

# Homework 9: Place Search iOS App

## 1. Objectives

- Become familiar with Swift language, Xcode and iOS App development.
- Practice the Model-View-Controller design pattern.
- Build a good-looking iOS app.
- Learn to use google place 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 OS X and iOS. First released in 2003, the latest stable release is version 9.0 and is available via the Mac App Store free of charge for OS X High Sierra users.

Features:

- Swift 4 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, OS X, 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/flexible/Node.js/>

### 3. Prerequisites

This homework requires the use of the following components:

#### 3.1 Download and install the latest version of 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 latest version of Xcode. 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.

Download the latest version of Xcode on your Mac free from the App Store.

To download the latest version of Xcode

- Open the App Store app on your Mac (by default it's in the Dock).
- In the search field in the top-right corner, type Xcode and press the Return key.
- The Xcode app shows up as the first search result.
- Click Get and then click Install App.
- Enter your Apple ID and password when prompted.
- Xcode is downloaded into your /Applications directory.

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

#### 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 OS X. 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/>

## 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 place information, save some as favorites, and post on Twitter. You should reuse the backend service (php/node.js script) you developed in HW8 and follow the same API call requirements (See Piazza @610 for more details).

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

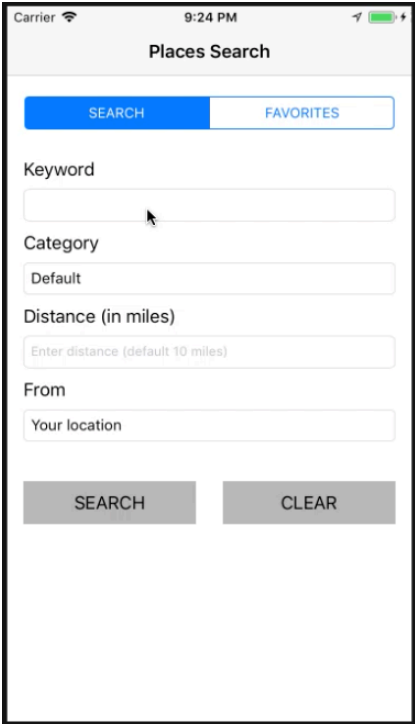
The image shows a mobile application interface titled "Places Search". At the top, there is a status bar with "Carrier", signal strength, "9:24 PM", and battery level. Below the title, there are two tabs: "SEARCH" (highlighted in blue) and "FAVORITES". The main form contains four input fields: "Keyword" (with a cursor), "Category" (with "Default" selected), "Distance (in miles)" (with placeholder text "Enter distance (default 10 miles)"), and "From" (with "Your location" selected). At the bottom, there are two buttons: "SEARCH" and "CLEAR".

Figure 1. The Place Search App

## 5. Implementation

### 5.1 Search Form

You must replicate the Search Form, as shown in Figure 1.

The interface consists of the following:

- **Keyword:** A 'UITextField' component allowing the user to enter the keyword.
- **Category:** A 'UITextField' allowing the user to choose a category. When the user taps on this field, a picker view should display at the bottom of the screen for selecting a category, as shown in Figure 2. Make sure you include all the categories in homework 8. See the hints section 6.3 for how to implement it.
- **Distance:** A 'UITextField' component allowing the user to enter the distance (in miles).
- **From:** A 'UITextField' component allowing the user to enter a location. When the user taps on this field, it goes to a full screen auto-complete mode, as shown in Figure 3. The user has to either choose one from the auto-complete or cancel. See the hints section 6.3 for how to implement it.
- A **SEARCH button** to get the input information of each field, after validation. If the validation is successful, then the places would be fetched from the server. However, if the validations are unsuccessful, appropriate messages should be displayed and no further requests would be made to the server.
- A **CLEAR button** to clear the input fields and rWhere do you mention eset them to default values if applicable.

