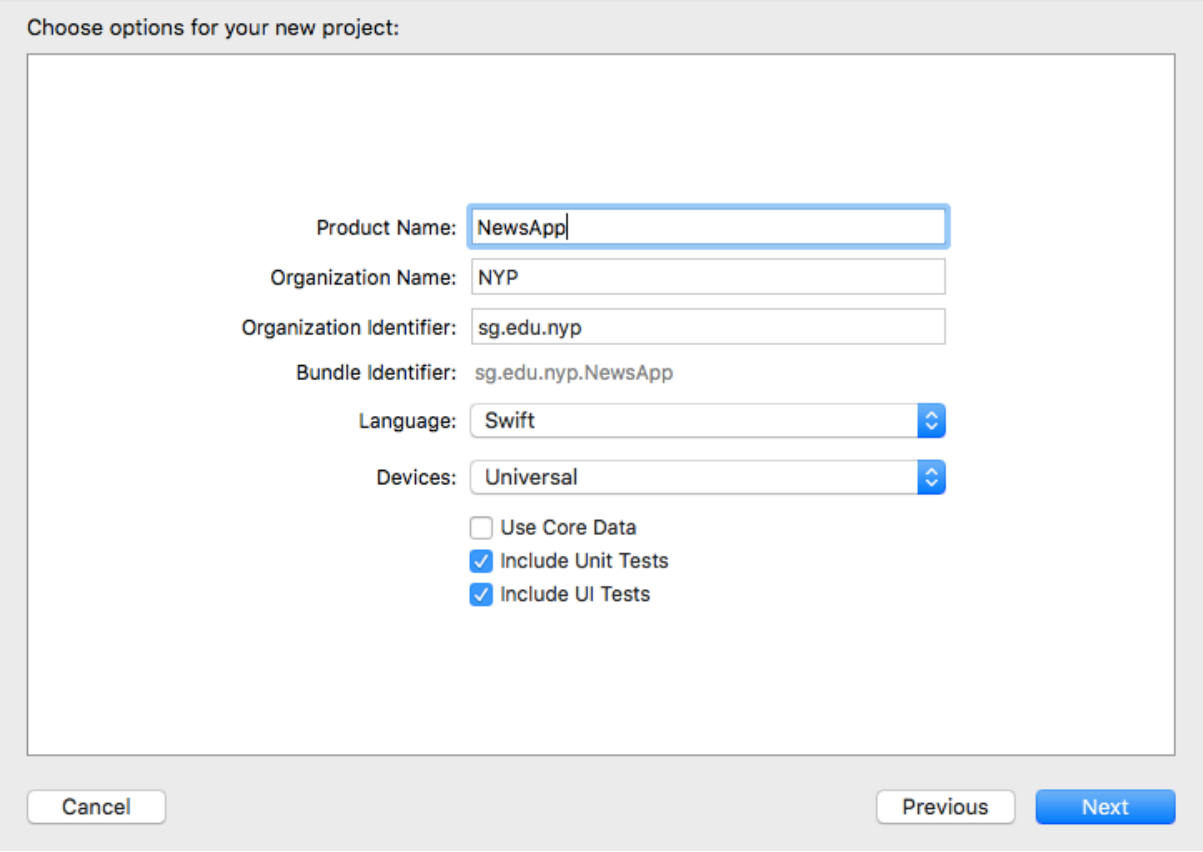
Practical 9: Working with JSON Web Service and SplitViewController

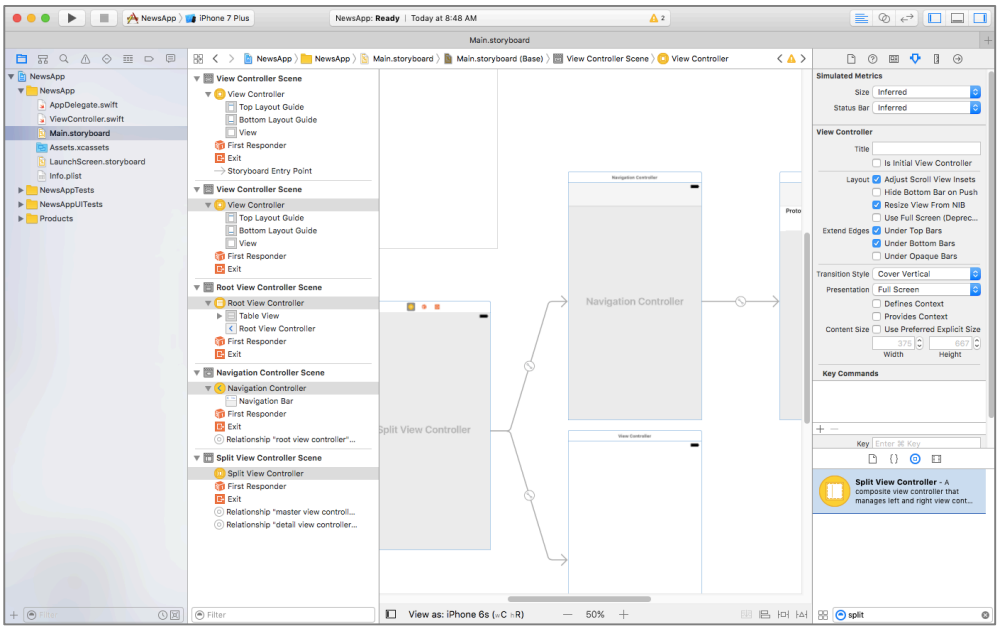
In this lab, we will learn how to consume a web service that provides information about iPhone 6 and returns JSON formatted data from within the iOS application. In the past, when we parse JSON in iOS apps, we used a third party library such as JSON framework. Since iOS5, Apple has added a JSON library in Cocoa, making it convenient for us, developers. We will learn to present its view controller using a split view controller.

