IST600 Mobile App Development & Design

Weather Mobile App Final Report

Liu, Chen

04-28-2015

Table of Contents

Planning

Initial research

There are a variety of weather mobile apps in Google Play. Those apps have great features and functionalities to satisfy users. However, according to my research, only a few of them have friendly user interface and human centered interactions, which means that a lot of them might be difficult to be navigated even though they provide enough functionalities. It is not convenient for new users. Therefore, I would like to do improvements on weather mobile apps.

The objectives include:

The mobile app allows people to check out the weather in multiple cities worldwide. The weather data is dynamic, which means that users can see the weather anytime.

The mobile app not only show the weather, temperature and humidity, but it also uses various icons to represent the weather accordingly. It will be easy to read and use.

Besides, the mobile app will have friendly user interfaces and human centered interactions. Users can find the information they want in a short time and limited clicks. It is easy to be navigated than other weather mobile apps in the market.

Development environment:

Android Studio

Open Weather Map API documentation:

https://developer.forecast.io/

Current weather API:

https://api.forecast.io/forecast/APIKEY/LATITUDE,LONGITUDE

Development plan (schedule)

Date	Tasks	Deliverables
02-24-2015	1. Did Initial research (Google Play, Yahoo weather	1. 100%

	API documentation)	2. 100%
02-28-2015	1. Drafted progress report #1 2. Drafted the UI design of the mobile app	1. 100% 2. 100%
03-05-2015	1. Complete project progress report #1 2. Learn how to use API (request, response, JSON) #1	1. 100% 2. 100%
03-07-2015	1. Learn how to use API (request, response, JSON) #2 and test the code in virtual machine 2. Develop the UI(xml) of the app 3. Develop the functionalities of the app #1	1. 100% 2. 100% 3. 100%
03-11-2015	1. Develop the functionalities of the app #2 2. Develop the functionalities of the app #3	1. 100% 2. 100%
03-14-2015	1. Draft project progress report #2	1. 100%
03-23-2015	1. Complete project progress report #2 2. Improve the functionalities of the app	1. 100% 2. 60%
04-04-2015	1. Draft project progress report #3 2. Wrap up the mobile app design and development	1. 100% 2. 100%
04-10-2015	1. Test the functionalities of the app #1 2. Test the functionalities of the app #2 (usability tests)	1. 100% 2. 100%
04-14-2015	1. Fix bugs and modify functionalities	1. 70%
04-25-2015	1. App enhancement	1. 80%
04-28-2015	1. Complete project progress report #3 2. Deliver the mobile app final version	1. 100% 2.100%

Scope of the project

The scope of this mobile app will not be too broad. On the contrary, I will narrow down and only focus on a few functionalities which have high usage frequency. They are basic and also they can totally satisfy users' needs.

The main functionalities include:

- View the weather information (temperature, humidity, wind speed etc.) of the cities that people added.
- Add new cities to the list or delete cities from the list.
- Change settings. Users can choose to use 24-hour time or select other languages the mobile app provides
- Widget on the homepage of the cell phone. There will be a small widget (half screen) shows the basic weather information.
- The background color will be changed according to the temperature. If the temperature is too high, the background color will become red or orange; if the temperature is too low, the background color will become blue or silver.

Mobile user analysis

Basically, every android users could be the users of my app. However, especially for people who would like to see the weather every day and decide what to wear tomorrow will be the target users for this app.

- The main target users include:
- Business men/women
- For business men or women, they need to view the weather information before they go to work every day, so that they will know what to wear, such as long sleeve t-shirt or short sleeve t-shirt.
- Mothers
- Every mother cares about their children, so another group of target users could be mothers. They will view weather information every morning to know how to prepare clothes for their children.
- Travelers
- Travelers will view destinations' weather information before they depart. Also, before they go back to home, they need to view weather information again.

Scenario analysis

- Screen and Interaction Analysis
  The users will use this mobile app on Android smart phones. All the information of this mobile app will be displayed full screen. Basically, the interactions include touch and click. For example, when users would like to view weather information, they click the icon to open this app; when they want to add a new city, they click the add icon and type in the city that they want.

- Usage Analysis
  Users can use this mobile app on the morning every day at home, on their way to travel, and other situations as long as they want to know weather information.

- Environment Analysis
  This mobile app only can be used on smart phones, not tablet devices. It will access to the Yahoo weather API to get the weather information. It sends requests, and then get responses from the API through the internet.