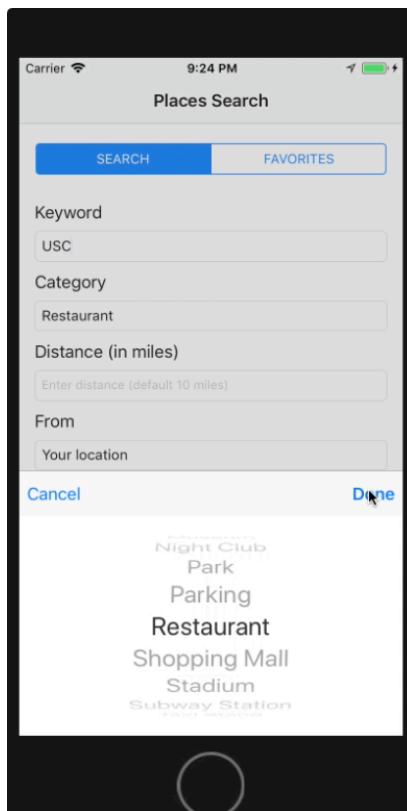


Figure 2: Choose category from a picker view

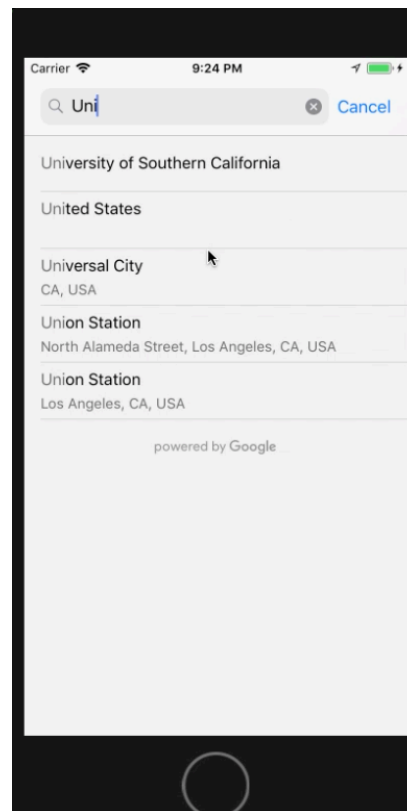
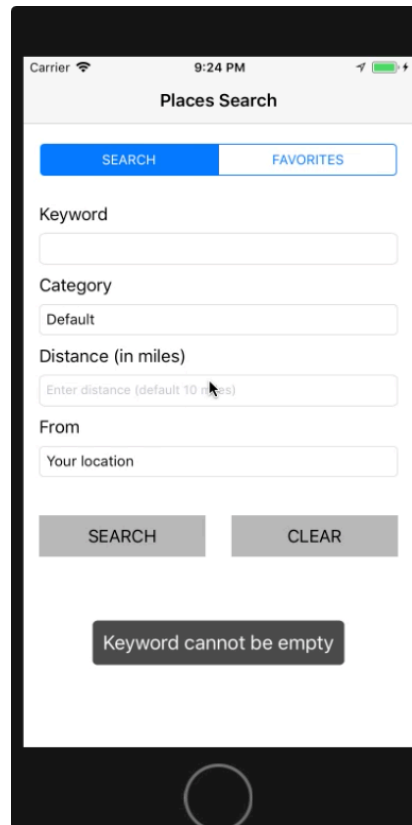


Figure 3: Autocomplete for places

The validation for empty keyword need to be implemented. If the user does not enter anything in the 'UITextField' or just enters some empty spaces, when he presses the 'SEARCH' button you need to display an appropriate message to indicate the error, as shown in Figure 4.

Once the validation is successful, you should execute an HTTP request to the PHP/Node.js script located in the GAE/AWS, which you have finished in Homework 8, and then navigate to the Search Result page.



**Figure 4: Validation for empty input**

## 5.2 Search Results

When the user taps the "SEARCH" button, your app should display a big spinner before it's ready to show the results page, as shown in Figure 5. Then after it gets data from your backend, hide the spinner and display the result page as a UITableView, as shown in Figure 6.

Each of the UITableViewCell should have the following:

- Category image
- Name of the place
- Address of the place
- A heart-shaped "Favorite" button

**See homework 8 for more details about these fields.**

Tap the favorite button would add the corresponding place into the favorite list, and a message should be displayed at the bottom of the app, as shown in Figure 7. Tap that button again would remove that place from the favorite list, and a similar message should also be displayed to indicate the place has been removed from the favorite list.

