CSCI 571: WEB TECHNOLOGIES

HOMEWORK 9: MOBILE APP DEVELOPMENT- ANDROID

Prof. Marco Papa

Designed and Developed by: Sunny Patel

CSCI 571: Web Technologies
Homework 9: Android Weather App

Table of Contents

1. OBJECTIVES **3**

2. BACKGROUND **4**

2.1 Android Studio 4
2.2 Android 4

3. PREREQUISITES **5**

4. HIGH LEVEL DESIGN **6**

5. IMPLEMENTATION **7**

5.1 App Icon and Splash Screen 7
5.2 Home Screen / Current Location / Weather Summary View 8
5.3 Searching for a new city 10
5.4 Detailed Weather Information view 12
5.4.1 Today tab *13*
5.4.2 Weekly tab *15*
5.4.3 Google tab *16*
5.5 Favorite cities 17
5.5.1 Adding to Favorites *17*
5.5.2 Removing from Favorites *18*
5.6 Temperature Toggle 19
5.7 Progress bar 20
5.8 Summary of detailing and error handling 21
5.9 Additional Info 21

6. IMPLEMENTATION HINTS **22**

6.0 Structuring your app 22
6.1 Icons 22
6.2 Getting current location 23
6.3 Third party libraries 23
6.3.1 Google Play services *23*
6.3.2 Volley HTTP requests *24*
6.3.3 Picasso *24*
6.3.4 Glide *24*
6.3.5 MPAndroidChart *24*
6.4 Adding tab indicators for favorite cities 24
6.5 Working with action bars and menus 24
6.6 Finding resource IDs dynamically 25
6.7 Converting number to date in Java 25
6.8 Adding Ripple effect on click of a card 25
6.9 Displaying ProgressBars 25
6.10 Dynamically changing ImageView 25
6.11 SearchBar and AutoCompleteTextView 25
6.12 Implementing Splash Screen 25
6.13 Adding the App Icon 26
6.14 Dynamic coloring using DrawableCompat 26

6.15 String manipulation in Java 26
6.16 Adding ellipsis to long strings 26
6.17 Floating Action Buttons 26
6.18 Adding tabs in Android 26
6.19 Adding a button to ActionBar 26
6.20 Implementing a Recycler view in android 26
6.21 Adding Toasts 26
6.22 To create the weekly table 27
6.23 To implement favorites 27
6.24 Passing variables to intent 27
6.25 Temperature Conversion formula 27
6.26 Location using Emulator Settings 27
7. WHAT TO UPLOAD TO GITHUB CLASSROOM 28

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.

The objective is to create an Android application as specified in the document below and in the video here.

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/

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.
- You must use the **emulator**. 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 cities to see weather summary, look at detailed information about them, pin those cities to favorites and post on Twitter about the weather.

You should reuse the Node.js backend service you developed in Homework 8 and follow the same API call requirements. In case you need to change something in Node, make sure you do not break your Angular assignment (or deploy a separate copy) as the grading will not be finished at least until 2 weeks later.

PS: This app has been designed and implemented in a Pixel 3XL emulator. It is highly recommended that you use the same virtual device to ensure consistency.

Demo will be on an emulator, no personal devices allowed, see the rules:

https://csci571.com/courseinfo.html#homeworks

5. Implementation

5.1 App Icon and Splash Screen

In order to get this icon/image, go to the icon's web page specified in the hints. Using advanced export option, set the colors and a correct size to download the PNG icon.

The app begins with a welcome screen (Figure 2) which displays the icon downloaded above. This is also where we credit Darksky for using their APIs and data.
This screen is called Splash Screen and can be implemented using many different methods. The simplest is to create a resource file for launcher screen and adding it as a style to AppTheme.Launcher (see hints).

This image is also the app icon as shown in Figure 1

Figure 1: App Icon

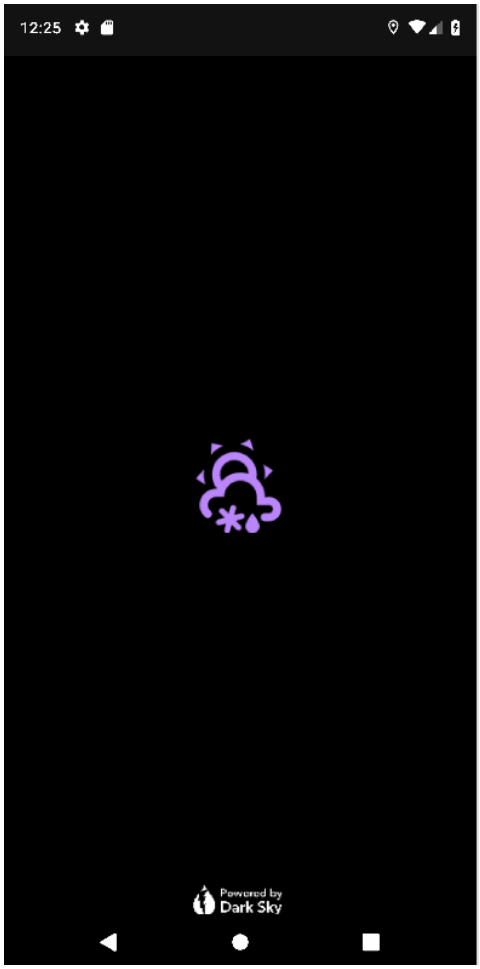


Figure 2: Splash Screen

5.2 Home Screen / Current Location / Weather Summary View

As soon as you open the app, the summary view of the **current location** is displayed. The tabs on the right swipe are described in a later section.
The location can either be fetched by the ip-api call or by using emulator settings and LocationManager.

Note: The current location tab always stays and can NOT be deleted/removed by any means.

