Homework 9
Event Finder App

Prof. Marco Papa

Developed and Designed by
Dr. Marco Papa, Aashreen Raorane and Jerry Allan Akshay

Homework 9: Event Search Android App

1. Objectives

- Become familiar with Java, JSON, Android Lifecycle, and Android Studio for Android app development.
- Build a good-looking Android app.
- Learn the essentials of Google's Material design rules for designing Android apps
- Learn to use the Google Maps APIs and Android SDK.
- Get familiar with third party libraries like Picasso, Glide 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 Nov 2018, 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 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, 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/nodejs

2.5 Microsoft Azure

The Azure cloud platform has more than 200 products and cloud services designed to help you bring new solutions to life—to solve today's challenges and create the future. Build, run, and manage applications across multiple clouds, on-premises, and at the edge, with the tools and frameworks of your choice.

To learn more about the Azure services, visit this page:
https://azure.microsoft.com/en-us/solutions/

To learn more about Azure support for Node.js visit this page:
https://docs.microsoft.com/en-us/azure/devops/pipelines/targets/webapp?view=azure-devops&tabs=yaml#deploy-a-javascript-nodejs-app

3. Prerequisites

This homework requires the use of the following components:

- Download and install Android Studio. Technically, you may use any 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.
- You must use the **emulator, preferably Pixel 5 with API 31**. Everything should just work out of the box.
- If you are new to Android Development, hints are going to be your best friends!

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 the events, look at information about it, save some as favorites and post on Facebook and Twitter about the event. You should reuse the Node.js backend service you developed in Homework 8 and follow the same API call requirements. These features among others are spread out over multiple activities and fragments. There is no hard requirement on the number of activities/fragments used as long as the behaviour of the app developed is similar to the reference implementation.

This homework makes use of the backend API's that you developed as part of Assignment 8. So, you can use the same Node.js backend as Homework #8. In case you need to change something in the Node backend, make sure you do not break your Angular assignment (or deploy a separate copy) as the grading for homework will not be finished at least until 1 week later.

We suggest you use Java. Kotlin is allowed but will not be supported in piazza.

PS: This app has been designed and implemented in a Pixel 5 emulator by using SDK API 30. It is highly recommended that you use the same virtual device and API to ensure consistency.

The demo will be on an emulator using Zoom recorded video, no personal devices allowed, The recording guidelines are on D2L

5. Implementation

5.1 App Icon and Splash Screen

In order to get the app icon/image, please see the hints section. The app begins with a welcome screen (**Figure 2**) which displays the icon provided in the hint above.

This screen is called Splash Screen and can be implemented using many different methods. The simplest is to create a resource file for the launcher screen and add it as a style to AppTheme.Launcher Please refer to **Figure 1** and **Figure 2**.

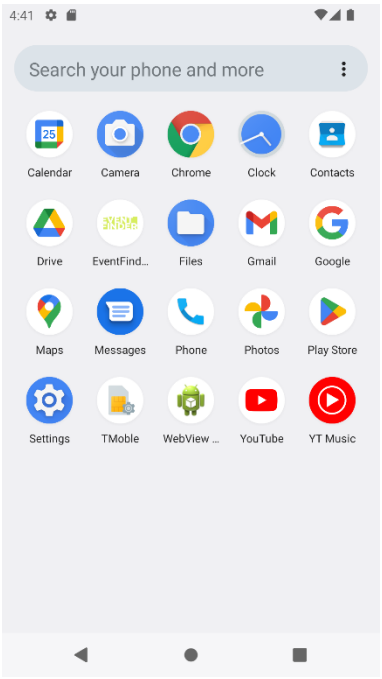

Figure 1: App Icon

Figure 2: The Event Finder Splash Screen

5.2 Search Form

The initial interface is shown in **Figure 3**. There are two tabs in this interface: Search and Favorites.

For the search tab, it has the following fields:

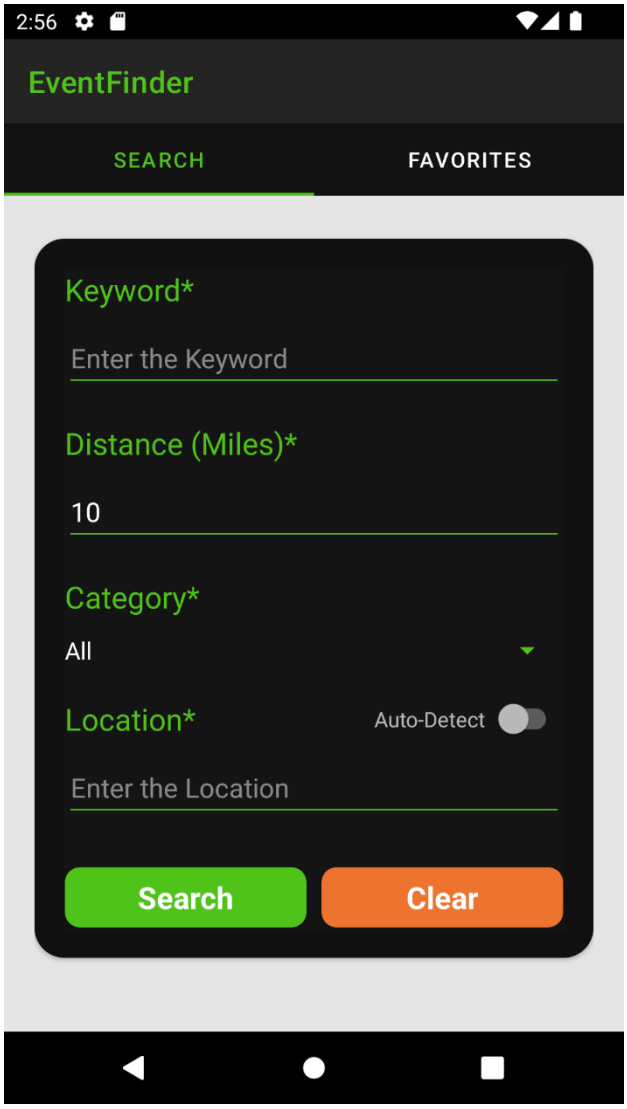

Figure 3: The Event Search App

- **Keyword**: An AutoCompleteTextView component allowing the user to enter the keyword. It provides the autocomplete function as shown in **Figure 4**. Make sure you use the same API as Homework 8. See section 6.3.5 for hints.
- **Distance**: An EditText (type:number) view allowing the user to enter the distance and the default value is 10.
  A Spinner for the user to select units: “miles” or “kilometers”.
- **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 5. Make sure you include all the categories in homework 8. Default is set to “All”
- **Location**: One Radio Button to select “Auto-Detect” location. On selecting the Auto-Detect location button, the text input for location is hidden and behind the scenes,

the current location of the device is fetched. You may use IP Info for this or play around with Android's Location Services API.