For the pagination, there should be two buttons “Prev” and “Next” below the results table, as shown in Figure 6. If it’s the first time to tap the “Next” button, it has to show the big spinner again while getting data from the server, as shown in Figure 8. Note that a “Prev” or “Next” button should be disabled if there’s no previous page or no next page.

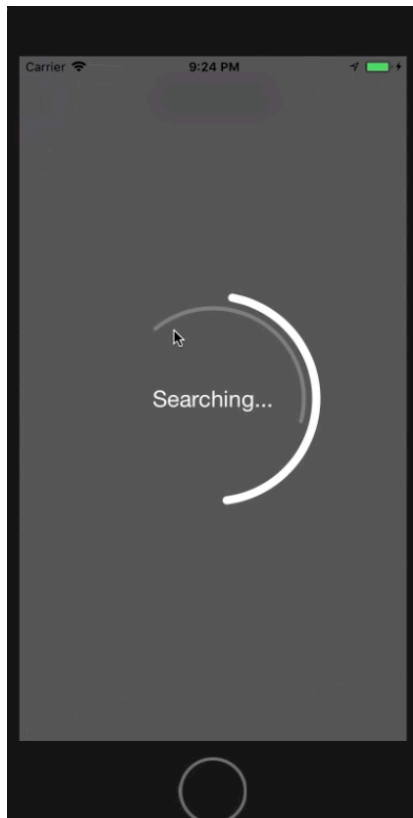


Figure 5: Display a big spinner while searching

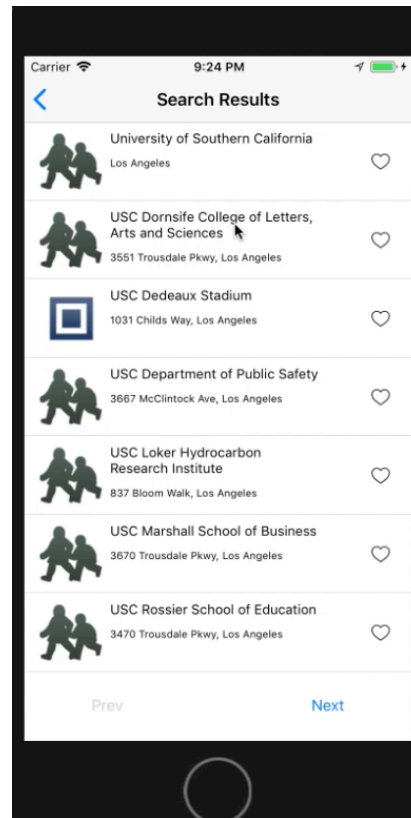


Figure 6: List of search results

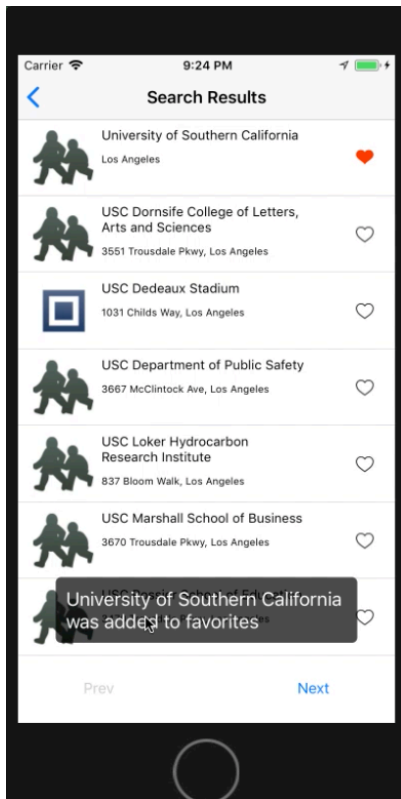


Figure 7: Message for adding to favorites

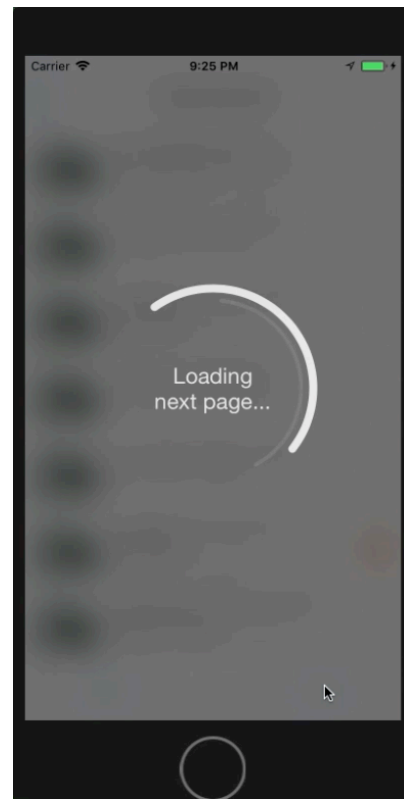


Figure 8: Loading next page

### 5.3 Place Details

Tap on a UITableViewCell in the results table, and the app should show details of that place with four tabs: Info tab, photos tab, map tab and reviews tab. Note that the spinner should be shown before you are ready to display all the place details.