Figure 3: The Home page for the app

This view is also called the summary view. The summary view has a selected amount of information which might be most useful for a user. It has the following fields

Card 1:

- **Icon**: "icon" property in "currently" json. The mapping of Darksky's icon field and our app's icon is given in the implementation hints.
- **Temperature**: "temperature" property in "currently" json. Make sure that the temperature float is exactly 2 digits after the decimal points followed by the Fahrenheit symbol.
- **Summary**: Summary field from the "currently" json. You must limit the summary field to upto 2 lines and terminate any longer strings with "…" at the end. See hints.
- **City**: Name of the city to which this card belongs. You can use either the autocomplete suggestion or the name returned by Darksky.
- **On click**: Clicking this card will open a new "detailed weather information view" described in section 5.4

Card 2:

This shows 4 selected values from the "currently" property. i.e. humidity, windspeed, visibility, pressure. Make sure all floats are terminated to exactly 2 digits after the decimal point. Also ensure that the correct icons are used, mapping is given in the hints.

- **"humidity**" is returned between 0 and 1 by the API - convert it to % by either mathematical operations or by dropping everything before "." followed by a % symbol.
- "**windSpeed**" property from currently json is used. The unit is mph.
- "**visibility**" property from "currently" json is used. The unit is km.
- "**pressure**" property from "currently" json is used. The unit is mb.

Card 3:

This card is used to display a quick overview of the next week's temperature. The predictions are available inside the "data" array of "daily" json. Each row in the table consists of:

- Date: "time" property converted to a MM/DD/YYYY format
- Icon: based on the "icon" property. The mapping of Darksky's icon field and our app's icon is given in the implementation hints.
- Minimum temperature: "temperatureLow" property in the json.
- Maximum temperature: "temperatureHigh" property in the json.
- **This must display all 8 days of information. You must use a ScrollView to achieve this table. See hints.**

Notes:

- JSON property names are written for reference. For exact structure please refer to HW8
- If any of the details are missing, you can use a default value of 0 or "N/A"
- All floats are rounded to exactly 2 digits after the decimal point.
- String manipulation in Java

5.3 Searching for a new city

- In a typical weather app, we need to provide only the city name to search. Our app will be a Google recommended “Searchable” app.
- On top right side, there will be a search button which opens a textbox where the user can type name of a city. See hints for icon.
- The user is provided with suggestions of city names using the places API.
- When the user taps on a suggestion, it is filled inside the search box and clicking enter/next takes the user to the next page.
- Before you get the data from your backend server, a progress bar should display on the screen as indicated in Figure 12.
- On the next page, the user will then be redirected to a new page/activity which will show the summary view (described above in section 5.2).

This component involves 2 things:

- Implementing a searchable – see hints
- Implementing autocomplete – see hints

Notes:

- **If your Node backend from Assignment 8 returns only the city name and not the state and country, you can still use that.**
- If you still have to change Node backend, make sure you don’t break the Angular app (if grading is still in progress) or deploy a newer instance of Node.
- **Reusing the summary view and the corresponding logic to dynamically set the value will make it much faster to implement this.**
- Do not forget the static “Search Results” label on top.
- **Favorites button:** This button will add/remove a city to/from the favorites. This can be implemented using a Floating Action Button. The icon of the button should also change based on whether the city currently belongs to favorites or not. For more details see video and implementation hints.
- **Clicking on the top card, also opens the same details view as described in the section 5.4**