Section 1: Create a Split View Controller

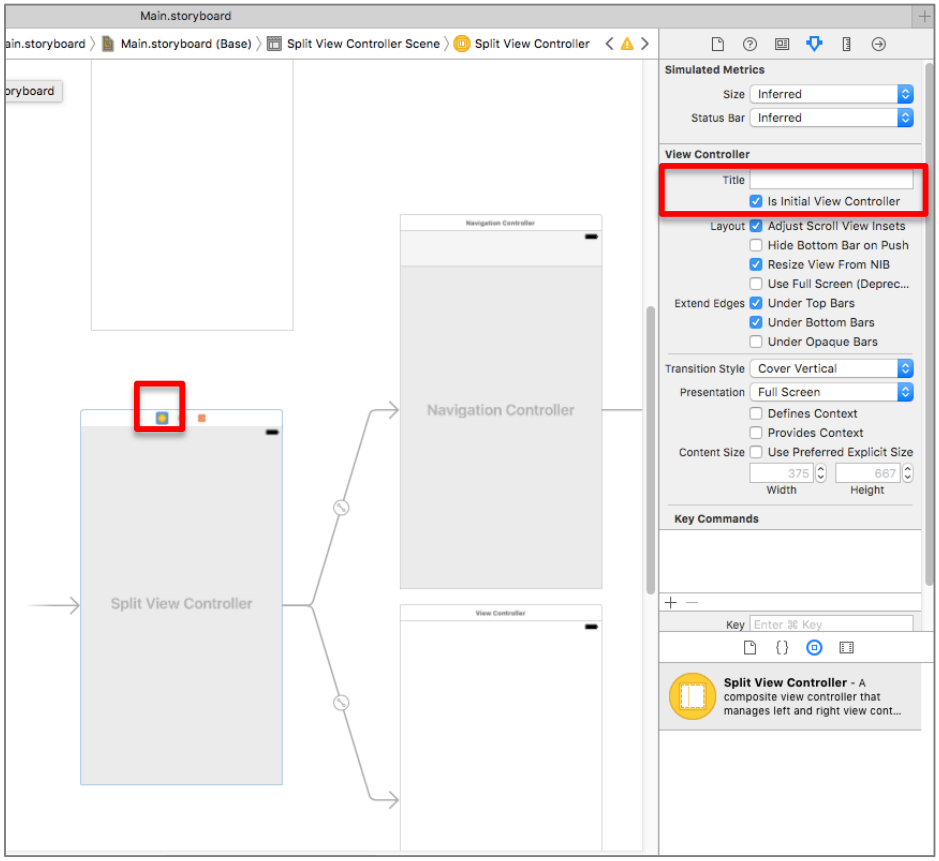
1. Using Xcode, click on **Create a new Xcode project** and select Single View Application.
2. Next, configure this project and name is as **NewsApp**. For device, choose **Universal**.



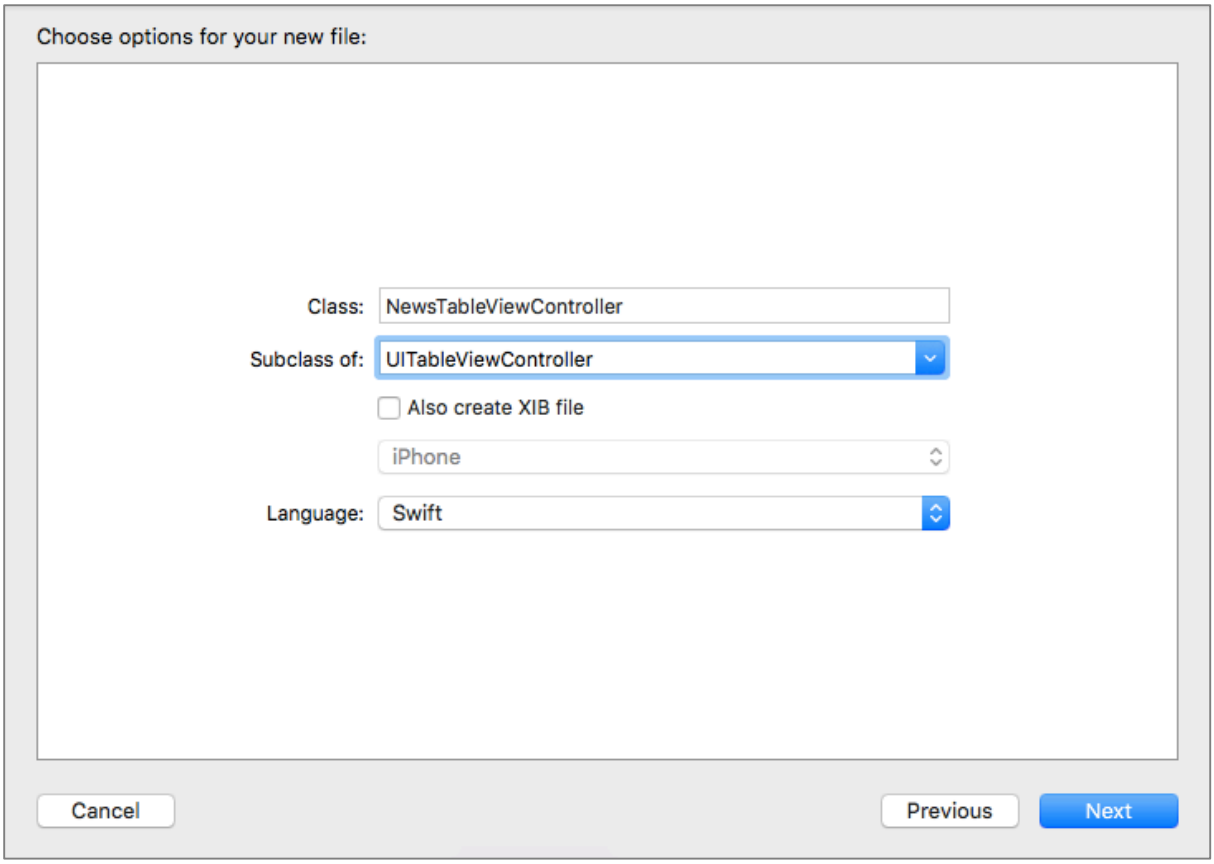
3. Open the **Main.storyboard** file and drag and drop a **SpiltViewController** into the storyboard.