Architectural design

Possible risks involve

- API
  I got weather information from forecast developer API, due to I failed bunch of times accessing to the open weather map's API. I have no idea why I failed. I registered and got an API key; however, I still failed.

- Development
  I did not have any experience with android mobile app development, even with java programming, therefore, making those elements work definitely will be a big challenge for me. I will do research first, or watch some tutorial videos online, practice by myself and then apply them on my app.
- Testing
  Different devices have different layouts and screen sizes. The GUI design and development might not fit all kinds of devices.

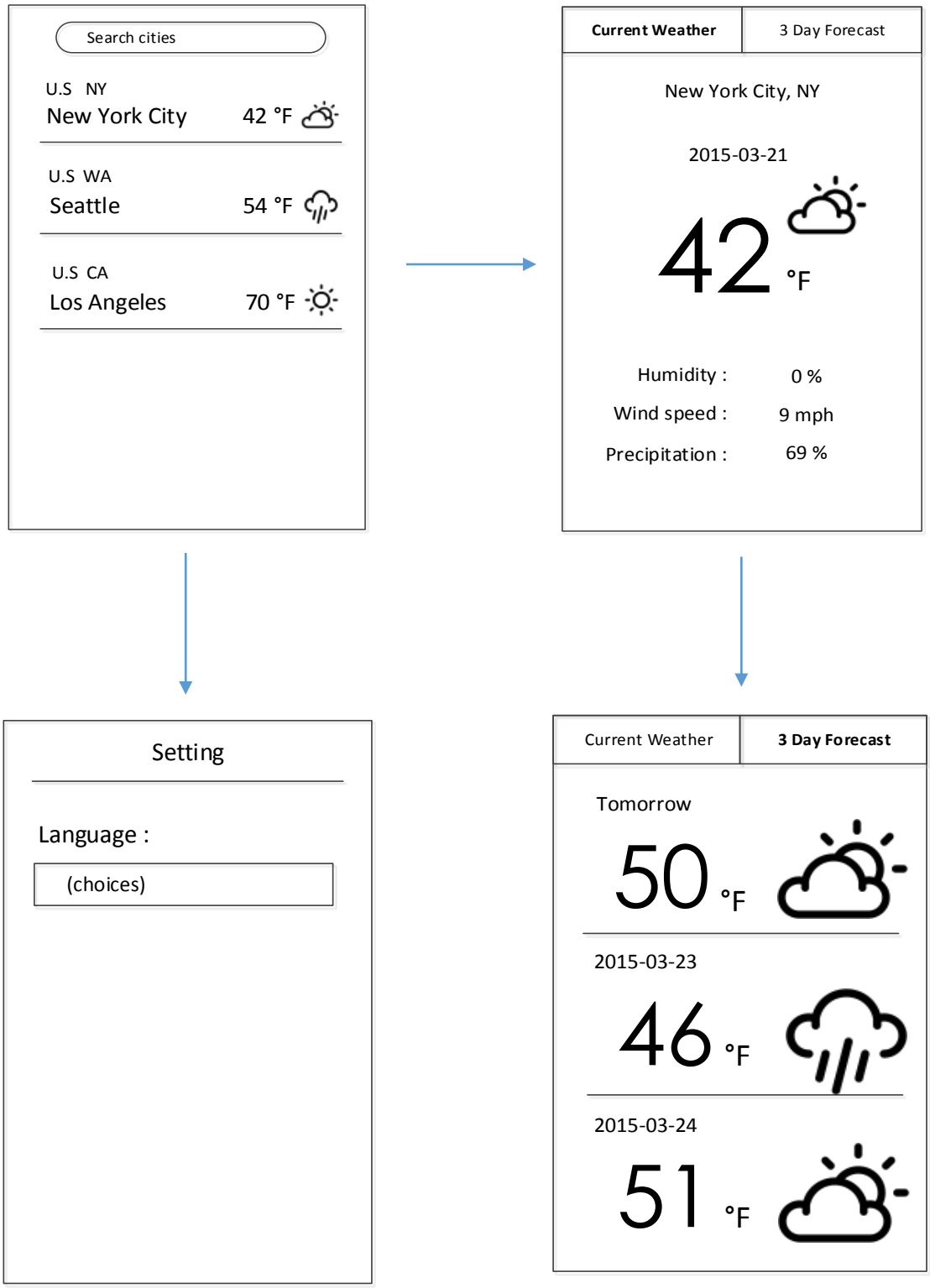
  Meanwhile, different users might have different opinions for this app and I need to consider which feedback I should listen and which I should not.