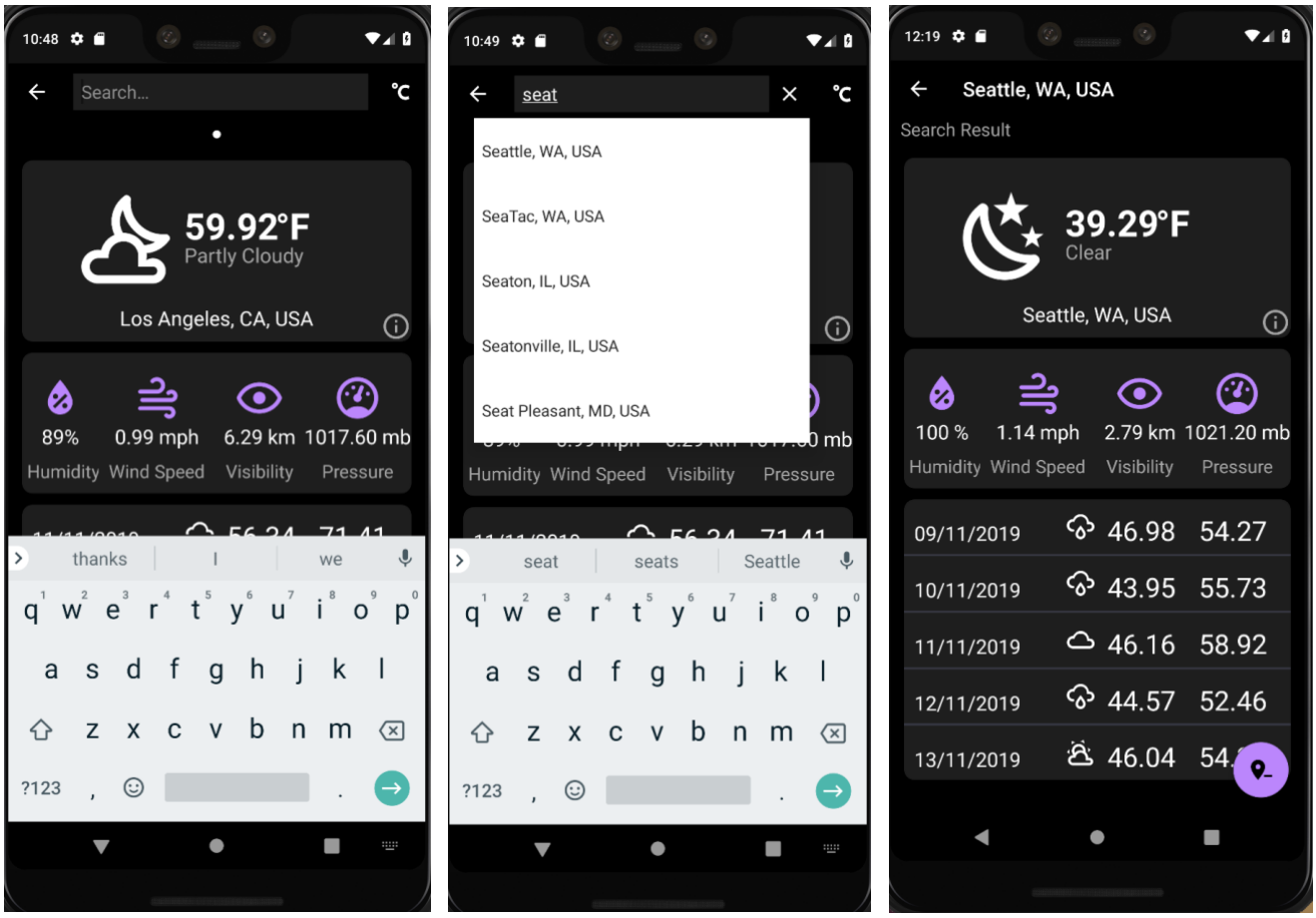


Figure 4: Flow for searching a city

5.4 Detailed Weather Information view

- Notice the information icon on card 1 in the weather summary view. When the user clicks on this card, a detailed view about the corresponding city is displayed.
- Add a ripple effect to the card.
- The detailed view has 3 tabs. These tabs are called “Fragments”. They can be implemented either with a tabbed activity or generated programmatically (see hints).
- The tabs are called “Today”, “Weekly” and “Photos”.
- There is a Tweet button in the Action Bar click which opens a browser with the Twitter intent similar to Hw8. See hints.
- The Tweet should be of the form: (Check Out CITY’s Weather! It is X°F! #CSCI571WeatherSearch)

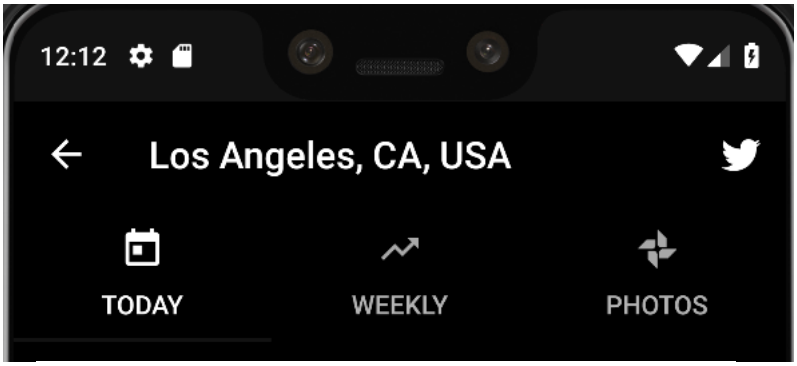


Figure 5: Tab view with back and Twitter buttons and city name

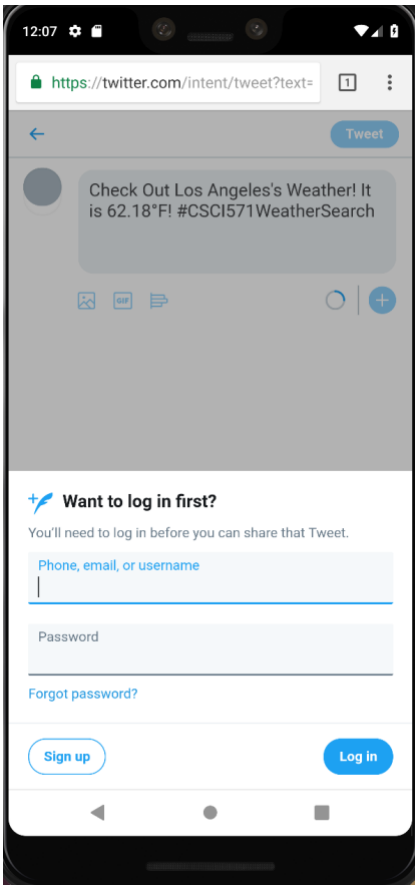
- 

Figure 6: Twitter intent

Notes:

- You can try to pass the weather JSON from the summary view to the Detailed View. You can also choose to query Darksky again. In any case, there should be a progress bar every time you are waiting for data.
- The default tab to be opened is the Today tab.
- Notice that the default colors of the icon and the tab name is gray and that for the active tab is white (see hints).
- The back button takes the user back to the summary view. See hints.
- The app bar must also contain the name of the city that the current view is for.

5.4.1 Today tab

The Today tab contains information from the “currently” property in JSON. It contains 9 cards with a gray background an icon, description and the value of that parameter. See hints
The 9 cards in this tab indicate:

1. **Wind Speed**: “windSpeed” property of the “currently” json. Unit: mph
2. **Pressure**: “pressure” property of the “currently” json. Unit: mb
3. **Precipitation**: “precipIntensity” property of the “currently” json. Unit: mmph
4. **Temperature**: “temperature” property of the “currently” json. Unit: F
5. **Icon**: See notes below
6. **Humidity**: “humidity” property of the “currently” json. Unit: %
7. **Visibility**: “visibility” property of the “currently” json. Unit: km
8. **Cloud Cover**: “cloudCover” property of the “currently” json. Unit: %
9. **Ozone**: “ozone” property of the “currently” json. Unit: DU

