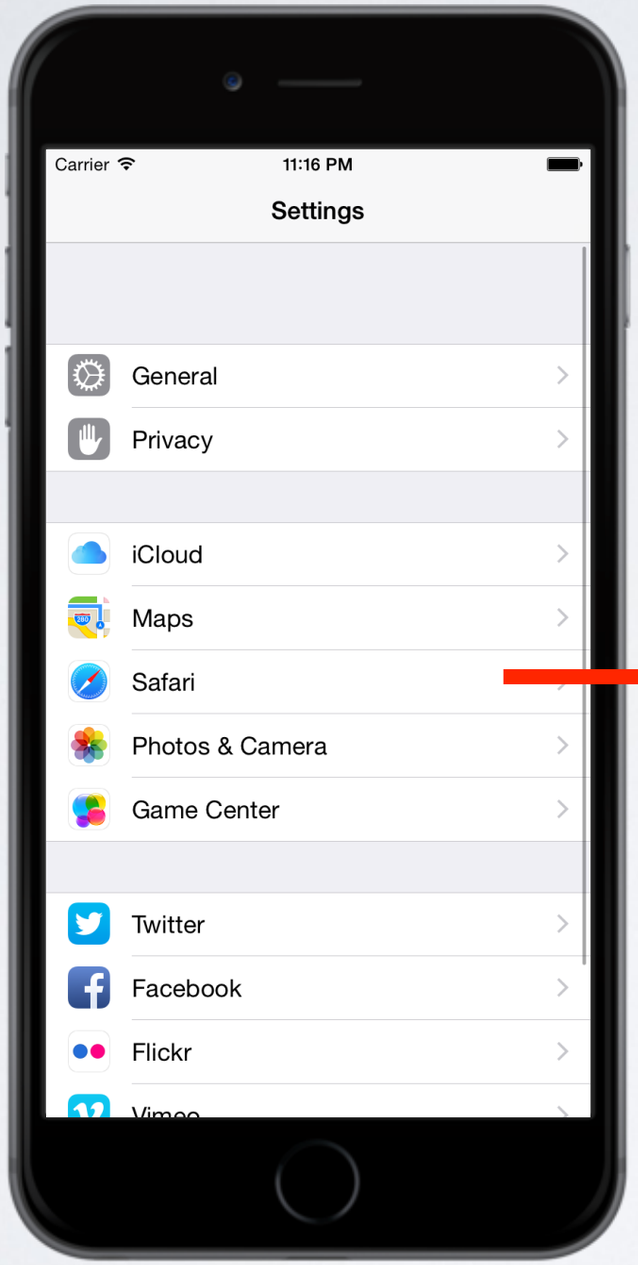


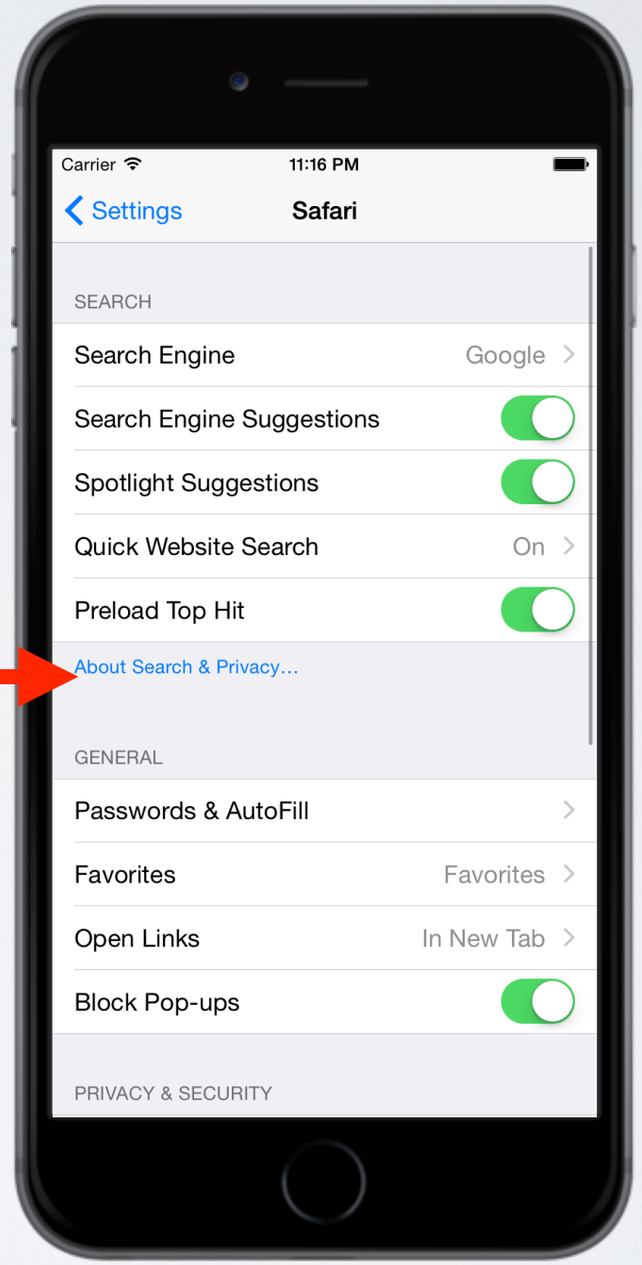
Lab #5 - Convert Payment App to Tabbed View