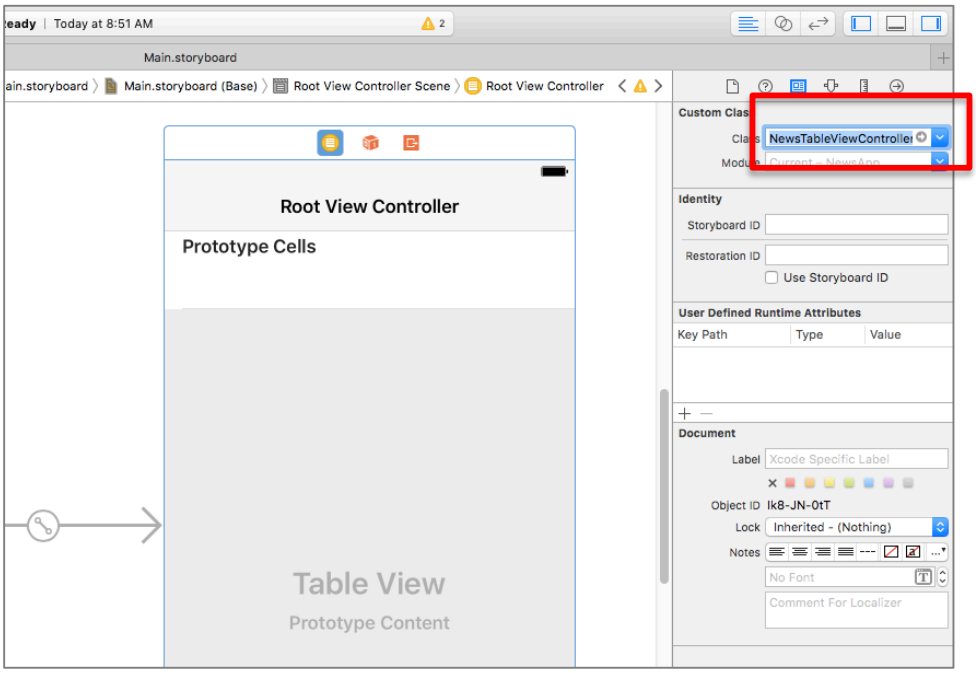
4. Select the **Split View Controller** and check the **Is Initial View Controller** at the Attribute Inspector.



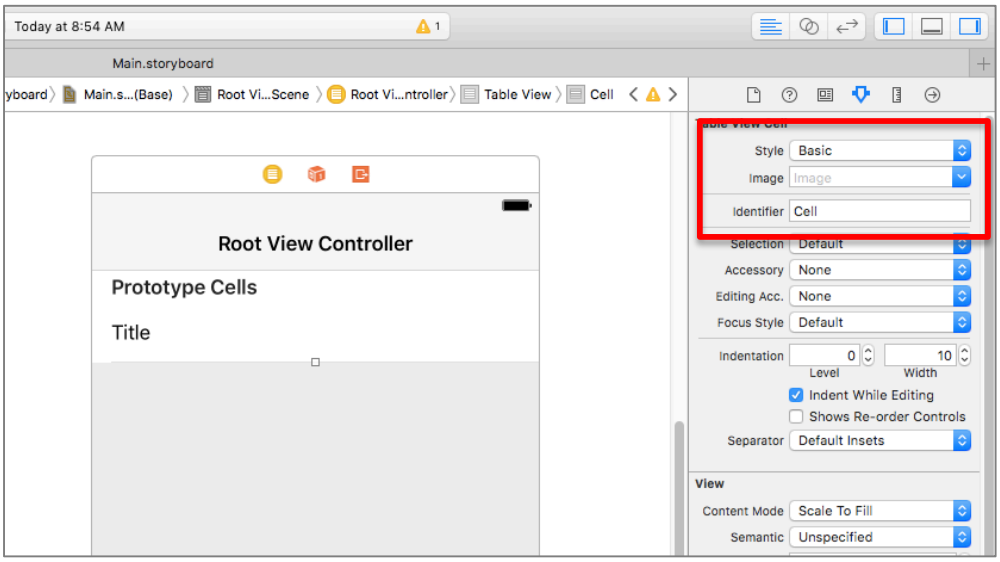
5. Delete the original View Controller from the interface builder and delete the **ViewController.swift** from the project navigator.
6. Right-click on the project folder and select Add File. Choose the **Cocoa Touch** Class.
7. Select **UITableViewController** from the drop down list and enter **NewsTableViewController** as the name of the class.



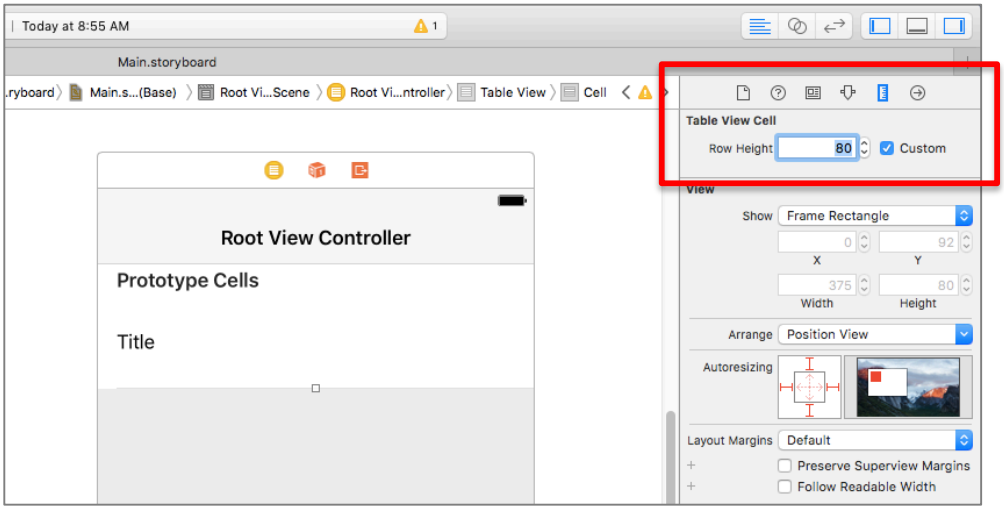
8. Open **Main.storyboard** and select the Root View Controller. Change the class under Custom Class to **NewsTableViewController** using the identity inspector.



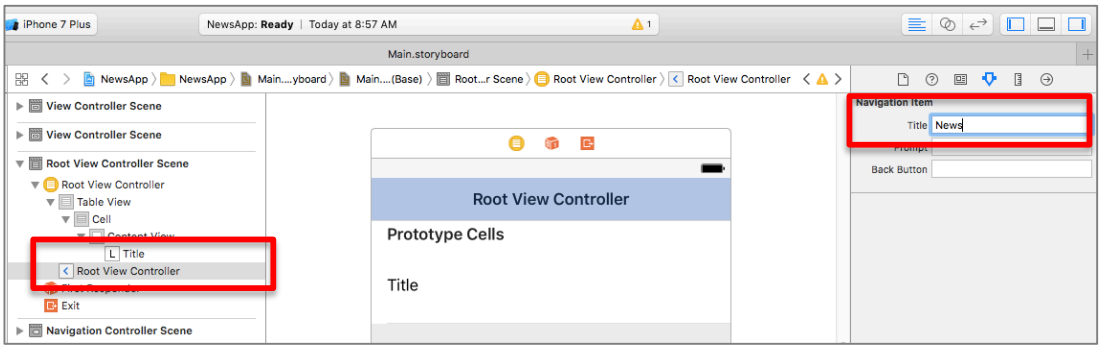
9. Next, to make sure the Prototype Cell in the Table View is given a reuse identifier, select the Prototype Cell and then change the Identifier to "**Cell**" and change the cell style to "**Basic**".