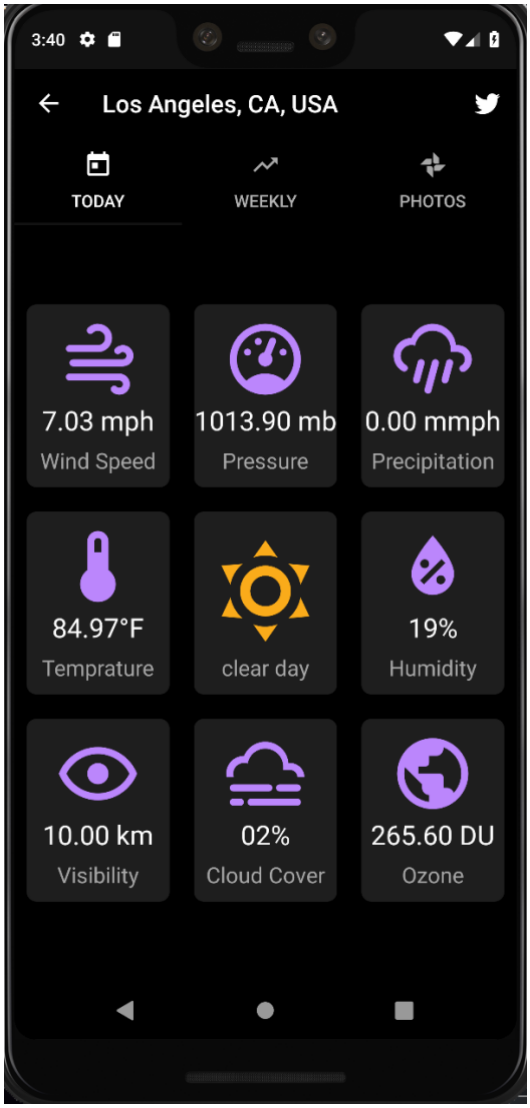


Figure 7: Today tab

Notes:

- For the “icon” card:
  - Based on the “icon” property of the “currently” json.
  - For the text below icon, replace “- “s with space and display the icon.
  - If the icon is “partly-cloudy-day” or “partly-cloudy-night” display only “cloudy day” and “cloudy night” correspondingly.
  - All possible values of icon can be found here: https://darksky.net/dev/docs#response-format
  - See hints for how to manipulate strings.
- Handle any missing values, otherwise the app can crash.
- Make sure all floating values are exactly 2 digits after the decimal point
- If a value is not present/malformed you can choose to display 0 or NA
- Make sure that the cards are well-styled and have appropriate sizes with centered images/texts

5.4.2 Weekly tab

This tab indicates the what the weather will look like in the coming week. The data belongs to "daily" properties' "data" array. There are 2 components to this page:

Card: This card is similar to the ones we used earlier. It contains:

- **Icon**: This image is the indicator for the week. See implementation hits for icon list.
- **Summary**: indicates the summary for week. From "summary" in "daily" json
- Make sure the alignment is good and the icon and text are correctly sized.

Graph view:

- This view uses **MPAndroidChart** 3rd party library. You must add this as a dependency.
- This chart contains two LineCharts.
- Extract the "temperatureLow" and "temperatureHigh" for each day from the "data" array and display the Lines in given color.
- Make sure you change the Legend, font and line colors.
- There are 3 hints which should be enough to implement this graph view.

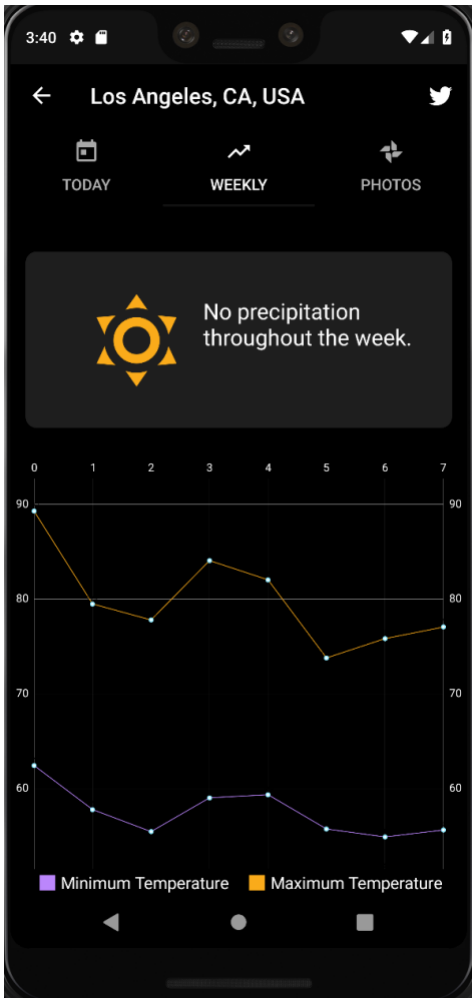


Figure 8: Weekly Tab

5.4.3 Google tab

Show **8** photos from Google custom search API. These photos are fetched using the name of the city. See Figure 9.

- You can use Volley Network ImageView, Picasso or Glide to load the image. See hints
- You must use a RecyclerView to implement this tab. A RecyclerView is the modern way of creating lists in an Android app.
- This is a very simple view with just one CardView having an ImageView. See hints

Notes:

- There are cases when the app crashes with an exception because of loading large images. Once implemented, you can turn off this functionality (of course it must work during the demo) to avoid dealing with frequent crashes.
- If your Node backend from Assignment 8 supports only fetching Seal images, then you will have to change it. If you have to change Node, make sure you don't break the Angular app (grading is still in progress) or deploy a newer instance of Node.

