Homework 9: Place Search Android App

1. Objectives

- Become familiar with Java, XML and Android Studio for Android app development.
- Build a good-looking Android app.
- Learn to use the Google Places APIs and Android SDK.
- Get familiar with third party libraries like Picasso and Volley.

2. Background

2.1 Android Studio

Android Studio is the official Integrated Development Environment (IDE) for Android application development, based on IntelliJ IDEA - a powerful Java IDE. On top of the capabilities you expect from IntelliJ, Android Studio offers:

- Flexible Gradle - based build system.
- Build variants and multiple apk file generation.
- Code templates to help you build common app features.
- Rich layout editor with support for drag and drop theme editing.
- Lint tools to catch performance, usability, version compatibility, and other problems.
- ProGuard and app-signing capabilities.
- Built-in support for Google Cloud Platform, making it easy to integrate Google Cloud Messaging and App Engine.

More information about Android Studio can be found at:
http://developer.android.com/tools/studio/index.html

2.2. Android

Android is a mobile operating system initially developed by Android Inc., a firm purchased by Google in 2005. Android is based upon a modified version of the Linux kernel. As of December 2013, Android was the number 1 mobile OS, in unit sales, surpassing iOS, while iOS was still the most profitable platform.

The Official Android home page is located at:
http://www.android.com/

The Official Android Developer home page is located at:
http://developer.android.com/

2.3 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 7.5 for .NET, and Apache Tomcat for Java.

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

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, microservices, 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 the 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:

- Download and install Android Studio. Technically, you may use any other IDE other than Android Studio such as Eclipse, but the latest SDKs may not be supported with Eclipse. We will not be providing any help on problems arising due to your choice of alternate IDEs.

  Recent versions of Android Studio come bundled with OpenJDK and you no longer need to set up your JDK separately.
- You will need to enable "Developer Options" and "USB debugging" if you plan to use an actual device for debugging. It is fairly simple to do so and is illustrated in this Stack Overflow post:

https://stackoverflow.com/questions/16707137/how-to-find-and-turn-on-usb-debugging-mode-on-nexus-4

Such setup is not needed if you plan to use the emulator. Everything should just work out of the box.

4. High Level Design

This homework is a mobile app version of Homework 8. In this exercise, you will develop an Android application, which allows users to search for a place, look at information about it, save some as favorites and post on Twitter about the place. You should reuse the backend service (PHP/Node.js) you developed in HW8 and follow the same API call requirements.

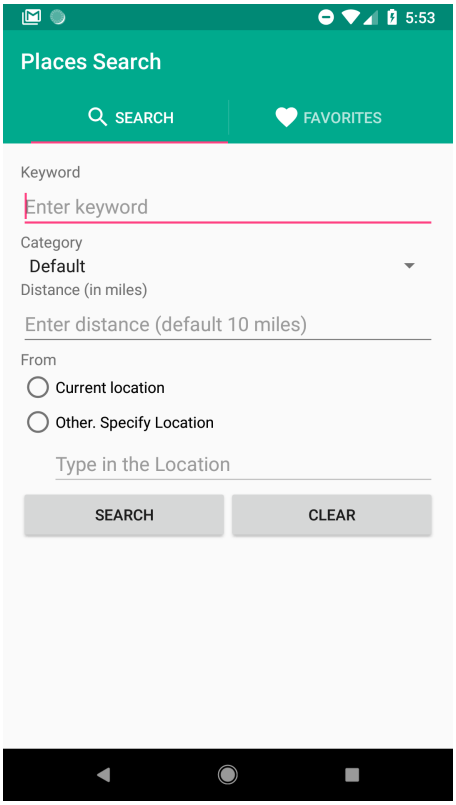

Figure 1: The Place Search App

5. Implementation

5.1 Search Form

You must replicate the Search form shown in Figure 1.
The interface consists of the following:

- **Keyword**: An EditText view allowing the user to enter the keyword.

- **Category**: A Spinner view allowing the user to choose a category. When the user taps on this field, a dropdown list should display for selecting a category, as shown in Figure 2. Make sure you include all the categories in homework 8.
- **Distance**: An EditText view allowing the user to enter the distance (in miles).
- **From**: An AutoCompleteTextView component allowing the user to enter a location. When the user enters text into this field, autocomplete suggestions should be presented as shown in Figure 3. The AutoCompleteTextView should only be enabled when the corresponding RadioButton is checked.
- **Search**: A 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 validation is unsuccessful, appropriate messages should be displayed and no further requests would be made to the server.
- **Clear:** A button to clear the input fields and reset them to default values when applicable. It should also remove any validation error messages.

