

# Homework 9: iOS Weather Search App

## Table of Contents

<b>1. Objectives .....</b>	<b>1</b>
<b>2. Background .....</b>	<b>2</b>
<b>2.1 Xcode.....</b>	<b>2</b>
<b>2.2 iOS .....</b>	<b>2</b>
<b>2.3 Swift .....</b>	<b>3</b>
<b>2.4 Amazon Web Services (AWS).....</b>	<b>3</b>
<b>2.5 Google App Engine (GAE) .....</b>	<b>3</b>
<b>3. Prerequisites .....</b>	<b>4</b>
<b>3.1 Download and install Xcode .....</b>	<b>4</b>
<b>3.2 Add your account to Xcode .....</b>	<b>4</b>
<b>3.3 Install CocoaPods .....</b>	<b>4</b>
<b>4. High Level Design .....</b>	<b>5</b>
<b>5. Implementation.....</b>	<b>6</b>
<b>5.1 Initial View.....</b>	<b>6</b>
<b>5.1.1 First Sub View .....</b>	<b>7</b>
<b>5.1.2 Second Sub View .....</b>	<b>8</b>
<b>5.1.3 Third Sub View.....</b>	<b>8</b>
<b>5.1.4 Current Location .....</b>	<b>9</b>
<b>5.2 Search Results Page.....</b>	<b>9</b>
<b>5.2.1 Twitter Button.....</b>	<b>10</b>
<b>5.2.2 Favorite's button .....</b>	<b>11</b>
<b>5.3 Weather Details .....</b>	<b>12</b>
<b>5.3.1 Today Tab .....</b>	<b>12</b>
<b>5.3.2 Weekly Tab.....</b>	<b>13</b>
<b>5.3.3 Google Photos Tab.....</b>	<b>14</b>
<b>5.4 Favorite List Implementation.....</b>	<b>15</b>
<b>5.5 Additional Info .....</b>	<b>16</b>
<b>6. Images / Icons .....</b>	<b>16</b>
<b>7. Material You Need to Submit .....</b>	<b>16</b>

## 1. Objectives

- Become familiar with the Swift language, Xcode and iOS App development.

- Practice the Model-View-Controller design pattern.
- Build a good-looking iOS app.
- Learn to integrate APIs and iOS SDK
- Manage and use third-party libraries by CocoaPods

## 2. Background

### 2.1 Xcode

Xcode is an integrated development environment (IDE) containing a suite of software development tools developed by Apple for developing software for macOS and iOS. First released in 2003, the latest stable release is version 11.1 and is available via the Mac App Store free of charge.

Features:

- Swift 5 support
- Playgrounds
- Interface Builder
- Device simulator and testing
- User Interface Testing
- Code Coverage

The Official homepage of the Xcode is located at:

<https://developer.apple.com/xcode/>

### 2.2 iOS

iOS (originally iPhone OS) is a mobile operating system created and developed by Apple Inc. and distributed exclusively for Apple hardware. It is the operating system that presently powers many of the company's mobile devices, including the iPhone, iPad, and iPod touch. It is the second most popular mobile operating system in the world by sales, after Android.

The Official iOS home page is located at:

<http://www.apple.com/iOS/>

The Official iOS Developer homepage is located at:

<https://developer.apple.com/ios/>

## 2.3 Swift

Swift is a general-purpose, multi-paradigm, compiled programming language created for iOS, macOS, watchOS, tvOS and Linux development by Apple Inc. Swift is designed to work with Apple's Cocoa and Cocoa Touch frameworks and the large body of existing Objective-C code written for Apple products. Swift is intended to be more resilient to erroneous code ("safer") than Objective-C and also more concise. It is built with the LLVM compiler framework included in Xcode 6 and later and uses the Objective-C runtime, which allows C, Objective-C, C++ and Swift code to run within a single program.

