Android 310 - Homework 1 (Due Apr 22)

Description

This is a preliminary version of Homework 1. Essentially to get you thinking about the first phase of the weather app. Also you may use this time building a “mock” version of the UI so when it comes to populating with real data, you already have an idea of what data you need and have available.

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=

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.

You should be using toolbars (action bar), and theming your app for “lollipop” type applications. Similar to homeworks from the 250 class.

However, I also don’t want you to get boxed in. So some things that are coming in future assignments that may influence your design:

- Eventually you will need to add the ability to add additional US cities
- Add ability to “show” current city as the default (or first) location.
- You should be able to navigate between cities. Looking at the play store some use spinners, some use view pagers. However, if you are pressed for time, displaying cities in a list with at least the current temp would be good enough.
- Don’t get too fancy or you may be spending unnecessary time on look and feel instead of the actual requirements for the application. As long as you meet the requirements for the HW, you will pass.
- No tablet support is required, but bonus points would be given if your design “translates” to tablets.

Use Cases:

- **You will be using fragments.** However, type XML configured or dynamic will be up to you. Obviously any fragment & activity navigation should be similar to homeworks from 250.

- Start planning/implementing a database, content provider(s) to hold your weather data. You will need to hold current conditions, plus forecast data. You will **not** be required to manage foreign keys, but for those that are into that type of thing, this is a good opportunity to possibly practice.
    - However, you likely will need 3 tables.
    - Since there is no real custom logic to any of the tables, I'd consider using one content provider and adding a "segment" to represent different tables. You should key on this to switch tables for the the different database operations. Hint: if the segment is the tablename, it might be even easier than you think.
    - Also you should be "replacing" data as you load it. No need to maintain historical data.
    - Therefore think about your "unique" keys. In the current condition table, city id will be unique. However, for the forecast tables, you will need to use the city id/period (ie 1, 2, 3) as the unique index. See the following page for more information: https://www.sqlite.org/lang_createindex.html

- Obviously you will be using loader(s) to populate the UI.

- Current conditions:
    - Display image provided in data. (you will need to plan to pull an image. However, it likely will not be part of the first hw).
    - Display current condition string (weather:description)
    - Display current temp (base:temp)
    - Display humidity. (base:humidity)
    - Wind speed (optionally direction in N, NE, etc) (wind:speed)
    - City Name (ideally showing in the title bar)

- For the forecast:
    - With the current conditions, you should display basic information for the 5 day forecast.
    - Consists of the daily image (again not for first HW), along with high/low temps and say day of week (ie WED, THU, etc)

- If you click on a “day” you should go to a detail page which will show the 3 hour forecasts.
    - You will be (eventually) displaying the image for the period
    - Temp
    - Condition string (weather:description)
    - And a time (in the user’s preferred format 12 or 24 hour)

- You may add additional data as you see fit, provided its available. Again, spending time with something not required may cause you to have less time meeting the requirements.

- I recommend using an IntentService to load all three feeds into the database either during one call, or optionally create multiple IntentService classes to load the data. However, I’d consider splitting the JSON parsing into a separate class or classes depending on its complexity. Use the JSONReader for parsing.

- Add logic to the IntentService to “query” the last time the data was loaded and if less than 10 min, skip fetching from the network. You likely will need a database column to track this time. This is current time in app, not something you will get from the feed.

- Implement the response cache as when we load images in a future assignment, they will be able to be cached. I know you can copy the code from the sample, but please don’t.

- Follow the best practices for security as shown in the lecture. Obviously HTTPS will not be able to be done, but you should implement the other security features.

- I would focus on getting current conditions to work first. Then add the five day forecast info to the screen. Finally I’d add the detailed forecast activity/fragment when you click the “daily” forecast.

Requirements (useful to setup your project)

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 22.0.0).
3. In build.gradle for project, you be using 1.1.0 of the build:gradle android plugin. This will require you to have AS 1.1 installed, otherwise the project does not compile or import correctly without manual intervention.
4. Make sure you create your activities as an ActionBarActivity
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 >.homework1. For example mine would be com.example.joeroger.homework7
7. Name the application: H311 <uw id> ie: H311 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: H311-<UW id>.zip ie H311-joeroger.zip

Points: 60 points total

30 points	10 each. Reading network and saving data for each feed
10 points	Displaying current conditions and 5 forecasts (with image place holders)
10 points	Displaying detail forecast
10 points	Misc (to be split amongst the two screens, but not sure its 5/5 or 7/3, etc).