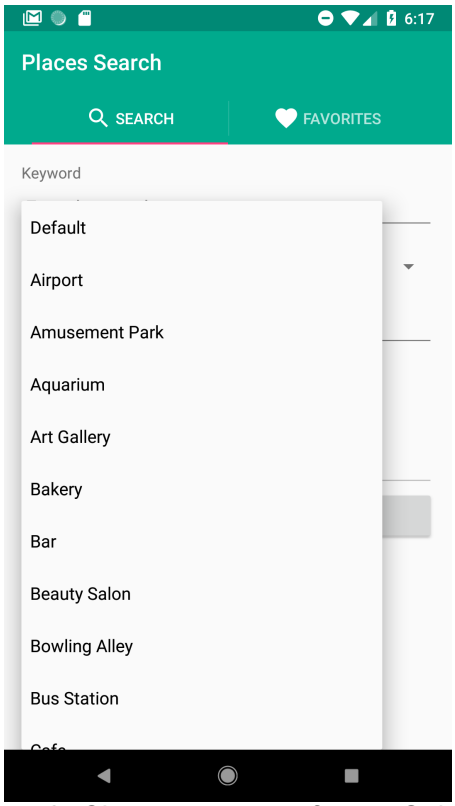

Figure 2: Choose category form a Spinner

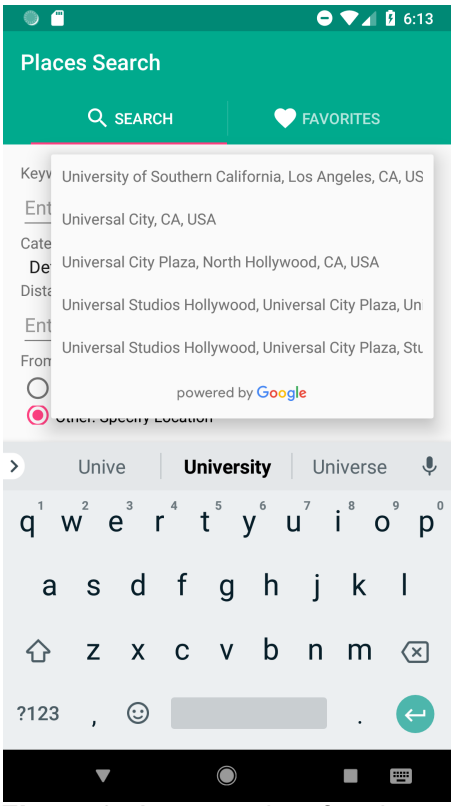

Figure 3: Autocomplete for places

The validation for empty keyword has to be implemented. If the user does not enter anything in the EditText or just enters some empty spaces, when he/she presses the Search button you need to display an appropriate message to indicate the error, as

shown in Figure 4. The same should be done when "**From**" location is not entered, and that option in enabled using the radio button as shown in Figure 5.