If the checkbox is not checked, it is expected that the user provides a location as an input string that is used as part of the search query - same as we did in Homework 8.

- **Search**: A 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 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.

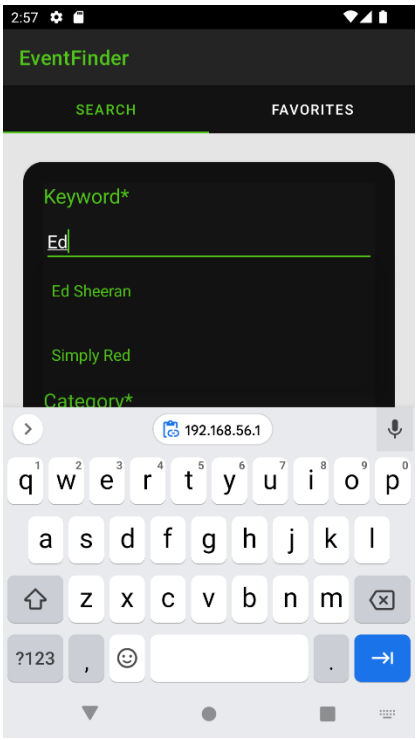

Figure 4: Autocomplete for keyword

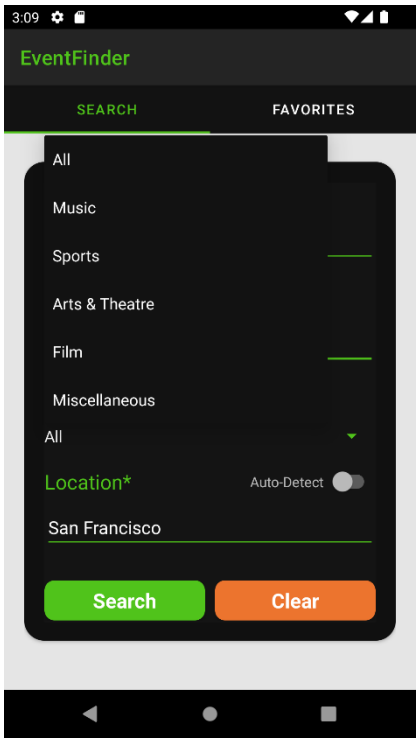

Figure 5: Category Spinner

The validation for an **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 6**. The same should be done when "**Location**" is not entered, and that option is enabled using the radio button.

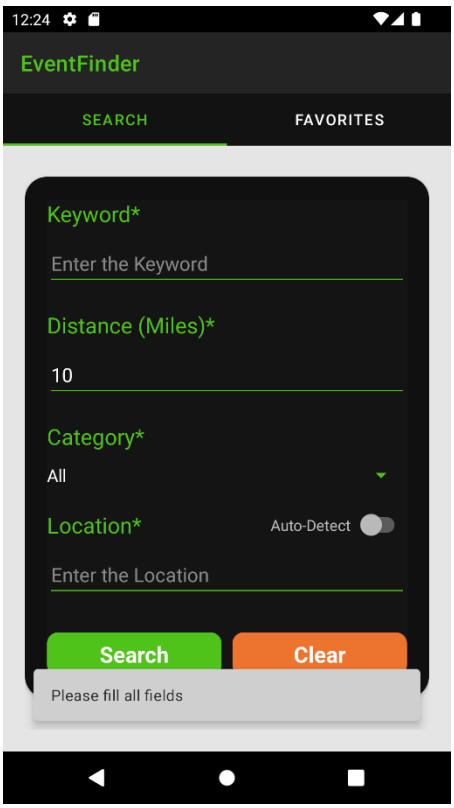

Figure 6: Validation error messages

5.3 Search Results

When the user taps the SEARCH button and all validations pass, your app loads the search results page. Before you get the data from your backend server, a progress bar should display on the screen, as shown in **Figure 7**. After you get the data from your backend, hide the ProgressBar and display the result page as a list using RecyclerView or ListView, as shown in **Figure 8**. The RecyclerView or ListView must be scrollable. They also provide a 'back button' to navigate back the search/favorite interface.

Each of the item in the list should have the following:

- Category image (See the mapping between segment and icons on section 6. 1)
- Name of the event
- Name of the venue
- Date and Time of the event
    - The time should be displayed in AM and PM format
- A heart-shaped "Favorite" button
  See homework 8 for more details about these fields.