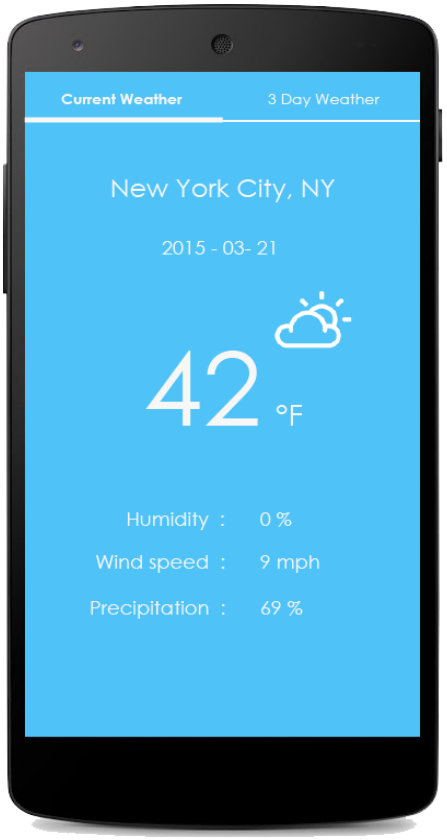
- Debug
  It might take a long time to debug and the mobile app might have few unexpected bugs. It takes time to do research and fix. I am not sure how long it will take, but I will try to make them be solved in a certain time and this project is still under control.

