HOMEWORK ASSIGNMENT 4

CSCI 571 – Fall 2024

Abstract

JSON, Node.js, MongoDB and the Tomorrow.io API on iOS

Assignment 4: iOS Weather Search App

Table of Contents

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 16.0 and is available via the Mac App Store free of charge.

Features:

- Swift 6 and Swift UI 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, iPadOS, 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 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 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 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 this assignment, Xcode version 10.x or later can be used. **Xcode version 16.0 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 6 and above 5.x are 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: To display toast messages in your app
- SwiftSpinner: For the big loading spinner.
- HighCharts ~> 9.2.2 : For implementing charts on weekly and data Tab.

3.4 Alternative to CocoaPods -> Swift Package Manager

To add dependencies to your Swift without worrying about PodFiles you can also use Swift Package Manager. **We recommend using the Package Manager instead of Cocoa Pods.**

In Toolbar, click File – Swift Packages – Add Package Dependency for adding all packages. Then add git repo link to add dependency.

The following URLs will be needed to add:

- https://github.com/SwiftyJSON/SwiftyJSON: To handle JSON data.
- https://github.com/Alamofire/Alamofire.git: To make HTTP requests.

- https://github.com/scalessec/Toast-Swift: To display toast messages in your app
- https://github.com/icanzilb/SwiftSpinner: For the big loading spinner.
- https://github.com/highcharts/highcharts-ios.git: For implementing charts.

4. High Level Design

This homework is a mobile app version of assignment 3. In this exercise, you will develop an iOS Mobile application, which allows users to search for weather details using the Tomorrow.io APIs and add locations in favorite list, and post on X (aka Twitter). You should reuse the backend service (node.js script) you developed in assignment 3 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

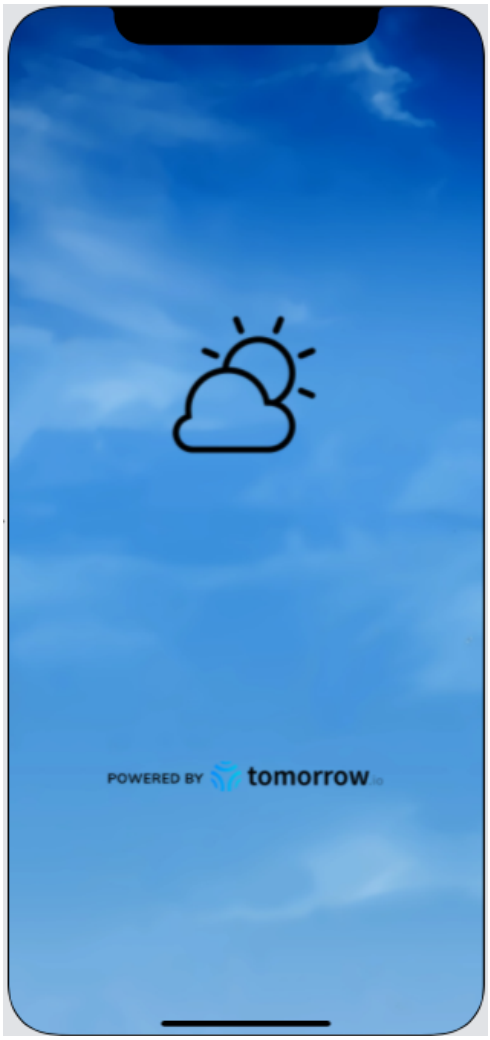


Figure 1.1: Launch Screen

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 Assignment 3. (Hint: The results can be shown as a 'UITableView' which is hidden/shown according to the response of the autocomplete API).