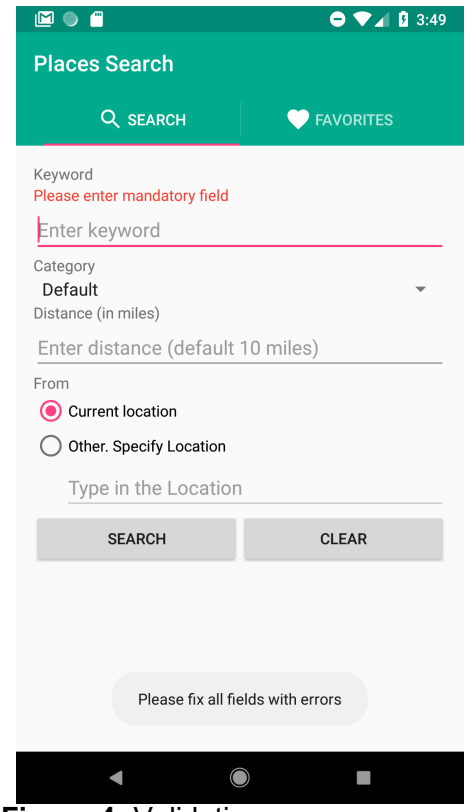

Figure 4: Validation error messages

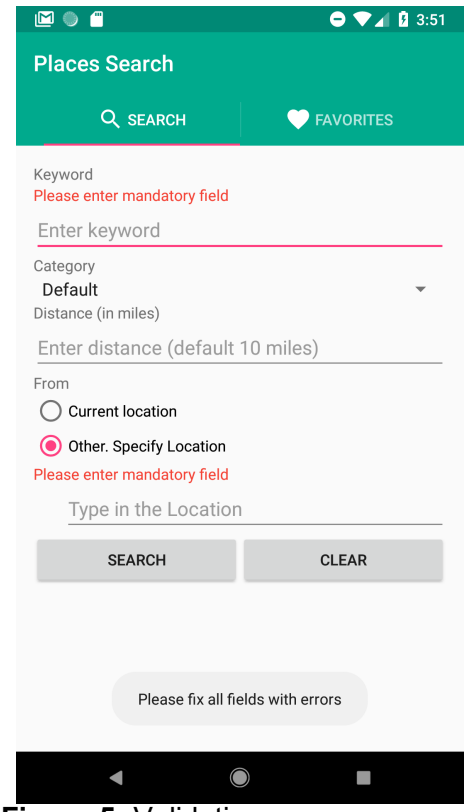

Figure 5: Validation error messages

5.2 Search Results

When the user taps the SEARCH button, your app should display a ProgressDialog before it is ready to show the results page, as shown in Figure 6. After you get the data from your backend, hide the ProgressDialog and display the result page as a list using RecyclerView, as shown in Figure 7.

Each of the item in the list 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.

Tapping the favorite button (the heart) would add the corresponding place into the favorites list, and a message should be displayed at the bottom of the app using a

Toast, as shown in Figure 8. Tapping the button again would remove that place from the favorites list, and a similar message should also be displayed to indicate the place has been removed from the favorites list.

For the pagination, there should be two buttons “PREVIOUS” and “NEXT” below the results table, as shown in Figure 7. If it is the first time a page is being fetched, you must use a ProgressDialog while the next page is being fetched, as shown in Figure 9. Note that the previous or next button(s) should be disabled if there’s no previous page or no next page.