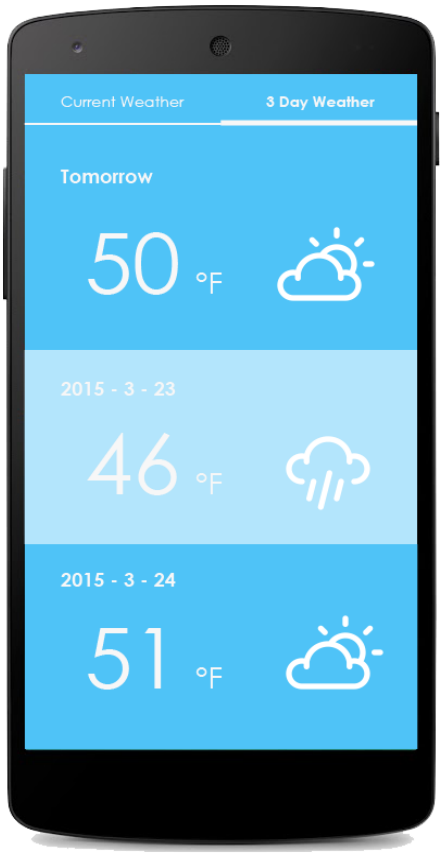
Navigation and User Interface Design

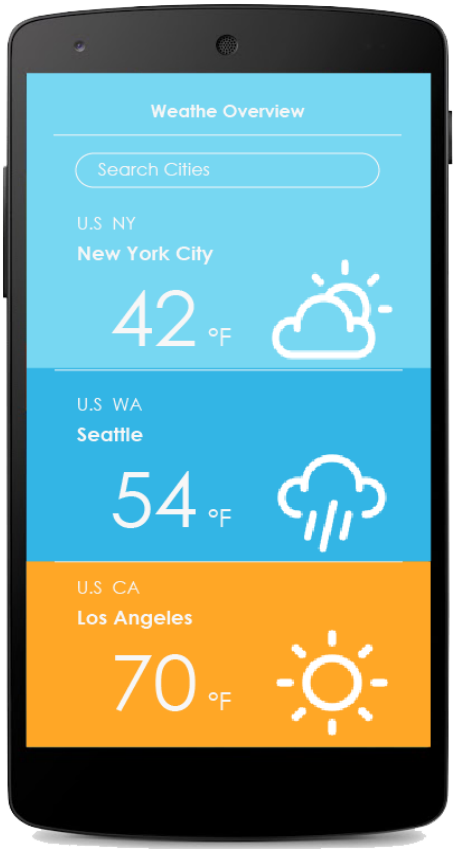
Wireframes

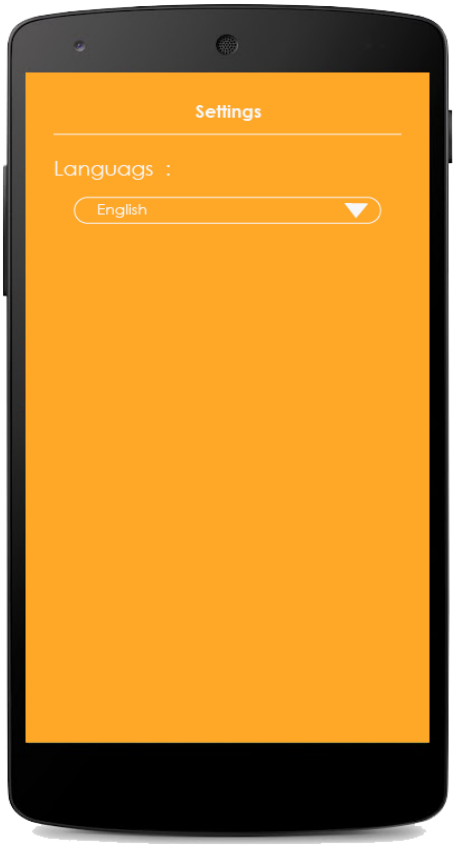
I designed wireframes for this weather mobile app which basically can display the information processes and how it works.


Search cities
U.S NY
New York City
42 °F
U.S WA
Seattle
54 °F
U.S CA
Los Angeles
70 °F
Current Weather
3 Day Forecast
New York City, NY
2015-03-21
42 °F
Humidity : 0 %
Wind speed : 9 mph
Precipitation : 69 %
Setting
Language :
(choices)
Current Weather
3 Day Forecast
Tomorrow
50 °F
2015-03-23
46 °F
2015-03-24
51 °F


Current Weather
3 Day Weather
New York City, NY
2015 - 03- 21
42 °F
Humidity : 0 %
Wind speed : 9 mph
Precipitation : 69 %


Current Weather
3 Day Weather
Tomorrow
50 °F
2015 - 3 - 23
46 °F
2015 - 3 - 24
51 °F


Weathe Overview
Search Cities
U.S NY
New York City
42 °F
U.S WA
Seattle
54 °F
U.S CA
Los Angeles
70 °F


Settings
Languags :
English

- When users open this mobile app, they will see the screen with all cities in the list. Users can add more cities in this list or remove those existing cities from the list.
- When users choose one city to read more details, they will see all the weather information about the city.
- Users also can see the weather information in the following three days by clicking the other tab on the top of this screen.
- Besides, from the homepage, users can also click "setting" page to change the language. However, the default language is English.

Colors

All the colors used in this mobile app is chosen from the Android Color official website.

http://developer.android.com/design/style/color.html

Icons, picture and sounds

- I will set a function that allow users to see weather icons along with the change of the weather, and try to get those icons from an icon website.
- There is no more other pictures and sounds will be used in this mobile app.

Implementation Details

Class files

ParseJSON.java

- This class file is responsible of parsing JSON from the Open Weather Map API

FetchJSON.java

- This class file is responsible of connecting the JSON file and fetching the JSON data from it.

XML files

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
                xmlns:tools="http://schemas.android.com/tools"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:paddingLeft="16dp"
                android:paddingRight="16dp"
                android:paddingTop="16dp"
                android:paddingBottom="16dp"
                tools:context=".MainActivity"
                android:background="#2b2b2b">


    <ListView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/citylistview"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_alignParentBottom="true"
        android:layout_below="@+id/setting"/>

    <ImageButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/setting"
        android:layout_alignParentTop="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true"
        android:background="@drawable/setting"/>

</RelativeLayout>
```

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:orientation="vertical"
              android:layout_width="match_parent"
              android:layout_height="match_parent"
              android:background="#2b2b2b">

    <ImageView
        android:layout_width="80dp"
        android:layout_height="80dp"
        android:id="@+id/imageView"
        android:layout_alignParentTop="true"
        android:layout_alignParentEnd="true"
        android:layout_alignParentRight="true"
        android:layout_marginRight="10dp"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Syracuse, NY"
        android:id="@+id/city_text"
        android:layout_alignParentTop="true"
        android:layout_marginLeft="20dp"
        android:textColor="#1ad6fd"
        android:textSize="18dp"
        android:layout_marginTop="20dp"/>
</RelativeLayout>
```

Implementation difficulties

1. I would like to use frame.xml to avoid of repeating similar codes. However, I faced some issues with it and finally I gave up. I chose to use listview element instead.
2. There is a search bar in the homepage which I never use it before. Thus, it takes time to learn how to program this element.
   After doing research, I figured out that in order to use this search bar, I should create a database to store those city information, such as names, latitude and longitude.
3. Even though JSON is an easy read file, it still takes time to learn and get data from it. In the end, I succeeded. I got the JSON data and reset the data on those textviews.
4. So far, I have tried so many times to reset background according to the temperature values. Among all those methods, I still could got errors saying that there is some something with the main thread.