UITableViews

GroundSpeed™
rapid web + mobile software

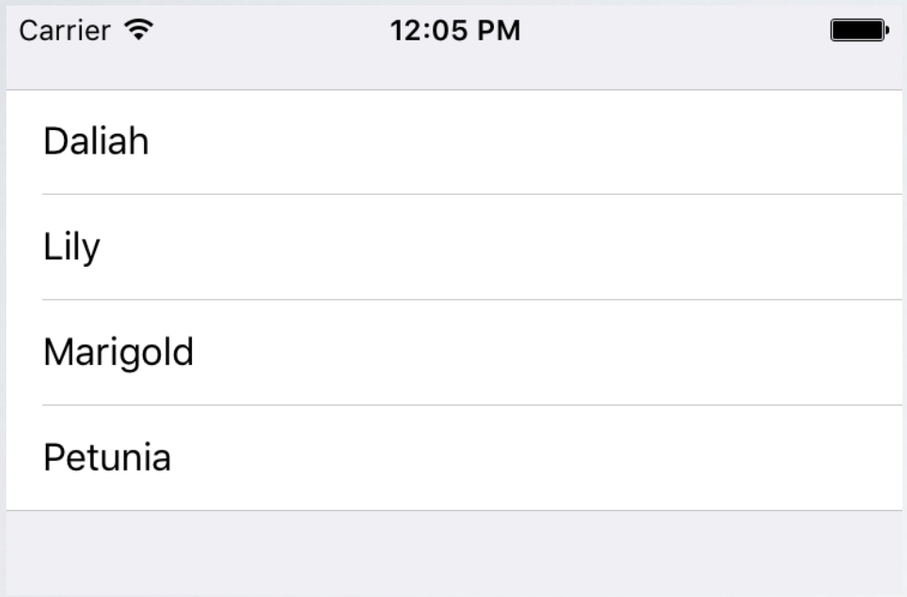
UITABLEVIEW OVERVIEW





UITABLEVIEW OVERVIEW

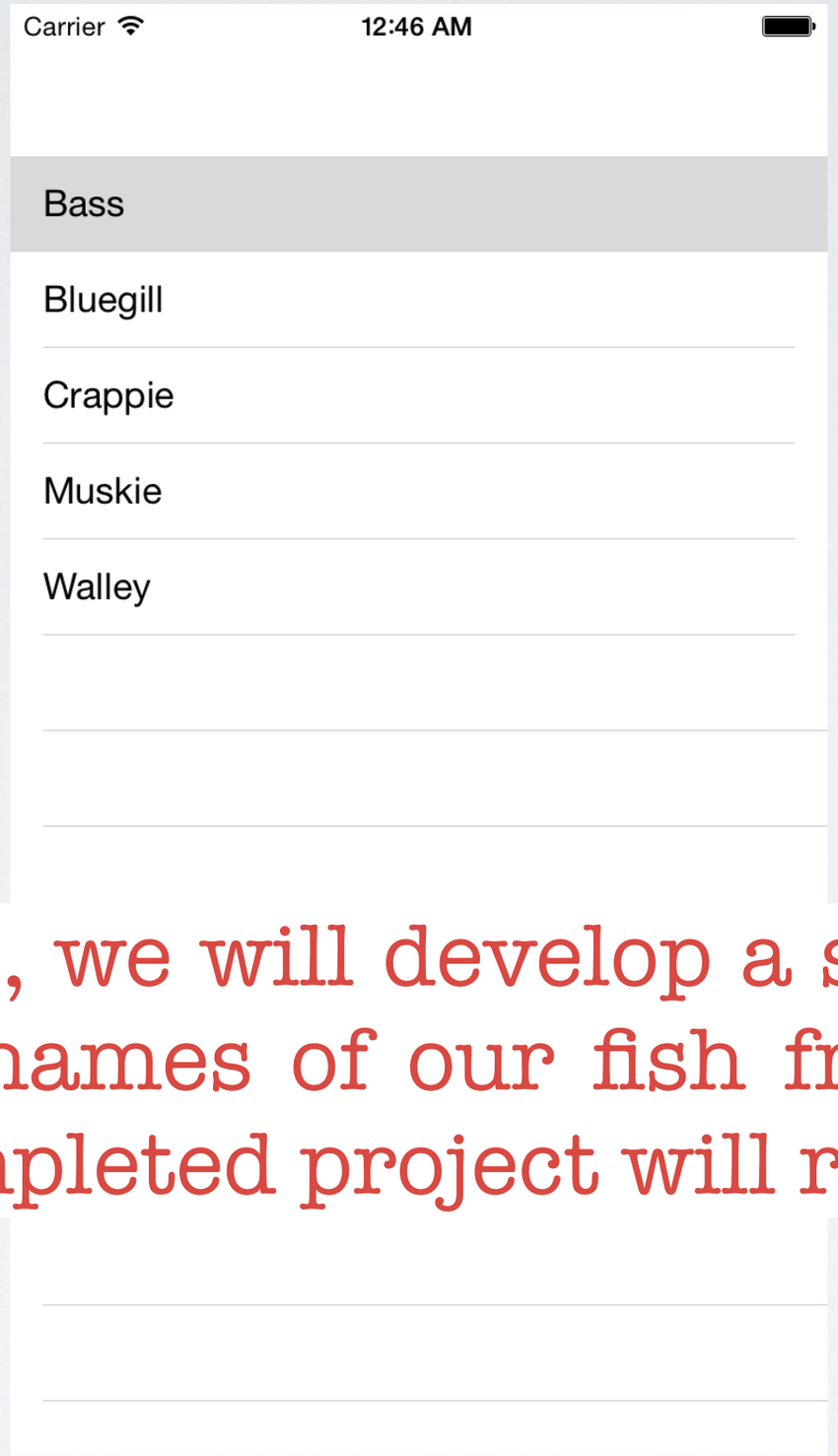
A Simple UITableView with 4 Rows



A Sectioned UITableView

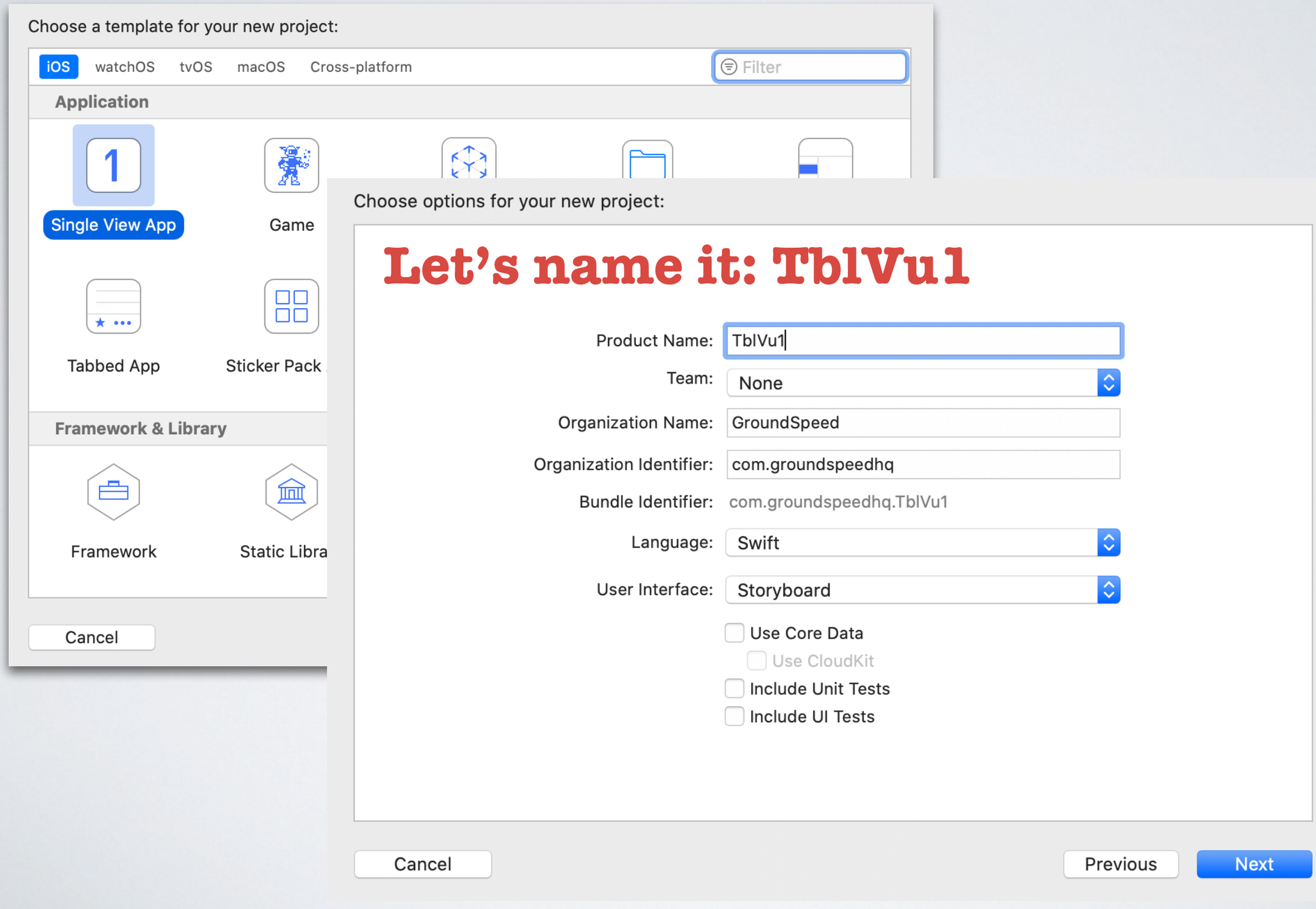
- Programmatically, the UIKit identifies rows and sections through their index number.
- Sections are numbered 0 through n-1 from the top of a table
- Rows are numbered 0 through n-1 within a section
- As Mentioned earlier, the table view comes in one of two basic styles: plain or grouped.