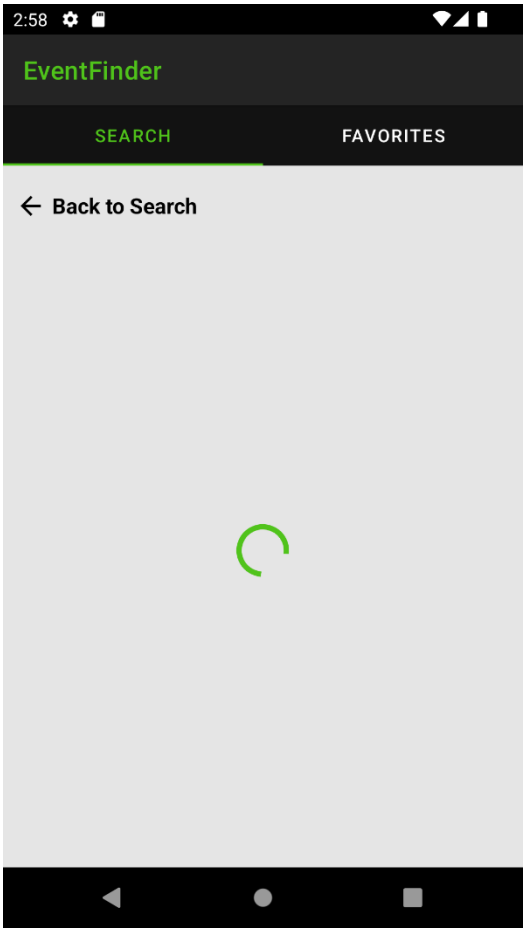

Figure 7: ProgressBar while fetching results

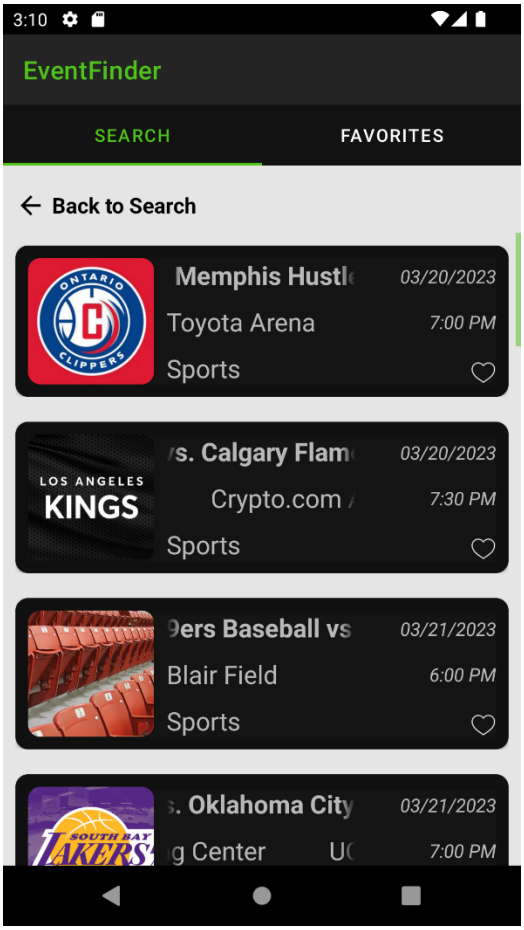

Figure 8: List of search results

Tapping the favorite button (the heart) would add the corresponding event into the favorites list, and a message should be displayed at the bottom of the app using a Toast, as shown in **Figure 9**. Tapping the button again would remove that event from the favorites list, and a similar message should also be displayed to indicate the event has been removed from the favorites list, as shown in **Figure 10**.

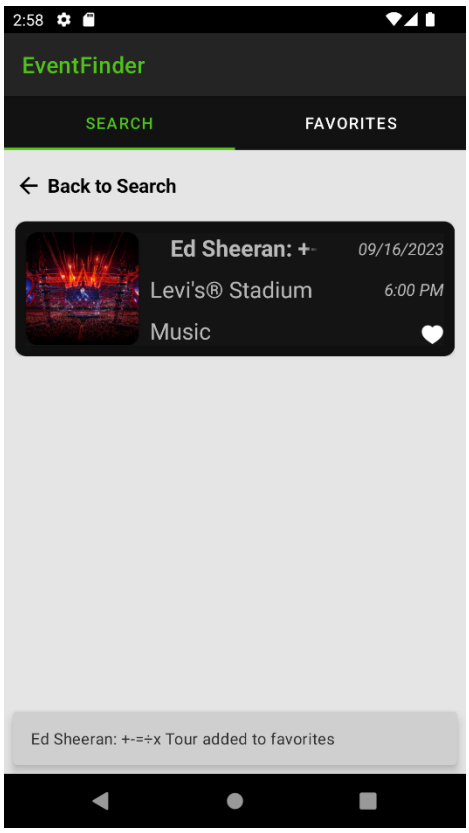

Figure 9: Message for adding favorites

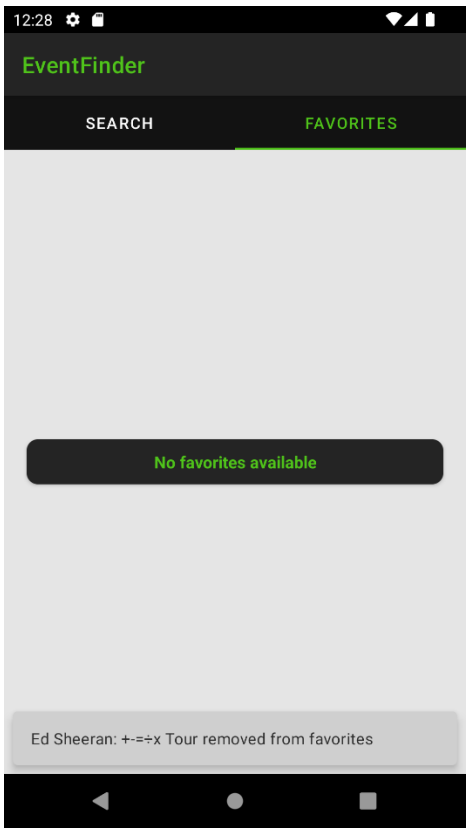

Figure 10: Message for removing favorites

NOTE: All the texts in the app should be a scrolling text i.e. whenever a text is encountered which does not fit in the given space (e.g. Event Name or Artists/Team section) it should scroll automatically.

5.4 Event Details

Tapping on an item in the result list should show details of that event with three tabs: Details, Artists, and Venue. Note that the ProgressBar should be shown on each tab before you are ready to display the corresponding tab.

The tabs should be attached to the ActionBar and a ViewPager should be used to host all the tabs, as shown in **Figure 11**. 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 event.