All the four tabs share the same Navigation Bar on the top, as shown in Figure 9. The navigation bar should include the following elements:

- Back button which navigates back to the searching results list
- Title, which is the name of the place
- Share button (✈) to share the place detail on Twitter. Once the button is tapped, a web page should be opened to allow the user to share the place information on Twitter, as shown in Figure 10. **See homework 8 for how it works and the format of the tweet content.**
- Favorite button to add/remove the place to/from the favorite list, and display a proper message at the bottom of the screen

#### 5.3.1 Info Tab

Show the fields listed in Figure 9 in the Info tab: Address, Phone Number, Price Level, Rating, Website, and Google Page. See homework 8 for more details about each field.



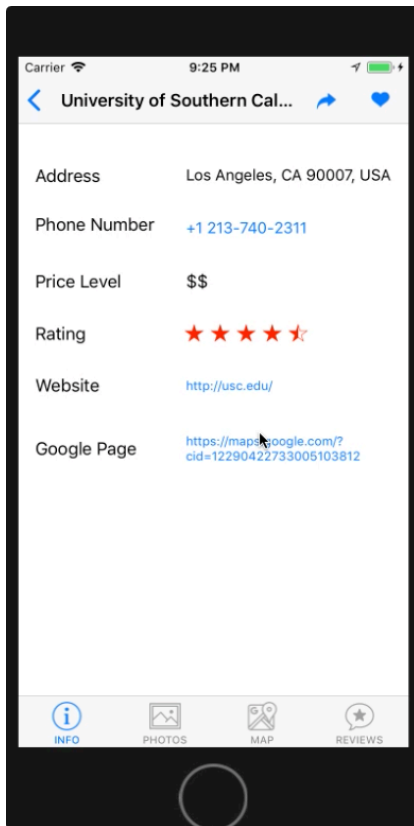


Figure 9: Place details and the Info Tab

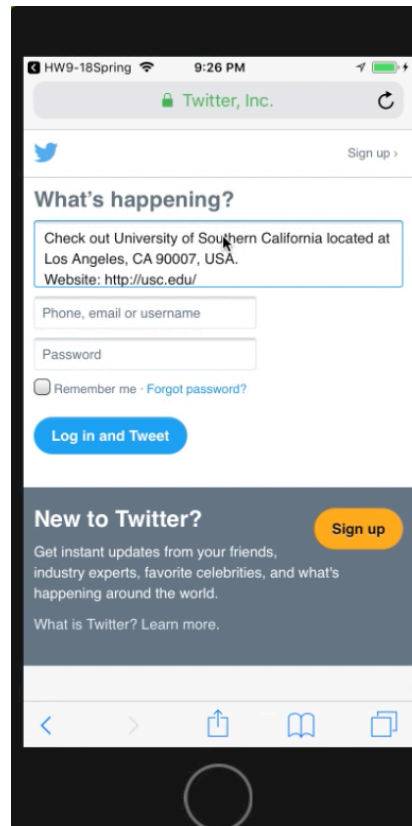
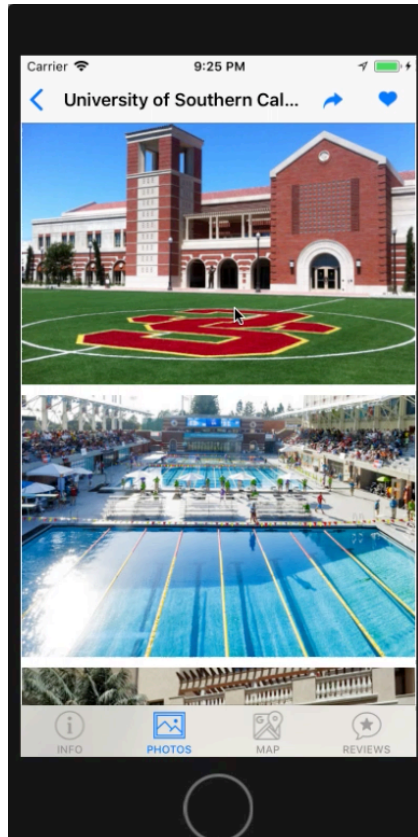


