Homework 9: Event 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 Maps 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 10.1 and is available via the Mac App Store free of charge.

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 you 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 event information, ticketing information, save events as favorites, 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. All the implementation details and requirements will be explained in the following sections.

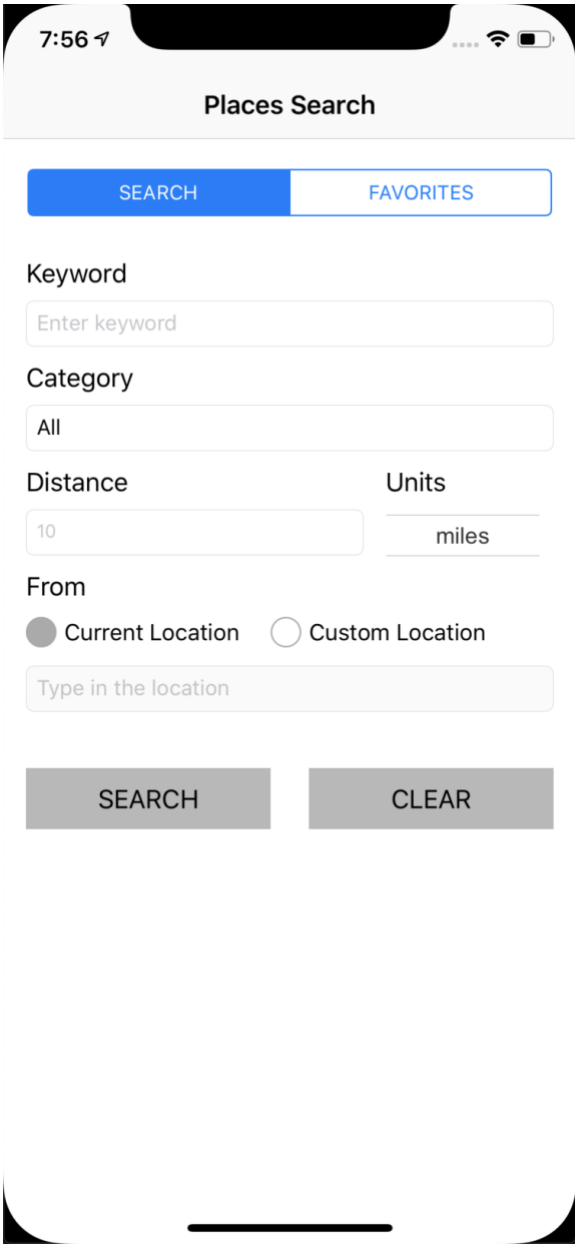

Figure 1. The Event Search App

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. It provides the autocomplete function as shown in Figure 3. Make sure you use the same API as Homework 8.
- **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.
- **Distance**: A 'UITextField' component allowing the user to enter the distance (in miles).
- **Units**: A 'UIPickerView' component allowing the user to select the units for the distance (miles or kms).
- **From**: Two 'UIButtons' that toggle between user's current location or a custom location. Below these two buttons is a 'UITextField' which should only be activated if the custom location option is chosen.
- A **SEARCH button** to get the input information of each field, after validation. If the validation is successful, then the events 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 set them to default values if applicable. It should remove all error messages (if present).