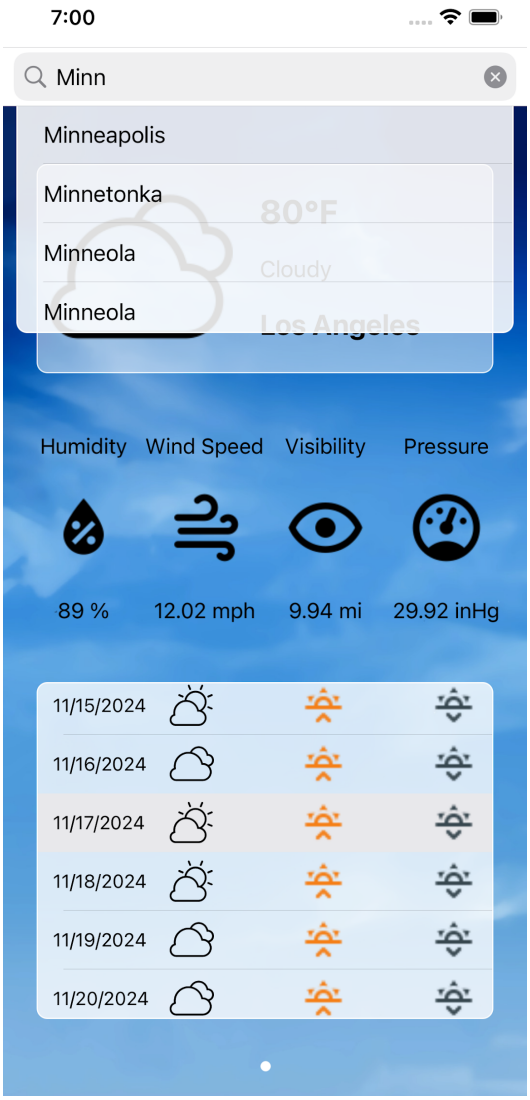


Figure 2: AutoComplete for Search

5.1.1 First Sub View

The first sub view consists of weather Icon according to the Weather Code from API, temperature, City Name and weather text according to weather code (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 (Humidity, windspeed, visibility and preassureSeaLevel) fixed weather properties as shown in (Figure 4).

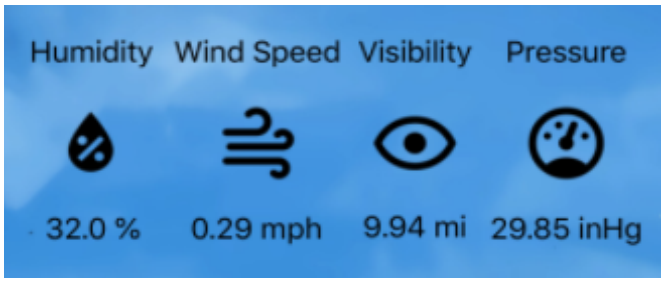


Figure 4: Second Sub View

5.1.3 Third Sub View

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

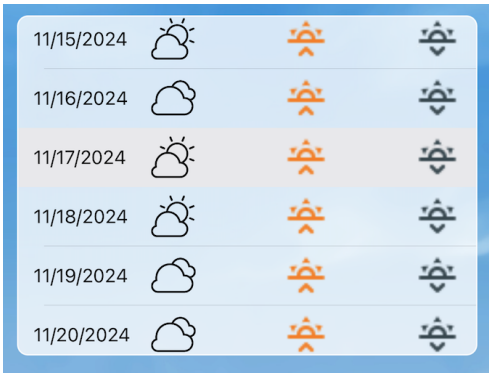


Figure 5: Third Sub View

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

5.2 Search Results Page

When the user taps any of the city from the autocomplete result, your app should display a *big spinner* (Hint: using SwiftSpinner, see 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 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 X Button

Share button (X/Twitter icon) to share the weather details on X (aka Twitter). Once the button is tapped, a web page should be opened (in Safari or your default browser for your emulator) to allow the user to share the weather information on X, as shown in Figure 8.