10. Click the Size Inspector, set **Custom** for the Row Height, and set the height of the cell to **80**.



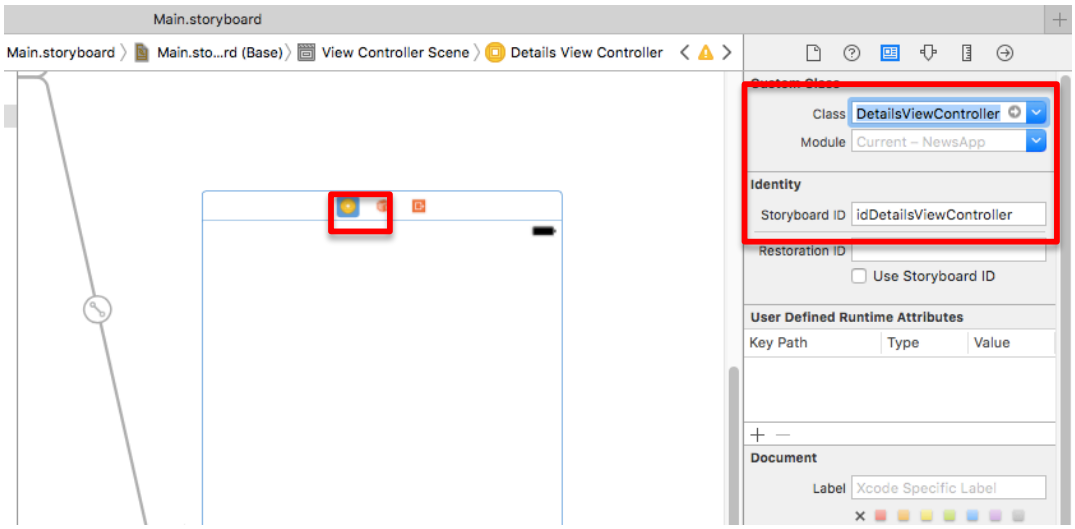
11. From the outline view, select the Root View Contoller and at Attribute Inspector, change the title to News.



12. Right-click on the project folder and select Add File. Choose the **Cocoa Touch** class.

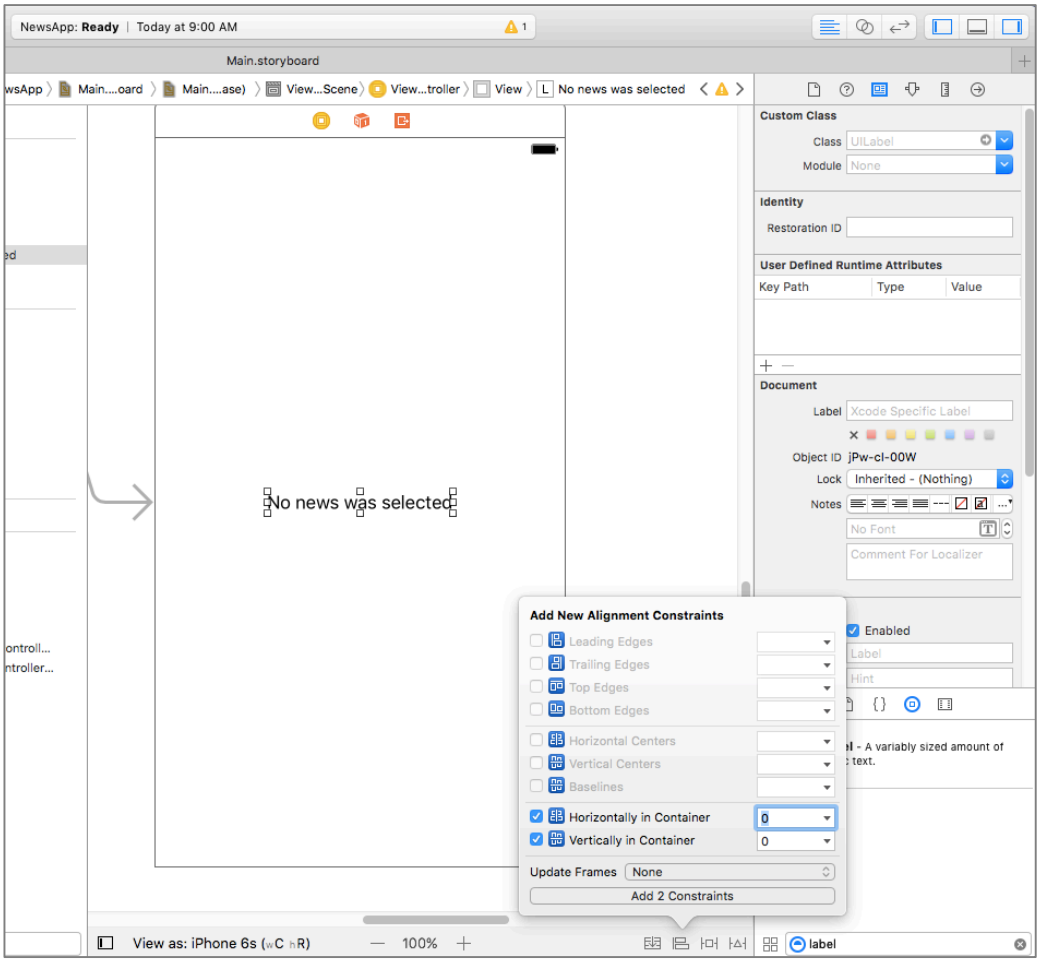
13. Select **UIViewController** from the drop down list and enter **DetailsViewController** as the name of the class.

14. Open **Main.storyboard** and select the View Controller. Change the class under Custom Class to **DetailsViewController** using the identity inspector and **idDetailsViewController** as the Storyboard ID.