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.

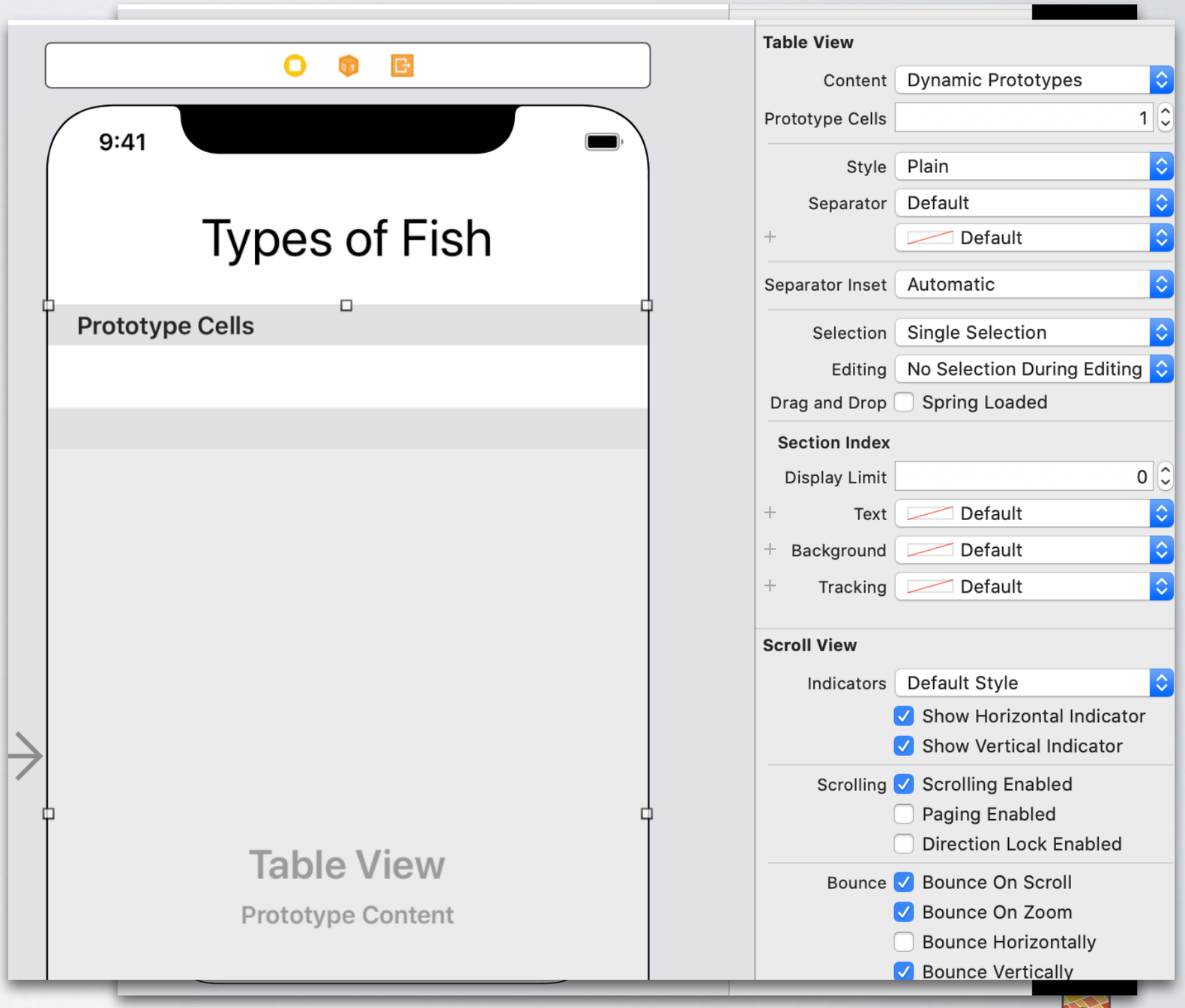
CREATE A SINGLE VIEW BASED APP.