Figure 10: Share place on twitter

### 5.3.2 Photos Tab

Use a “UICollectionView” to display the place photos, as shown in Figure 11. See the Places API for iOS about how to get place photos:

<https://developers.google.com/places/iOS-api/photos>



**Figure 11: Photos Tab**

### 5.3.3 Map Tab

As shown in Figure 12, there are three elements in this tab:

- From: a “UITextField”, when it is tapped, it goes into the full screen auto-complete mode
- Travel mode: use a segmented control
- Map: use the Google Map SDK for iOS:

<https://developers.google.com/maps/documentation/iOS-sdk/>

Initially, there’s a marker on the map indicating the location of the place. Once the user taps the “From” field and selects a place entry from the auto-complete, the app fills the “From” field with the name of the place selected and displays a path from the start location to the destination, as shown in Figure 13.

Switching among different travel modes should update the paths on the map, as shown in Figure 14.

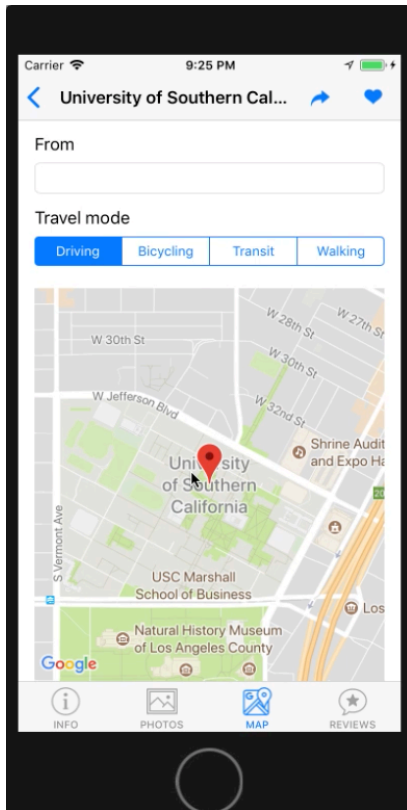


Figure 12: Map Tab

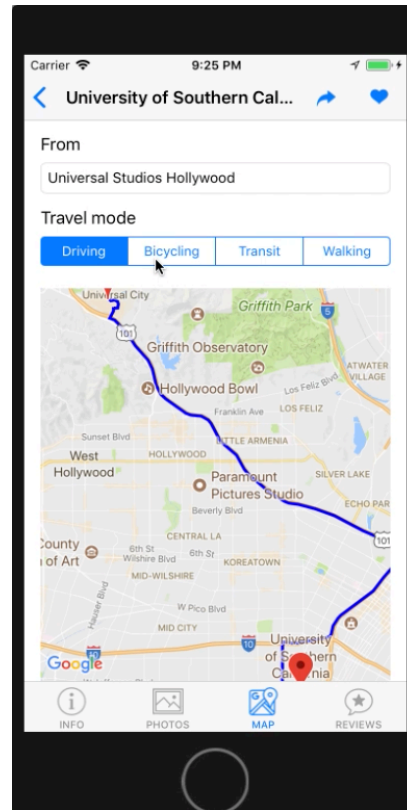


Figure 13: Show the path on the map

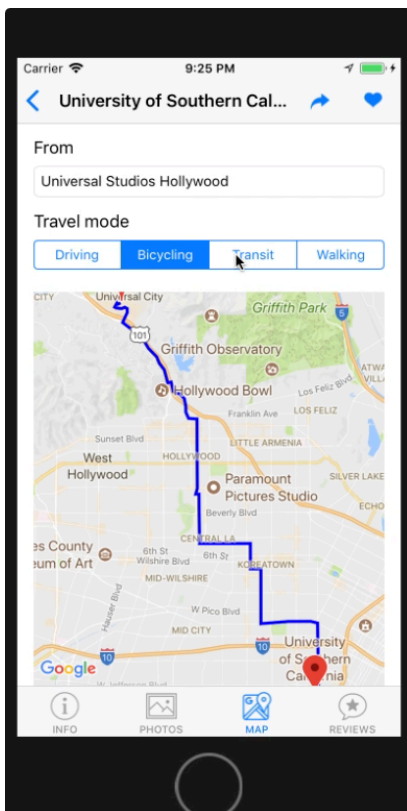


Figure 14: Switch to a different travel mode

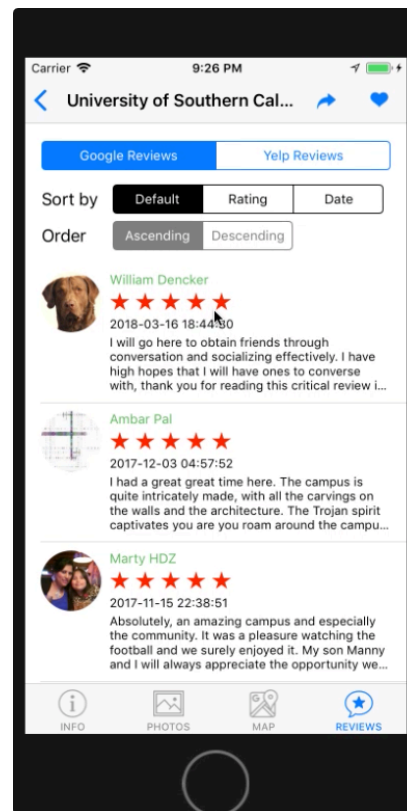


Figure 15: Reviews Tab

### 5.3.4 Reviews Tab

This tab displays the Google reviews and Yelp reviews of the place, same as homework 8. As shown in Figure 15, you should use a segmented control to switch between google reviews and yelp reviews. And use two more segmented controls to allow users to sort the reviews by Default/Rating/Date in Ascending/Descending order.

The reviews are shown in a “UITableView”. Note that each of the cell can be tapped and then a web page should be opened and navigates to the tapped review, as shown in Figure 16.

In the case of no reviews data, show “no reviews” in the center of the screen, as shown in Figure 17.