- A **favorite button** to add/remove the event to/from the favorite list, and display a Toast at the bottom of the screen. See video for more detail.
- A **facebook button**, to share the event details on Facebook. Once the button is tapped, a web page should open to allow the user to share the event information on Facebook, as shown in **Figure 12(a)**. This should work the same as homework 8.
- A **twitter button**, to share the event details on Twitter. Once the button is tapped, a web page should open to allow the user to share the event information on Twitter, as shown in **Figure 12(b)**. This should work the same as homework 8.

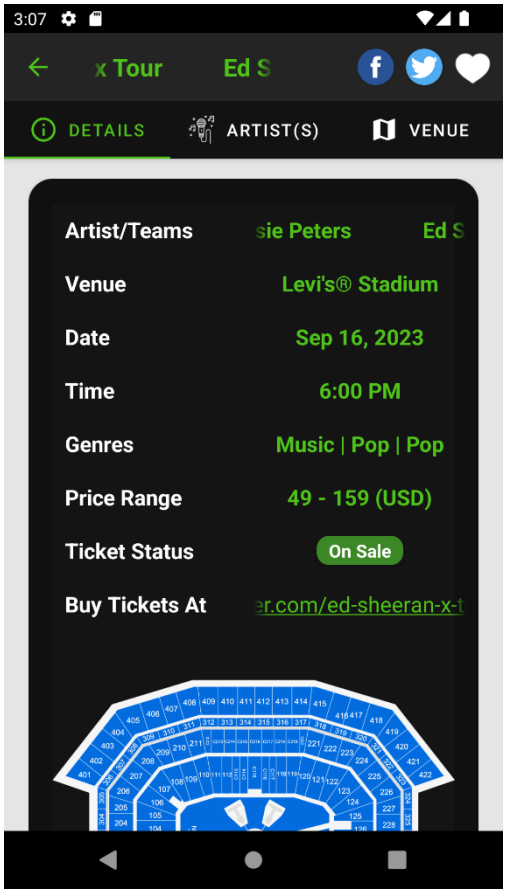

Figure 11: Event details

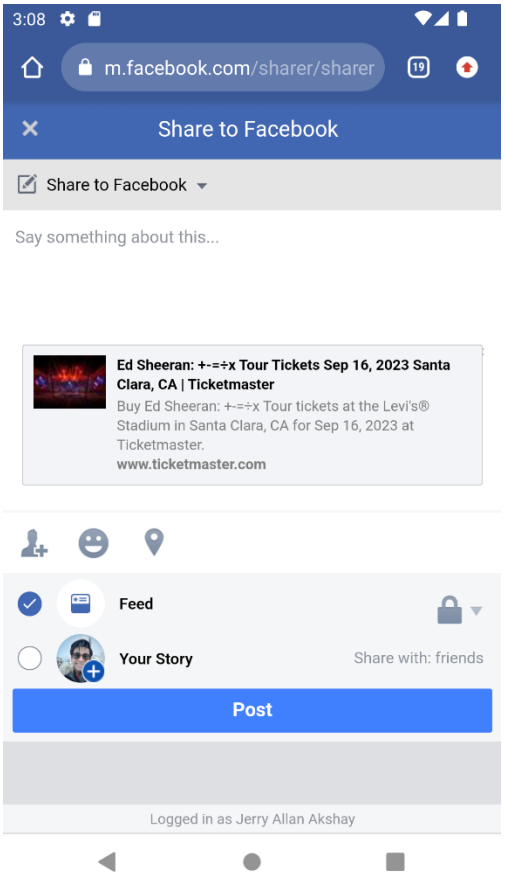

Figure 12(a): Share Event on Facebook

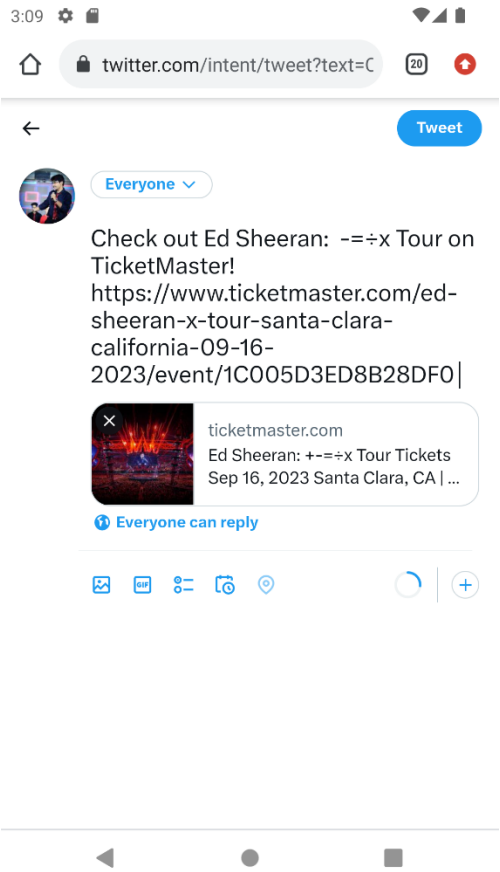

Figure 12(b): Share Event on Twitter

5.4.1 Event Tab

The fields in **Figure 11** should be shown in the Event tab. See homework 8 for more details about each field.

5.4.2 Artist(s) Tab

Same as in homework 8. For each artist show

- Artist Name
- Followers - The followers count should be displayed in M(Millions) or K(Thousands) depending on the value.
- Popularity – with a red progress ring around the popularity number
- Link to the Artist's Spotify page
- Top 3 popular albums' cover

For the events that are not related to music category, the Artist tab will not display artist info. Rather it should display a message "Artist/Music data unavailable."

See **Figure 13**, **14** and **15**. You could use Volley Network ImageView, Picasso or Glide to load the image. See more on section 6.3.