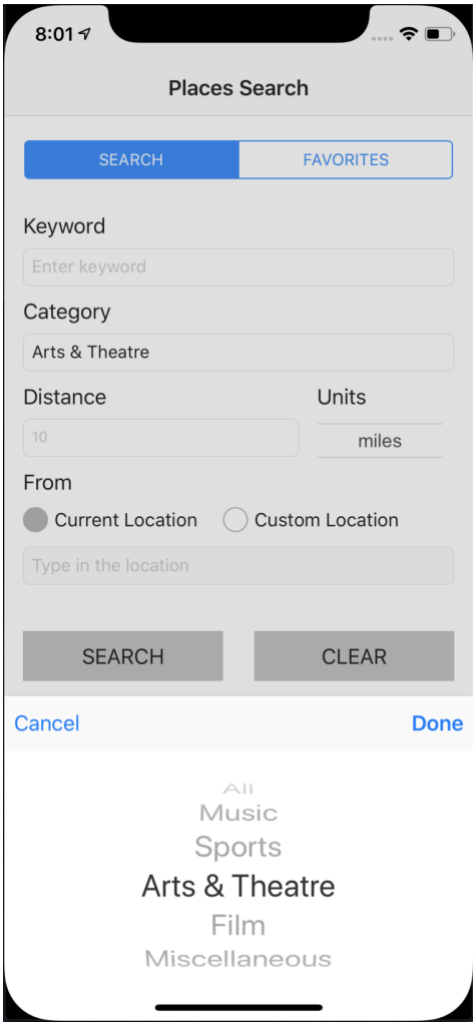

Figure 2: Choose category from a picker view

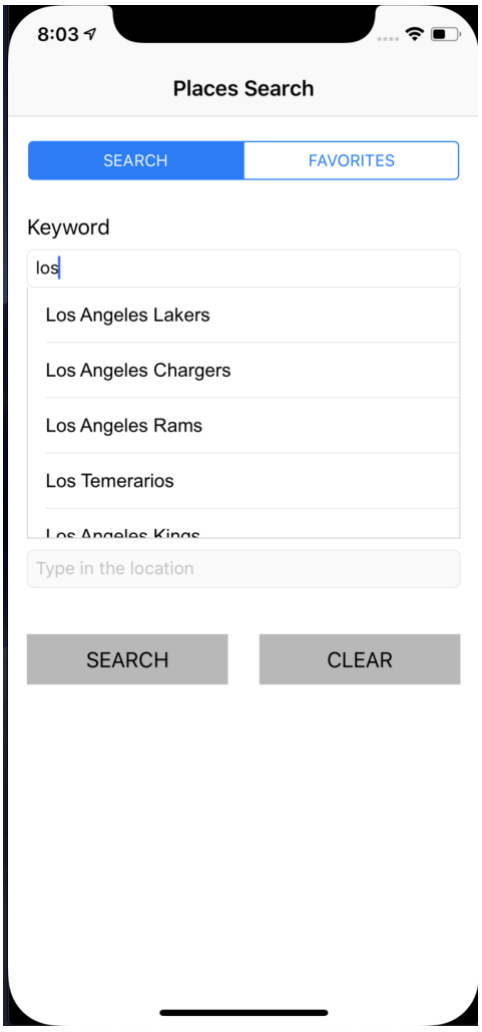

Figure 3: Autocomplete for keywords

The validation for empty keyword and empty location (if custom location is selected) needs 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 Node.js script located in the GAE/AWS/Azure, which you have finished in Homework 8, and then navigate to the Search Results page.

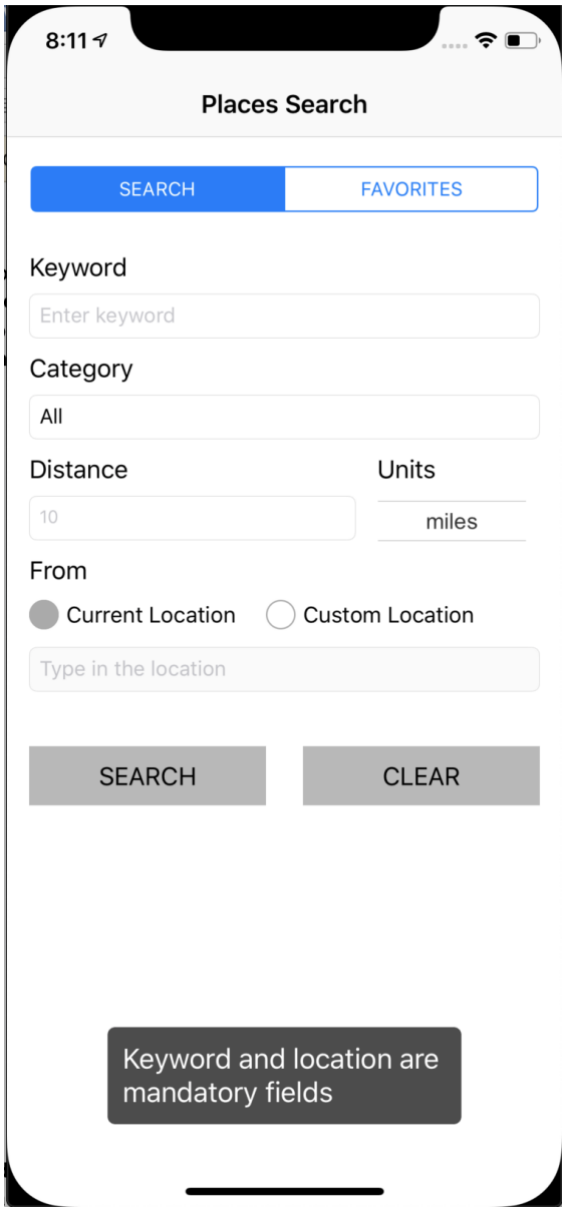

Figure 4: Validation for empty input

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 6. Then after it gets data from your backend, hide the spinner and display the result page as a UITableView, as shown in Figure 5.

Each of the UITableViewCell’s should have the following:

- Category image
- Name of the event
- Name of the venue
- Data and time of the event
- A heart-shaped “Favorite” button

See homework 8 for more details about these fields.

Tap the favorite button would add the corresponding event into the favorite list, and a message should be displayed at the bottom of the app, as shown in Figure 7. Tapping that button again would remove that event from the favorite list, and a similar message should also be displayed to indicate that the events has been removed from favorites as shown in Figure 8.

