ITIS/ITCS 4180/5180 Mobile Application Development
Homework 3

Basic Instructions:

1. In every file submitted you MUST place the following comments:
    a. Assignment #.
    b. File Name.
    c. Full name of the student.
2. Everybody should submit the assignment individually.
3. Your assignment will be graded for functional requirements and efficiency of your submitted solution. You will lose points if your code is not efficient, does unnecessary processing or blocks the UI thread.
4. Please download the support files provided with this assignment and use them when implementing your project.
5. Export your Android project and create zip file which includes all the project folder and any required libraries.
6. Submission details: The file name should follow the following format: **Group#_HW03.zip**
7. **Failure to follow the above instructions will result in point deductions.**

Homework 3 (100 Points)

In this assignment you will build a simple weather application that will check the weather. You will be learning the usage of GSON to store data into shared preferences.

Initial Setup and API Description

You should use the Weather Underground API for getting the weather information. The API of interest is the Current Weather Data API which is based on the city name and state initials. You need to create an account in order to create an API key. Follow the below steps:

- Go to https://www.wunderground.com/signup?mode=api_signup and create a new account.
- After clicking on the activation link, login to your account, surf this link (https://www.wunderground.com/weather/api) click on **"View Billing"**
- Keep the **"STRATUS PLAN"** selected, select the Developer option, and click on **"Purchase Key".**
  - Fill in the given form with required information
  - Click on "Purchase Key"
- Now that your key is generated, you can use that to make API calls.

We will be using **wunderground hourly API** and **forecast10day API** for this app. For information related to the API please check https://www.wunderground.com/weather/api/d/docs?d=data/hourly .

The API details is as follows:

- Endpoint: http://api.wunderground.com/api/<<api_key>>/hourly/q/<<state>>/<<city_name>>.json
- Arguments:
  - api_key: this is the key that was generated earlier.
  - state: this should be a 2 letter initial representing a state. Example: CA.
  - city_name: this should be the city name. Space is replaced by "_". Example: San_Francisco

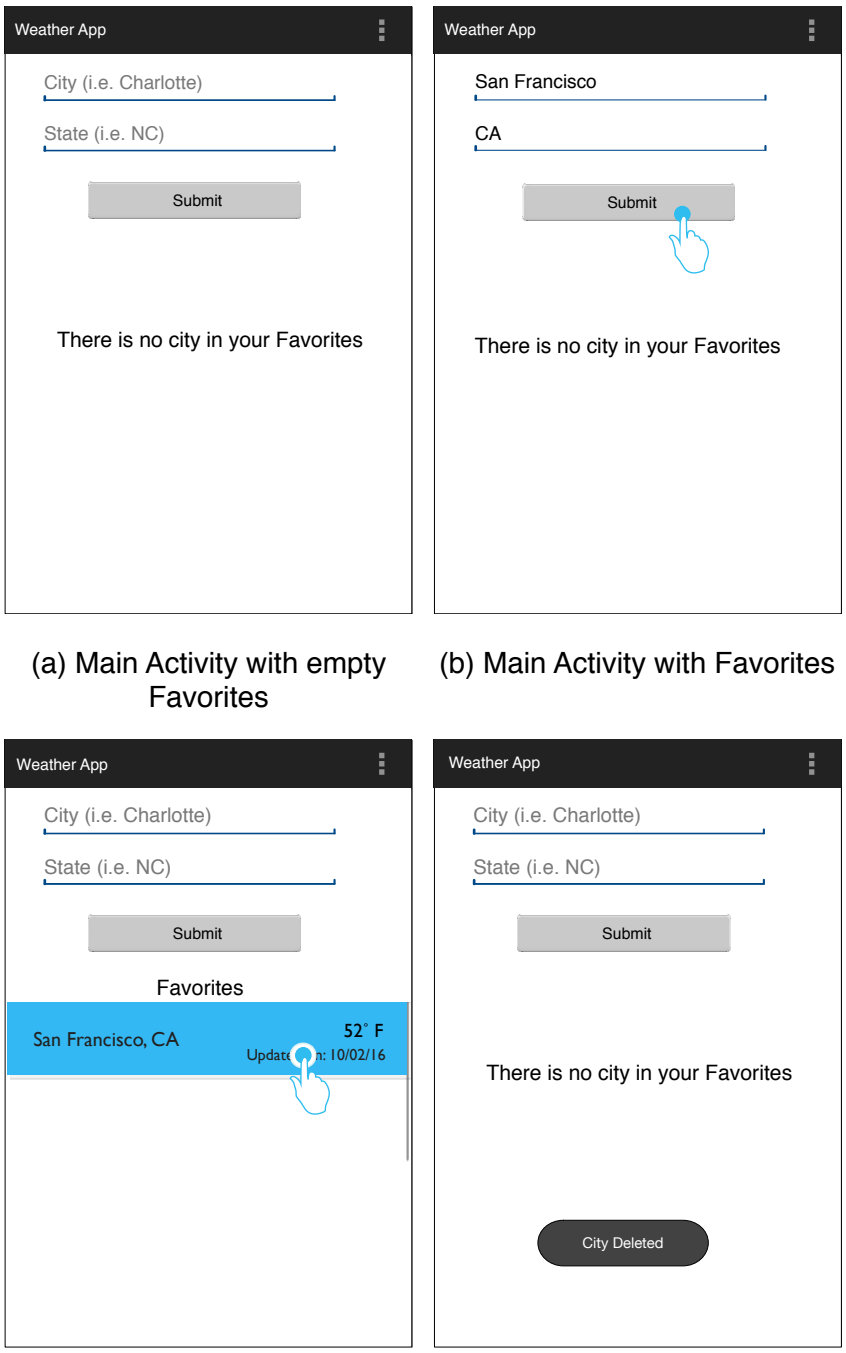
For example to retrieve the weather for San Francisco, the url should be setup as follows (replace api_key with your key):
http://api.wunderground.com/api/<<Your_Key>>/q/CA/San_Francisco.json