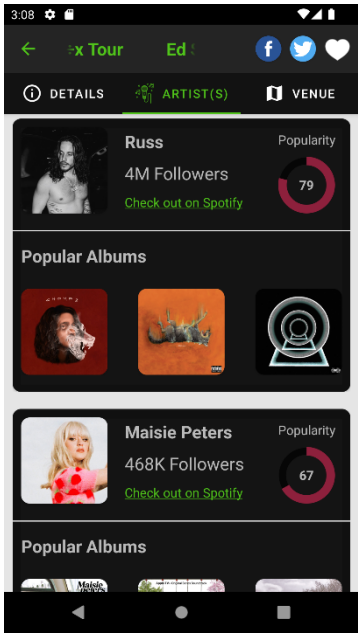

Figure 13: Artists Tab with Music Artist

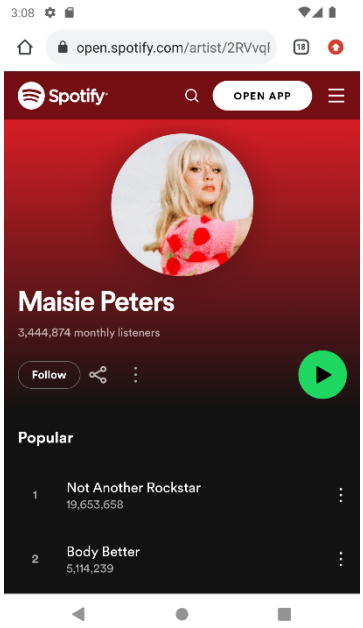

Figure 14: Artist's Spotify page

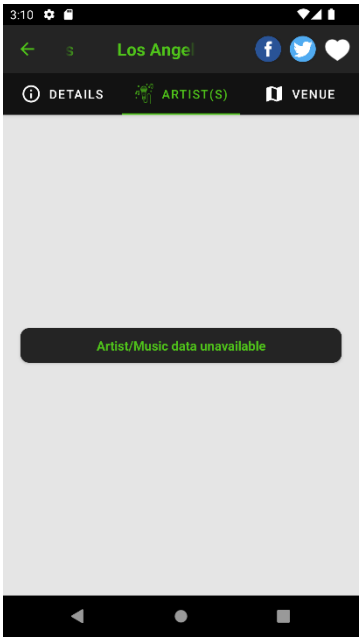

Figure 15: Artists/Teams tab for an event that is not music-related

5.4.3 Venue Tab

As shown in **Figure 16** and **17**, there are two elements in this tab:

- **Details of the venue table with map:** Same as homework 8 with below mentioned fields.
    - Name
    - Address
    - City/State
    - Contact Info
    - Map - you should render a google map with a maker centered in the map of the venue location. The maps should be rendered using the Google Maps SDK for Android. https://developers.google.com/maps/documentation/android-api/

- **Other details:** Same as homework 8, with below mentioned fields.
    - Open Hours
    - General Rule
    - Child Rule

This view should be scrollable since the details of the venue table may be too long.

Paragraphs such as the Venue Rules section should have “show more/show less” functionality for the complete area of the text as instructed in HW 8.

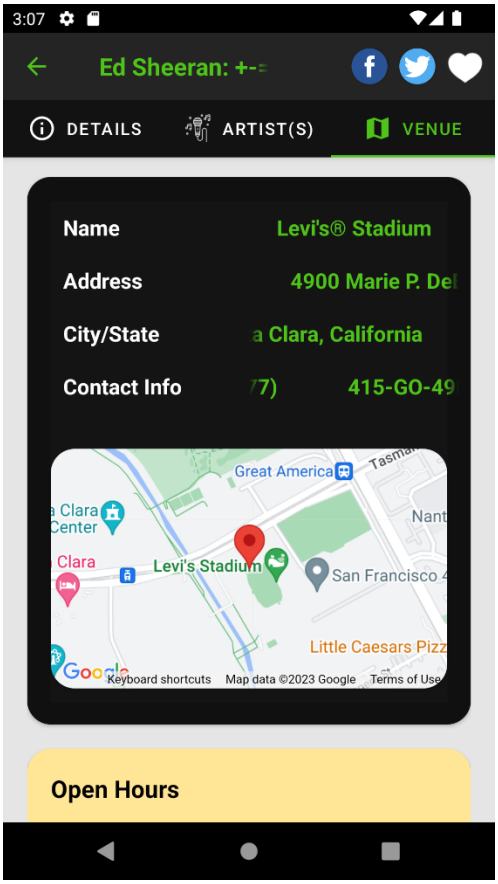

Figure 16: Venue Info with map

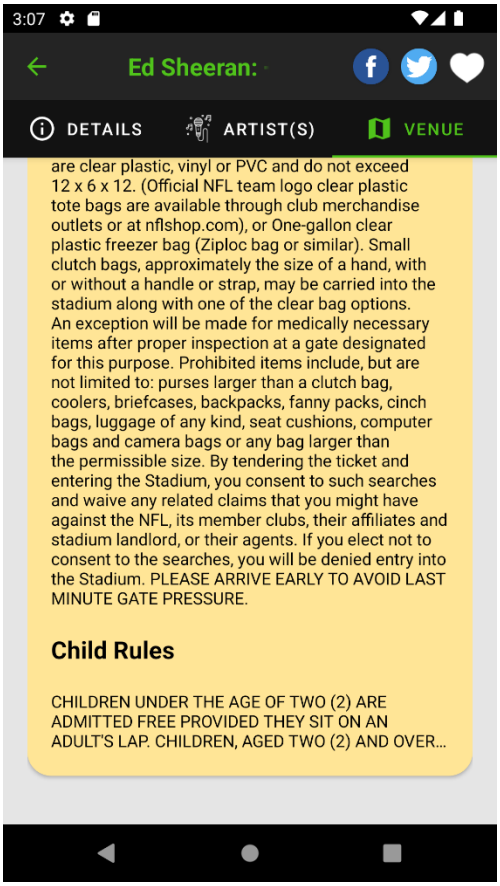

Figure 17: Other Venue Info

5.5 Favorite list

Use Tabs with a ViewPager on the main screen to switch between the search page and the favorite page. The favorite events should be displayed in a list using a RecyclerView/ListView. Each of the items in the list includes an event catalog image, event name, venue name and category and date/time, as shown in **Figure 18**. If there are no favorite events, "No Favorites" should be displayed at the center of the screen, as shown in **Figure 19**.