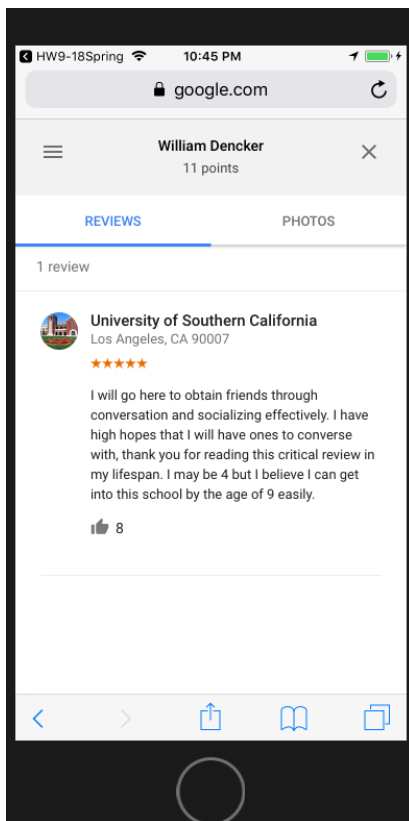


Figure 16: webpage for the tapped review

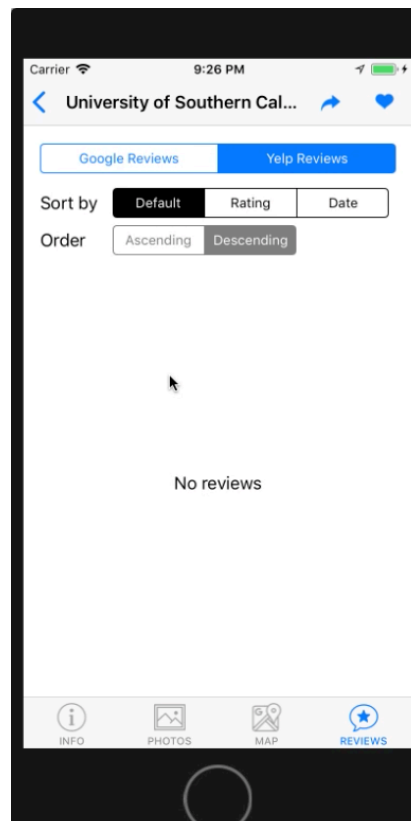


Figure 17: No reviews

### 5.4 Favorites list

Use a segmented control in the main screen to switch between the search page and the favorites page. The favorite places should be displayed in a “UITableView”. Each of the “UITableViewCell” includes a place’s catalog image, place name and address, as shown in Figure 18. If there are no favorite places, a “No Favorites” should be displayed at the center of the screen, as shown in Figure 19.

Swiping a cell allows the user to remove a place from the favorites list, as shown in Figure 20. A message should be displayed at the bottom if the place is removed.