Part 1: Main Activity (40 Points)

The Activity UI should match the UI presented in Figure 1. Below are the requirements:

1. The activity should initially display **two Edit texts** to enable the user to enter the city and the state. Under the submit button there is a list of **Favorite Cities.** Initially the list will be empty, so you should display a message, "There is no city in your Favorites".
2. Use a RecyclerView to display the list. You should store the Favorites list (**city, state, storing date, and the temperature at storage time**) into shared preferences using the GSON library, https://github.com/google/gson.

3. After typing city and state, press the submit button. It should start the **CityWeather activity.** You should show a progress dialog, with a message “Loading Hourly Data” while your app is loading the hourly data from the API call.
4. If the api returns an error, then toast the error message describing the error encountered.



(a) Main Activity with empty Favorites

(b) Main Activity with Favorites

(c) Long Pressing to delete

(d) Deleted

Figure 1, Main Activity

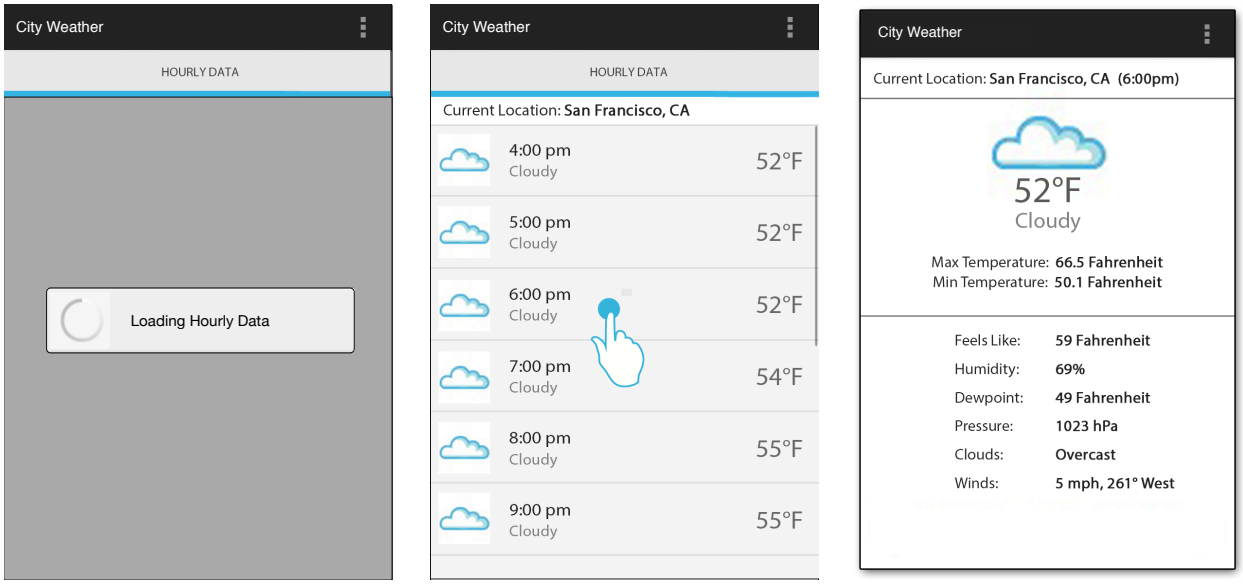
5. Long pressing on a particular city in the favorite list should delete the stored record of the selected favorite item. You should show a TOAST confirming the deletion.