Like in search results, pressing the favorite icon here should remove the event from the favorites list. See video for more detail.

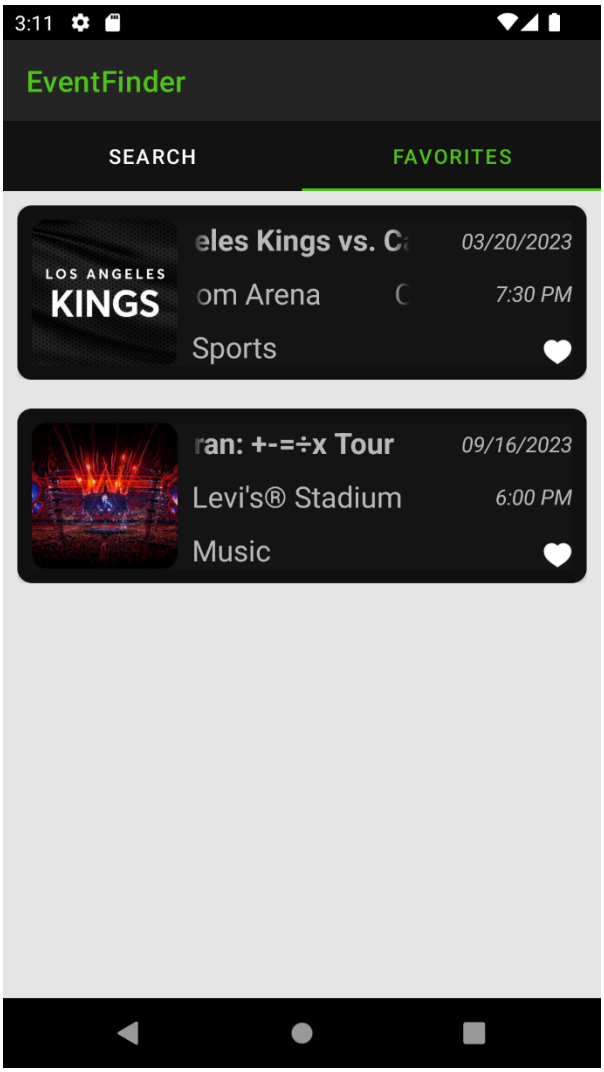

Figure 18: Favorite list

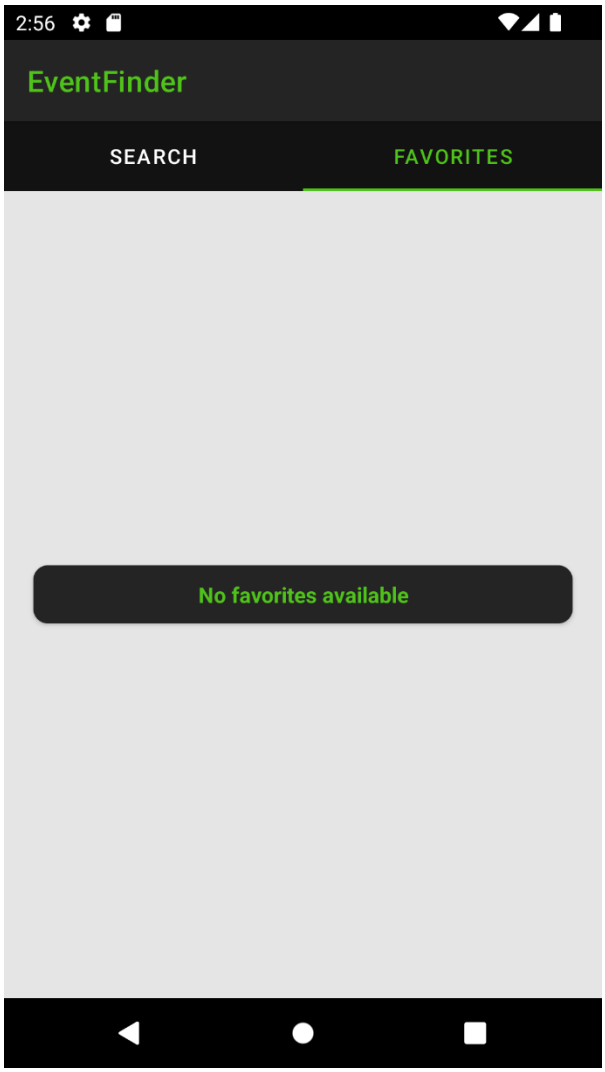

Figure 19: No favorites

5.6 Error handling

If no events are found given a keyword, a “no results” should be displayed, as shown in **Figure 20** and **21**. If for any reason an error occurs (no network, API failure, cannot get location etc.), an appropriate error message should be displayed at the bottom of screen using a Toast.