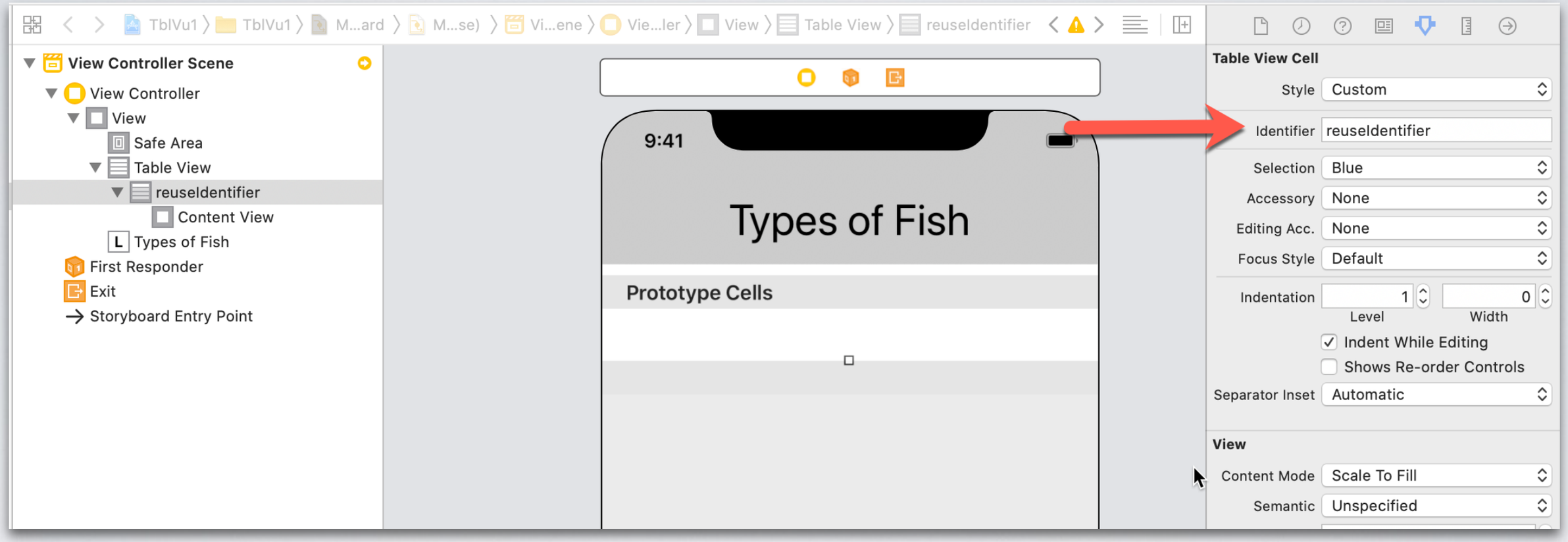
GroundSpeed™
rapid web + mobile software