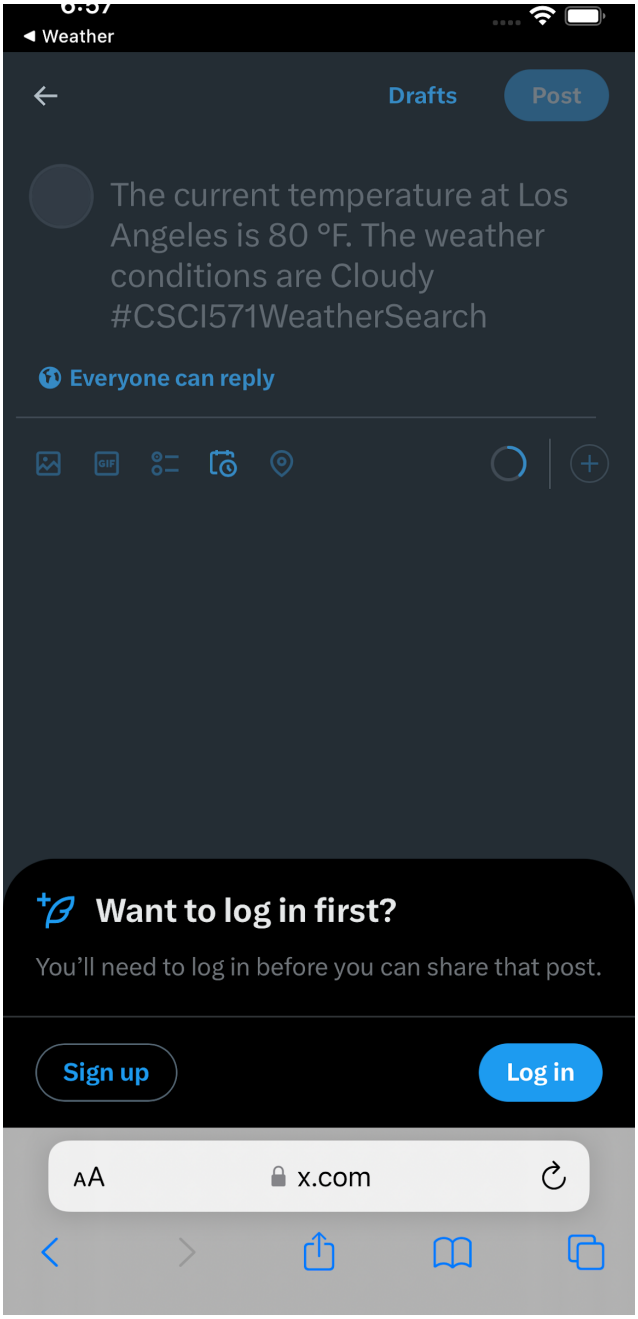


Figure 8: X Tweet

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 removing the city to fav list

Note: The state of the fav button should be consistent. I.e., if the user searches for a city that is already in the favorite list, then on search results page, the state of the favorite button should be ('x') not ('+'). Favorites should be stored in the MongoDB Atlas database in the cloud, as implemented in assignment 3.

5.3 Weather Details

Tapping on the first sub view in the should show weather details with three tabs named as: **Today**, **Weekly** and **Weather Data** (Figure 10).

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 a corresponding icon, property name and its value with units.