The Official Swift homepage is located at:

<https://developer.apple.com/swift/>

## 2.4 Amazon Web Services (AWS)

AWS is Amazon's implementation of cloud computing. Included in AWS is Amazon Elastic Compute Cloud (EC2), which delivers scalable, pay-as-you-go compute capacity in the cloud, and AWS Elastic Beanstalk, an even easier way to quickly deploy and manage applications in the AWS cloud. You simply upload your application, and Elastic Beanstalk automatically handles the deployment details of capacity provisioning, load balancing, auto-scaling, and application health monitoring. Elastic Beanstalk is built using familiar software stacks such as the Apache HTTP Server, PHP, and Python, Passenger for Ruby, IIS for .NET, and Apache Tomcat for Java.

The Amazon Web Services homepage is available at: <http://aws.amazon.com/>

## 2.5 Google App Engine (GAE)

Google App Engine applications are easy to create, easy to maintain, and easy to scale as your traffic and data storage needs change. With App Engine, there are no servers to maintain. You simply upload your application and it's ready to go. App Engine applications automatically scale based on incoming traffic. Load balancing, micro services, authorization, SQL and noSQL databases, memcache, traffic splitting, logging, search, versioning, roll out and roll backs, and security scanning are all supported natively and are highly customizable.

To learn more about GAE support for PHP visit this page:

<https://cloud.google.com/appengine/docs/php/>

To learn more about GAE support for Node.js visit this page:

<https://cloud.google.com/appengine/docs/standard/nodejs/>

### 3. Prerequisites

This homework requires the use of the following components:

#### 3.1 Download and install Xcode

To develop iOS apps using the latest technologies described in these lessons, you need a Mac computer (macOS Sierra 10.12.4 or later) running the Xcode (10.x or later). Xcode includes all the features you need to design, develop, and debug an app. Xcode also contains the iOS SDK, which extends Xcode to include the tools, compilers, and frameworks you need specifically for iOS development.

For HW-9, Xcode version 10.x or later can be used.

**Xcode version 10.3 is strongly recommended. It can be downloaded from the following link:**

<https://developer.apple.com/download/more/?name=Xcode>

You may use any other IDE other than Xcode, but you will be on your own if problems come up.

**Swift 4.2 is strongly recommended to use for this app.**

#### 3.2 Add your account to Xcode

When you add your Apple ID to the Xcode Accounts preferences, Xcode displays all the teams you belong to. Xcode also shows your role on the team and details about your signing identities and provisioning profiles that you'll create later in this document. If you don't belong to the Apple Developer Program, a personal team appears.

Here is detailed documentation:

<https://developer.apple.com/library/iOS/documentation/IDEs/Conceptual/AppStoreDistributionTutorial/AddingYourAccounttoXcode/AddingYourAccounttoXcode.html>

#### 3.3 Install CocoaPods

CocoaPods is a dependency manager for Swift and Objective-C Cocoa projects. It has over ten thousand libraries and can help you scale your projects elegantly. You can install dependencies using it, we will need to install many third-party modules and frameworks using it.

CocoaPods is built with Ruby and is installable with the default Ruby available on macOS. We recommend you use the default Ruby. Using the default Ruby install can require you to use 'sudo' when installing gems.

Run the command below in your Mac terminal:

```
$ sudo gem install cocoapods
```

Once you have created your Xcode project, you can start to integrate CocoaPods into your project.

Further guides on how to integrate CocoaPods are available at: <https://cocoapods.org/>.

The following pods will be needed:

- [SwiftyJSON](#): To handle JSON data.
- [Alamofire](#): To make HTTP requests.
- [Toast-Swift](#) ~> 4.0.0\*: To display toast messages in your app
- [SwiftSpinner](#) : For the big loading spinner.
- [Charts](#): For implementing charts on weekly Tab.

## 4. High Level Design

This homework is a mobile app version of Homework 8. In this exercise, you will develop an iOS Mobile application, which allows users to search for weather details using the dark sky APIs and add locations in favorite list, and post on Twitter. You should reuse the backend service (node.js script) you developed in HW8 and follow the same API call requirements.