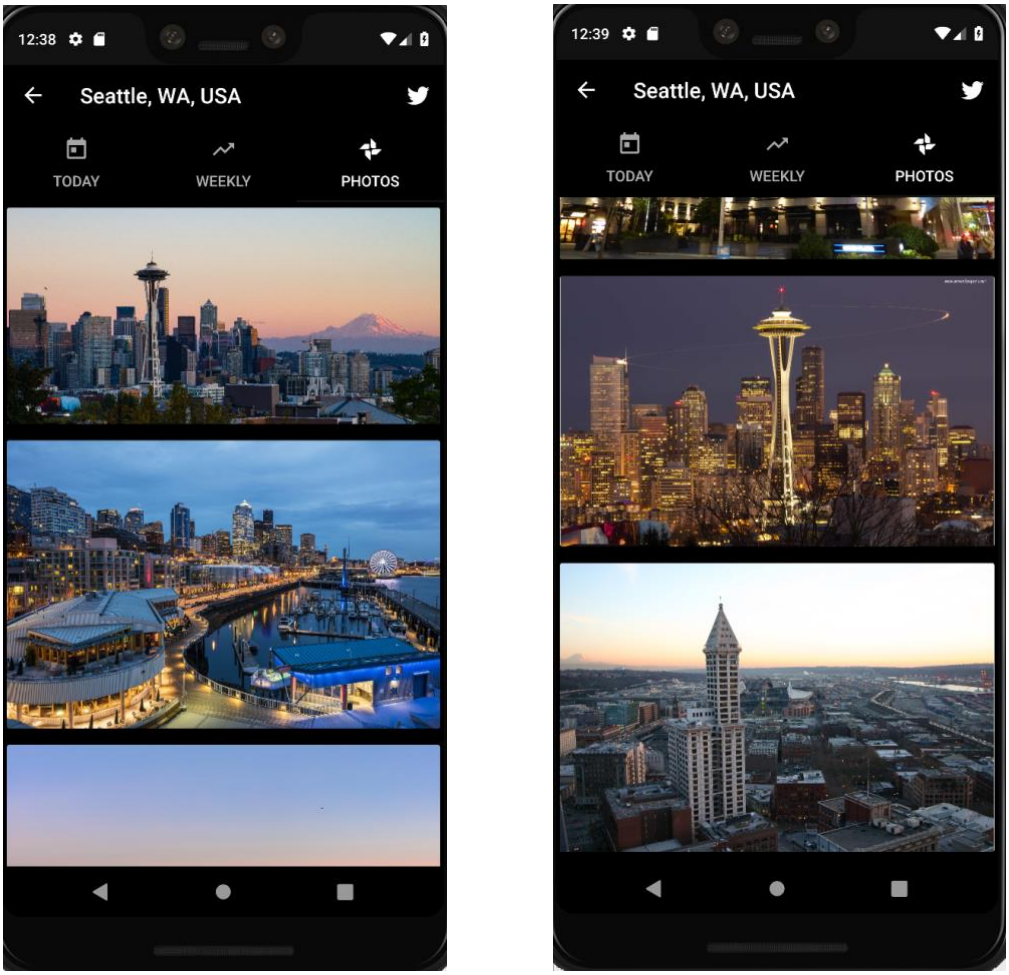


Figure 9: Photos tab

5.5 Favorite cities

This will be one of the most complicated parts of the assignment. The idea is to give users an ability to add a city to Favorites. The favorite cities are persisted even after the user has closed the app.
The favorite cities are added as a Dynamic page/tab/fragment on the home page.

Again, consider reusing the view and the code that you wrote for search and the home location tabs. The view, onclick behaviors are all the same with additional buttons.

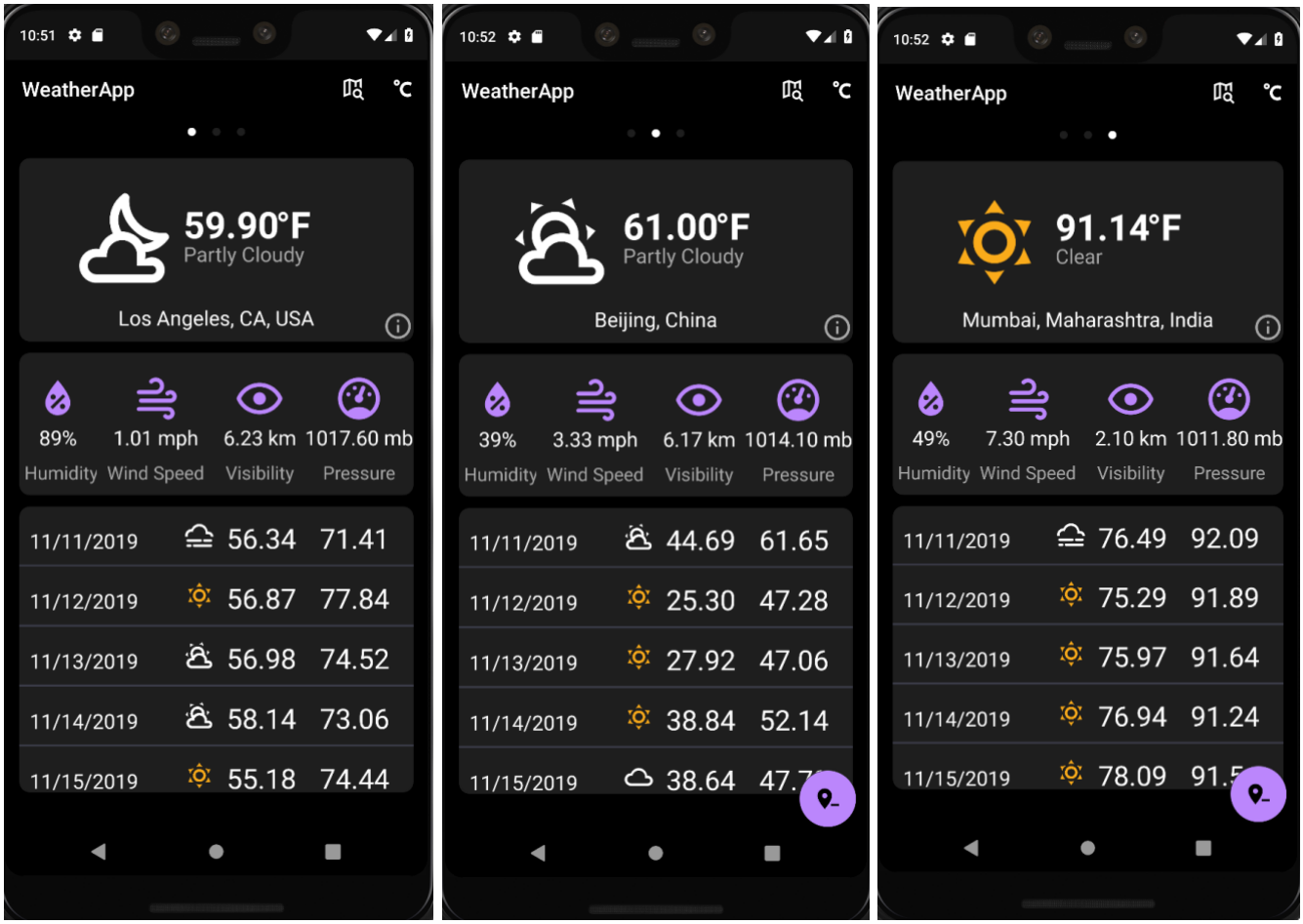


Figure 10: Favorite cities – Left to right: Default location without remove button, two cities added to favorites. Also notice the dot indicators on top

5.5.1 Adding to Favorites

Once a user has searched for a city, the summary view contains a FloatingActionButton which indicates “add to favorites” – icon of a map pin with a plus sign.