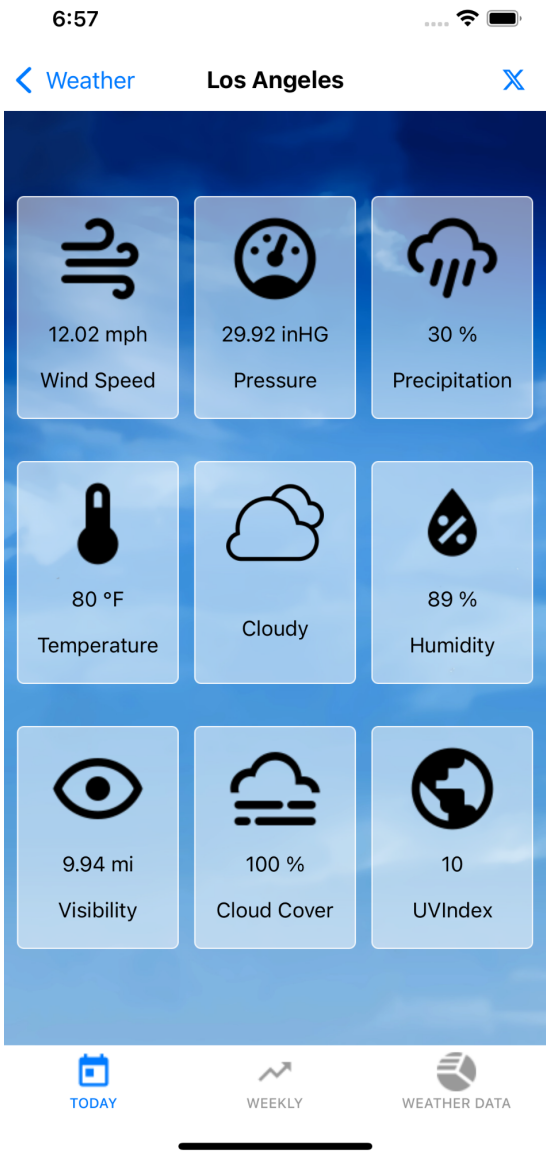


Figure 10: Today Tab

5.3.2 Weekly Tab

The Weekly Tab consists of the following:

1. Weekly Weather Summary View having the weather text according to the weather code and the corresponding icon with current temperature displayed on it.

2. An Area Chart showing minimum and maximum temperature trend for next 15 days from current date (including current date). You will be using HighCharts pod (mentioned in 3.3)