15. From the Object Library, drag and drop a **Label** and type in “No news was selected” and make sure they are placed in the center of the view.

16. Click on the **Align** button at the bottom-right side of the Interface Builder, check both the **Horizontal center in container** and **Vertical center in container** constraints, and click **Add 2 Constraints**.

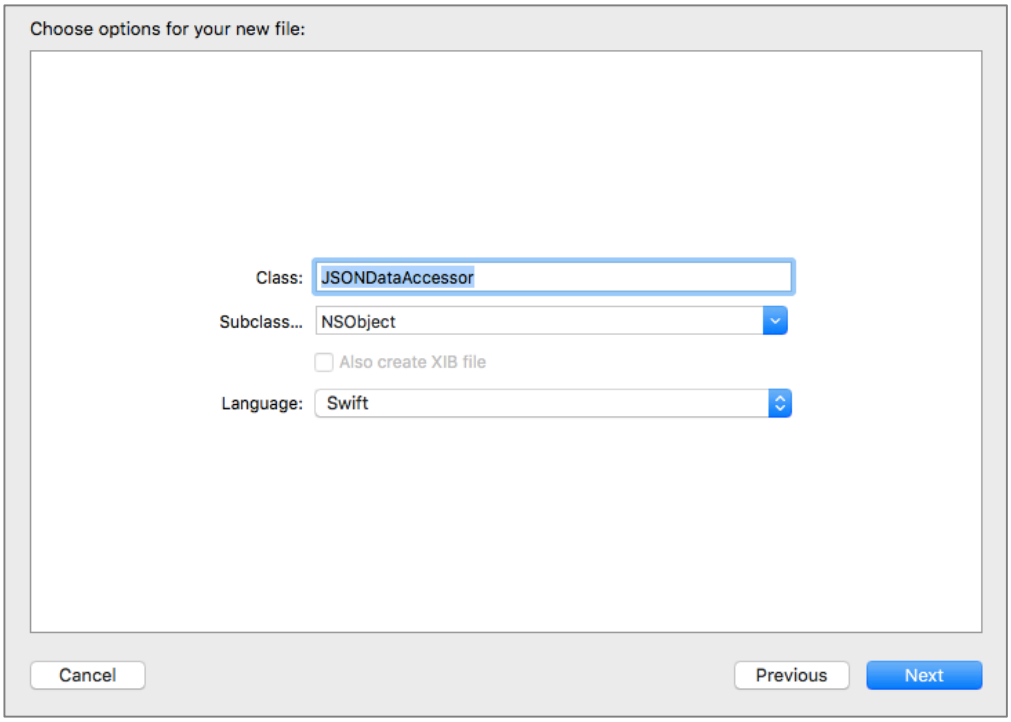


17. From the Object Library, drag and drop a **Web View**. Click on the Pin button and enable the trailing, leading, top and bottom constraints.

Section 2: Parsing JSON from the Web

*In this section, we will be populating data to the Table View using **UITableViewDataSource**. We will be consuming a web service that provides the information about iPhone 7 from Google Apis. But to simplify our task of reading results from the JSON object, we are going to implement a very simple wrapper.*

18. Right-click on your **NewsApp** folder and add a new **Cocoa Touch** class.
19. Name the class **JSONDataAccessor**, and the subsclass as **NSObject**.



20. Add the following codes into your JSONDataAccessor class:

```
//
//  JSONDataAccessor.swift
//  NewsApp
//
//  Created by KIM FOONG CHOW on 28/9/16.
//  Copyright © 2016 NYP. All rights reserved.
//

import UIKit

class JSONDataAccessor : NSObject
{
    var data : Any

    init (_ data: Any)
    {
        self.data = data
    }

    // Allows the developer to access a dictionary property
    // using an indexer '[...]'
```

```
    //
    subscript (_ key: String) -> JSONDataAccessor?
    {
        get
        {
            let dictionary = data as? [String: Any]
            if dictionary != nil
            {
                return JSONDataAccessor(dictionary![key])
            }
            else
            {
                return nil
            }
        }
    }

    // Allows the developer to access an array
    // using an indexer '[...]'
    //
    subscript (_ index: Int) -> JSONDataAccessor?
    {
        get
        {
            let array = data as? [Any]
            if array != nil
            {
                return JSONDataAccessor(array![index])
            }
            else
            {
                return nil
            }
        }
    }

    // Counts the number of items in a JSON array
    //
    var count : Int
    {
        get
        {
            let array = data as? [Any]
            if array != nil
            {
                return (array!).count
            }
            else
            {
                return 0;
            }
        }
    }
}
```

21. In the **NewsTableViewController.swift**, implement the viewDidLoad method.

    We made use of the **dispatch_async** to get iOS to execute the code to load data from googleUrl. Since the loading of data may take some time, using the **dispatch_async** with **dispatch_get_global_queue** is a simple way of keeping your application responsive while the loading occurs in a background thread. Once the loading is complete, we call **dispatch_async** again with **dispatch_get_main_queue** to execute another piece of code in the main UI thread.

    Some background on Grand Central Dispatch (GCD)

Grand Central Dispatch (GCD) dispatch queues are a powerful tool for performing tasks. Dispatch queues let you execute arbitrary blocks of code either asynchronously or synchronously with respect to the caller. You can use dispatch queues to perform nearly all of the tasks that you used to perform on separate threads. The advantage of dispatch queues is that they are simpler to use and much more efficient at executing those tasks than the corresponding threaded code.

```
class NewsTableViewController: UITableViewController {

    let googleUrl = URL (string:
"https://ajax.googleapis.com/ajax/services/feed/find?v=1.0&q=iphone7")

    override func viewDidLoad() {
        super.viewDidLoad()

        // Queue an asynchronous task on the background thread,
        // so that viewDidLoad will end immediately, while the
        // task programmed inside the dispatch will run at a
        // later time.
        //
        DispatchQueue.global(qos: .background).async
        {
            do
            {
                let data = try Data (contentsOf: self.googleUrl!)

                // Queue another asynchronous task on the main thread,
                // so that we can perform any updates to the UI.
                //
                DispatchQueue.main.async {
                    do
                    {
                        let json = JSONDataAccessor(try
                            JSONSerialization.jsonObject(with: data,
                            options: .mutableLeaves))

                        print (json.data)

                        let firstArticle =
                            json["responseData"]?["entries"]?[0]

                        print ("Title :\(firstArticle?["title"]?.data
as! String)")
                        print ("Link :\(firstArticle?["link"]?.data as!
String)")
                    }
                    catch
                    {
                        print ("Error parsing JSON data.")
                    }
                }

            }
            catch
            {
                print ("Error retrieving data from server.")
            }
        }
}
```