The main scene of this app is like that in Figure 1. The launch screen of the App is shown in Figure 1.1. All the implementation details and requirements will be explained in the following sections.



Figure 1: Weather App

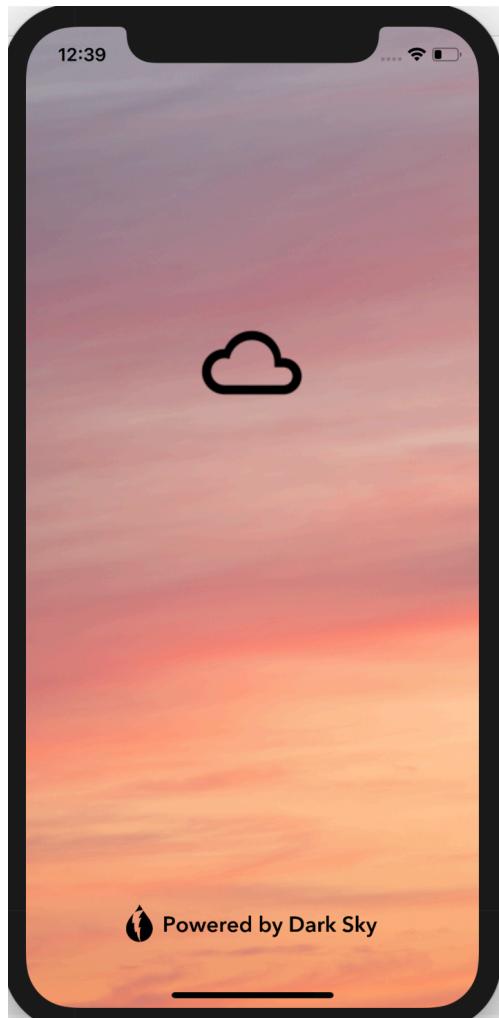


Figure 1.1: Launch Screen

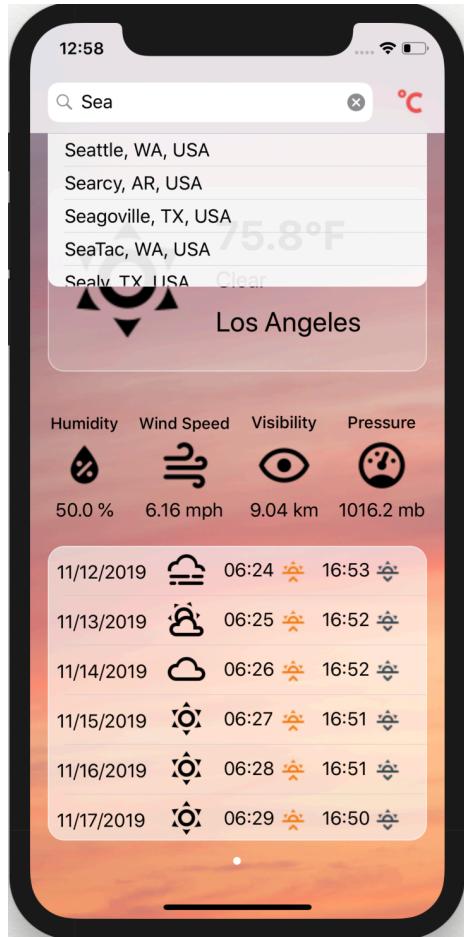
## 5. Implementation

### 5.1 Initial View

You must replicate the Initial View as shown in Figure 1. It should **always** display the weather details of the Current location automatically.

The interface consists of the following:

- **Search Bar:** A '`UISearchBar`' component allowing the user to enter the **city** name. While typing in the search bar, the results should be provided using the autocomplete API.  
Make sure you use the same API as Homework 8. (Hint: The results can be shown as a '`UITableView`' which is hidden/shown according to the response of the autocomplete API).