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**

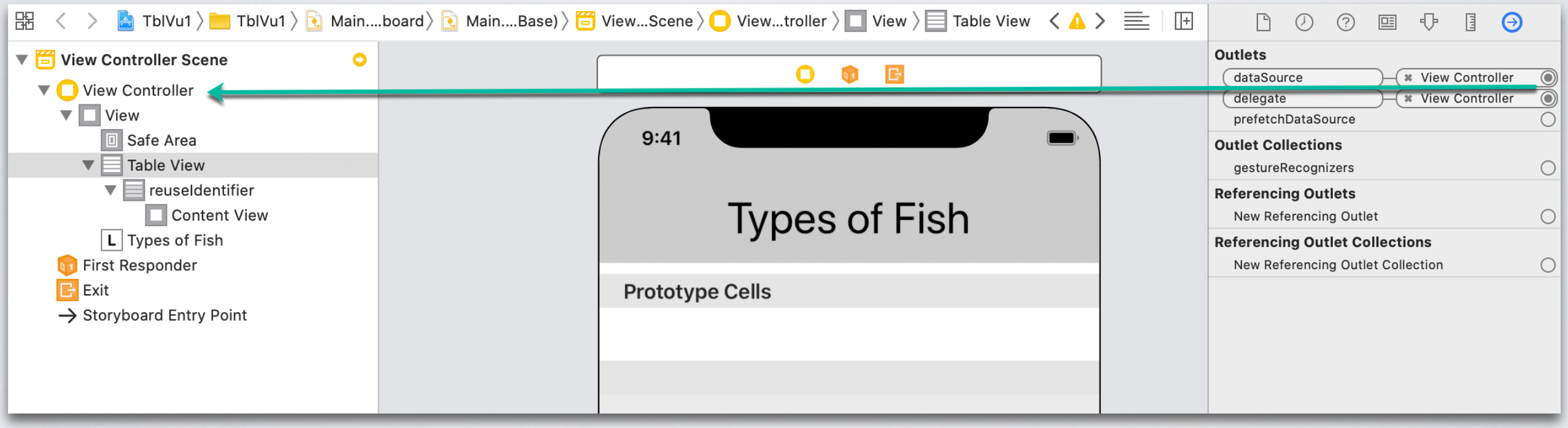


UPDATE PROTOTYPE CELL WITH REUSE IDENTIFIER



DATASOURCE & DELEGATE

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



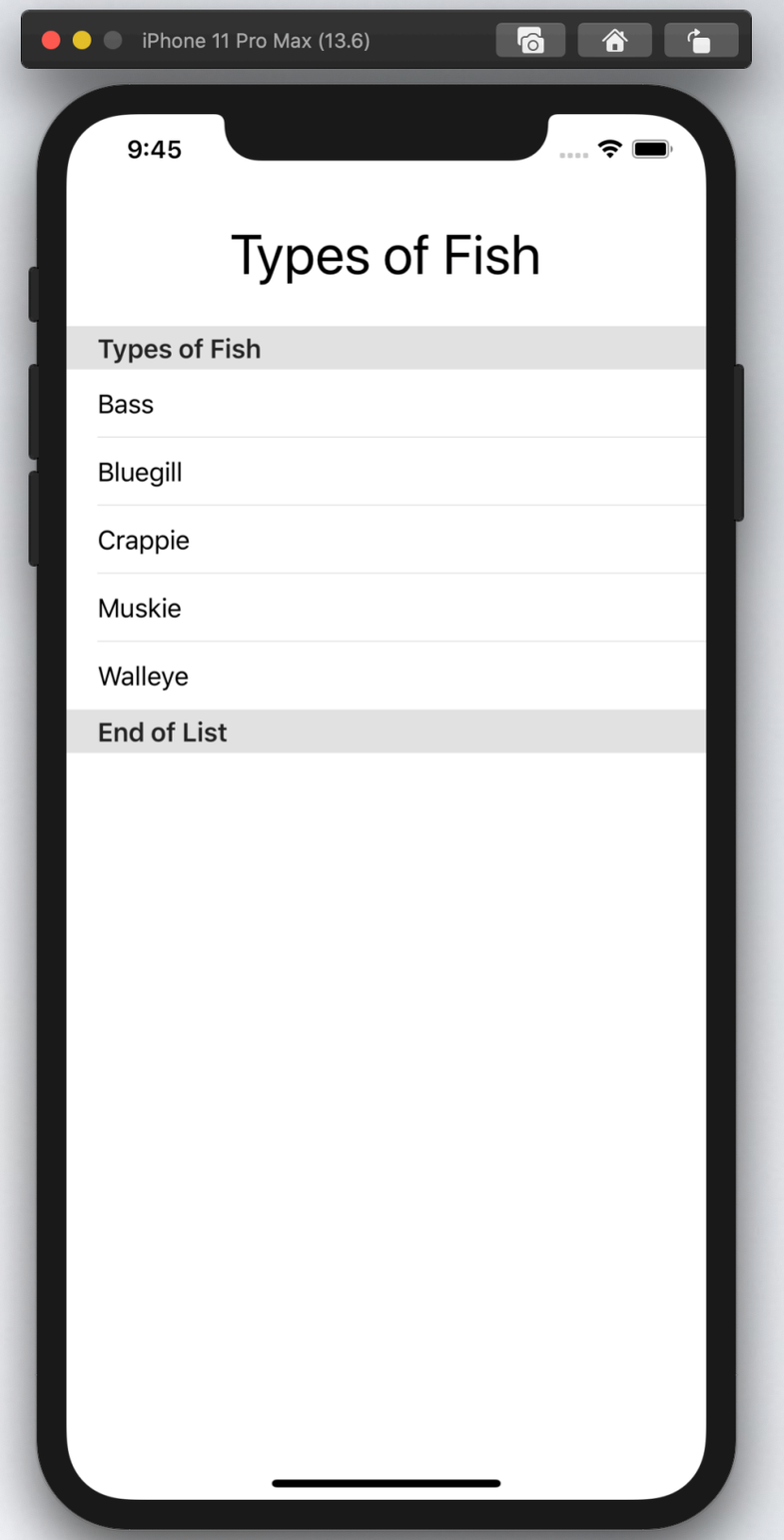
OPEN THE VIEWCONTROLLER.SWIFT FILE AND INSERT THE FOLLOWING CODE

```