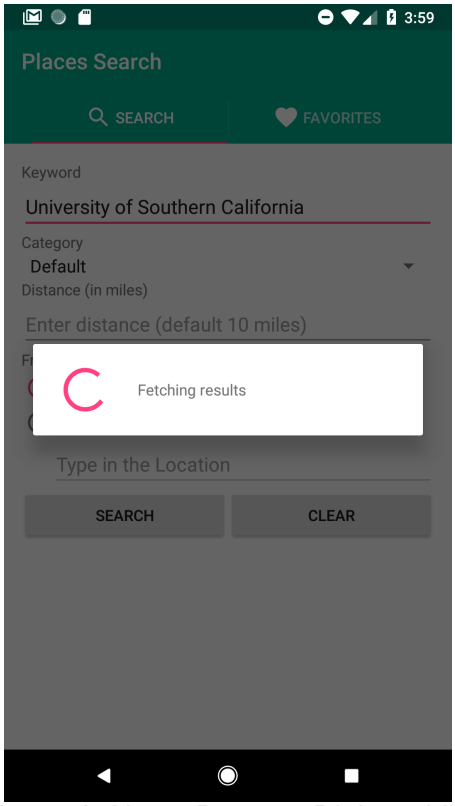

Figure 6: Use a ProgressDialog while fetching results

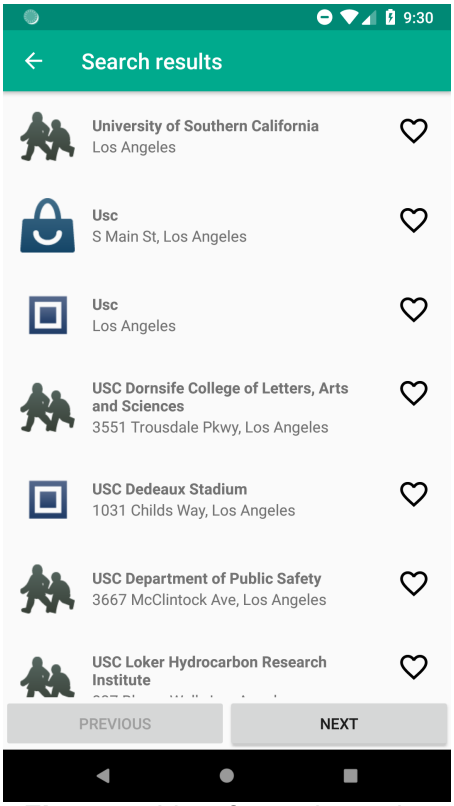

Figure 7: List of search results

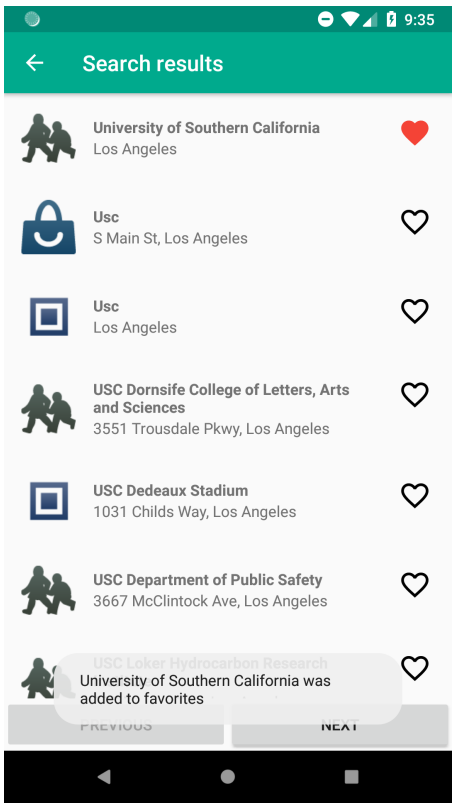

Figure 8: Message for adding favorites

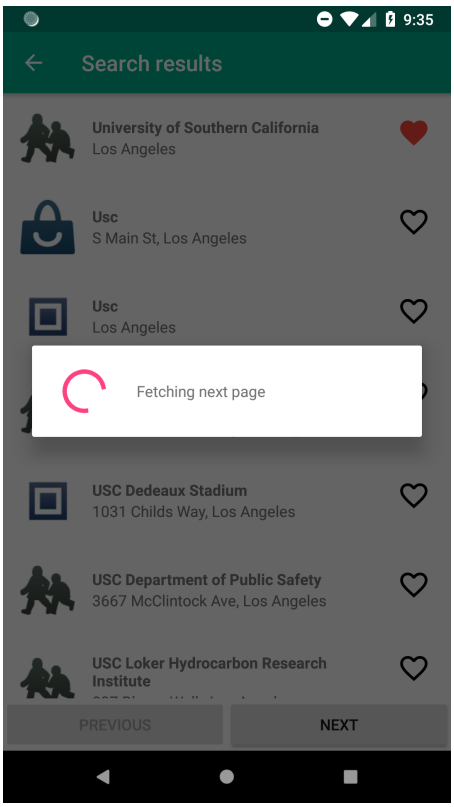

Figure 9: Loading next page

5.3 Place Details

Tapping on an item in the RecyclerView list in the results table should show details of that place with four tabs: Info, Photos, Map and Reviews. Note that the a ProgressDialog should be shown before you are ready to display all the place details.

The tabs should be attached to the ActionBar and a ViewPager should be used to host all the tabs, as shown in Figure 10. The users should be able to switch between tabs by both swiping and tapping on a tab. The ActionBar should also include the following elements:

- A **back button,** which navigates back to the search results list.
- A **title**, which is the name of the place.
- A **share button** (↗), to share the place details on Twitter. Once the button is tapped, a web page should open to allow the user to share the place information on Twitter, as shown in Figure 11. This should work the same as homework 8.
- A **favorite button** to add/remove the place to/from the favorite list, and display a Toast at the bottom of the screen.