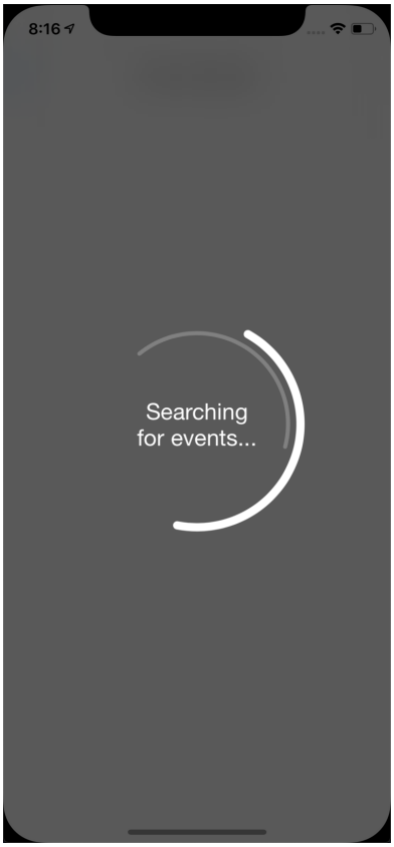

Figure 5: Display a big spinner while searching

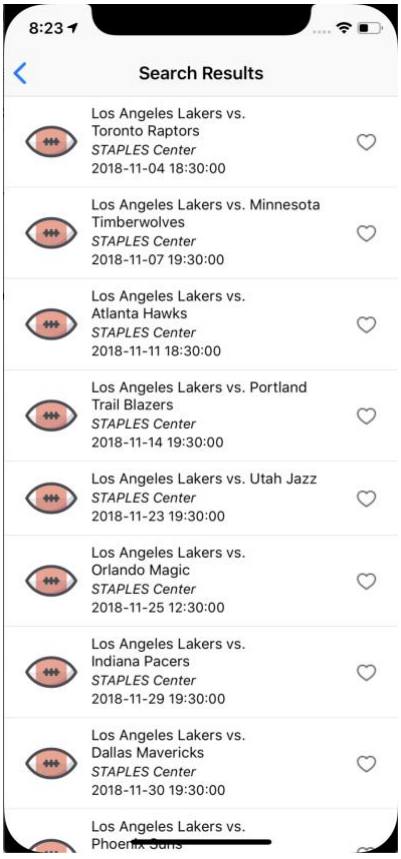

Figure 6: List of search results

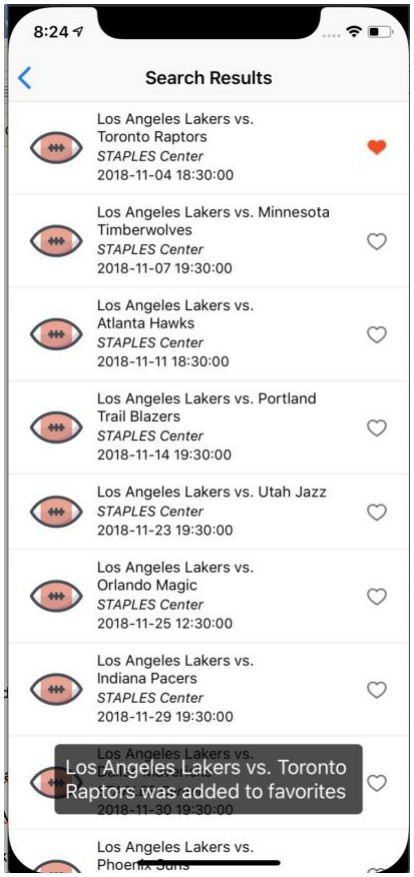

Figure 7: Message for adding to favorites

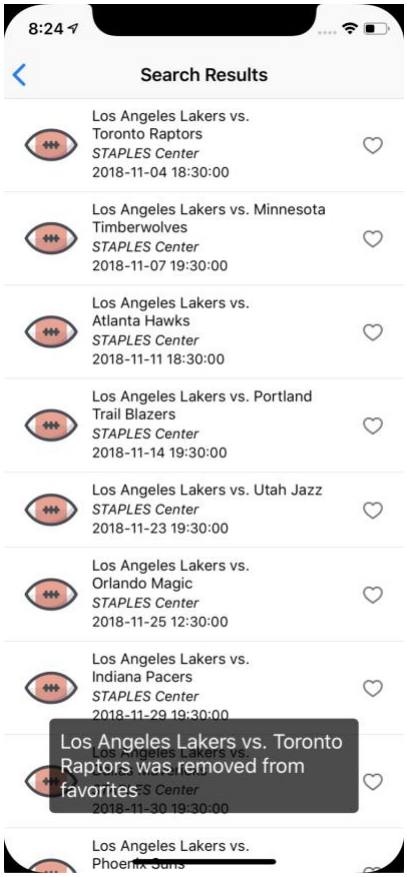

Figure 8: Message shown for removing from favorites

5.3 Event Details

Tapping on a UITableViewCell in the results table should show details of that event with four tabs: Event Info, Artist Info + Photos, Venue Details and Upcoming event. Note that a spinner should be shown before you are ready to display information in each tab.