**Figure 2: AutoComplete for Search**

- **Temperature Conversion Button:** A '`UIBarButtonItem`' allowing the user to switch temperature unit from °C to °F or vice versa. When the user taps on this button, the conversion should be done in **ALL** the places in the entire app where the temperature has been displayed.

### 5.1.1 First Sub View

The first sub view consists of weather Icon, temperature, weather summary and the City Name (Figure 3). The entire view needs to be clickable. On Clicking anywhere on the sub view, user should be navigated to the details page.



**Figure 3: First Sub View**

#### 5.1.2 Second Sub View

The second sub view consists of 4 fixed weather properties as shown in (Figure 4).

Humidity	Wind Speed	Visibility	Pressure
50.0 %	6.16 mph	9.04 km	1016.2 mb

**Figure 4: Second Sub View**

#### 5.1.3 Third Sub View

Third sub view is a scrollable UITableView, with each cell displaying the data for next 7 days. The data consists of the date, weather summary icon, sunrise time, sunrise icon, sunset time and sunset icon. Note that the time is according to **PST** time zone.

11/12/2019		06:24		16:53	
11/13/2019		06:25		16:52	
11/14/2019		06:26		16:52	
11/15/2019		06:27		16:51	
11/16/2019		06:28		16:51	
11/17/2019		06:29		16:50	

**Figure 5: Third Sub View**

### 5.1.4 Current Location

To obtain the current location, standard location service provided by apple should be used. More details on standard location service can be found here:

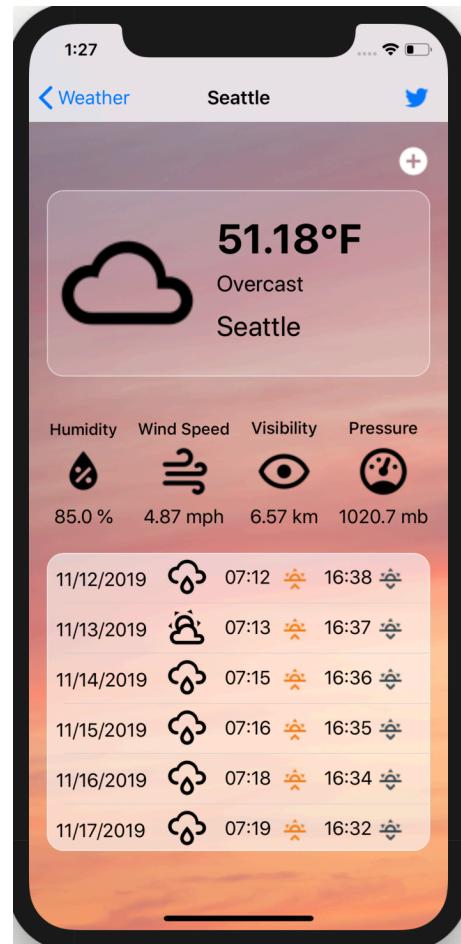
[https://developer.apple.com/documentation/corelocation/getting\\_the\\_user\\_s\\_location/\\_using\\_the\\_standard\\_location\\_service](https://developer.apple.com/documentation/corelocation/getting_the_user_s_location/_using_the_standard_location_service)

### 5.2 Search Results Page

When the user taps any of the city from the autocomplete result, your app should display a *big spinner* (Figure 6) before it's ready to show the search results page. Then after it gets data from your backend, it should hide the spinner and display the result page with exactly the same layout as the initial page, shown in Figure 7.