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!

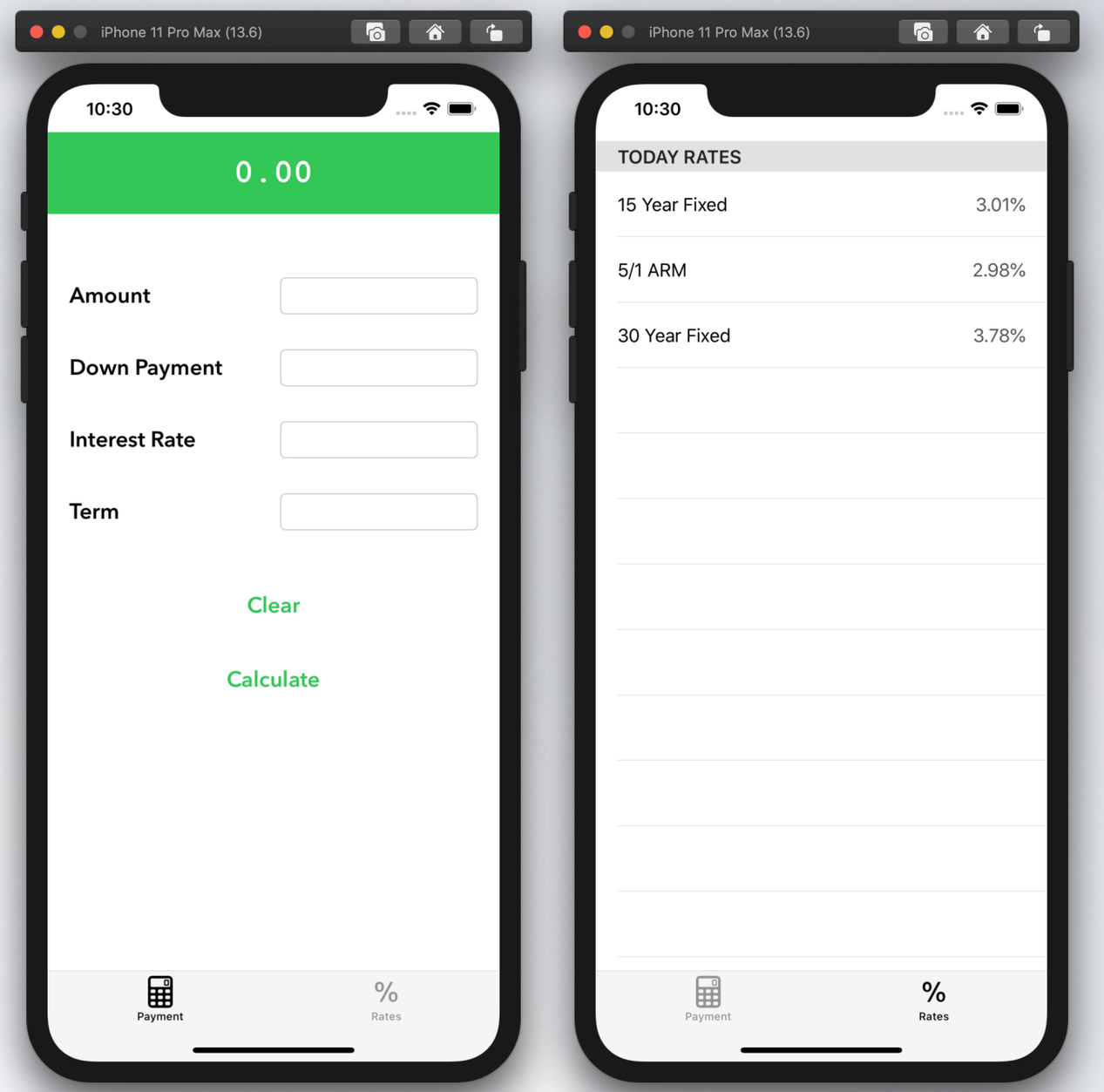


GroundSpeed™
rapid web + mobile software

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

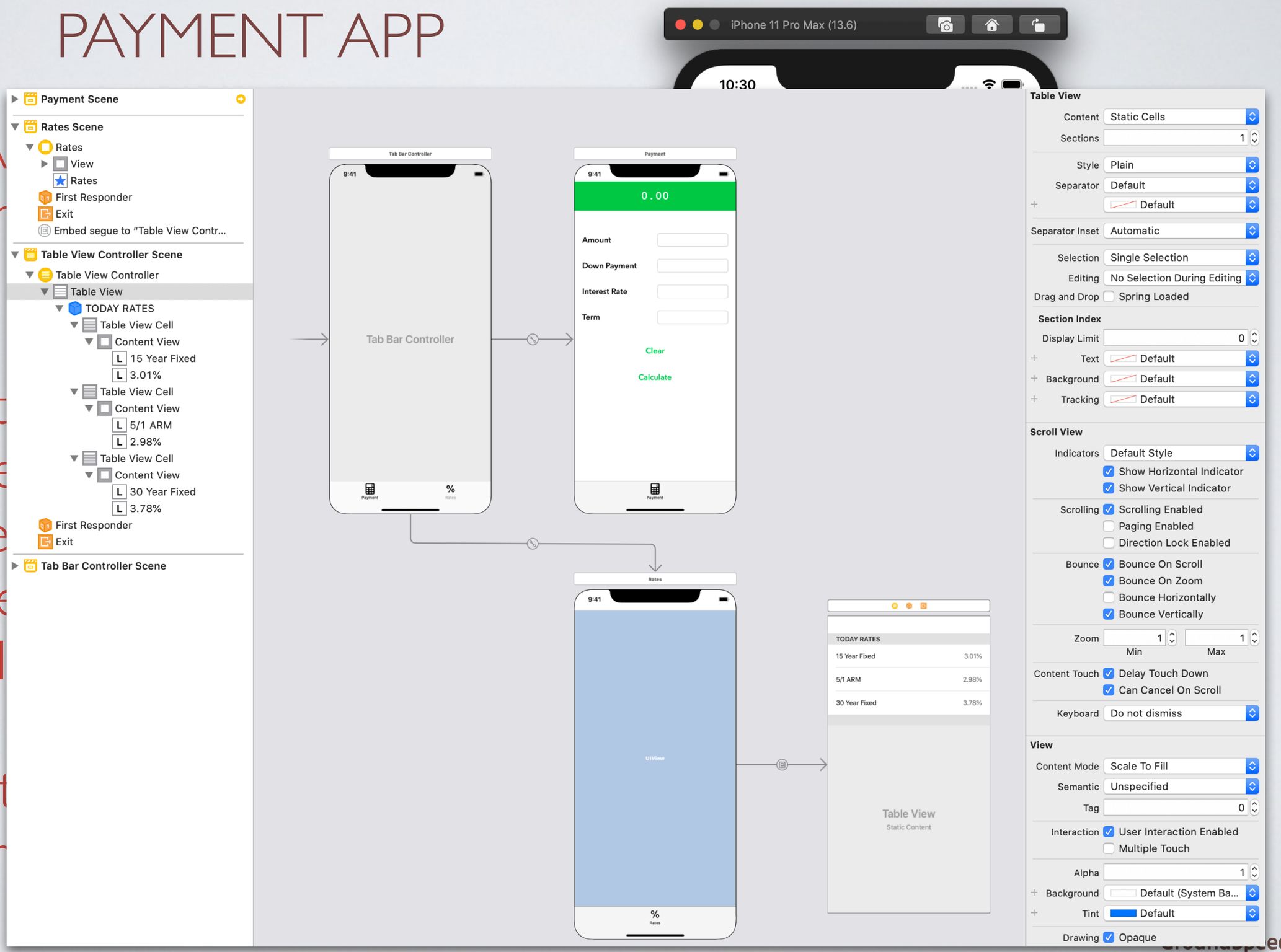


Download Images At:
http://sl.gshq.co/percent
http://sl.gshq.co/calculator

GroundSpeed™
rapid web + mobile software

ADD A TAB BAR CONTROLLER TO THE AMORTIZATION PAYMENT APP

1. Open A
2. Select th
3. Change
   cells
4. Set Sect
5. Set Style
6. Add the
   of the se
7. Create I
   to each
   the right
8. Add sor



Lab #7 - GitHub Walkthough

Don Miller

don@GroundSpeedHQ.com

@donmiller

http://github.com/donmiller

http://github.com/groundspeed

GroundSpeed™
rapid web + mobile software