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
- **Share button (Twitter icon**) to share the event detail on Twitter. Once the button is tapped, a web page should be opened to allow the user to share the event 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 event to/from the favorite list and display an appropriate message at the bottom of the screen.

5.3.1 Event Info Tab

Show the fields listed in Figure 9 in the Info tab: Artist/Team(s), Venue, Time, Category, Price Range, Ticket Status, Ticketmaster Ticket Link and Seat Map Link. Clicking on the ticketmaster URL will open the Ticketmaster URL in a Safari window. The seat map link will be the URL of the staticMap property and when user click on this URL, it will open Safari to show the seat map.

See homework 8 for more details about each field.

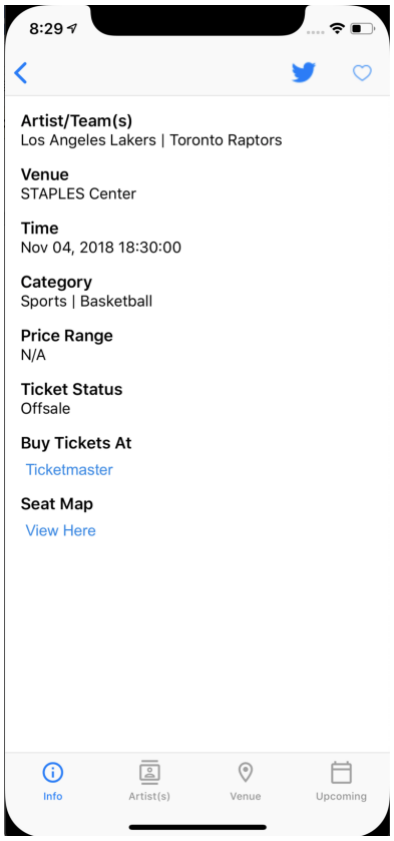

Figure 9: Event details and the Info Tab

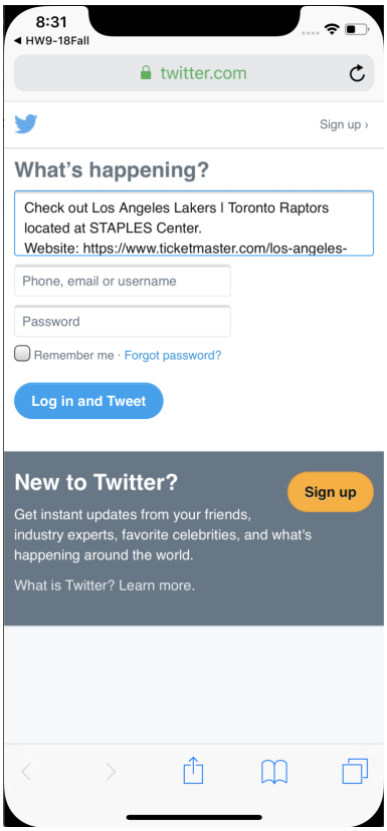

Figure 10: Share place on twitter

5.3.2 Artist Info Tab

Use a "UICollectionView" to display the artist's photos and info (if applicable), as shown in Figure 12. But for multiple artists in homework #9, you could only show the **first two** artist's music profiles (if applicable) and corresponding photos (at most 8 photos for each artist/team).

See Figure 11, Figure 12 and Figure 13 for more details.

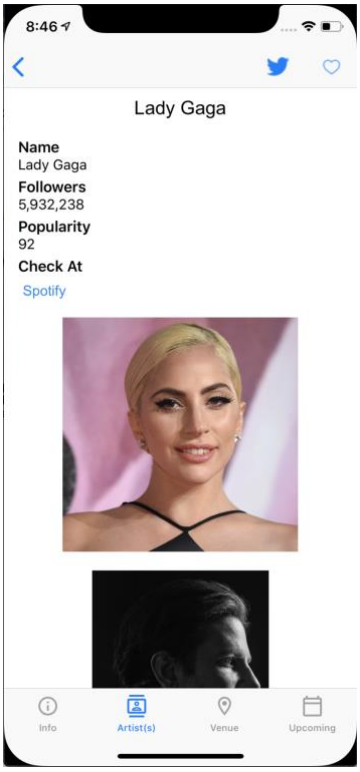

Figure 11: Artist Tab with Music Artist

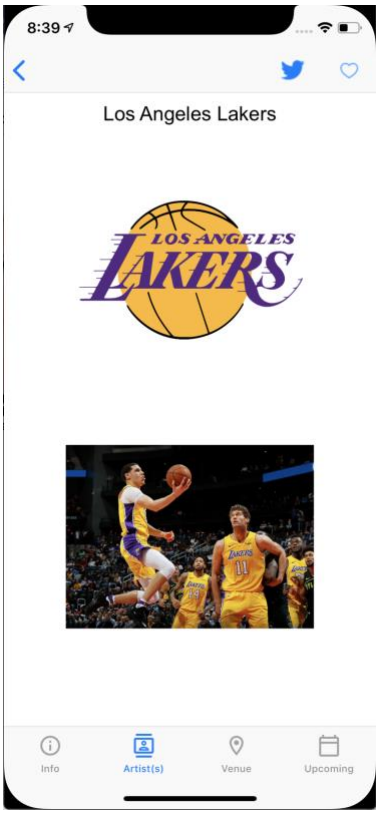

Figure 12: First Team Photos

Figure 13: Second Team Photos

5.3.3 Venue Info Tab

As shown in Figure 14 and 15, there are two elements in this tab:

Details of the venue table: Same as Homework #8.
Map: Same as homework 8, you should render a google map with a maker centered in the map of the venue location.