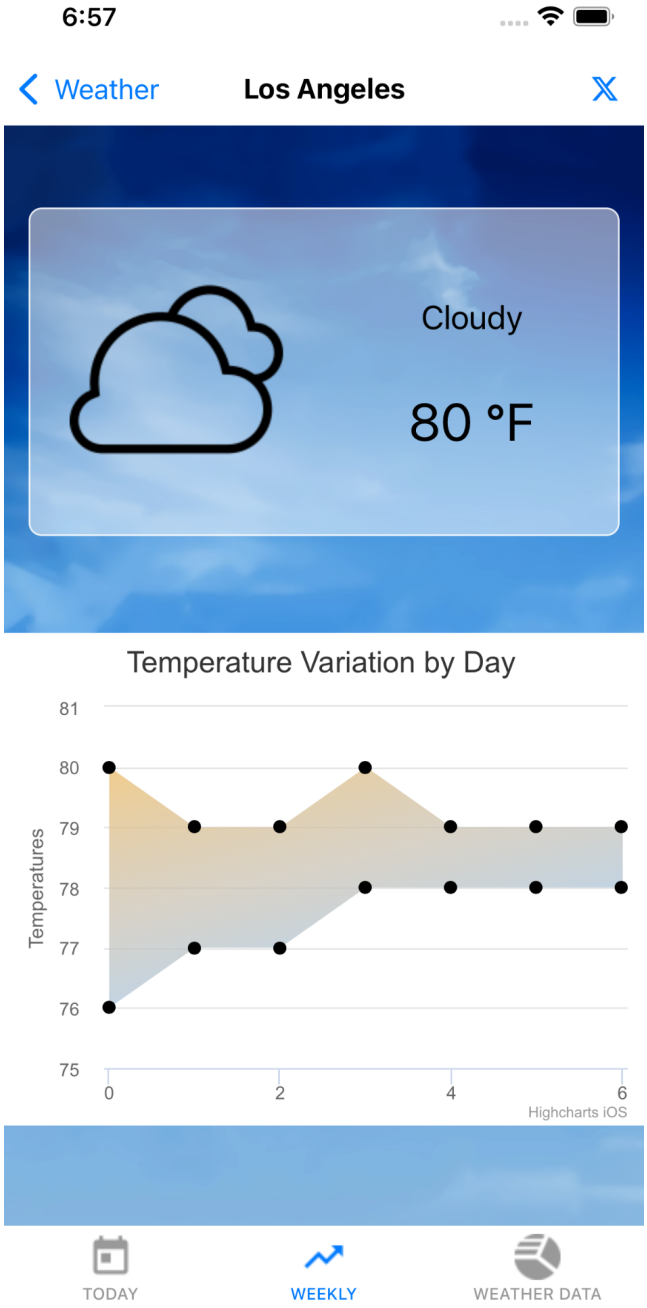


Figure 11: Weekly Tab

5.3.3 Weather Data Tab

The Weather Data Tab consists of the following:

1. Key Data Summary View having 3 data displayed. precipitationProbability, humidity and cloudCover with the icons displayed which are same as the Today Tab.

2. An **Activity gauge** chart from HighCharts pods. Which will contain the percentage value of precipitationProbability, humidity and cloudCover with 3 different gauges.

You can find more info about different charts in HighCharts pods here:
https://www.highcharts.com/demo/ios

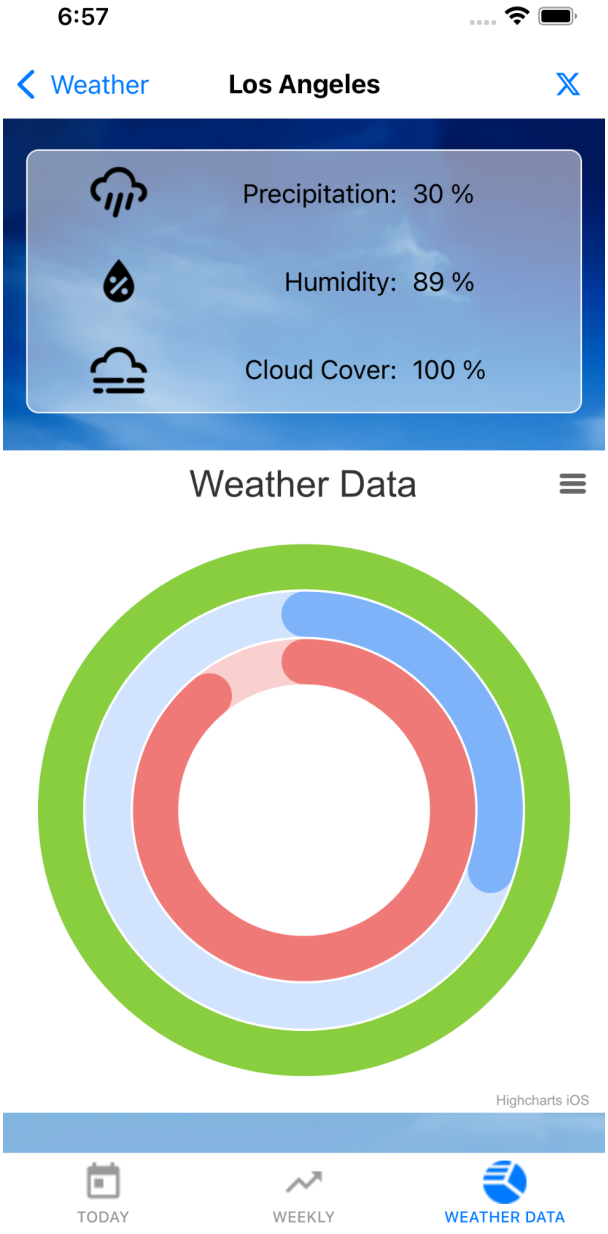


Figure 12: Google Photos Tab

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. See:

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.
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 items not specified in the document, rubric, piazza, or the video, you can make your own decisions. But keep the following points in mind:

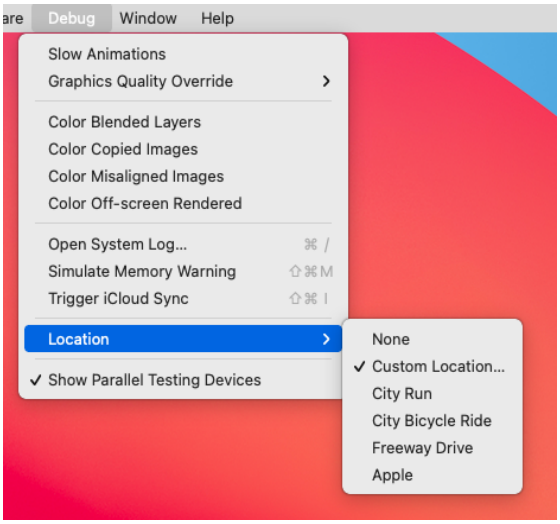
- Always display a proper message and don't crash the app, if an error occurs.
- You can only make HTTP requests to your backend (Node.js script on AWS/GAE/Azure)
- All HTTP requests should be asynchronous.

6. Set your own location in emulator

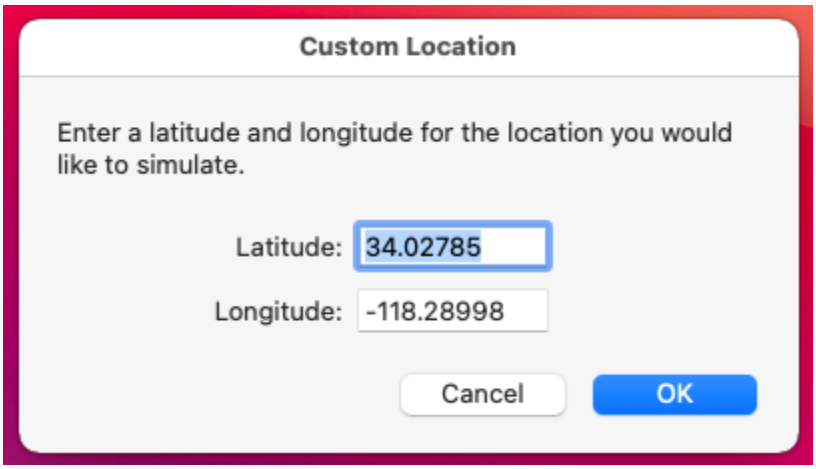
Follow these steps to add your own location inside your emulator as it defaults to some fixed points around the world.

Step 1: Open your emulator.

Step 2: Go to Debug → Location → Custom Location



Step 3: Set your location using Latitude and Longitude.



7. Images / Icons

All the icons and Images necessary for this assignment will be provided on DEN.

8. Material You Need to Submit

You will have to demo your submission using a video recorded with **Zoom**. Details are included in a Final Presentation document on D2L Brightspace. **Demo is done on a MacBook using the simulator, and not a physical mobile device.**

You should run **Xcode** → **Product** → **Clean Build Folder**, to remove all compiled object files.

Then generate the following files:

1. Your **mobile source code** as a **ZIP file**. This file should contain your front-end code (Swift or SwiftUI), by zipping the top iOS folder containing your project; name your zip file **Assignment4-<username>_mobile.zip**.
2. Your **back-end source code** (NodeJS files, possibly the same as Assignment #3), as a **ZIP file** of the NodeJS top folder; name your zip file **Assignment4-<username>_backend.zip.**
3. Your **video file** in **.mp4 format**, created using Zoom and the *Final Presentation* guidelines available on D2L Brightspace. Name the recording as **Assignment4-<username>.mp4**.

Then click **File Upload**. Note: make sure that BOTH the 2 .ZIP files and the .MP4 file (3 files) are **uploaded on the SAME submission**, as every submission will overwrite the previous one.

IMPORTANT

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