5.5.2 Removing from Favorites

"Remove from favorites" is the same FloatingActionButton as add, but with a different icon of marker with a minus sign. This button appears on 2 places:

- On the corresponding dynamically added page on the home screen
- On the search result page when the city is searched again.

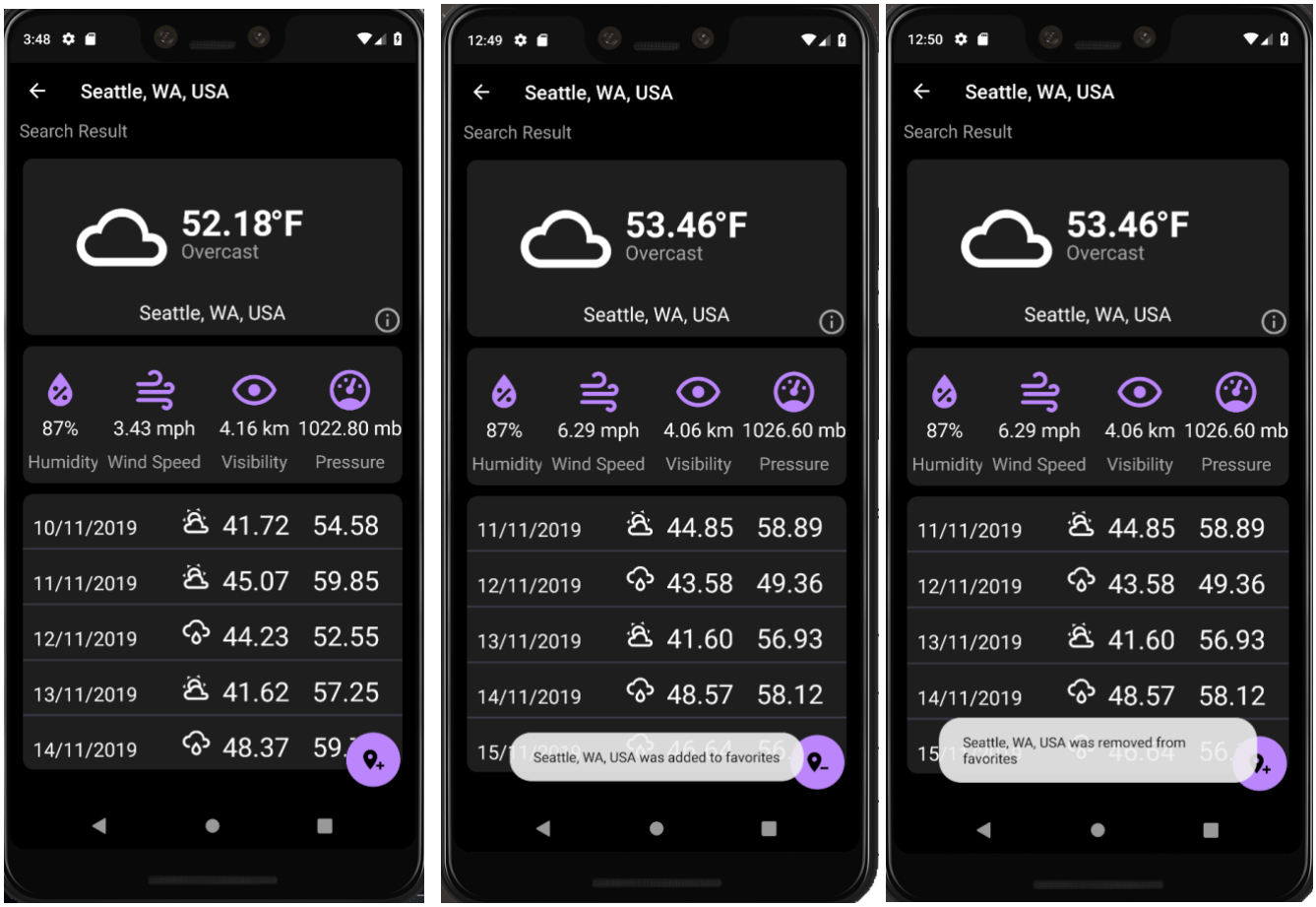


Figure 11: Toast messages on adding/removing a city from favorite

Notes:

- Display "CITY was added/removed to/from favorites" using a Toast Hints.
- Make sure that the add/remove icon toggles correctly.
- The favorite cities must persist across sessions i.e. Opening and closing should retain the view. You can use SharedPreferences to do this like LocalStorage in hw8. See hints
- **The favorite cities are also indicated with dots to indicate the active tab.** See hints
- **The favorites tab must update dynamically when the user comes back to home screen from a search query**. E.g. If the favorites are empty and the user searches for Seattle, from the search summary view if the user clicks the "add to favorite" button and navigates back to the home screen, instead of just current location there should now also be the Seattle tab! And similarly, removing a favorite city must delete the tab.
- **The dynamic tabs can be implemented in multiple ways. See hints**

5.6 Temperature Toggle

- Clicking on the F or C button changes the temperature throughout the app to the relevant unit. See hints
- This acts like a toggle which indicates the mode of the app. This can be achieved by passing around an intent variable or by persisting it with the SharedPreferences and extracting it on every Activity. See hints
- The default unit is Fahrenheit, and it can be flipped back and forth between F and C
- This toggle affects:
    - The values on the card1 in the summary view.
    - The values in the card3 in the summary view.
    - The values in the middle left card in “today tab”
    - The values in the graph in “weekly” tab.
    - And the F/C conversion icon itself.