22. Build and Run your application. You will see the results are printed out in the console.

23. Before we create the array to store the list of news, we must first create a subclass of **NSObject** named **News**. An instance of News class represents an item with a set of attributes. To create a new class in Xcode, right-click on the project folder at the project navigator panel, choose **New File…**. Then select **Cocoa Touch** class, click **Next** and name the class **News**.

```
import UIKit

class News: NSObject {
    var contentSnippet : String
    var link : String
    var title : String
    var url : String
}
```

24. Add the codes below to implement the designated initializer.

```
class News: NSObject {
    var contentSnippet : String
    var link : String
    var title : String
    var url : String


    init(content: String, link: String, title: String, url: String)
    {
        self.contentSnippet = content;
        self.link = link;
        self.title = title;
        self.url = url;

        super.init()
    }
}
```

25. In the **NewsTableViewController.swift**, create an array to store the list of news.

```

```

```
class NewsTableViewController: UITableViewController {

    let googleUrl = NSURL (string:
"https://ajax.googleapis.com/ajax/services/feed/find?v=1.0&q=iphone6s")

    // Declare an array of news with no items inside
    var newsList : [News] = []

    ...
}
```

26. Now, we going to modify the **viewDidLoad** method to put the news objects into newsList.

```
    override func viewDidLoad() {
        super.viewDidLoad()

        // Queue an asynchronous task on the background thread,
        // so that viewDidLoad will end immediately, while the
        // task programmed inside the dispatch will run at a
        // later time.
        //
        DispatchQueue.global(qos: .background).async
        {
            do
            {
                let data = try Data (contentsOf: self.googleUrl!)

                // Queue another asynchronous task on the main thread,
                // so that we can perform any updates to the UI.
                //
                DispatchQueue.main.async {
                    do
                    {
                        let json = JSONDataAccessor(try
                            JSONSerialization.jsonObject(with: data,
                            options: .mutableLeaves))

                        print (json.data)

                        let firstArticle =
                            json["responseData"]?["entries"]?[0]

                        print ("Title :\(firstArticle?["title"]?.data
as! String)")
                        print ("Link :\(firstArticle?["link"]?.data as!
String)")

                        // Clear all the news from our list first.
                        //
                        self.newsList.removeAll()

                        // Get the total number of articles
                        //
                        let entries = json["responseData"]?["entries"]
                        let count = entries?.count

                        // Loop through each article, create a new News
                        // object for each article and add it to our
                        // list.
                        //
                        for var i in 0 ..< count!
                        {
```

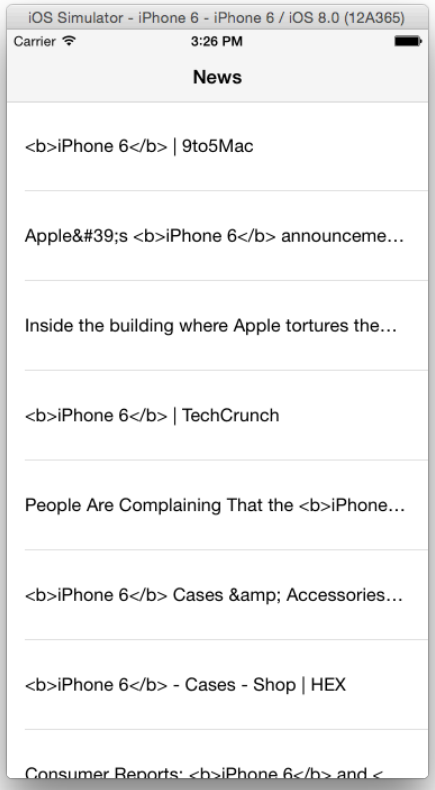
```
                            let newsEntry = entries?[i]
                            let news = News(
                                content:
                                    newsEntry?["contentSnippet"]?.data
                                    as! String,
                                link: newsEntry?["link"]?.data
                                    as! String,
                                title: newsEntry?["title"]?.data
                                    as! String,
                                url: newsEntry?["url"]?.data as! String)

                            self.newsList.append(news)
                        }

                        // Tells the tableView to refresh itself.
                        //
                        self.tableView.reloadData()

                    }
                    catch
                    {
                        print ("Error parsing JSON data.")
                    }
                }

            }
            catch
            {
                print ("Error retrieving data from server.")
            }
        }
    }
```

27. In the **NewsTableViewController.swift**, add the following codes:

```
    // One of the functions for UITableViewDataSource.
    // Tells the table view how many sections there are.
    // In our case, there's only 1 section.
    //
    override func numberOfSections(in tableView: UITableView)
        -> Int
    {
        return 1;
    }

    // One of the functions for UITableViewDataSource.
    // Tells the table view how many items in the given section.
    // Since we have only 1 section, this function will only be
    // called once by the tableView where the section = 1.
    // So we just return the number of items in the newsList array.
    //
    override func tableView(_ tableView: UITableView,
        numberOfRowsInSection section: Int) -> Int
    {
        return self.newsList.count;
    }


    // One of the functions for UITableViewDataSource.
    // Creates or reuses a UITableViewCell with the data at the
    // row position and returns it.
    //
    override func tableView(_ tableView: UITableView, cellForRowAt
indexPath: IndexPath) -> UITableViewCell {
        let cell = tableView
            .dequeueReusableCell(
                withIdentifier: "Cell",
```

```
                for: indexPath)

        let news = newsList[indexPath.row]
        cell.textLabel!.text = news.title
        return cell
    }
```

28. Build and Run your application. If you are unable to see anything on the screen, tap on the **< News** button at the top left corner to go to the table view.