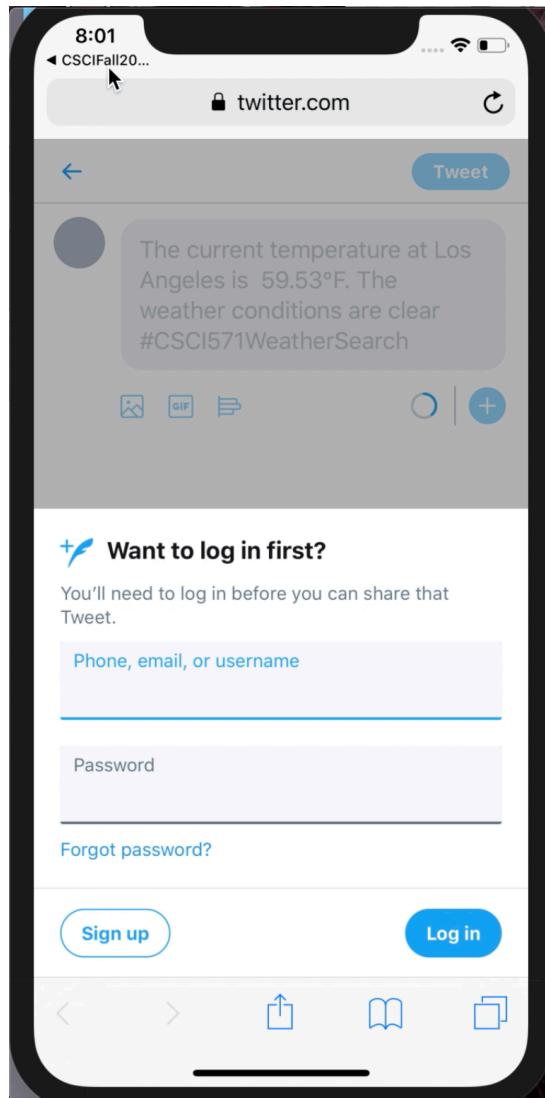
**Figure 6: Display a big spinner while searching**



**Figure 7: Search Result Page**

### 5.2.1 Twitter Button

**Share button (Twitter icon)** to share the weather details on Twitter. Once the button is tapped, a web page should be opened (in Safari) to allow the user to share the weather information on Twitter, as shown in Figure 8.



**Figure 8: Twitter Button**

### 5.2.2 Favorite's button

Tapping the Favorite button (the '+' button) would add the corresponding Weather city into the Favorite list. A message should be displayed at the bottom of the app and the Favorite button should be changed to 'x' button, as shown in Figure 9. Tapping that button again would remove that city from the favorite list, and a similar message should also be displayed to indicate that the city has been removed from favorite list and the button is changed back to '+' button as shown in the video.