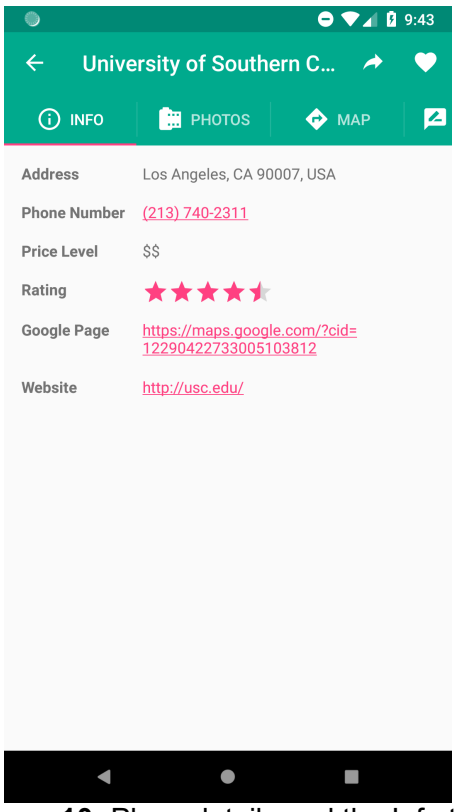

Figure 10: Place details and the Info tab

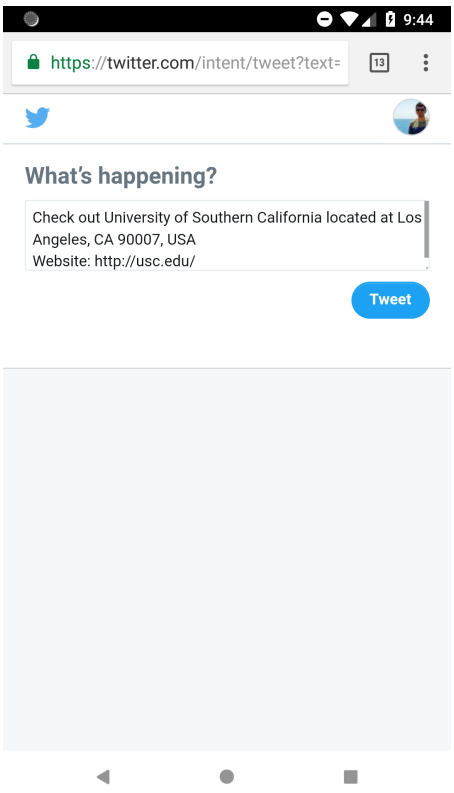

Figure 11: Share place on Twitter

5.3.1 Info Tab

The fields in Figure 10 should be shown in the Info tab: Address, Phone Number, Price Level, Rating, Google Page and Website. See homework 8 for more details about each field.

5.3.2 Photos Tab

You need to show the photos of the place as shown in Figure 12. The Places API for Android documentation shows how one can fetch the photos of the place.
https://developers.google.com/places/android-api/photos

Figure 12: Photos Tab

5.3.3 Map Tab

As shown in Figure 13, there are three elements in this tab:

- **From**: an EditText view, when text is entered, autocomplete suggestions should be shown.
- **Travel mode:** a Spinner to pick the travel mode.
- **Map:** the maps should be rendered using the Google Maps SDK for Android. https://developers.google.com/maps/documentation/android-api/

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 suggestions, 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 14.

Switching among different travel modes should update the paths on the map, as shown in Figure 15.

