Android 310 - Homework 2 (Due May 20th)

Description

Homework 2 is a continuation of homework 1. Essentially expanding on certain features to make the weather app more robust.

If you do not feel you completed homework 1 such that you can reuse it for homework 2, you may use my solution as a foundation for your homework 2.

Weather API:

For this assignment we are using the **free** tier of OpenWeatherMap. Once you create an account, you should have an API key to issue queries. To verify your API key append it to the end of the following queries for Seattle.

Current Conditions:
http://api.openweathermap.org/data/2.5/weather?id=5809844&units=imperial&APPID=

Current Conditions query by city name:
http://api.openweathermap.org/data/2.5/weather?q=Seattle&units=imperial&APPID=

Current Conditions query by lat/long:
http://api.openweathermap.org/data/2.5/weather?lat=47.61&long=-122.33&units=imperial&APPID=

or you may need to use the “find” api to search a radius and take the first city found.

Current Conditions query by multiple cities:
http://api.openweathermap.org/data/2.5/group?id=5809844,703448&units=imperial&APPID=

5-day (every 3 hours)
http://api.openweathermap.org/data/2.5/forecast?id=5809844&units=imperial&APPID=

Daily Forecast:
http://api.openweathermap.org/data/2.5/forecast/daily?id=5809844&units=imperial&APPID=

These are also the apis likely to focus on for the app. You are free to play with other apis, etc. Note, the “daily” forecast for Seattle seems buggy, but for the purposes of this app, we will use it.

Postman (REST Client)

https://chrome.google.com/webstore/search/postman?hl=en

This is the rest client for Chrome I demoed in class. If you have ever tried to hit a rest service, these tools make investigating what is happening much easier.

There are tools for other platforms and postman may also exist for firefox. However, you are on your own. Some may be better, so if you have a favorite by all means keep using it.

Design direction:

I want to give you a bit of flexibility on the “design” direction/navigation of the app. Obviously I’m not expecting anything as fancy as the apps in the play store, but they will give you ideas on how to present certain data. Also, the app requirements may cause deviations.

However, your app should strive to follow material design. Therefore, if you are unsure of how something should look, consult the material design guidelines. It should help “guide” you along with providing specifications for certain look and feel. Obviously you may have to take a liberty or two when presenting weather data.

Use Cases:

- Load images. Use the picasso library. However, you should “pre-fetch” images using the “fetch” api when you load the weather data. One thing to note is you likely will be pulling the same image several times in a forecast/daily forecast, so ideally should filter out duplicates.
    - I thought about other libraries, but Volley adds more complexity than I’d like to have you use for the HW and Glide doesn’t use the HttpResponseCache.

- Add background loading.
    - At minimum should be done on some periodic basis. Ie refreshing data based on a period such as an hour. Use the job scheduler for Android 5.0+ and the alarm manager 4.4 and below per samples/lecture from 4/15.
    - Advanced options would notice a shift in location to ensure current location is updated in the background to avoid the shift when starting the app showing previous current location. 4/29 lecture.

- Switch between cities (if not implemented in HW1)
    - Add a mechanism to switch between cities.
    - The easiest solution is placing cities in a listview, however should make sure current city is first. One way to do that is a “hidden” sort criteria in the database, that the user does not see.

- Add favorite cities:
    - Add a mechanism to add favorite cities to the application.
    - Should use the Google Places API to present a list of cities to the user via the autocomplete api demonstrated during the 4/22 lecture. Should restrict the autocomplete list to “cities” and bias the list to cities in the continental US.
    - Either use the getPlaceById to get the “place” to get the lat/long of the city, or use the geocoder api, to use the description to get the “address” and lat/long, or feed the city name directly to open weather api via the query.
    - Ideally also update the current conditions to fetch all cities with city ids in one query, using the group query.

- When app starts first time, should show the weather for current location. You should show current location as “first” location for subsequent uses of the app. However, depending on how you manage backgrounding ops, this may flash your previous current city briefly.
    - You are not required to update current location in the background. However, to avoid that location lookup that may happen when the app is launched, it is best solved by updating the location in the background periodically. Ie flashing Seattle, when the user is now in Bellevue due to the small lag that may happen getting the location, fetching the city data, etc.
    - Note: current location may be an added city, and ideally you avoid showing a city twice. They may require tracking in the database of which cities are favorites and which city is the current city. However, if you show the city twice it will not count against you provided the current city is always shown first.

Requirements (useful to setup your project)

(changes in bold)

1. In build.gradle under “app” use compileSdkVersion 22, targetSdkVersion 22, minimumSdkVersion 16, buildToolsVersion “22.0.1”
2. Use the support library and app compat v7 (version **22.1.1**).
3. In build.gradle for project, you be using 1.1.0 or 1.2.2 of the build:gradle android plugin. This will require you to have AS 1.1 or 1.2 installed, otherwise the project does not compile or import correctly without manual intervention.
4. Make sure you create/update your activities to derive from **AppCompatActivity**
5. All newly added fragments must be “support.v4” fragments or child fragments will NOT work on API 16.
6. Name your package: com.example.<your uw id >.homework2. For example mine would be com.example.joeroger.homework2
7. Name the application: H312 <uw id> ie: H312 joeroger

Delivery Instructions

Close the project in AS. (Leaving it open potentially will re-add some files).
Copy the project directory cp -R <original> <new>
Remove the following files/directories in the copy:
(note commands are unix so windows users will need to adjust)

```
rm *.iml
rm app/*.iml
rm local.properties
rm -rf .gradle/                    (this is new to cut zip size down)
rm -rf .idea
rm -rf build
rm -rf app/build
```

The project is expected to compile (no need to include an apk). Some effort will be made to get it to compile assuming the issue is an import problem, however, if you following the requirements for setting up the project this should not happen. However, if it is a “code” problem it will impact your grade. Best way to verify everything works is create the zip, then import the directory used to make the zip and verify your app is working. If it is, then it should work for me.

Name your zip file: H312-<UW id>.zip ie H312-joeroger.zip

Points: 60 points total

10 points	Load images
10 points	Add periodic background loading
10 points	Add/favorite multiple cities and ability to switch between cities
10 points	Use Google Places API to provide user choice of cities.
15 points	Add ability to show weather for current location
5 points	Ensure current location is displayed first