APIs, code examples or libraries to be used

API

String apiKey = "084fa305d2db857db56883843b8c8d48";

String forecastUrl = "https://api.forecast.io/forecast/" + apiKey +

"/" + latitude + "," + longitude;

Code examples

Java files

```
public void popularcitylist(){
    mycities.add(new Cities("Syracuse, NY","partly_cloudy"));
    mycities.add(new Cities("New York City, NY","cloudy"));
    mycities.add(new Cities("Chicago, IL","cloudy"));
    mycities.add(new Cities("Boston, MI","cloudy"));
    mycities.add(new Cities("Los Angeles, CA","sunny"));
    mycities.add(new Cities("San Francisco, CA","sunny"));
    mycities.add(new Cities("Seattle, WA","partly_cloudy"));
    mycities.add(new Cities("Atlanta, GA","partly_cloudy"));
    mycities.add(new Cities("Austin, TX","sunny"));
    mycities.add(new Cities("Houston, TX","sunny"));
    mycities.add(new Cities("Honolulu, HA","sunny"));
}

public void populatelistview() {
    ArrayAdapter<Cities> adapter = new MyListAdapter();
    ListView list = (ListView) findViewById(R.id.citylistview);
    list.setAdapter(adapter);
}


public class MyListAdapter extends ArrayAdapter<Cities>{
    public MyListAdapter(){
        super(MainActivity.this,R.layout.item_view,mycities);

    }
```

```
public class MyListAdapter extends ArrayAdapter<Cities>{
    public MyListAdapter(){
        super(MainActivity.this,R.layout.item_view,mycities);

    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent){
        //make sure we have a view to work with
        View itemView = convertView;
        if(itemView == null){
            itemView = getLayoutInflater().inflate(R.layout.item_view,parent,false);
        }

            //find the cities to work with
            Cities currentCity = mycities.get(position);
            //fill the view

            ImageView imageView =(ImageView)itemView.findViewById(R.id.imageView);
            imageView.setImageResource(currentCity.getIconId());

            //fill the cities
            TextView cityText = (TextView) itemView.findViewById(R.id.city_text);
            cityText.setText(currentCity.getCityname());

            return itemView;
    }

}
```

Detail.java

```
private void getForecast(double latitude, double longitude) {

    String apiKey = "084fa305d2db857db56883843b8c8d48";
    String forecastUrl = "https://api.forecast.io/forecast/" + apiKey +
            "/" + latitude + "," + longitude;

    if (isNetworkAvailable()) {
        toggleRefresh();

        OkHttpClient client = new OkHttpClient();
        Request request = new Request.Builder()
                .url(forecastUrl)
                .build();

        Call call = client.newCall(request);
        call.enqueue(new Callback() {
            @Override
            public void onFailure(Request request, IOException e) {
                runOnUiThread(() -> { toggleRefresh(); });
                alertUserAboutError();
            }

            @Override
            public void onResponse(Response response) throws IOException {...}
        });
    }
    else {
        Toast.makeText(this, "Network is unavailable!",
                Toast.LENGTH_LONG).show();
    }
}
```

```
private void updateDisplay() {
    mTemperatureLabel.setText(mCurrentWeather.getTemperature() + "");
    mTimeLabel.setText(mCurrentWeather.getFormattedTime());
    mHumidityValue.setText(mCurrentWeather.getHumidity() + "");
    mPrecipValue.setText(mCurrentWeather.getPrecipChance() + "%");
    mSummaryLabel.setText(mCurrentWeather.getSummary());

    Drawable drawable = getResources().getDrawable(mCurrentWeather.getIconId());
    mIconImageView.setImageDrawable(drawable);
}

private CurrentWeather getCurrentDetails(String jsonData) throws JSONException {
    JSONObject forecast = new JSONObject(jsonData);
    String timezone = forecast.getString("timezone");
    Log.i(TAG, "From JSON: " + timezone);

    JSONObject currently = forecast.getJSONObject("currently");

    CurrentWeather currentWeather = new CurrentWeather();
    currentWeather.setHumidity(currently.getDouble("humidity"));
    currentWeather.setTime(currently.getLong("time"));
    currentWeather.setIcon(currently.getString("icon"));
    currentWeather.setPrecipChance(currently.getDouble("precipProbability"));
    currentWeather.setSummary(currently.getString("summary"));
    currentWeather.setTemperature(currently.getDouble("temperature"));
    currentWeather.setTimeZone(timezone);

    Log.d(TAG, currentWeather.getFormattedTime());

    return currentWeather;
}
```