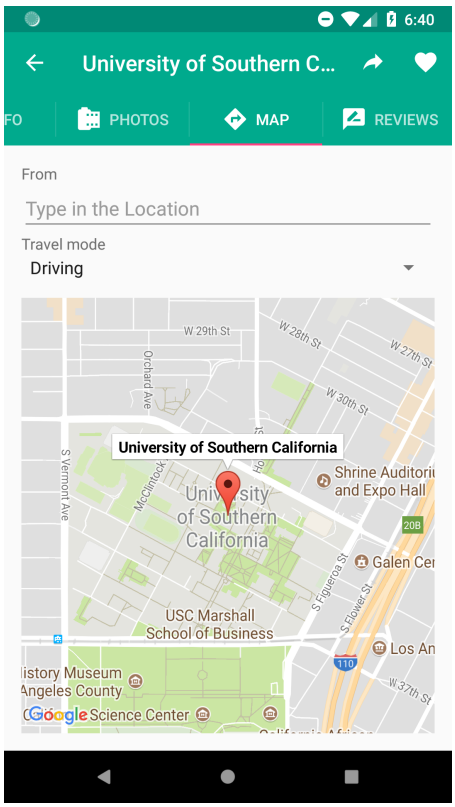

Figure 13: Map tab

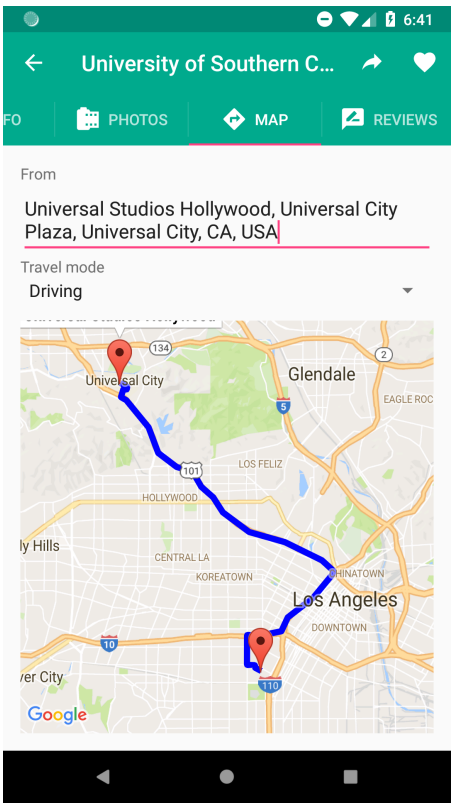

Figure 14: Path shown on the map

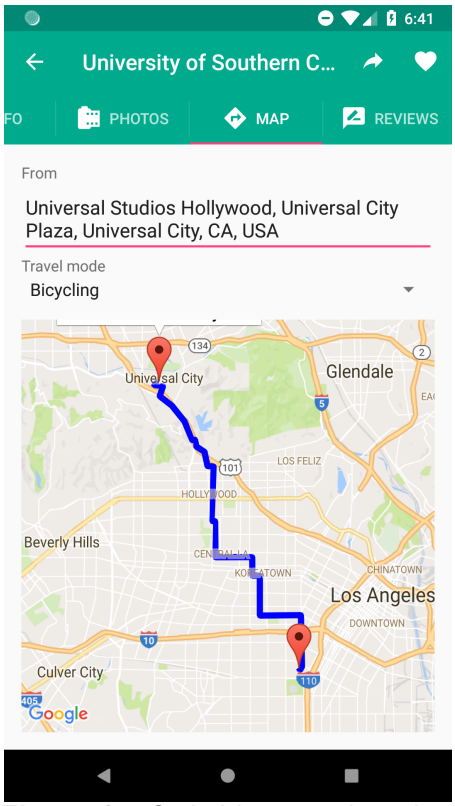

Figure 15: Switching travel modes

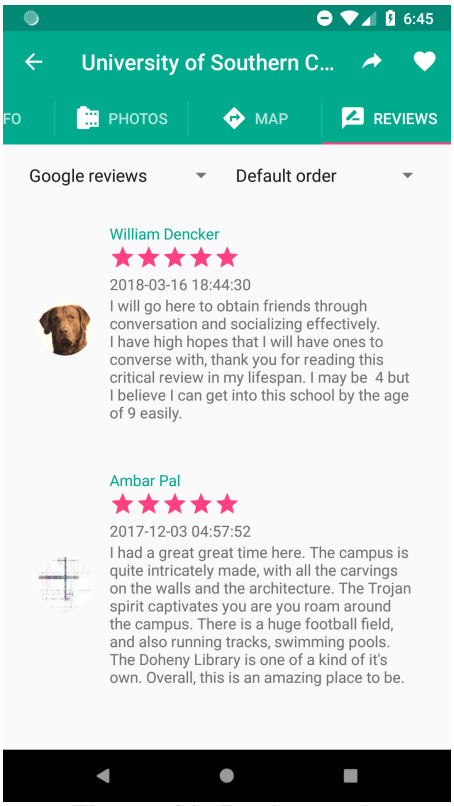

Figure 16: Reviews tab

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 16, you should use a Spinner to switch between Google reviews and Yelp reviews. Another Spinner allow users to sort the reviews by default order, rating and review date in ascending or descending order.

The reviews are shown in a list using a RecyclerView. Note that each of the cells can be tapped and then a web page should be opened and navigates to the tapped review, as shown in Figure 17.

In the case of no reviews data, show "no reviews" in the center of the screen, as shown in Figure 18.