Use the **Google Map SDK** for iOS:
https://developers.google.com/maps/documentation/iOS-sdk/

The view should be scrollable as details of the venue may be long.

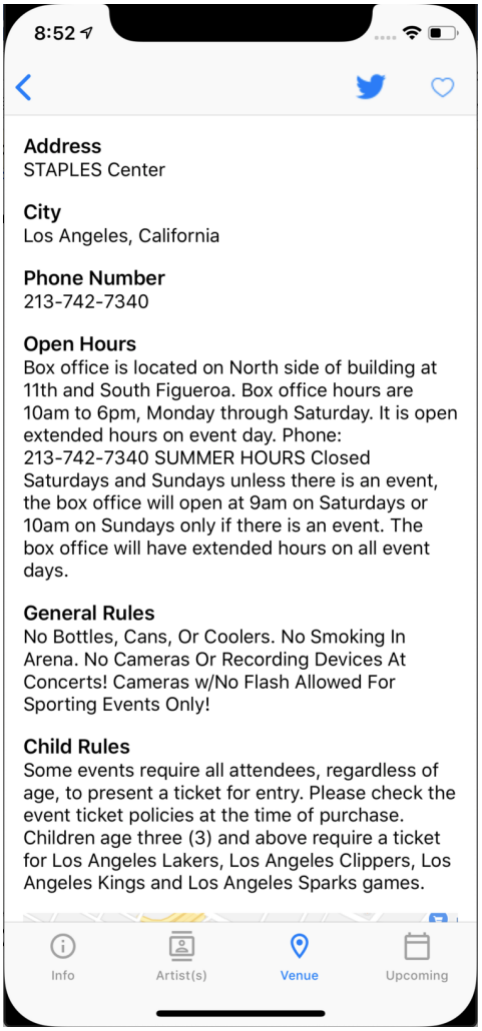

Figure 14: Venue Info Tab

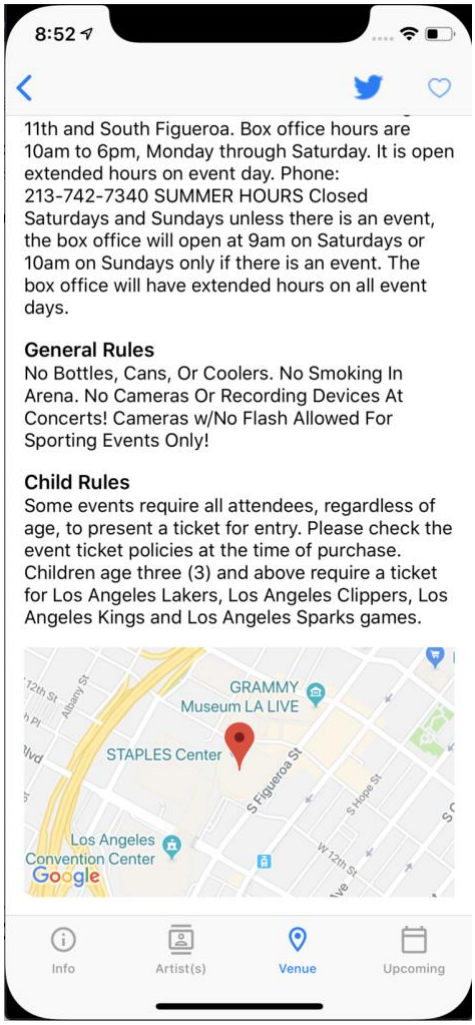

Figure 15: Venue Map

5.3.4 Upcoming Tab

This tab displays the upcoming events of the venue, same as homework 8. Please note, in homework #9, you need to get **at most 5 (first 5)** upcoming events from your server.

As shown in Figure 16, you should use two segmented controls – one to switch between which parameter to sort the upcoming events on and the other to decide in what order to sort it on.

The events are shown in a “UITableView”. Note that each of the cell can be tapped and then a new Safari instance should show the corresponding event’s Songkick link as shown in Figure 17.

In the case of no events data, show “No upcoming events” in the center of the screen, as shown in Figure 18.