Libraries

```
import android.content.Intent;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ImageButton;
import android.widget.ImageView;
import android.widget.ListView;
import android.widget.TextView;

import java.util.ArrayList;
import java.util.List;


import android.content.Context;
import android.content.Intent;
import android.graphics.drawable.Drawable;
import android.media.Image;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
```

```
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.ImageButton;
import android.widget.ImageView;
import android.widget.ProgressBar;
import android.widget.TextView;
import android.widget.Toast;

import com.squareup.okhttp.Call;
import com.squareup.okhttp.Callback;
import com.squareup.okhttp.OkHttpClient;
import com.squareup.okhttp.Request;
import com.squareup.okhttp.Response;

import org.json.JSONException;
import org.json.JSONObject;

import java.io.IOException;

import butterknife.ButterKnife;
import butterknife.InjectView;


import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.TimeZone;
```

Testing plan/Deployment

Planned test

I conducted a testing plan to see if the mobile app can connect to the server and get JSON data in while I am programming. Also, I will fix bugs during the development progress as many as possible. Besides, there is another test plans that let me test the entire mobile app.

Here is the basic schedule:

White Box Testing

Testing Case	Testing Result	Solution
View weather information of the default city	Working	
View the weather condition image (if it is right or wrong)	Working	(those icons on the homepages are wrong ones, but this is not hard to be fixed)
View another city's (default city) weather information by change the city's name or latitude and longitude	Working	

View current weather information	Working	
View other cities current weather information	Working	
View future weather information	Not working	I could not be able to access the data from the server (JSON). Without getting the JSON data, I cannot parse it and set it on those texts.
Select another language	Not working	First time, I used a spinner to set languages names, which allowed users to select. However, it did not work. Then I used buttons for users to select another language, however, there are few bugs with the java code

Black box testing

Testing Case	Testing Result	Solution
Randomly select a city and view its weather information	Working	
Randomly select a city and see if the search result is correct or not (test search bar functionality)	Not working	The search bar does not work well and also there is no database in the app
Randomly select a language and see if it works or not	Not working	There are few bugs with the java code

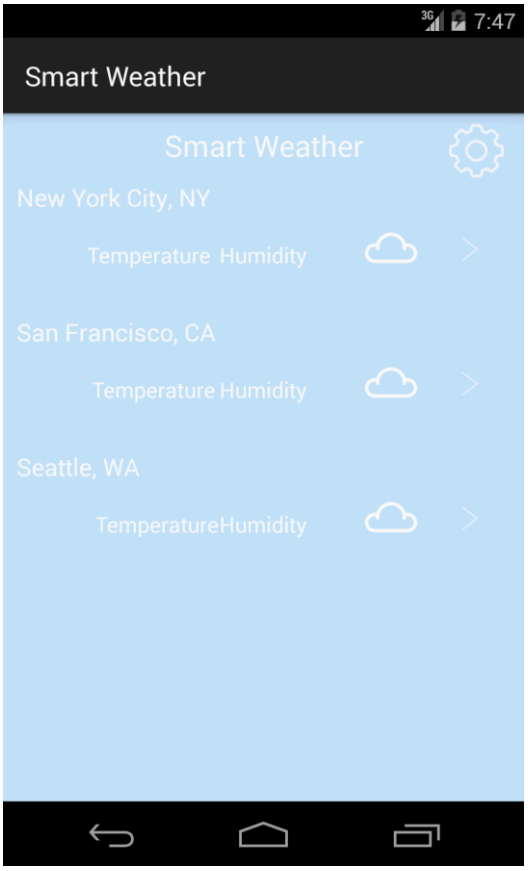
Functional testing

As I said above, few functionalities work well.