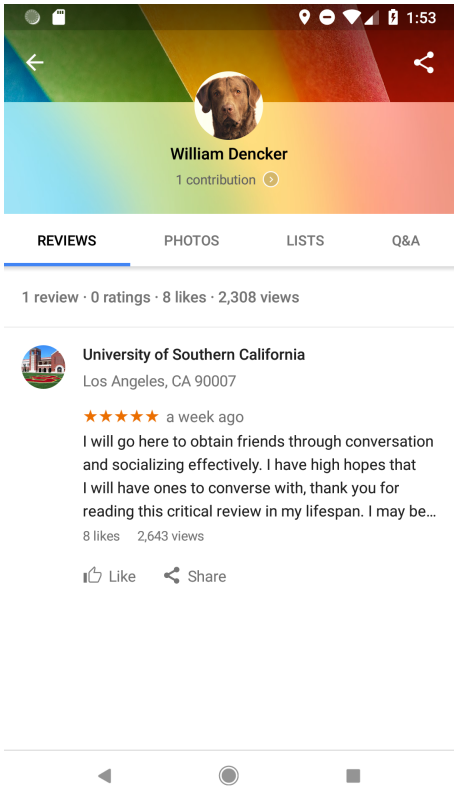

Figure 17: Webpage for the review

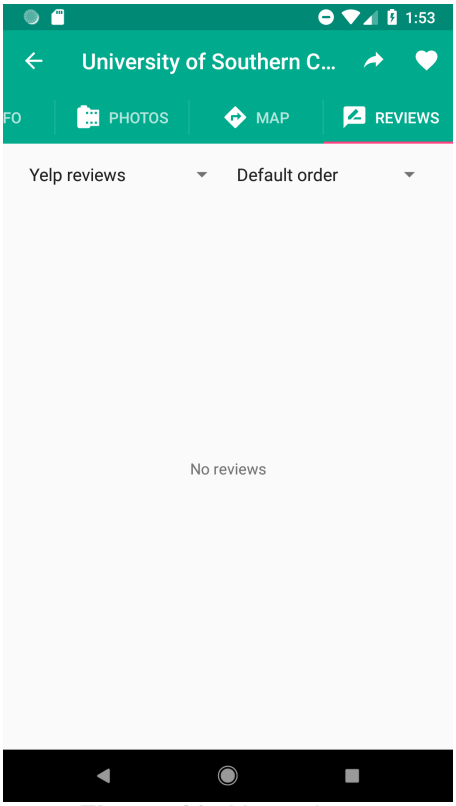

Figure 18: No reviews

5.4 Favorites list

Use Tabs with a ViewPager on the main screen to switch between the search page and the favorites page. The favorite places should be displayed in a list using a RecyclerView. Each of the items in the list includes a places catalog image, place name and address, as shown in Figure 19. If there are no favorite places, "No Favorites"

should be displayed at the center of the screen, as shown in Figure 20.

Like in search results, pressing the favorite icon here should remove the place from the favorites list.

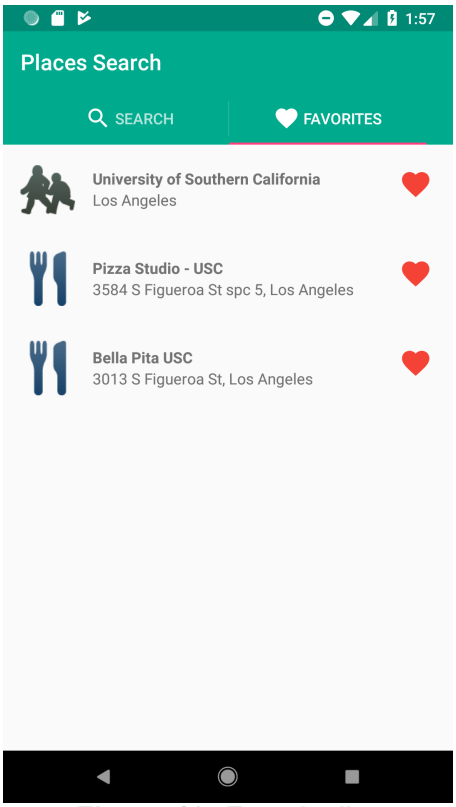

Figure 19: Favorite list

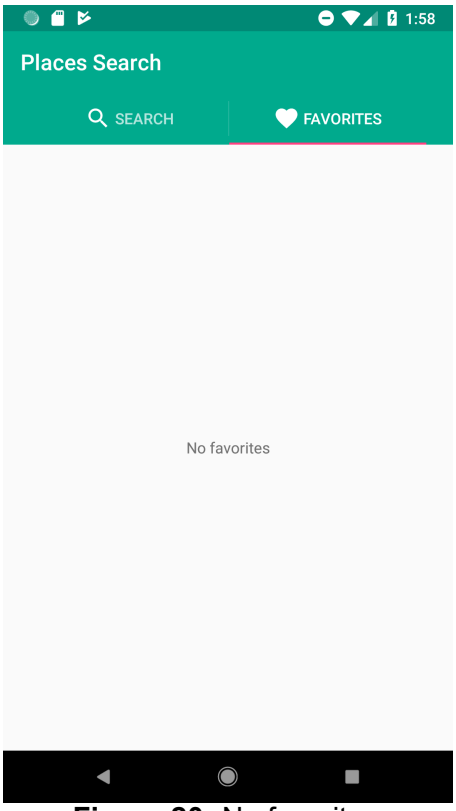

Figure 20: No favorites

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 an error occurs (no network, API failure, cannot get location etc.), an appropriate error messages should be displayed at the bottom of screen using a Toast.