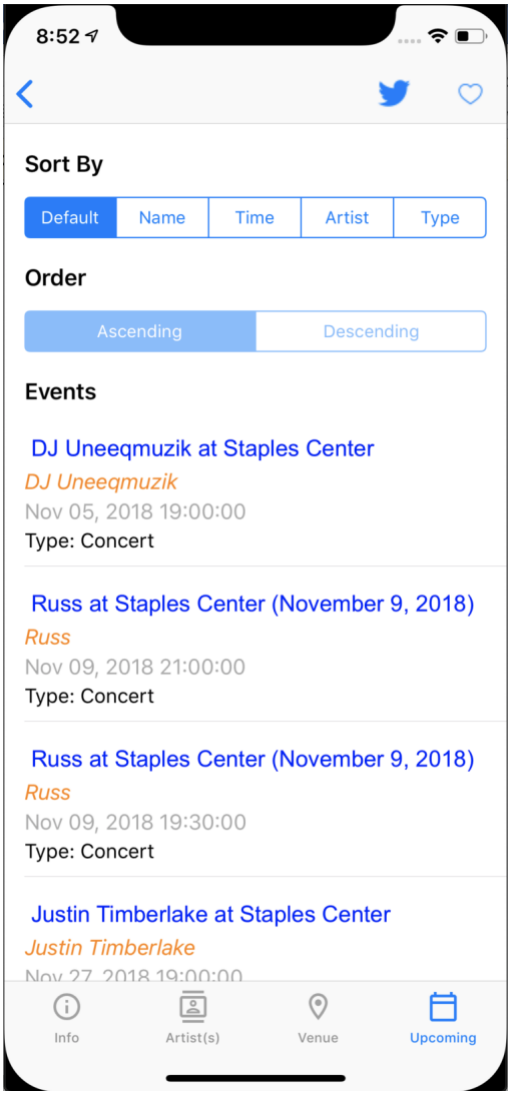

Figure 16: Upcoming Events Tab

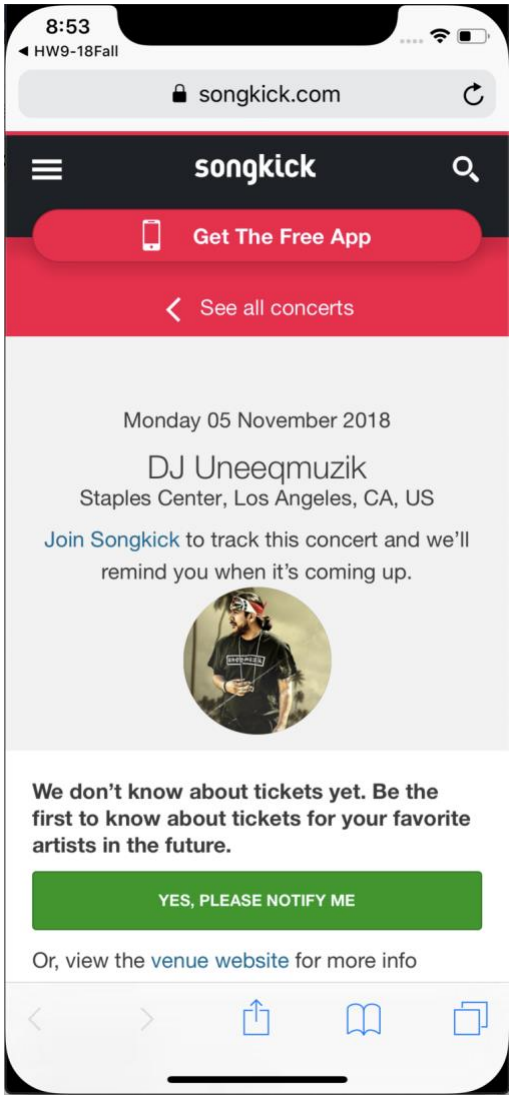

Figure 17: Songkick Page

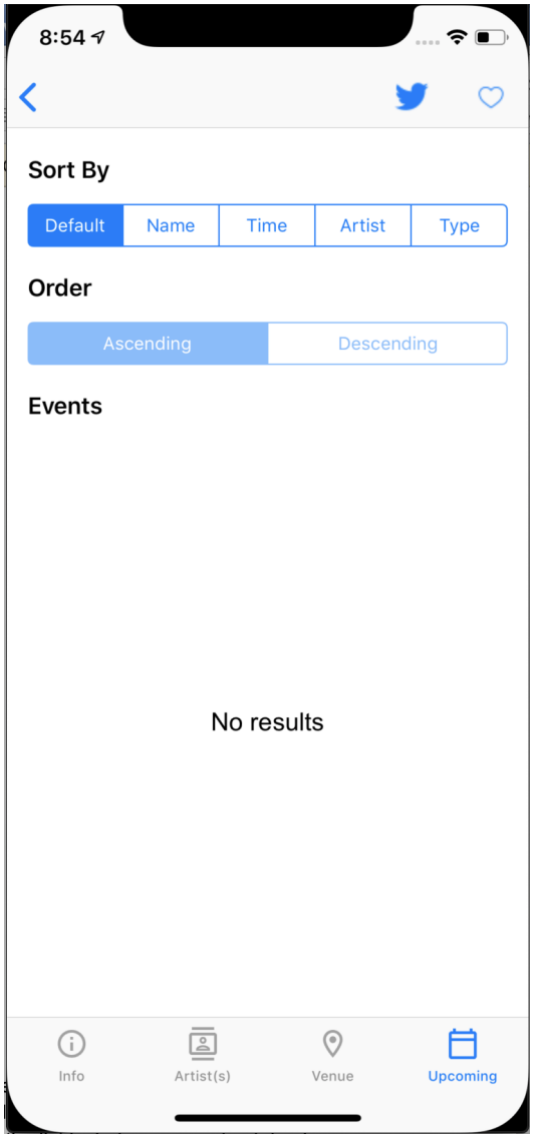

Figure 18: No upcoming events

5.4 Favorites list

Use a segmented control in the main screen to switch between the search page and the favorites page. The favorite events should be displayed in a “UITableView”. Each of the “UITableViewCell” has the exact same structure as the search results cell, but has **NO favorite button** as show in Figure 19. If there are no favorites, a “No Favorites” should be displayed at the center of the screen, as shown in Figure 20.

Tapping on a cell should have the same behavior as tapping on a cell in the search results page.

The user should also be able to remove an event by left-swiping a cell, and clicking the delete option as shown in Figure 21.