- The mobile app can connect to the internet/server
- The mobile app can send request to the API server
- The mobile app can get JSON data from the API server
- The mobile app can parse JSON data and display all the data on the screen
- Users can view every city's weather information
- Users can jump from main page to the setting page
- Once I click the item of the list on the home screen, it will go to the weather detail information page.

User acceptance testing

The first version



User 1	Result	Feedback
Select city function	Working	Not very convenient
Setting function	Working	No content
View city detail weather information	Working	This is good

User 2	Result	Feedback
Select city function	Working	I would like to add more cities if possible
Setting function	No……	
View city detail weather information	Working	The colors are too light

User 3	Result	Feedback
Select city function	Working	good
Setting function	No……	
View city detail weather information	Working	

Adjustment/corrections to be conducted

- UI design:
  The UI design does not look good due to the size of the icon. I will modify those icon or reset the size of them to make it look appropriately according to the size of the screen. Besides, the xml file is using relative layout, so some elements will look differently after modifications. I will do more tests to ensure this mobile app has friendly user interfaces.

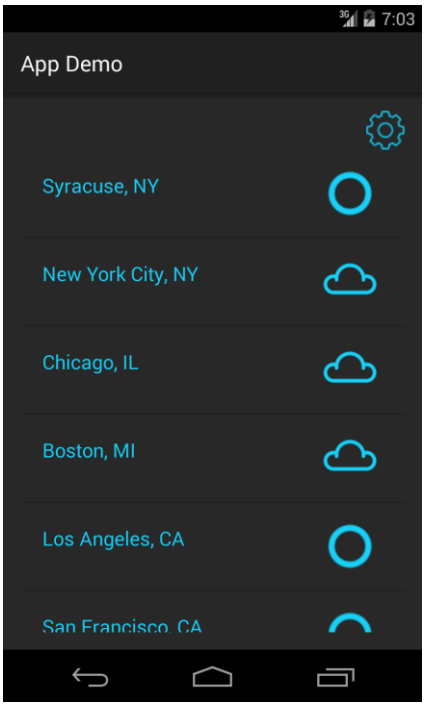
Other testing activities to be conducted

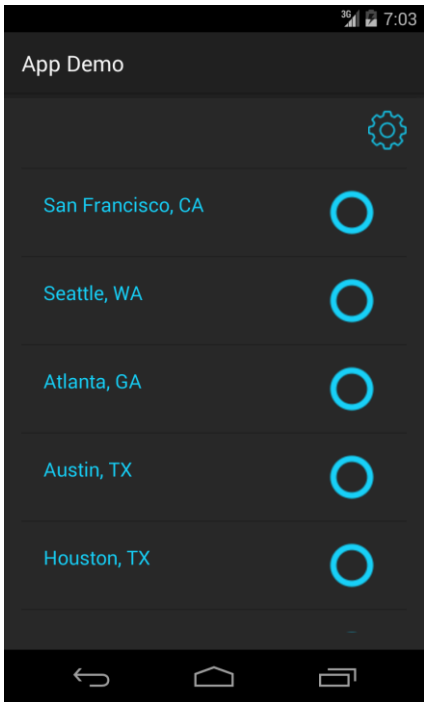
- User experience
  The user experience might be different when I test the app on a real device, due the resolution and the screen size. I will definitely do adjustments based on the data that I get.

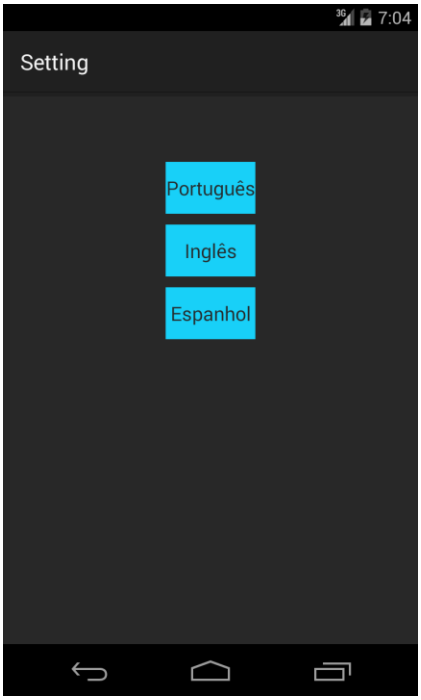
- Functionalities
  Few functionalities might be changed due to technical reasons. For an instance, I planned to display the basic weather information in a small widget in the home screen of the device. If this functionality cannot be implemented eventually, I probably will cancel it.