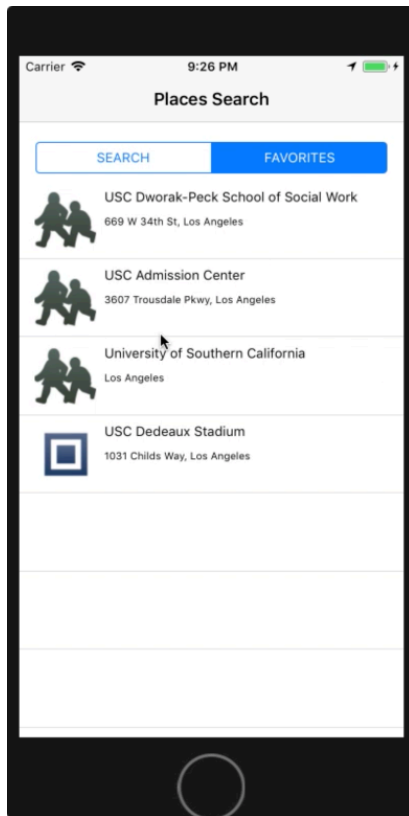


Figure 18: Favorite list

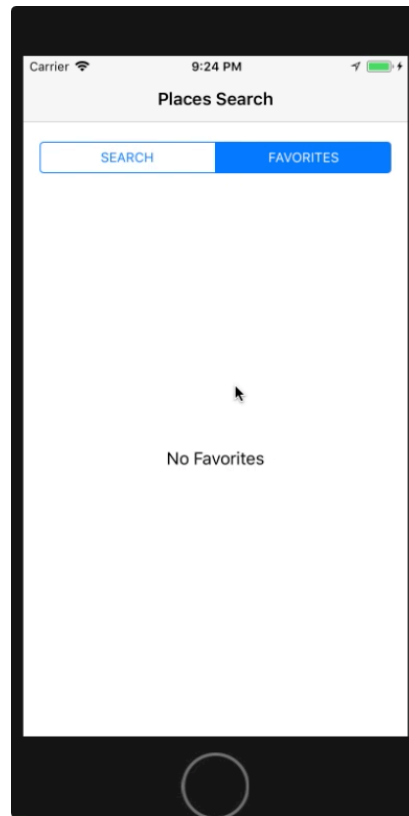


Figure 19: No Favorites

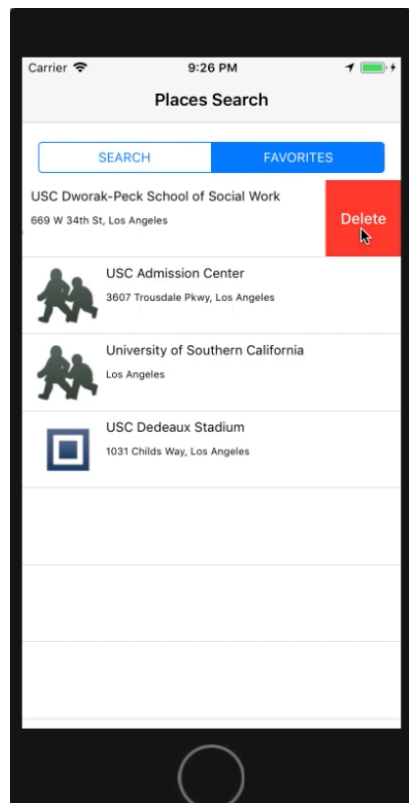


Figure 20: Swipe to remove a place

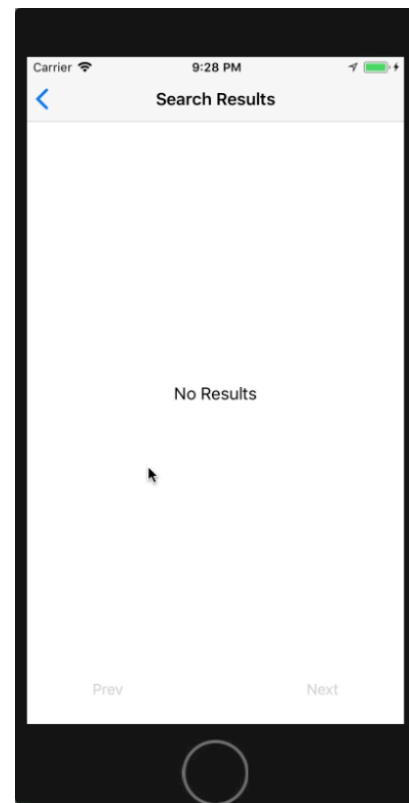


Figure 21: No search results

## 5.5 Error handling

If no places are found given a keyword, a “no results” should be displayed, as shown in Figure 21.

If for any reason (no network, API failure, cannot get location etc.) an error occurs, an appropriate error messages should be displayed at the bottom of screen.

## 5.6 Additional

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

- Always display a proper message and don't crash if an error happens.
- You can only make HTTP requests to your backend (PHP/Node.js script on AWS/GAE), or use a Google Places API for iOS or Google Map SDK for iOS.
- All HTTP requests should be asynchronous.

# 6. Implementation Hints

## 6.1 Images

The images needed for this homework are available here:

<http://cs-server.usc.edu:45678/hw/hw9/images/ios/forward-arrow.png>

<http://cs-server.usc.edu:45678/hw/hw9/images/ios/favorite-filled.png>

<http://cs-server.usc.edu:45678/hw/hw9/images/ios/favorite-empty.png>

<http://cs-server.usc.edu:45678/hw/hw9/images/ios/google-maps.png>

<http://cs-server.usc.edu:45678/hw/hw9/images/ios/info.png>

<http://cs-server.usc.edu:45678/hw/hw9/images/ios/picture.png>