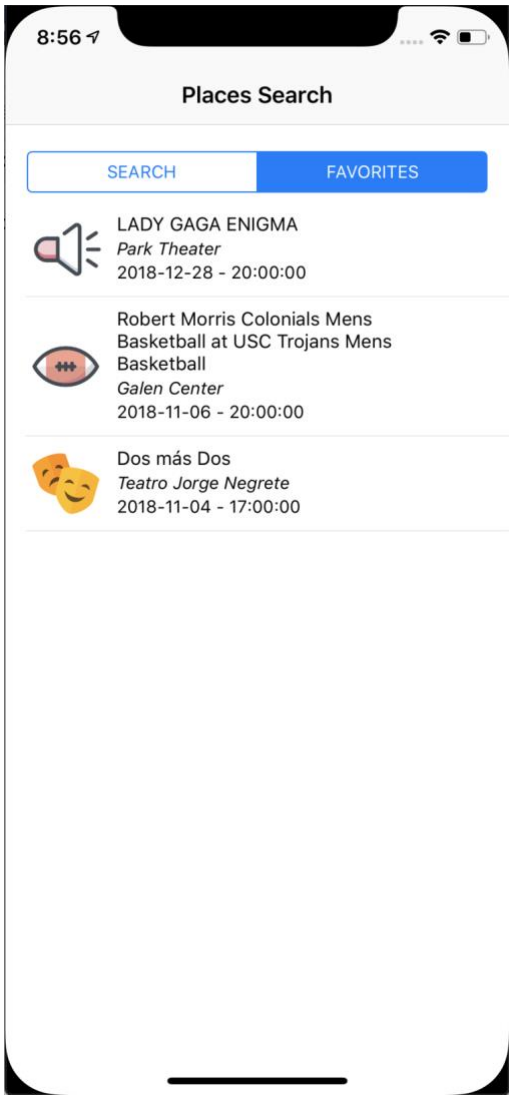

Figure 19: Favorite list

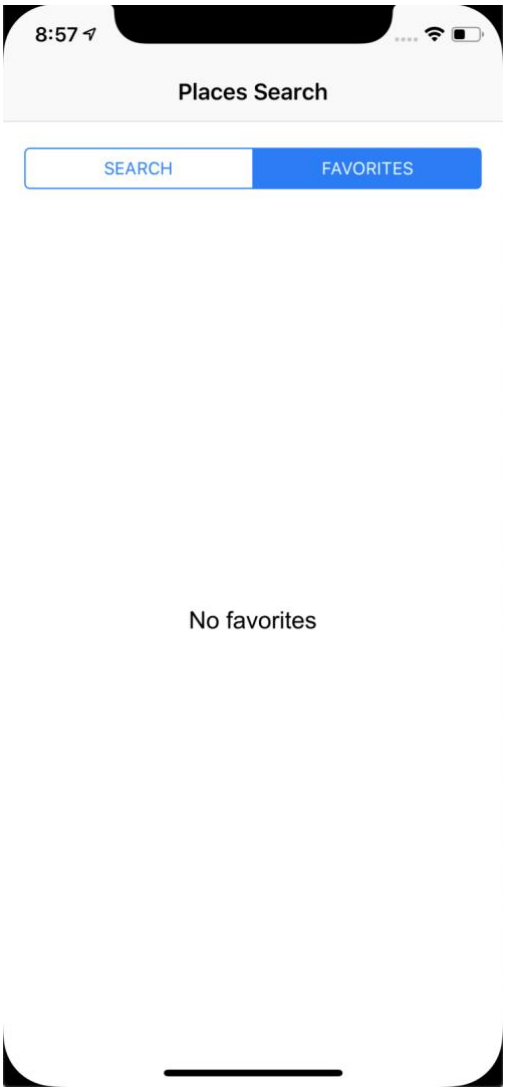

Figure 20: No Favorites

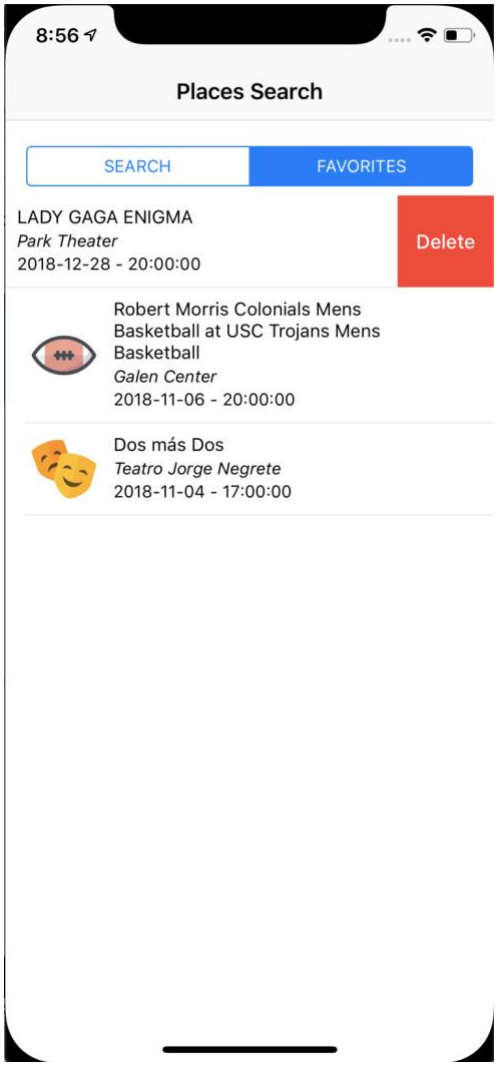

Figure 21: Swipe to remove a place

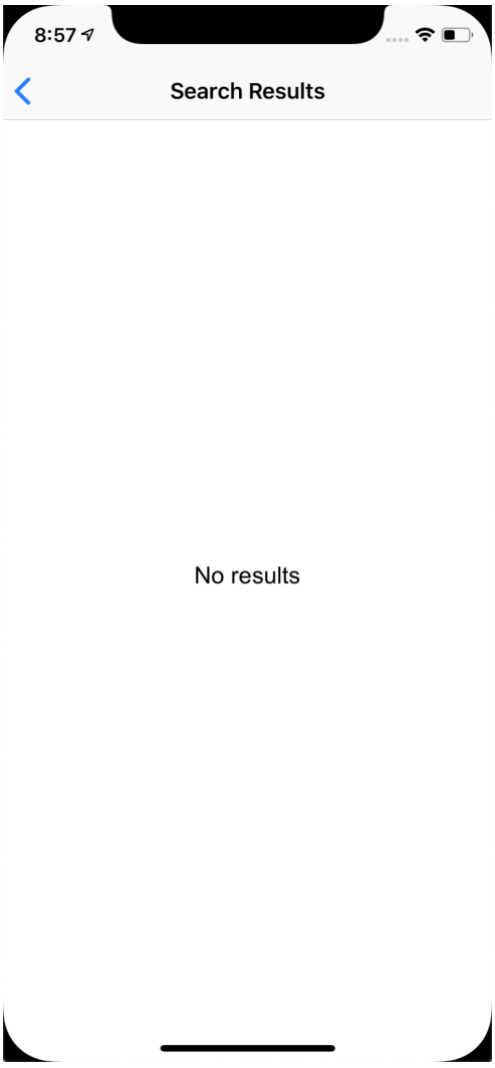

Figure 22: No search results

5.5 Error handling

If no events are found given a keyword, a “no results” should be displayed, as shown in Figure 22.

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 occurs.
- You can only make HTTP requests to your backend (Node.js script on AWS/GAE/Azure) or use 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://csci571.com/hw/hw9/images/ios/imagesets.zip

Favorites

favorite-empty.imageset
favorite-filled.imageset

Category Icons

Music:	music.imageset
Sports:	sports.imageset
Arts & Theatre:	arts.imageset
Film:	film.imageset
Miscellaneous:	miscellaneous.imageset

Tab Icons

Event Tab:	info.imageset
Artists Tab:	contact.imageset
Venue Tab:	location.imageset
Upcoming Tab:	calendar.imageset

Twitter

twitter.imageset

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 Show messages and Spinners

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

https://cocoapods.org/pods/EasyToast

https://github.com/icanzilb/SwiftSpinner

6.3.2 Picker view for the event search category

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

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

6.3.3 Kingfisher for downloading images from the web and caching

https://github.com/onevcat/Kingfisher

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.