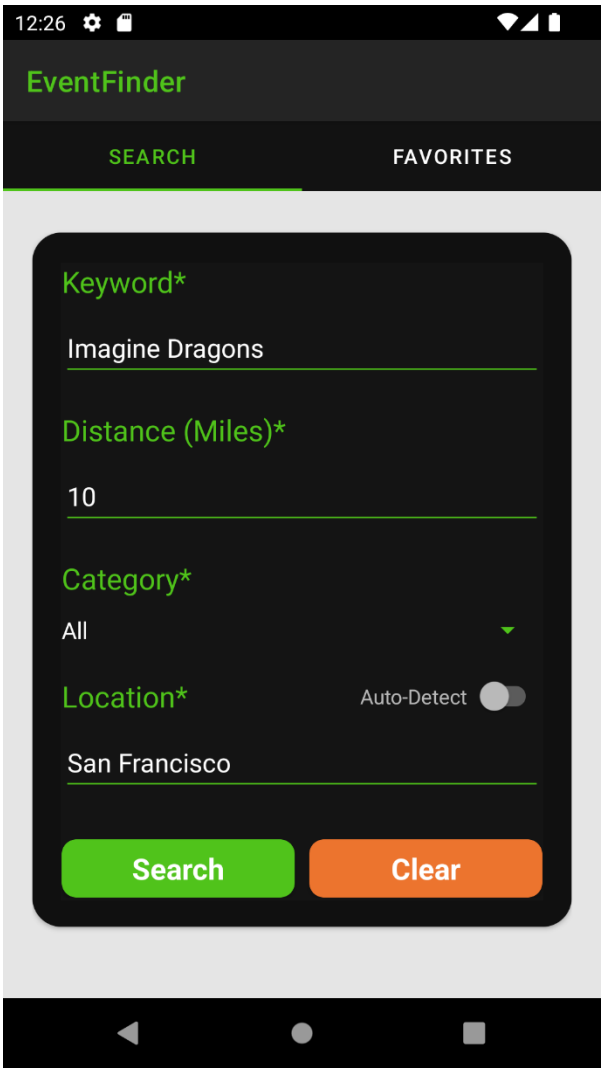

Figure 20: Search Inputs that have no results

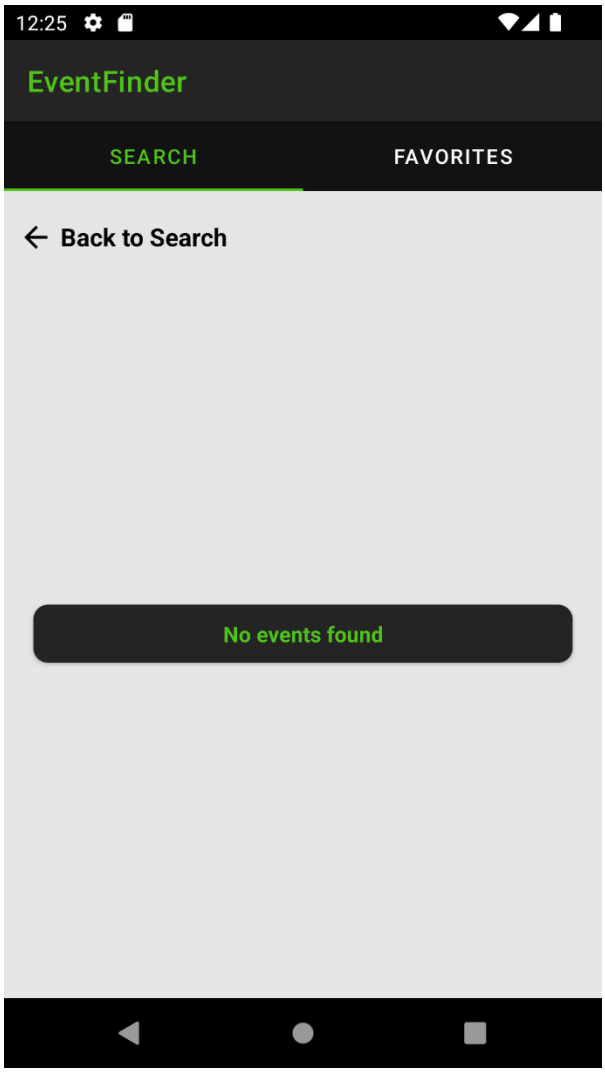

Figure 21: "No events found" message

5.7 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.
- Always display a loading message if the data is loading.
- You can only make HTTP requests to your backend Node.js on AWS/GAE/Azure and use the 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.
- Make sure there are no conditions under which the app crashes
- Make sure all icons and texts are correctly positioned as in the video/screenshots
- Make sure the screens and toasts are correctly displayed.

- Make sure the styling for different features matches the video/screenshots.
- All API calls are to be made using Node.js backend.

6. Implementation Hints

6.1 Icons and Images

The images used in this homework are available in the zip file in the D2L dropbox folder and will be linked in Piazza post for HW 9. The videos also will be uploaded on D2L and the youtube video linked on Piazza post.

Furthermore, please refer to the following websites and search for additional icons.

1. https://materialdesignicons.com
2. https://icons8.com/icons/

You can choose to work with XML/png/svg/jpg versions. We recommend using XML as it is easy to modify colors by setting the Fill Colors.

6.2 Third-party libraries

Sometimes using 3rd party libraries can make your implementation much easier and quicker. Some libraries you may have to use are:

6.2.1 Volley HTTP requests

Volley can be helpful with asynchronous http request to load data. You can also use Volley network ImageView to load photos in Google tab. You can learn more about them here: https://developer.android.com/training/volley/index.html

6.2.2 Picasso

Picasso is a powerful image downloading and caching library for Android.
http://square.github.io/picasso/

6.2.3 Glide

Glide is also a powerful image downloading and caching library for Android. It is similar to Picasso. You can also use Glide to load images..
https://bumptech.github.io/glide/

6.3 Working with action bars and menus

https://developer.android.com/training/appbar/setting-up

https://stackoverflow.com/questions/38195522/what-is-oncreateoptionsmenumenu-menu

6.4 Implementing Splash Screen

There are many ways to implement a splash screen. This blog highlights almost all of them with examples:
https://android.jlelse.eu/the-complete-android-splash-screen-guide-c7db82bce565

6.5 Adding the App Icon

https://dev.to/sfarias051/how-to-create-adaptive-icons-for-android-using-android-studio-459h

6.6 Adding ellipsis to long strings

https://stackoverflow.com/questions/6393487/how-can-i-show-ellipses-on-my-textview-if-it-is-greater-than-the-1-line

6.7 Adding a button to ActionBar

https://developer.android.com/training/appbar/actions
https://stackoverflow.com/questions/12070744/add-back-button-to-action-bar
https://stackoverflow.com/questions/34110565/how-to-add-back-button-on-actionbar-in-android-studio