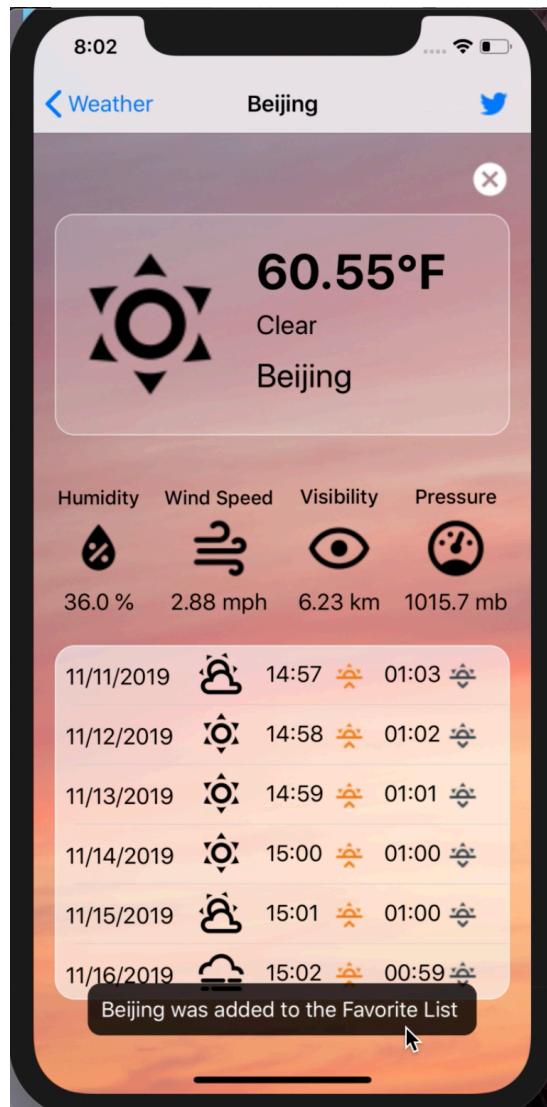


Figure 9: Message on adding the city to fav list

The state of the fav button should be consistent i.e. If the user searches for a city that is already in the fav list, then on search results page, the state of the favorite button should be ('x') not ('+').

## 5.3 Weather Details

Tapping on the first sub view in the should show weather details with three tabs named as: Today, Weekly and Photos (of the city) from google (Figure 10). Note that a spinner should be shown before you are ready to display information in Today tab and Photos tab. (refer video)

All the three tabs share the same **Navigation Bar on the top**. The navigation bar should include the following elements:

- **Back button** which navigates back to the search results page.
- **Twitter button**
- **City Name** as Title of the header.

### 5.3.1 Today Tab

The today tab contains a collection view having 9 items as shown in figure 10. Each Item has corresponding icon, property name and its value with units.

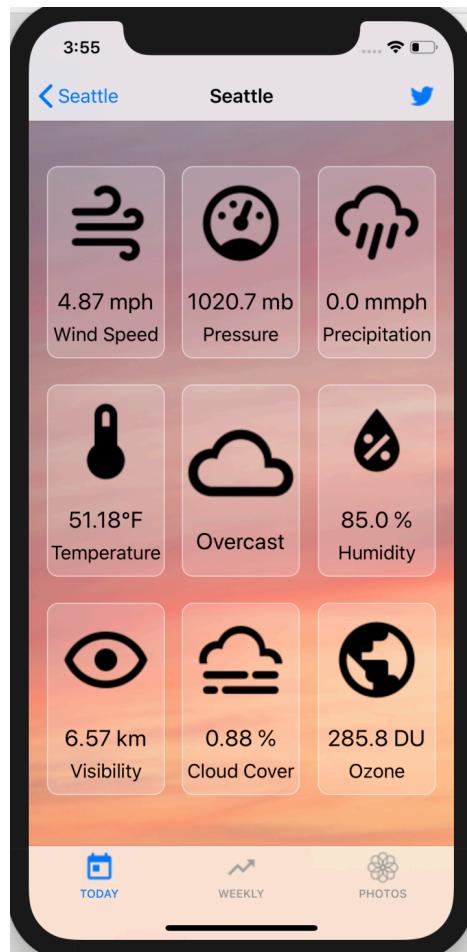


Figure 10: Today Tab

### 5.3.2 Weekly Tab

Weekly Tab consists of the following:

1. Weekly Weather Summary View having the weather summary of the entire week and the corresponding icon. It is the ‘summary’ and ‘icon’ fields of ‘daily’ key in the JSON result.
2. A Line chart showing minimum and maximum temperature trend during the week. You will be using charts pod (mentioned in 3.3)