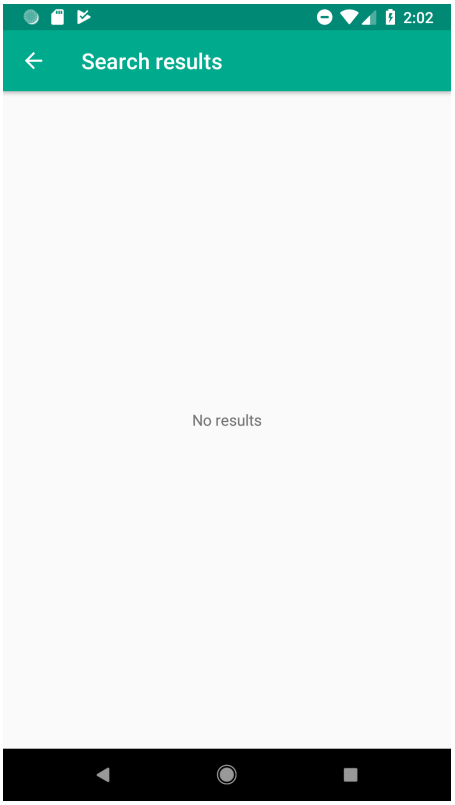

Figure 21: No search results

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) on AWS/GAE or use the Google Places API for Android or Google Map SDK for Android.
- All HTTP requests should be asynchronous and should not block the main UI thread. You can use third party libraries like Volley to achieve this in a simple manner.

6. Implementation Hints

6.1 Images

The images needed for the homework as provided as vector drawables and are available here:

http://cs-server.usc.edu:45678/hw/hw9/images/android/heart_fill_red.xml
http://cs-server.usc.edu:45678/hw/hw9/images/android/heart_fill_white.xml
http://cs-server.usc.edu:45678/hw/hw9/images/android/heart_outline_black.xml
http://cs-server.usc.edu:45678/hw/hw9/images/android/heart_outline_white.xml

http://cs-server.usc.edu:45678/hw/hw9/images/android/info_outline.xml
http://cs-server.usc.edu:45678/hw/hw9/images/android/maps.xml
http://cs-server.usc.edu:45678/hw/hw9/images/android/photos.xml
http://cs-server.usc.edu:45678/hw/hw9/images/android/review.xml
http://cs-server.usc.edu:45678/hw/hw9/images/android/search.xml
http://cs-server.usc.edu:45678/hw/hw9/images/android/share.xml

6.2 Getting current location

For your location fetching code to work, you must request the permission from the user. You can read more about requesting permissions here:
https://developer.android.com/training/permissions/requesting.html

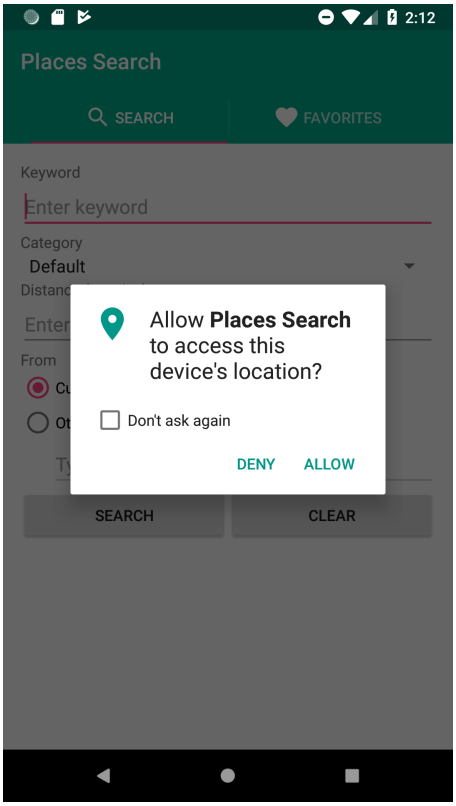

Figure 22: Requesting location permission

For those using the emulator, you may need to mock the location. This can be done from the emulator settings.

6.3 Third party libraries

Almost all functionalities of the app can be implemented without using third party libraries, but sometimes using them can make your implementation much easier and quicker. Some libraries you may have to use are:

6.3.1 Google Play services

You will need this for various features like getting the current location and using Google Maps in your app.

You can learn about setting it up here:
https://developers.google.com/android/guides/setup

6.3.2 HTTP requests

Volley and/or Picasso can be helpful with asynchronously loading data and images into ImageViews. You can learn more about them here:

https://developer.android.com/training/volley/index.html
http://square.github.io/picasso/

6.3.3 Working with the AutoCompleteTextView

Working with the AutoCompleteTextView to show the places suggestions might be a little challenging. This tutorial goes over how it is done so that you get an idea of how to go about it.

http://www.zoftino.com/google-places-auto-complete-android

7. What to Submit

You should also ZIP all your source code (the java/ and res/ directories excluding the vector drawables that we provide to you) and submit the resulting ZIP file by the end of the demo day.

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, on the Announcement page and on Piazza.

****IMPORTANT****

All videos are part of the homework description. All discussions and explanations on 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.