6.8 Implementing a Recycler view in android

https://developer.android.com/guide/topics/ui/layout/recyclerview

https://stackoverflow.com/a/40217754

https://abhiandroid.com/materialdesign/recyclerview-gridview.html

6.9 Adding Toasts

https://stackoverflow.com/questions/3500197/how-to-display-toast-in-android

https://developer.android.com/guide/topics/ui/notifiers/toasts

6.10 Passing variables to intent

https://stackoverflow.com/questions/2405120/how-to-start-an-intent-by-passing-some-parameters-to-it

6.11 Formatting Text using HTML in TextView

https://stackoverflow.com/a/27462961

6.12 Open Link in browser
https://www.tutorialkart.com/kotlin-android/android-open-url-in-browser-activity/

6.13 Back press behavior on Back button
https://stackoverflow.com/a/27807976

6.14 SearchBar and AutoCompleteTextView
To implement the search functionality, these pages will help:
https://www.youtube.com/watch?v=9OWmnYPX1uc
https://developer.android.com/guide/topics/search/search-dialog

Working with the AutoCompleteTextView to show the suggestions might be a little challenging. This tutorial goes over how it is done to help implement it.
https://www.journaldev.com/9574/android-autocompletetextview-example-tutorial
https://developer.android.com/reference/android/widget/AutoCompleteTextView

In order to link your Search Bar with autocomplete suggestions, these links might help:
https://www.dev2qa.com/android-actionbar-searchview-autocomplete-example/

6.15 Implementation For Reservations
https://www.journaldev.com/9412/android-shared-preferences-example-tutorial

6.16 Implementing Dialogs
https://mkyong.com/android/android-custom-dialog-example/
https://androidexample.com/Custom_Dialog_-_Android_Example/index.php?view=article_discription&aid=88&aaid=111
https://medium.com/@suragch/creating-a-custom-alertdialog-bae919d2e

6.17 Sectioned RecyclerView Adapter
https://github.com/luizgrp/SectionedRecyclerViewAdapter/tree/master/app/src/main/java/io/github/luizgrp/sectionedrecyclerviewadapter/demo/example1
https://github.com/luizgrp/SectionedRecyclerViewAdapter/blob/master/app/src/main/res/layout/section_ex1_header.xml
https://github.com/luizgrp/SectionedRecyclerViewAdapter/blob/master/app/src/main/res/layout/section_ex1_item.xml
https://github.com/luizgrp/SectionedRecyclerViewAdapter/blob/master/app/src/main/res/layout/fragment_ex1.xml

6.18 Rounded buttons/Images
https://stackoverflow.com/a/13872391

6.19 GridView
https://abhiandroid.com/ui/gridview#:~:text=In%20android%20GridView%20is%20a,item%20by%20clicking%20on%20it.

6.210 Using Recyclerview inside ScrollView
https://stackoverflow.com/questions/27083091/recyclerview-inside-scrollview-is-not-working

6.21 Swiping to delete feature in RecyclerView
https://www.journaldev.com/23164/android-recyclerview-swipe-to-delete-undo#code

https://github.com/xabaras/RecyclerViewSwipeDecorator

6.22 Drag and Reorder feature in RecyclerView
https://www.journaldev.com/23208/android-recyclerview-drag-and-drop

6.23 Create Swipeable tabs with ViewPager2 and TabLayout
https://medium.com/busoft/how-to-use-viewpager2-with-tablayout-in-android-eaf5b810ef7c
https://developer.android.com/guide/navigation/navigation-swipe-view-2

6.24 Getting current location
For your location fetching code to work, you must request permission from the user (See **Figure 22**). You can read more about requesting permissions here:
https://developer.android.com/training/permissions/requesting.html

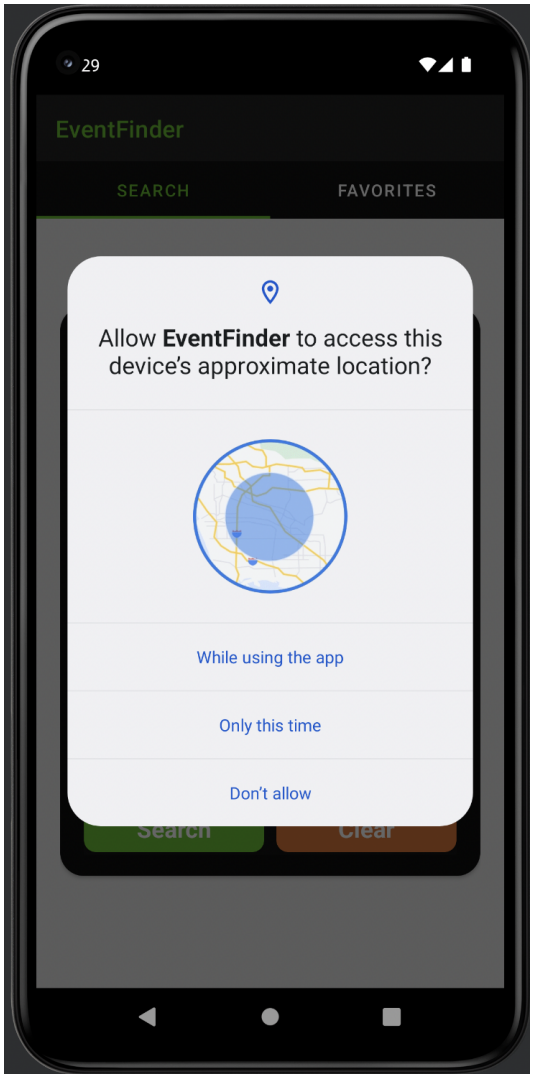

Figure 22: Requesting location permission.

You may need to mock the location in your emulator. This can be done from the emulator settings.

7. Files to Submit

You should ZIP all your source code (without image files and third-party modules) and submit the resulting ZIP file to the DEN D2L Dropbox folder.

You will also have to submit a video of your assignment demo to the same DEN D2L Dropbox folder. You must demo your submission using Zoom. Details for how to create the video are on DEN D2L.

Demo is done on a MacBook or Windows PC using the Android emulator, and not on a physical mobile device.

****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.