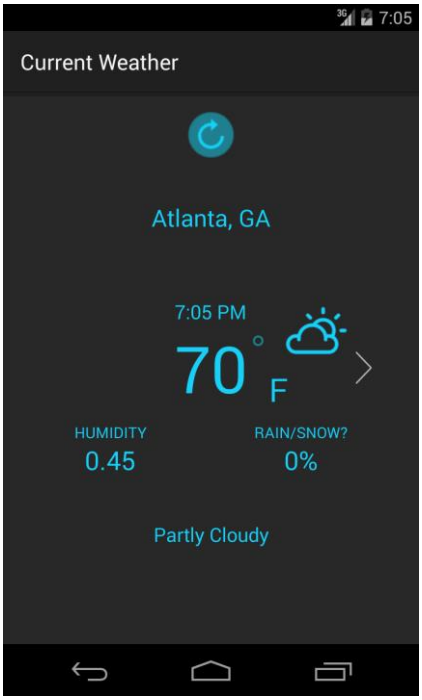
Final enhancement

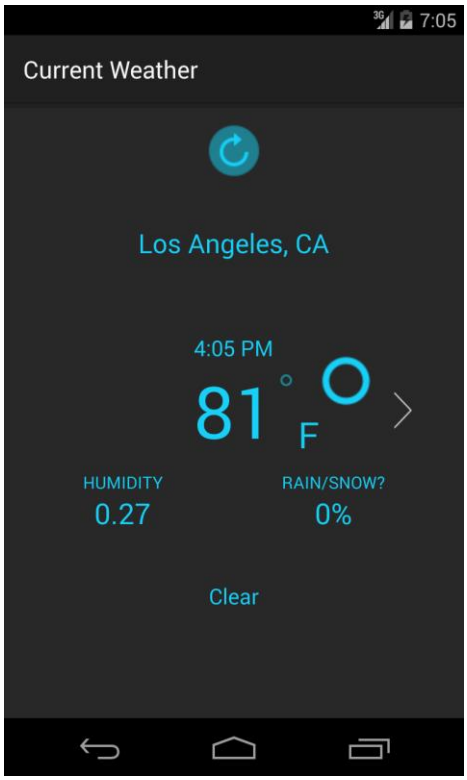
I redesign the UI that used the dark color for the background and the light color to make comparison, so users can see the texts/information clearly. Here is the screen shoot






7:04
Setting
Português
Inglês
Espanhol


7:05
Current Weather
Atlanta, GA
7:05 PM
70 °
F
HUMIDITY
0.45
RAIN/SNOW?
0%
Partly Cloudy


7:05
Current Weather
Los Angeles, CA
4:05 PM
81 °
F
HUMIDITY
0.27
RAIN/SNOW?
0%
Clear

User manual

I wrote this user manual from users' perspective. They would like to know how to use

Select city:

Open the app → select a city/scroll down to get more options

View weather information:

Open the app → select a city → click the item → view weather information

Open the app → view weather information (weather icon)

Change languages:

Open the app → click the setting icon → choose the language you want

View further weather information:

Open the app → select a city → click the item → view weather information → click forward icon to view weather information

Code Structure & Description

MainActivity.java (activity_main.xml, item_list.xml): display city list view

Detail.java (activity_detail.xml): display detail weather information, connect to the internet, send requests to the server, get response, and set the text views

CurrentWeather.java: get and parse JSON data

Future Perspectives

Few functionalities can be improved or accomplished

- Make the multi languages functionality works
- Users are able to select more cities from the search bar
- Users are able to add more cities in the list

Summary

After taking this course, I got much more confidence in Android development. I am so proud that I have learned a lot from this class and my classmates' projects.

I used to self-learning Android development and did not make any apps during that time. However, in this class, I developed a functional mobile app, even though it is not perfect. I appreciate the

professor spent a lot of time working with us, fixing bugs, discussing our ideas and providing us the devices.

Working on assignments, I learned more about a variety of elements of Android and how to make them work. I learned Java when I was in college; however, I did not use it due to I did consulting work. This course is a good opportunity to practice this skill and now I also know how to continue self-learning after this course.

Working on the project, I practiced the rapid development process, doing testing and enhancement. Meanwhile, I developed the second version, which has many improvements. The value of designing and developing this mobile app are to clarify the design process, created those files by myself, do research to fix bugs and test it with target users.