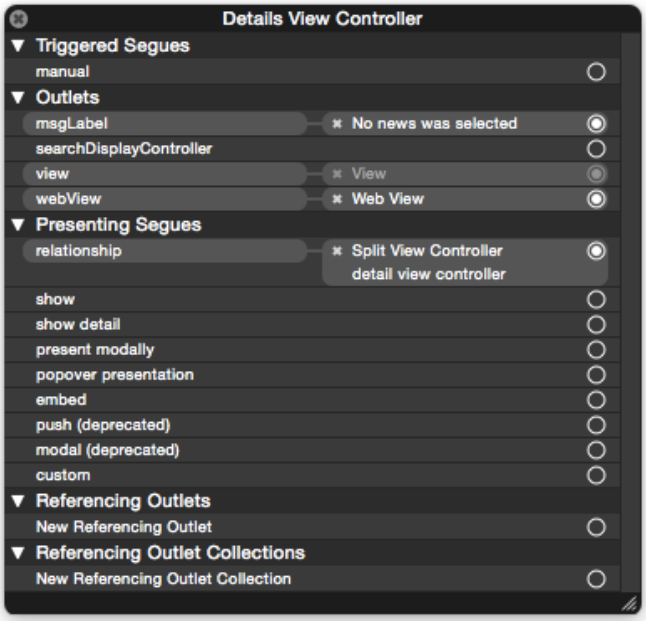


Section 4: Using UIWebView

29. In the project navigator, select the **DetailsViewController.swift** file and add in the following codes as shown:

```
import UIKit

class DetailsViewController: UIViewController {

    @IBOutlet weak var webView : UIWebView!
    @IBOutlet weak var msgLabel : UILabel!

}
```

30. Click on the Main.Storyboard. Select the **Details View Controller** and connect the outlets as shown.



Section 3: Link the Left with the Right View Controller

1. In the project navigator, select the **DetailsViewController.swift** file and add in the following codes as shown:

```
import UIKit

class DetailsViewController: UIViewController {

    @IBOutlet weak var webView : UIWebView!
    @IBOutlet weak var msgLabel : UILabel!

    override func viewDidLoad() {
        super.viewDidLoad()
        webView.isHidden = true
        msgLabel.isHidden = false;
    }


    override func viewDidAppear(_ animated: Bool) {
        super.viewDidAppear(animated)

        if(newsURL != nil)
        {
            // If the newsURL is set by the first view controller
            // then ask the webView to load and display that URL.
            let req = URLRequest(url: newsURL)
            webView.loadRequest(req)

            webView.isHidden = false
            msgLabel.isHidden = true
        }

    }
}
```

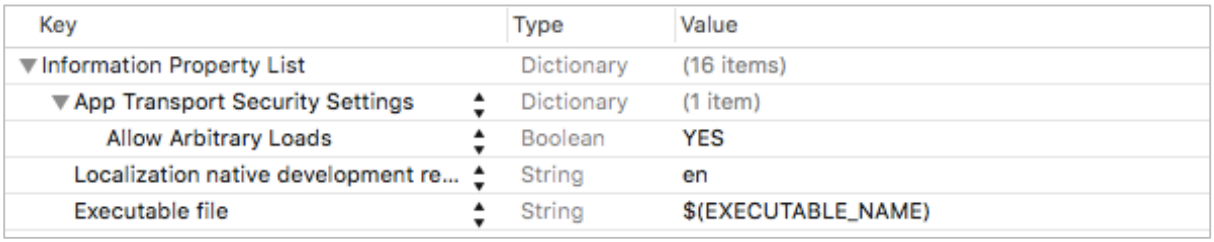
2. Next, In the **NewsTableViewController.swift**, override the **didSelectRowAtIndexPath** to notify the news selection delegate of the new news.

```
    // This function is triggered when the user clicks on
    // an item on the tableView. The indexPath tells us
    // which row the user clicked on.
    //
    override func tableView(_ tableView: UITableView,
        didSelectRowAt indexPath: IndexPath)
    {
        // Gets the news at the selected row.
        let selectedNews = newsList[indexPath.row]

        // We load an instance of the view controller
        // from the storyboard with the id "idDetailsViewController"
        //
        let detailsViewController = UIStoryboard(name: "Main",
            bundle: nil)
            .instantiateViewController(withIdentifier:
                "idDetailsViewController")
            as! DetailsViewController

        // Set the newsURL.
        detailsViewController.newsURL =
            URL(string: selectedNews.link)
```

```
        // Show the view controller of the detail screen.
        self.showDetailViewController(detailsViewController,
            sender: self)
    }
```

3. In your project navigator, select the Info.plist file, and add the following entries:

Key	Type	Value
App Transport Security Settings	Dictionary	
Allow Arbitrary Loads	Boolean	YES



Note: The above is required for navigating to non-https URLs.

4. Try and run the application.
5. Next, we are going to make some changes to the behavior of the split view controller. We want to achieve the following:
   - When in portrait mode, always show the primary view controller (the table view controller) first.
   - When in landscape mode, always show both primary (table view) and secondary (detail view) view controllers side-by-side, regardless of the size of the iOS device.
6. Right on the project folder and select Add File. Choose the **Cocoa Touch** class. Select **UIViewController** from the drop down list and enter **ContainerViewController** as the name of the class.
7. Open the **ContainerViewController.swift** and declare a **SplitViewController** and method **setEmeddedViewController**.

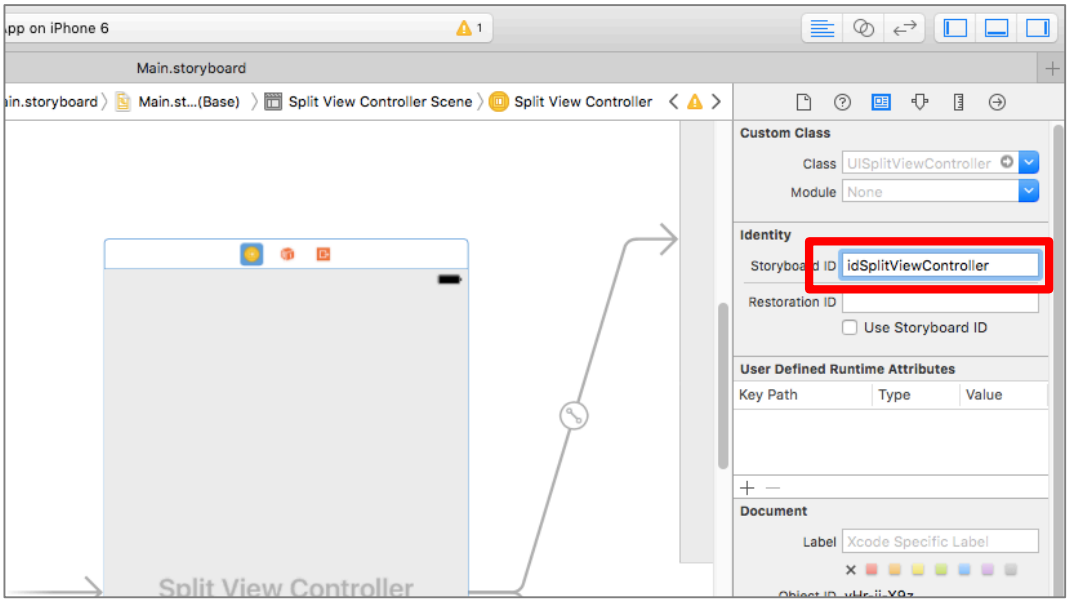
```
import UIKit

class ContainerViewController: UIViewController {

    var viewController: UISplitViewController!

    func setEmbeddedViewController(splitViewController:
        UISplitViewController!)
    {
        if (splitViewController != nil)
        {
            viewController = splitViewController

            // The following code adds a view controller
            // into a container view controller.
            //
            self.addChildViewController(viewController)
            self.view.addSubview(viewController.view)
            viewController.didMove(toParentViewController: self)
```

```
        }
    }
}
```