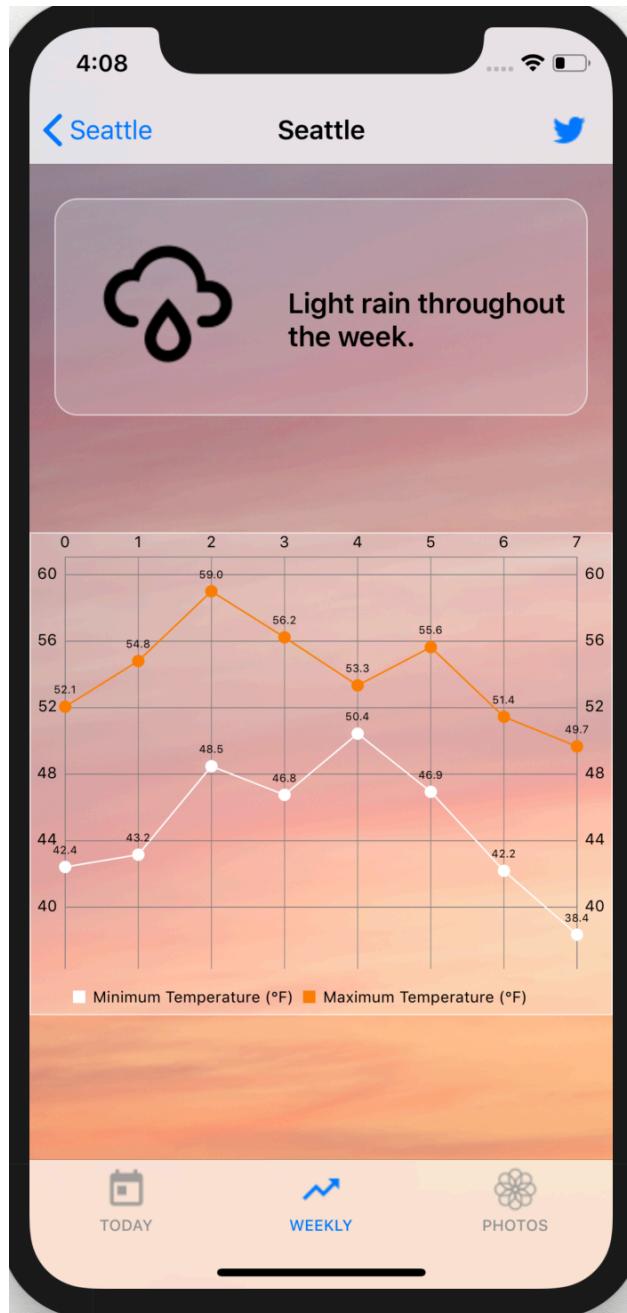


Figure 11: Weekly Tab

**Note: The graph units (°C/ °F ) should depend on the state of the button selected on the initial page.**

### 5.3.3 Google Photos Tab

Google Images Fetched using the name of the city and shown as 'UIScrollView' with vertical scroll. You should use google custom search API for image search.