Figure 13: Changes due to temperature toggle

5.7 Progress bar

Every time the user has to wait before he can see the data, you must display a progress bar. The progress bar is constant for every screen and just says “Fetching Weather”. One idea would be to display the progress bar by default (where needed) and then hide it as soon as the data is received/view is prepared.
See hints for progress bar related ideas and styling.

Note:
Based on your implementation, you might need to put progress bars on different places. During demo, there should NOT be any screen with default/dummy/placeholder values or an empty screen.

Figure 12: Progress bar

5.8 Summary of detailing and error handling

1. Make sure there are no conditions under which the app crashes
2. All floating-point numbers must have EXACTLY 2 digits after the decimal point.
3. Make sure the units are correct
4. Make sure all icons and texts are correctly positioned/centered as in the video/screenshots
5. Humidity and CloudCover must be converted to % using String or Math operations
6. Make sure the graph legends and axes are correctly displayed.
7. Make sure there is a progress bar every time there is nothing to show
8. The favorites button work as expected and must dynamically update the tabs on the home screen
9. Make sure you use the right icons as listed in the hints.

5.9 Additional Info

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.

6. Implementation Hints

6.0 Structuring your app

There are many ways you could do this. The easiest way would be to create following Activities:
(This is just a suggestion and NOT a mandate)

- Main
- Searchable
- Details

And following Fragments

- Favorites
- Tabs

Try to reuse as much of your code as you can - there are multiple places which share the same logic – Home view, favorites view and search view are the same. All the details pages are essentially the same.

All the hints below are just pointers, feel free to research and find a different way of doing things. You do NOT have to stick to these resources.

6.1 Icons

6.1.1 Icons

The images used in this homework are taken from https://materialdesignicons.com/ .

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

Icon Name	Usage
weather-partly-snowy-rainy	App Icon and Splash screen
Darksky attribution logo	https://darksky.net/dev/docs/terms#tos-attribution
weather-windy	Wind Card
gauge	Pressure Card
weather-pouring	Rain Card
thermometer	Temperature Card
water-percent	Humidity card
eye-outline	Visibility card
weather-fog	Cloud Cover Card
earth	Ozone card
information-outline	Information icon for card on summary views
map-marker-plus	Add to favorites
map-marker-minus	Remove from favorites

map-search-outline	Search bar icon
calendar-today	For “Today” tab
trending-up	For “Weekly” tab
google-photos	For “Photos” tab
twitter	Tweet Icon

6.1.2 Mapping between Darksky summarys’ to app’s icons

Icon Name	**Darksky Summary**
weather-sunny	Default icon and for clear-day (Use yellow color)
weather-night	Clear-night
weather-rainy	rain
weather-snowy-rainy	sleet
weather-snowy	snow
weather-windy-variant	wind
weather-fog	fog
water-cloudy	cloudy
weather-night-partly-cloudy	partly-cloudy-night
weather-partly-cloudy	partly-cloudy-day

6.2 Getting current location

In order to get the current location, you can use either ip-api or location services.
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

You may need to mock the location in your emulator. This can be done from the emulator settings. See 6.11

6.3 Third party libraries

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

6.3.1 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 Volley HTTP requests

Volley can be helpful with asynchronously 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

6.3.3 Picasso

Picasso is a powerful image downloading and caching library for Android.

http://square.github.io/picasso/

If you decide to use RecycleView to display the photos with Picasso Please use version 2.5.2 since latest version does not support RecycleView well.
https://github.com/codepath/android_guides/wiki/Displaying-Images-with-the-Picasso-Library

6.3.4 Glide

Glide is also powerful image downloading and caching library for Android. It is similar to Picasso. You can also use Glide to load photos in Google tab.

https://bumptech.github.io/glide/

6.3.5 MPAndroidChart

In order to create graphs we will use the MPAndroidChart library. Full code and documentation can be found here -https://github.com/PhilJay/MPAndroidChart

For our purposes following these links should suffice:
Creating a LineChart: https://weeklycoding.com/mpandroidchart-documentation/setting-data/
Styling the Legend: https://github.com/PhilJay/MPAndroidChart/wiki/Legend
Styling the label: https://stackoverflow.com/questions/28632489/mpandroidchart-how-to-set-label-color

6.4 Adding tab indicators for favorite cities

https://stackoverflow.com/questions/20586619/android-viewpager-with-bottom-dots
https://stackoverflow.com/questions/38459309/how-do-you-create-an-android-view-pager-with-a-dots-indicator

6.5 Working with action bars and menus

https://developer.android.com/training/appbar/setting-up
https://stackoverflow.com/questions/38195522/what-is-oncreateoptionsmenumenu-menu

6.6 Finding resource IDs dynamically

This can be helpful in setting the 5 days table in the summary view.
https://stackoverflow.com/questions/4427608/android-getting-resource-id-from-string

6.7 Converting number to date in Java

https://stackoverflow.com/questions/535004/unix-epoch-time-to-java-date-object

6.8 Adding Ripple effect on click of a card

https://stackoverflow.com/questions/26942434/ripple-effect-on-android-lollipop-cardview

6.9 Displaying ProgressBars

https://stackoverflow.com/a/28561589
https://stackoverflow.com/questions/5337613/how-to-change-color-in-circular-progress-bar

6.10 Dynamically changing ImageView

This will be used for icon fields based on summary property.
https://stackoverflow.com/questions/2974862/changing-imageview-source

6.11 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 so that you get an idea of how to go about it.
https://www.truiton.com/2018/06/android-autocompletetextview-suggestions-from-webservice-call/

In order to link your Search Bar with autocomplete suggestions, these links might help:
https://www.dev2qa.com/android-actionbar-searchview-autocomplete-example/https://stackoverflow.com/questions/34603157/how-to-get-a-text-from-searchview

6.12 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

6.13 Adding the App Icon

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

6.14 Dynamic coloring using DrawableCompat

https://medium.com/@hanru.yeh/tips-for-drawablecompat-settint-under-api-21-1e62a32fc033