<http://cs-server.usc.edu:45678/hw/hw9/images/ios/review.png>

## 6.2 Get current location and favorites storage

Use “CLLocationManager” to get current location. Since sometimes it fails to get the current location in the iPhone simulator, you might have to set a custom location by setting “Debug – Location – custom location...” in the simulator.

Use “UserDefaults” to store favorites data.

## **6.3 third-party modules**

### **6.3.1 Place autocomplete**

Google Places API: <https://developers.google.com/places/iOS-api/autocomplete>

### **6.3.2 Http requests**

As learned in the class, you may use “Alamofire” and “AlamofireSwiftJSON” to make http requests and parse JSON. Also “AlamofireImage” is useful for downloading images.

### **6.3.3 Show messages and Spinners**

Install “EasyToast” by CocoaPods to display messages in your app. For the big spinner, install “SwiftSpinner” by CocoaPods.

### **6.3.4 Picker view for the place search category**

To implement the picker view in Figure 2, install “McPicker” by CocoaPods. See:

<https://github.com/kmcgill88/McPicker-iOS>

### **6.3.5 Rating stars**

To implement the rating stars in the Info Tab of place details, install “Cosmos” by CocoaPods. See:

<https://github.com/evgenyneu/Cosmos>

## **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.

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

**\*\*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.