Part 2: CityWeather activity (50 Points)

The CityWeather activity is responsible for retrieving the Hourly Data by calling the Wunderground api with the results based on the "cityname" and "state name" sent from the MainActivity. The Requirements is as follows:

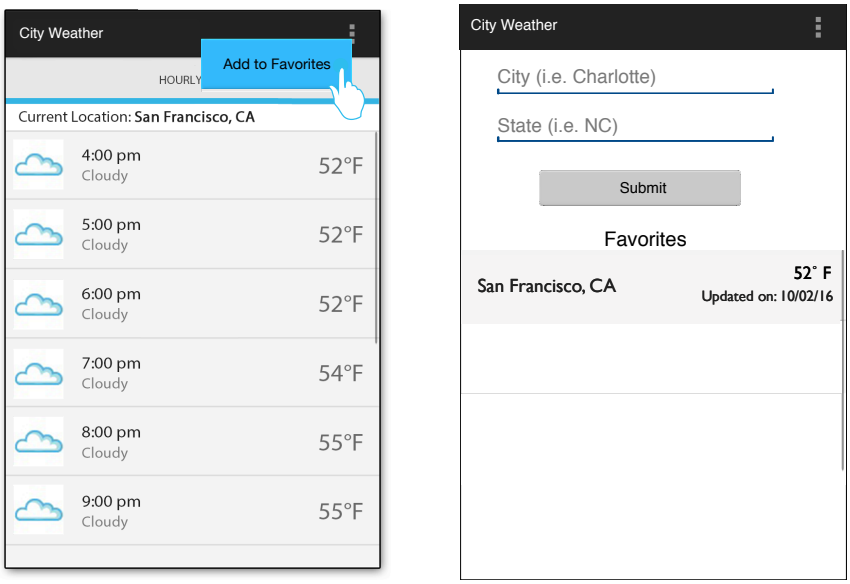
1. Form the URL to call the API.
   Example: Values sent from Main Activity:
   city_name - San_Francisco
   state_initial - CA
   URL to call:
   http://api.wunderground.com/api_key/hourly/q/state_initial/city_name.json
   The URL will be formed as below:
   http://api.wunderground.com/api_key/hourly/q/CA/San_Francisco.json
2. If "cityname", "statename" have invalid values then API will return "No cities match your query". In that case display a Toast message to the user indicating the error and finish the current and navigate back the previous activity after 5 seconds.
3. Use a thread pool or AsyncTask to communicate with the corresponding API and to parse the result. Do not use the main threads to download the results.
4. Each hour forecast details are in the hourly_forecast array. The required information is as follows:
   - All the **timing** details are under FCTTIME element.
   - Current Hour **temperature** is under **temp** element. Use the value from english attribute.
   - **Dewpoint** value is under dewpoint. Use the value from english attribute.
   - Use the value from **condition** for **Clouds**.
   - **Icon URL** is present under icon_url
   - **Wind Speed** value is under **wspd** (Use the value from english attribute) and **Wind Direction** is under **wdir**.
   - **Climate Type** is stored under **wx**.
   - **Humidity** is under humidity.
   - **Feels Like** is present under **feelslike**.
   - **Pressure** is under **mslp**. Use the value from metric attribute.
5. Create a weather class containing the following string variables: time, temperature, dewpoint, clouds, iconUrl, windSpeed, windDirection, climateType, humidity, feelsLike, maximumTemp, minimumTemp and pressure.
6. The progress dialog should be dismissed after the parsing is completed, and the hourly data items should be displayed in a list. This list should be created using a customized ListView. Each item in the ListView should contain a TextView showing the *time*, ImageView displaying the corresponding *weather icon(from icon_url)*, using the iconURL string that is retrieved from "icon_url." See Figure 2(b).
7. You can use Picasso Library or Async Task to retrieve and display the thumbnails. Check http://square.github.io/picasso/
8. **Clicking on any item in ListView should start the Details Activity** and send the ArrayList of weather objects as Extras to the Details Activity.

9. You should add a Menu having a button named “Add to Favorites”. Tapping “Add to Favorites” should add the current City, State, Date and Temperature to your Favorites List (Figure 4). Show a TOAST saying “Added to Favorites”. If the city is already stored in the favorites then simply update the stored record to reflect the



(a) CityWeather Activity (b) Hourly data (c) Details Activity

Figure 2, CityWeather Activity and DetailsActivity



(a) Add to Favorites (b) MainActivity

Figure 3, Add to Favorites

current date and temperature. Display a toast indicating “Updated Favorites Record”.

10. Use GSON to store the weather information required into the shared preferences for managing Favorite List.

Part 3: DetailsActivity (10 Points)

1. While tapping on any item of ListView in CityData activity you should start DetailsActivity. It will show the data for that particular hour as shown in Figure 2(c).

Keep Up Great Works!