6.15 String manipulation in Java

https://www.guru99.com/java-strings.html

6.16 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

6.17 Floating Action Buttons

https://developer.android.com/guide/topics/ui/floating-action-button

6.18 Adding tabs in Android

https://www.spaceotechnologies.com/create-multiple-tabs-using-android-tab-layout/

6.19 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/38195522/what-is-oncreateoptionsmenumenu-menu

6.20 Implementing a Recycler view in android

https://www.learningsomethingnew.com/how-to-use-a-recycler-view-to-show-images-from-storage
https://developer.android.com/guide/topics/ui/layout/recyclerview

6.21 Adding Toasts

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

6.22 To create the weekly table

https://stackoverflow.com/questions/4427608/android-getting-resource-id-from-string
https://stackoverflow.com/questions/535004/unix-epoch-time-to-java-date-object
https://developer.android.com/reference/android/widget/ScrollView

6.23 To implement favorites

- Please read about how ViewPagers work: https://developer.android.com/training/animation/screen-slide
- Set OffScreenLimit to 10: https://stackoverflow.com/questions/11650152/viewpager-offscreen-page-limit
- And then follow these links:
    - https://medium.com/@rajatsingh1695/making-dynamic-tabs-and-fragments-in-android-using-json-data-70a59eb6f0e0
    - https://stackoverflow.com/questions/34306476/dynamically-add-and-remove-tabs-in-tablayoutmaterial-design-android
    - https://stackoverflow.com/questions/10396321/remove-fragment-page-from-viewpager-in-android

6.24 Passing variables to intent

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

6.25 Temperature Conversion formula

https://www.rapidtables.com/convert/temperature/how-fahrenheit-to-celsius.html

6.26 Location using Emulator Settings

If you choose to use location services to get the current location, then you can use the following emulator settings.

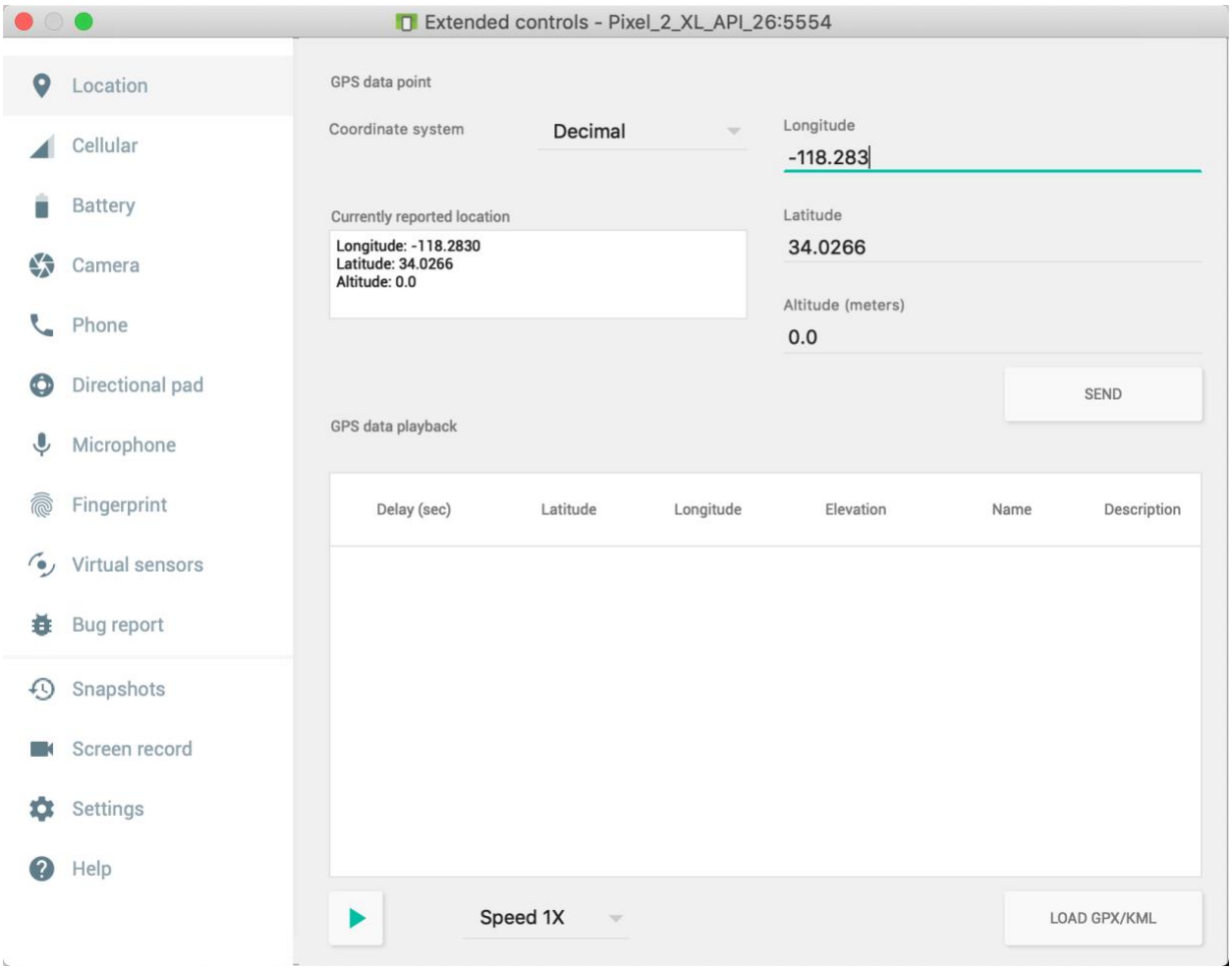


Figure 14: Location Setting of Emulator

7. What to Upload to GitHub Classroom

You should also ZIP all your source code (the java/ and res/ directories excluding the vector drawables that were downloaded) 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. **Demo is done a laptop/ notebook/MacBook or Windows PC using the emulator, and not 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.