8. Finally, go to **AppDelegate.swift**. Make the following changes:

```
class AppDelegate: UIResponder, UIApplicationDelegate
    , UISplitViewControllerDelegate
{

    var window: UIWindow?


    func application(application: UIApplication,
         didFinishLaunchingWithOptions launchOptions:
         [NSObject: AnyObject]?) -> Bool
    {

        // Override point for customization after application launch.
        // Instance a new SplitViewController from the
        // storyboard named 'idSplitViewController'
        //
        let splitViewController = UIStoryboard(name: "Main",
            bundle: nil)
            .instantiateViewController(withIdentifier:
"idSplitViewController")
            as! UISplitViewController
        splitViewController.delegate = self
        splitViewController.preferredDisplayMode =
            UISplitViewControllerDisplayMode.allVisible;

        // Create a new container view controller to
        // hold the split view controller created above.
        //
        let containerViewController = ContainerViewController();
        containerViewController
            .setEmbeddedViewController(splitViewController:
splitViewController)

        // Set the current application window's view controller
        // to the container view controller.
        //
        window!.rootViewController = containerViewController;

        return true
    }

    ...
}
```

9. Click on the **Main.Storyboard** and add in the **idSplitViewController** to the Storyboard Id.



10. Lastly, we are going to implement the new method introduced in iOS 8, ***viewWillTransitionToSize***. This method is called when the size of the currently presented view controller is about to change, and it is the best shot we have in order to control an upcoming orientation change.

11. In the **ContainerViewController.swift**, add in the following codes:

```
class ContainerViewController: UIViewController {

    ...

    // Overriding this method allows us apply our own size classes.
    // to child view controllers.
    //
    // In this case, we are trying to force smaller iOS devices to
    // behave like large devices in landscape mode, where we want
    // to show both the primary & secondary view controllers
    // of the split view side-by-side
    //
    override func viewWillTransition(to size: CGSize,
        with coordinator:
        UIViewControllerTransitionCoordinator)
    {

        if (size.width > size.height)
        {
            self.setOverrideTraitCollection(
                UITraitCollection(horizontalSizeClass:
                    UIUserInterfaceSizeClass.regular),
                forChildViewController: viewController)
        }
        else
        {
            self.setOverrideTraitCollection(nil,
                forChildViewController: viewController)
        }

        super.viewWillTransition(to: size,
            with: coordinator)
    }
```

```
}
```

12. Build and run the application.

13. To change the default view to the NewsTableViewController whenever your app starts up on portrait mode, go to **AppDelegate.swift** and add in the following codes:

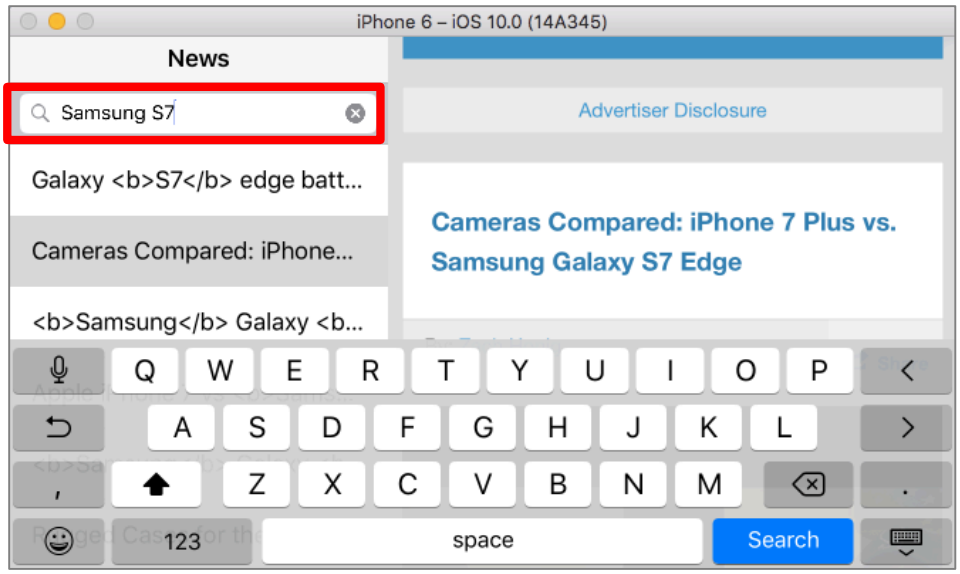
```
class AppDelegate: UIResponder, UIApplicationDelegate
    , UISplitViewControllerDelegate
{
    ...

    // Returning true from this function forces
    // the table view controller to display first when
    // we are in portrait mode. By default, the secondary
    // view controller (in this case, the detail view controller)
    // will display first.
    //
    func splitViewController(
        _ splitViewController: UISplitViewController,
        collapseSecondary secondaryViewController:
        UIViewController,
        onto primaryViewController:
        UIViewController) -> Bool
    {
        return true
    }

    ...
```

```
}
```

14. Build and run the application.

Challenge

1. Try to implement a search function by dropping the UISearchBar into your TableViewController (just above the Cell). Then write codes to reload the news based on the text that the user entered into the UISearchBar.



You may need to implement the relevant delegates and methods. Refer to the following links:

https://developer.apple.com/reference/uikit/uisearchbar
https://developer.apple.com/reference/uikit/uisearchbardelegate
https://grokswift.com/swift-tableview-search-bar/