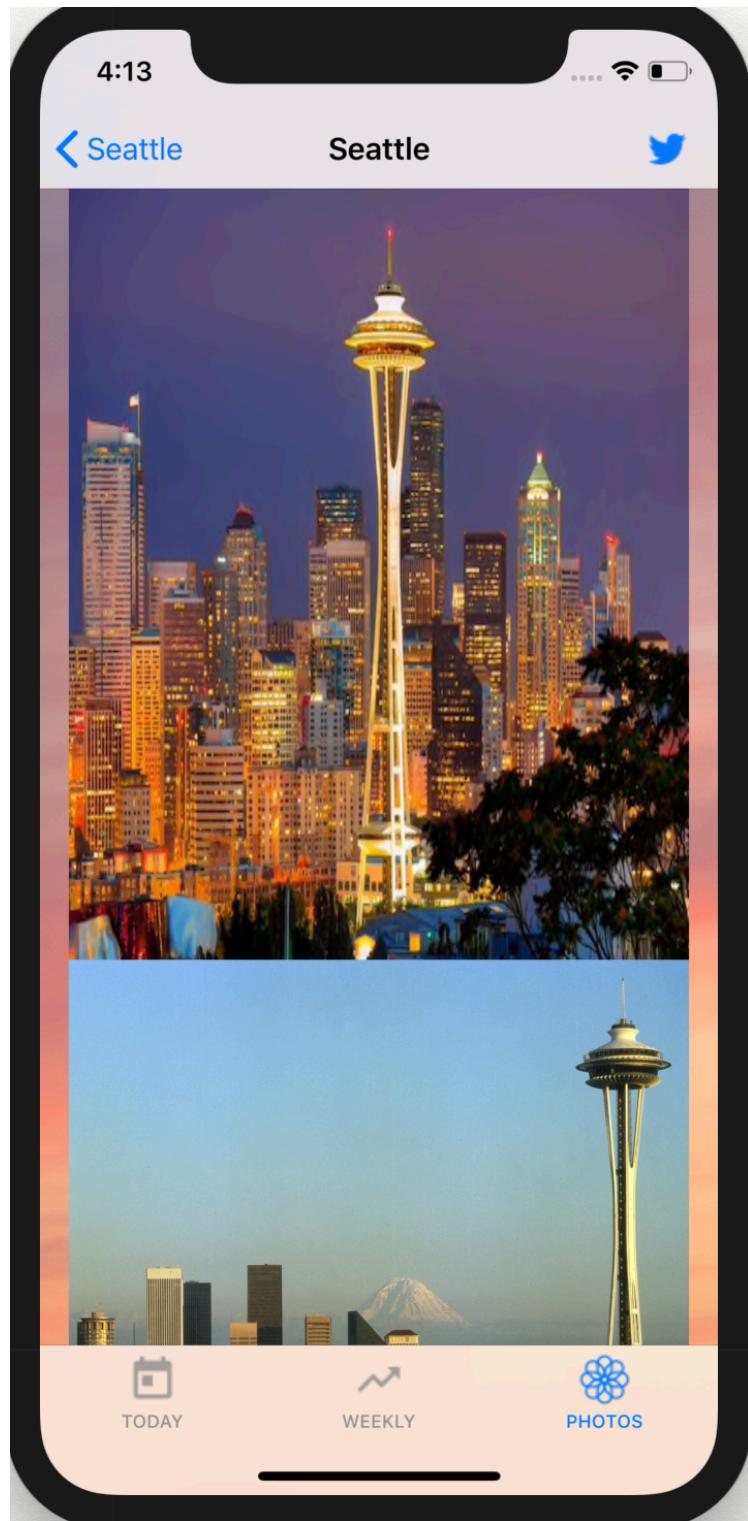


Figure 12: Google Photos Tab

#### 5.4 Favorite List Implementation

Use **UIPageControl** and **UIScrollView** in the initial view to display the weather details of the cities adding the favorite list.

Use “**UserDefaults**” to store favorite List data.

<https://developer.apple.com/documentation/foundation/userdefaults>

Key Points to note :

1. The favorite list should persist even after closing the app. The items should be removed from fav list only if the user removes them (see video).
2. You may need to pass data between different views. You can refer this link to learn about data passing between Views in Swift : <https://learnappmaking.com/pass-data-between-view-controllers-swift-how-to/>.
3. The horizontal scroll of the views should replicate exactly as shown in the video

## 5.5 Additional Info

For things not specified in the document, grading guideline,piazza, or the video, you can make your own decisions. But keep the following points in mind:

- Always display a proper message and don't crash the app, if an error occurs.
- You can only make HTTP requests to your backend (Node.js script on AWS/GAE/Azure)
- All HTTP requests should be asynchronous.

## 6. Images / Icons

[Click](#) to download the images needed for this HW. **All** the icons needed for the app are included in the downloads.

## 7. Material You Need to Submit

Unlike other exercises, you will have to “demo” your submission “in person” during a special grading session. Details and logistics for the demo will be provided in class, in the Announcement page and in Piazza.

**Demo is done on a MacBook using the emulator, and not on a physical mobile/tablet device.**

**You should also ZIP all your source code (without image files and third-part modules) and push the resulting ZIP file by the end of the demo day to GitHub Classroom.**

### **\*\*IMPORTANT\*\*:**

All videos are part of the homework description. All discussions and explanations in Piazza related to this homework are part of the homework description and will be accounted into grading. So please review all Piazza threads before finishing the assignment.